Int. J. Appl. Math. Comput. Sci., 2005, Vol. 15, No. 2, 205–220

ANT ALGORITHM FOR FLOW ASSIGNMENT IN CONNECTION-ORIENTED NETWORKS

KRZYSZTOF WALKOWIAK

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

This work introduces ANB (**A**nt Algorithm for **N**on-**B**ifurcated Flows), a novel approach to capacitated static optimization of flows in connection-oriented computer networks. The problem considered arises naturally from several optimization problems that have recently received significant attention. The proposed ANB is an ant algorithm motivated by recent works on the application of the ant algorithm to solving various problems related to computer networks. However, few works concern the use of ant algorithms in the assignment of static flows in connection-oriented networks. We analyze the major characteristics of the ANB and try to explain its performance. We report results of many experiments over various networks.

Keywords: ant algorithms, flow assignment, computer networks

1. Introduction

In recent years, we have been observing a growing role of computer networks. Due to increasing expectations for the quality of service and traffic engineering capabilities, new technologies like ATM (Asynchronous Transfer Mode) and MPLS (MultiProtocol Label Switching) are introduced to overcome the disadvantages of old protocols, e.g., the IP. Most of the new techniques apply the connection-oriented model, where routing decisions are made once, while establishing a virtual connection. Network resources must be provisioned effectively to facilitate low-cost, reliable services. Furthermore, current networks are large, and the bandwidths of links and the volume of traffic grow. Therefore, methods of network optimization, both dynamic (on-line) and static (off-line), are indispensable for the development of robust networks that can fulfil high expectations of the users. In this work we focus on capacitated static optimization of connection-oriented flows. We assume that traffic demands are known and must all be satisfied. Furthermore, we consider a capacitated problem with capacity constraint, i.e., the flow of each link cannot exceed the capacity of that link. The capacitated network design problems are much more difficult than the corresponding uncapacitated ones (Gendron *et al.*, 1998). Solving static problems is crucial for efficient design of new networks or updating the existing networks (Kasprzak, 2001; Grover 2004).

Since various optimization problems encountered in computer networks are NP-complete and some of the existing algorithms are not adequate to tackle the increasing complexity of such problems for large networks, a range of heuristic approaches are developed to deal with these problems. Some of the most promising approaches are algorithms taking inspiration from physics, biology or social sciences. The most significant examples are: genetic algorithms—simulated annealing, tabu-search, neural nets, and ant systems. These heuristics apply a certain amount of repeated trials and employ one or more agends operating using a mechanism of competition-cooperation. Applications of these algorithms cover many combinatorial optimization problems. For some examples, refer to (Colorni *et al.*, 1996; Elbaum and Sidi, 1996). In our opinion the exploitation of biological concepts might lead to new approaches for many classical network optimization problems.

In this work we focus on ant algorithms, a method proposed in 1991 by Dorigo *et al.* (1991), also called an ant system (AS) or ant colony optimization (ACO). For clarity, in the remainder of the paper we will use the term 'ant algorithm' to refer to a broad class of ant-based algorithms. The ant algorithm is a simulation of agents that cooperate to solve an optimization problem by means of communication. The inspiration comes from research on the behavior of real ants. Ants are social insects living in colonies. Their behavior is determined by the survival

of the whole colony while the importance of individual ants is not so significant. Ants can cooperate effectively in a group to perform some tasks. For instance, almost blind ants are able to find the shortest paths from their colony to feeding sources and back. It was observed that a moving ant lays some pheromone (in variable quantities) on the ground, hence marking the path it follows. Next, ants moving towards the feeding area can detect the pheromone left by the previous ant, decide with a high probability to follow it, and reinforce the selected trail with their own pheromones. This form of indirect communication mediated by pheromone placing is called *stigmergy*. Another principal aspect of real ants' behavior is a coupling between the *autocatalityc* (positive feedback) mechanism and an implicit evaluation of solutions, i.e., the more ants follow a trail, the more attractive that trail becomes for being followed. A comprehensive treatment of ants' behavior and its impact on ant algorithms can be found in (Colorni *et al.*, 1996; Dorigo *et al.*, 1991; 1999). Researchers have found many applications of ant algorithms that cover problems such as the traveling salesman problem, the quadratic assignment problem, job scheduling, vehicle routing, graph coloring, or network routing (Dorigo *et al.*, 1999).

Our discussion in this article focuses on the application of an ant algorithm to a non-bifurcated flow assignment (NBFA) problem. Connection-oriented network techniques like ATM, Frame Relay and MPLS are modeled using the non-bifurcated multicommodity flow. These techniques have gained much attention in recent years. Most optimization problems related to ATM and MPLS are similar to the NBFA problem. However, since NBFA is NP-complete, for relatively large networks only heuristic algorithms can be applied. In this work we discuss and evaluate a new ant algorithm called ANB that solves the NBFA problem. Our starting point is an algorithm proposed in (Walkowiak, 2001b).

Network flows are usually optimized according to additive or bottleneck weights (Szeto *et al.*, 2002; Kar *et al.*, 2000). The former kind of metrics assume that the cost function is computed as a sum of weights over all links. In the latter approach, the objective function is given by the maximum (or minimum) value of link weights. In this work we concentrate on network optimization applying additive weights. Additive metrics arise in many settings, e.g., cost, end-to-end delay, jitter, survivability (Kasprzak, 2001; Walkowiak, 2003a). The ANB algorithm minimizes the overall network flow—the function defined as the sum of the link flows of all the links in the network. The link flow is simply a sum of the bandwidths of the paths which traverse that link. However, ANB can be easily adapted to use also objective functions like network cost, delay, lost flow.

The paper is organized as follows: In Section 2 we briefly introduce non-bifurcated multicommodity flows. In Section 3 we present background and previous research in the field of ant algorithms applied to network problems. Section 4 contains a detailed description of ANB. In Section 5 we show the results of extensive simulations and study three main issues that arise in applying the ANB algorithm: the tuning of the algorithm parameters, convergence and comparison with other heuristics proposed previously in the literature. In the last section we draw some conclusions and outline directions for future research.

2. Multicommodity Flows

Routing algorithms used to assign network flows can be classified as static or dynamic, and centralized or distributed. Static routing assumes that the network conditions are time-invariant. Dynamic routing is applied in networks where demands frequently change. Centralized algorithms are usually applied in legacy routing systems and may have problems with scalability and inordinate demand for managing decisions requiring human attention. Distributed systems usually work locally in each network node and use only locally available information (Kassabalidis *et al.*, 2001). In this work we focus on static routing using the centralized approach. We assume that the network and all demands to be sent in the network are known *a priori*.

Various network or transportation problems can be modeled as multicommodity flow problems. The multicommodity flow problem consists in finding routes for all such commodities which minimize (or maximize) a performance function (e.g., a delay, a cost) such that a set of constraints (e.g., arc capacity constraints) is satisfied (Assad, 1978; Fratta *et al.*, 1973; Girard and Sanso, 1998; Gendron *et al.*, 1998; Grover, 2004; Kasprzak, 2001, Ott *et al.*, 2001; Pióro *et al.*, 2003; Pióro and Medhi, 2004).

We are given a network (G, c) where $G = (N, A)$ is a directed graph with n nodes and m arcs, and $c : A \rightarrow \mathbb{R}^+$ is a function that defines capacities of the arcs. We assume that all commodities included in a set P are numbered from 1 to p, where p denotes the number of all commodities. For the k-th commodity, s_k denotes a source and t_k denotes the destination of the commodity. Each commodity of the flow requirement Q_k must be routed from the node s_k to the node t_k through a given network. A multicommodity flow is a set of functions

$$f^k : A \rightarrow \mathbb{R}^+ \cup \{0\}, \quad k = 1, \ldots, p \tag{1}$$

for which the flow of the k-th commodity in the arc (x, y) $f^k(x, y)$ for $k = 1, \ldots, p$ satisfies the following condi-

tions:

$$\sum_{y\in D(x)} f^k(x,y) - \sum_{y\in B(x)} f^k(y,x) = \begin{cases} Q_k & \text{for } x = s_k, \\ -Q_k & \text{for } x = t_k, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

$$f^k(x,y) \geq 0 \text{ for } (x,y) \in A \quad k \in P, \quad (3)$$

where $D(x) = \{y : y \in N \text{ and } (x,y) \in A\}$ is a set of destination nodes of edges leaving the node x, and $B(x) = \{y : y \in N \text{ and } (y,x) \in A\}$ is a set of source nodes of edges entering the node x. The condition (2) is called the conservation of the flow at nodes. The condition (3) is a non-negativity of the flow in directed edges. The definition of the multicommodity flow given by (1)–(3) is called the node-path notation.

Here $f(x,y)$ denoting the flow of the arc (x,y) is defined as

$$f(x,y) = \sum_{k\in P} f^k(x,y). \quad (4)$$

Multicommodity flows are of two types: bifurcated and non-bifurcated. The former is a flow in which one commodity can be transported using many paths. Each path carries only a part of the commodity. For the non-bifurcated flow each commodity flows along one path only. An example of the bifurcated flow is the flow of the Internet using the TCP/IP protocols. Connection-oriented networks like ATM, MPLS, and Frame Relay are examples of networks applying non-bifurcated multicommodity flows.

Many optimization problems related to computer networks can be modeled as multicommodity flow problems. Kasprzak (2001) specifies the most important problems of computer network design:

- Flow Assignment (FA),
- Capacity and Flow Assignment (CFA),
- Topology, Capacity and Flow Assignment (TCFA).

The most common method to optimally solve problems of bifurcated multicommodity flow allocation is the linear programming approach (Bienstock, 2002; Grover, 2004; Ott *et al.*, 2001; Pióro *et al.*, 2003; Pióro and Medhi, 2004). Also, some heuristic algorithms are developed for the optimization problem considered. In (Bienstock, 2002; Burns *et al.*, 2003; Kasprzak, 2001; Murakami and Kim, 1996), algorithms based on the Flow Deviation method proposed in (Fratta *et al.*, 1973) are developed for various problems related to bifurcated multicommodity flows. Gendron *et al.* (1998) propose to apply the Lagrangean relaxation to solve capacitated network problems. Another popular method used for the optimization of bifurcated multicommodity flows are soft optimization techniques (Corne *et al.*, 2000; Grover, 2004). Other interesting algorithms for the multicommodity flow assignment problem are Extremal Flows (EF) (Cantor and Gerla, 1974) and Gradient Projection (GP) (Schwart and Cheung, 1976).

In this work we focus on a static flow assignment FA problem for non-bifurcated flows. The global multicommodity flow denoted by $\underline{f} = [f_1, f_2, \ldots, f_m]$ is defined as a vector of flows in all arcs according to constraints (1)–(4). Let Ψ_b be a set including all vectors $\underline{f}$ describing non-bifurcated multicommodity flows. An important constraint in the optimization of computer networks is the capacity constraint defined as follows:

$$\forall j \in A : f_j \leq c_j. \quad (5)$$

The inequality (5) guarantees that in every arc the flow does not exceed the capacity. Let Ψ_{bc} denote a set of all non-bifurcated flows $\underline{f} \in \Psi_b$ for which the condition (5) holds. In the rest of the paper we call a flow *feasible* if the capacity constraint (5) is satisfied.

The non-bifurcated multicommodity flow assignment (NBFA) problem can be formulated as follows:

$$\min_{\underline{f}} \; K(\underline{f}) = \sum_{j\in A} f_j, \quad (6)$$

subject to

$$\underline{f} \in \Psi_{bc}. \quad (7)$$

The objective function (6) is the overall flow in the network. In the problem (6)–(7) we wish to minimize the overall flow over all feasible non-bifurcated flows. According to (Karp, 1975), the NBFA problem is NP-complete.

In this work we apply an equivalent representation of non-bifurcated multicommodity flow assignment called link-path formulation (Fratta *et al.*, 1973; Pióro *et al.*, 2003). It is obtained by providing for each commodity $i \in P$ a set of paths $\Pi_i = \{\pi_i^k : k = 1, \ldots, l_i\}$ from the node s_k to the node t_k. For a non-bifurcated multicommodity flow, the commodity can use only one path π_i^k. Let x_i^k denote a $0/1$ variable, which equals one if π_i^k is the path for the commodity i and is equal to 0 otherwise. Another binary variable a_{ij}^k indicates whether or not the path π_i^k uses the arc $j \in A$. Using this representation of the multicommodity flow, the NBFA problem is as follows:

$$\min_{\underline{f}} \; K(\underline{f}) = \sum_{j\in A} f_j \quad (8)$$

subject to (2), (3), and

$$\sum_{\pi_i^k \in \Pi_i} x_i^k = 1, \quad \forall i \in P, \quad (9)$$

$$x_i^k \in \{0,1\}, \quad \forall i \in P,\ \pi_i^k \in \Pi_i, \tag{10}$$

$$f_j = \sum_{i \in P} \sum_{\pi_i^k \in \Pi_i} a_{ij}^k x_i^k Q_i, \tag{11}$$

$$f_j \leq c_j, \quad \forall j \in A. \tag{12}$$

The objective function (8) is the cost of the flow in the network. To simplify the problem we assume that the cost of every commodity is the same. Therefore, in (8) we sum the flows of all links in the network. Since we consider the non-bifurcated multicommodity flow, the condition (9) states that each commodity can use only one primary route. The constraint (10) ensures that decision variables x_i^k are binary. The condition (11) is a definition of a link flow. Finally, (12) is a capacity constraint. If we change the constraint (10) to

$$0 \leq x_i^k \leq 1, \quad \forall i \in P,\ \pi_i^k \in \Pi_i,$$

we will obtain the bifurcated multicommodity flow problem.

The NBFA problem, like many other network design problems, is very complex and numerically intractable even for networks with a small number of nodes. Notice that the NBFA problem is an integer 0–1 problem with linear constraints. However, the size of the problem is very large even for relatively small networks. For instance, for a sample network having 10 nodes and 42 arcs the average number of routes between a node pair is 237. For the link-path formulation the number of binary variables representing the selected route is about 90^{237}.

A popular method to solve 0–1 problems is the branch-and-bound approach. Such algorithms were applied to many problems related to NBFA (Kasprzak, 2001; Walkowiak, 2002; 2004). Nevertheless, branch-and-bound algorithms are intractable for networks of medium and large sizes. The only way to solve the NBFA problem by an exact algorithm that can produce an optimal solution is to reduce the problem size and consider only a part of all possible routes. Such an approach, called the Path Generation technique, is discussed in (Pióro *et al.*, 2003). Another possible method to reduce the size of the problem considered is the hop-limit approach (Herzberg *et al.*, 1995).

Some heuristic algorithms have been developed for solving the non-bifurcated multicommodity flow problem. One of the most substantial ones is the Flow Deviation (FD) algorithm proposed in (Fratta *et al.*, 1973). The FD algorithm and its modifications have proven their effectiveness in many network design problems (Bienstock, 2002; Burns *et al.*, 2003; Kasprzak, 2001; Murakami and Kim, 1996; Walkowiak, 2002, 2003a, 2003b). Also, a genetic algorithm has been proposed for the problem considered (Walkowiak, 2001a). However, it should be mentioned that most papers on flow optimization focus on bifurcated multicommodity flows. Much fewer works consider problems of non-bifurcated flow allocation formulated as 0–1 integer problems. Therefore, generally, there are not many algorithms and methods for such problems.

3. Related Work

In this section we focus on example applications of ant algorithms to various network problems. The related work on multicommodity flows is provided in the previous section.

First, we briefly present the main ideas of the Ant Colony Optimization algorithm, which is a foundation of many other ant algorithms. In ACO meta-heuristic a colony of artificial ants cooperate in finding good solutions to discrete optimization problems. Cooperation is a major element of ACO algorithms. The choice is to allocate the computational resources to a set of relatively simple agents that communicate indirectly by *stigmergy*. Artificial ants have some common features with real ants—they employ pheromone trails, which are used for the shortest path finding in real ant colonies. However, artificial ants have been enriched with some capabilities which are not encountered in nature. Actually, ACO is an engineering approach to the design and implementation of software systems for the solution of optimization problems. Therefore, artificial ants are provided with some capabilities that make them more efficient. The most important ideas of ACO derived from real ants are the use of a colony of cooperating individuals, a (artificial) pheromone trail for local stigmergetic communication, a sequence of local moves to find the shortest paths, and a stochastic decision policy using local information and no lookahead. The major differences between artificial ants used in ACO and real ants are: the discrete nature of artificial ants (they live in a discrete world and have discrete states); artificial ants have internal state and memory to save past actions; artificial ants can deposit a pheromone according to the function of the quality associated with the solution found; artificial ants can update pheromone trails after having generated a solution; an ACO system can use some extra capabilities like local optimization and backtracking (Dorigo *et al.*, 1999).

ACO consists of a finite size colony of cooperating artificial ants altogether seeking high quality solutions to the optimization problem. Each individual ant constructs a solution, or an element of it, starting from an initial state chosen according to the problem considered. Each ant gathers information on its own performance and on special characteristics of the problem in order to change the problem representation, as seen by the other ants. Each ant leaves an amount of pheromone according to the quality of its solution that is expressed as the shortest path

through a graph representing the optimization problem. A single ant can find a feasible solution, but a much better performance is achieved by a set of cooperating ants. The moves of each ant are selected according to a local optimization procedure that involves: ant private information, pheromone trail and problem-specific local information. Artificial ants have memory storing information about ant history. This mechanism can be used to avoid cycles by backtracking the ant from already visited nodes or to save ant routes. It makes finding feasible solutions easier. Two approaches are used for releasing a pheromone by an ant: an on-line step-by-step update and a global online delayed update. In the former method each ant updates the pheromone on each visited link according to a selected formula. The latter approach assumes an update of the pheromone trails after completing the ant's route and generating the whole solution. Ants make use of local available ant-decision tables including information for the decision of each incoming ant to direct their search towards the most promising areas of the solution space. After generating a solution and depositing pheromone information, the ant is deleted from the system. In order to encourage ants to find new routes representing solutions, the pheromone gradually evaporates. Consequently, old solutions, if not reinforced by new ants, successively become less important. Beside ants' operations acting from a local perspective, a special *daemon* can be used, which applies a global representation of the problem in order to improve the performance of the ant algorithm. A major point in the application of the ant system to optimization problems is a paper problem representation. For that reason the first step to implement an ACO algorithm is to develop a representation of the problem considered proper to the ant algorithm. Problems related to computer networks have many features that allow the application of the ACO. The most meaningful of them are: computation distribution, asynchronous evolution of the network status, and graph representation (Dorigo *et al.*, 1999).

The paper by Schoonderwoerd *et al.* (1997) is the earliest work including an application of the ant algorithm to a routing problem. Schoonderwoerd *et al.* developed an ant-based control (ABC) algorithm and employed it to a model of the British Telecom telephone network. Each node of the network has the same functionality as a crossbar switch with limited capacity, while network links have unlimited capacity. The major goal of the ABC system is to find routes for new connections to obtain load balancing of the network and avoid the congestion of the network or the rejection of demands. Ants are created at regular temporal intervals from all the nodes towards randomly selected destination nodes. Ants deposit a pheromone step-by-step on the links they traverse. The pheromone is associated with routing information. After that, ants moving in the opposite direction of the ant considered apply the pheromone information. An assumption is made that the analysed network is cost-symmetric. It simplifies the approach and ants do not have to remember their routes. The amount of the pheromone on the link selected by an ant is reinforced according to a factor that is a function of the ant's age. For the normalization of the pheromone values, which are used as probabilities, the pheromone of other links decays proportionally. Routing tables for new calls are generated according to ant-decision tables. This means that new calls build routes starting from the source node and choosing sequentially neighbor nodes with the highest probability value until the destination node is reached. The node capacity is updated accordingly to the call setup or termination.

Di Caro and Dorigo (1998a; 1998b) developed a system called the AntNet for distributed routing in connectionless networks. In the AntNet real-time trips experienced by ants and a local-based statistical model are applied to estimate the path quality. This means that ants move through the same real network and can have delays like real data packets. The global update of the pheromone is used. Once a path has been found, ants lay down the on the visited nodes an amount of the pheromone proportional to the quality of the found path. Ants move back to source nodes using high priority queues to enable fast propagation of pheromone information. In addition to the pheromone information, local heuristic information representing the current link queues is used in the ant-decision table. Di Caro and Dorigo also developed an enhanced version of the AntNet for connection-oriented networks (Dorigo *et al.*, 1999).

White *et al.* (1998) developed the Routing By Ants (RBA) system for routing in connection-oriented networks. The RBA algorithm applies a very similar approach to the classical Ant System (AS) proposed in (Dorigo *et al.*, 1991). An ant-based algorithm for dynamic packet-switched networks is also presented in (Subramanian *et al.*, 1997).

The above examples of the ant algorithm focus on dynamic routing problems. The work (Varela and Sinclair, 1999) is an attempt to apply the ant algorithm to a static routing problem. It presents the problem of routing and wavelength-allocation (RWA) in a multi-wavelength all optical virtual-wavelength-path routed transport network. The objective function is the network wavelength requirement that denotes a total number of distinct wavelengths used in the network, which is equal to the maximum number necessary on any link. Ants move from every source to every destination, one link per algorithm step. When all ants reach their destination and die, a new cycle of the algorithm starts. Varela and Sinclair developed three major versions of the ant algorithm according to three update strategies: local update, global/distance and global/occupancy. In the first version, each ant is attracted

by a pheromone of its own type left by previous ants and repelled by other ants' pheromone. The pheromone is updated on each algorithm step. The other two versions apply the global update strategy. In the global/update the pheromone is updated inversely proportionally to the length of the ant's path. The global/occupancy strategy assumes that the weight of repulsion depends on the link utilization, i.e., the number of ants of any type using the link considered. All variants use pheromone evaporation. Simulations show that the best results are produced by the global/occupancy algorithm. Varela *et al.* applied a backtracking operation, with each ant keeping a "tabu" list of previously visited nodes. Backtracking prevents dead-ends and cycles. When an ant is blocked, it analyzes its "tabu" list and tries to proceed from the previous location. This capability requires each ant's memory to include a list of nodes visited in order.

Garlick and Barr (2003) also consider the RWA problem with dynamic traffic, in which the number of wavelengths per fiber is fixed. The objective is to minimize connection blocking using an ant-colony optimization (ACO) algorithm. The main goal of the algorithm is to quantify the importance of combining path-length and congestion information in making routing decisions. The ACO algorithm reaches lower blocking rates than an exhaustive search over all available wavelengths for the shortest path.

An overview of many ant algorithms applied to various static and dynamic optimization problems can be found in (Dorigo *et al.*, 1999). Generally, most of the literature on the application of ant algorithms to network problems we have found concerns either dynamic routing or the RWA problem. To the best of our knowledge, the first work presenting an ant algorithm applied to the problem of static flow assignment in connection-oriented networks with capacity constraint is the work (Walkowiak, 2001b). In this paper we present a much more comprehensive study of the algorithm proposed in (Walkowiak, 2001b) together with the results of extensive numerical experiments.

4. Ant Algorithm for the Non-Bifurcated Multicommodity Flow Assignment Problem

In this section we introduce our approach to the NBFA problem using an ant algorithm called hereafter the ANB (**A**nts for **N**on-**B**ifurcated Flows) algorithm. The overall framework of the ant algorithm for connection-oriented flow problems was proposed in (Walkowiak, 2001b). The ANB algorithm presented below is a continuation of that work. Moreover, we also depict a second ant algorithm—ANIBS (ANB with Initial Solution)—which is a slight modification of ANB. The only difference between ANB and ANBIS is the initialization of pheromone values. All other procedures are the same. Therefore, if we refer to ANB, also the ANBIS algorithm is kept in mind.

We use the link-path representation of the NBFA problem presented in Section 2. For the sake of simplicity, we introduce the following function:

$$\varpi(x) = \begin{cases} 0 & \text{for} \quad x \leq 0, \\ x & \text{for} \quad x > 0. \end{cases}$$

When we tackle an optimization problem with an ant algorithm, either the formulation of the algorithm should guarantee that all constraints are satisfied or constraints of the problem must be introduced in the objective function using a penalty method. To introduce the capacity constraint (12), we modify the objective function of the network flow and use the penalty method. The modified objective function of the NBFA problem is as follows:

$$K'(\underline{f}) = \sum_{j \in A} \Big(f_j + Pn\big(\varpi(f_j - c_j)\big)^2 \Big). \tag{13}$$

The Pn parameter is a penalty factor. It should be noted that to find a feasible solution of the NBFA problem, it is very important to select the value of the penalty factor in a correct way. The algorithm should be forced to find feasible solutions. In the objective function K' we must 'promote' feasible solutions and 'punish' solutions violating the capacity constraint. Other constraints of NBFA connected with the non-bifurcated nature of the multicommodity flow are satisfied due to the formulation of the ANB algorithm.

We begin the presentation of the algorithm by introducing the notation. We will keep the same notation in the remainder of the paper.

Indices:

- i used as a subscript, denotes the number of the ant considered,
- j used as a subscript, denotes the number of the node considered,
- t used as a superscript, denotes the number of the current iteration (cycle) of the algorithm.

Algorithm parameters:

- p number of ants which equals the number of commodities,
- Q_i bandwidth requirement for the commodity associated with the ant i,
- Pn penalty factor used to scale the penalty function in the objective function,
- R constant used in the pheromone updating rule,

ρ evaporation coefficient; at the end of each cycle t the level of the pheromone on all links is reduced by ρ (in the remainder of the paper this parameter is also labeled as RHO),

α parameter that allows the user to control the relative importance of the pheromone given by the variable ε_{ij}^t (in the remainder of the paper this parameter is also labeled as ALPHA),

β parameter that allows the user to control the relative importance of the local heuristic information represented by η_{ij}^t (in the remainder of the paper this parameter is also labeled as BETA),

NOI number of iterations of the algorithm.

Variables

π_i^t route used by the ant i in the iteration t,

τ_{ij}^t amount of the i-th ant pheromone laid on the arc to the j-th node in the iteration t,

L_i^t length of the route of the i-th ant selected in the cycle t, i.e., a sum of metrics $(f_k + Pn(\varpi(f_k - c_k))^2)$ of the arcs k belonging to the route π_i^k followed by the ant in the cycle t,

ε_{ij}^t weight of attraction of the neighbor node j for the i-th ant in terms of the pheromone laid on the arc j,

A_i^t set of nodes allowed to the ant i in the time cycle t (this set is associated with the tabu list),

η_{ij}^t visibility—local heuristic information of selecting the node j by the ant i in the iteration t (visibility is based on strictly local information and it measures the attractiveness of the next node to be selected),

γ_{ij}^t ant decision probability of selecting the node j by the i-th ant in the cycle t (the ant can select only among all allowed nodes; since the variable γ_{ij}^t is a probability, it is normalized).

The number of ants is equal to the number of commodities in the network, i.e., each commodity has its own ant. Each ant deposits its own pheromone. An ant is characterized with the attributes of the commodity: the source and destination nodes, and the bandwidth requirement. Each ant possesses some memory to store the route being traversed. Every cycle of the algorithm starts with placing each ant in its source node and initiating the ant's memory. The framework of ANB and ANBIS is shown in Fig. 1. The most substantial features of this algorithm are explained in the following. It must be noted that ANB and ANBIS are very similar. The only difference is included in the `InitializePheromoneValues` (τ) procedure.

`InitializePheromoneValues` (τ): For the ANB algorithm all pheromone values are set to the same positive constant value 1 in the first iteration of the algorithm, i.e., $\tau_{ij}^1 = 1$. For ANBIS we apply a feasible (in terms of the capacity constraint) solution in the following way:

```
 1 InitializePheromoneValues(τ)
 2 for t=1 to NOI do
 3   for i=1 to p do InitializeAnt(i)
 4   while(ExistAnts()==TRUE) do
 5     for i=1 to p do
 6       if (ExistAnt(i)==TRUE) then
 7        MoveAnt(i)
 8       end if
 9     end for
10   end while
11   PheromoneUpdate(τ)
12   DaemonActions()
13 end for

//i – index of analyzed ant
14 procedure MoveAnt(i)
15   j=SourceNodeOfAnt(i)
16   do
17     A_i=CreateAllowedNodesSet(i,j)
18     if (A_i^t==∅) then j=0
19     else
20      for k∈A_i^t do
21       α_ik^t=ComputeAttraction(i,j,k)
22       η_ik^t=ComputeLocalVisibility(i,j,k)
23      end for
24      for k∈A_i^t do
     γ_ik^t=ComputeDecisionProbability(i,j,k)
25       j=SelectNextNode(γ)
26     end else
27     if (j==0) then
28       FindShortestRoute(a)
29       j=DestinationNodeOfAnt(i)
30     end if
31   while(j!=DestinationNodeOfAnt(i))
32 end procedure

//i – index of analyzed ant; j – number of
current node; k – number of //allowed node
for the ant i which is in node j
33 procedure ComputeLocalVisibility(i,j,k)
34   d=DestinationNodeOfAnt(i)
35   if (k==d) then
36    if (ResidualCapacity(j,k)>Q_i) then
      η_ik^t=1
37    else η_ik^t=2·n
38   else
39    if (ResidualCapacity(j,k)>Q_i) then
40    η_ik^t=LengthOfFeasibleShortestPath(i,k)
41    else η_ik^t=2·n
42   end else
43   return η_ik^t
44 end procedure
```

Fig. 1. ANB description in pseudo-code.

All pheromone values are also initially set to 1. However, after that, each ant follows the path given by the feasible solution found by another algorithm. Next, we update the pheromone according to formulas given below. This means that the pheromone values are initiated according to a feasible solution.

The main loop of the program (the lines 2–13) is repeated for a given number of iterations. However, some other termination criteria, e.g., when the evaluation of the best found solution gives a positive result can be applied. The variable t denoting the current cycle is a global variable available in all procedures.

`InitializeAnt (i)`: Each ant is placed in its source node. The memory of the ant is cleared. The loop from the line 4 to the line 10 is repeated until all ants reach their destination nodes.

`MoveAnt (i)`: An important part of the ANB algorithm. This procedure is responsible for moving the ant through the network. The detailed actions of an ant are given in the lines 14–32. The ant continues its trip across the network until it reaches the destination node (the lines 16–31). According to the current state and the position in the network, the ant applies its decision policy and selects a next node to move to (the line 25). The ant is attracted to a node of those adjacent to its current node excluding nodes contained in the tabu list. The set A_i^t, created in the line 17, includes all nodes allowed to the ant i. The weight of attraction of the node j for the i-th ant is given by

$$\varepsilon_{ij}^t = \frac{\tau_{ij}^t}{\sum\limits_{j \in A_i^t} \tau_{ij}^t}. \tag{14}$$

Moreover, in the ant decision table we apply some local heuristic information η_{ij}^t of selecting the node j by the ant i. The calculation of η_{ij}^t is done by the procedure `ComputeLocalVisibility` (the line 22) discussed below. The values of η_{ij}^t are normalized in much the same way as ε_{ij}^t.

The ant decision probability of selecting the node j by the i-th ant is calculated as follows:

$$\gamma_{ij}^t = \frac{(\varepsilon_{ij}^t)^\alpha (\eta_{ij}^t)^\beta}{\sum\limits_{j \in A_i^t} (\tau_{ij}^t)^\alpha (\eta_{ij}^t)^\beta}. \tag{15}$$

The parameters α and β are used to find a trade-off between local heuristic information and pheromone intensity.

Ants follow a route from a particular source node to a particular destination. The route must not contain loops. Therefore, each ant records its route in memory to maintain a tabu list of nodes the ant must not revisit. However, a loop may occur when the ant reaches a 'dead end' (Varela and Sinclair, 1999). In order to overcome this problem, we allow the ant to backtrack to the source node and find the shortest route using the shortest path algorithm skipping the pheromone information (the line 28). We apply the Dijkstra algorithm with the hop metric, i.e., each arc has a weight equal to 1. As a matter of fact, in our algorithm, we let the ant try 10 times to overcome the loop. If after 10 attempts the ant cannot reach the destination node, we apply the shortest path algorithm to compute the ant's route.

`ComputeLocalVisibility (i,j,k)`: This procedure is responsible for providing some local, heuristic information to the ant. While we were developing and testing the algorithm, we noticed that the selection of local information is very important for correct functioning of the algorithm. Therefore, we introduced some improvements to the initial version of the algorithm (Walkowiak, 2001b). The modifications yielded much better results. The detailed actions of the procedure are shown in the lines 33–43. We use two kinds of local information: the residual capacity of arcs and the distance from the current node to the destination node of he ant.

`ResidualCapacity (j,k)`: It returns the residual capacity of the arc (j,k), i.e., the difference between the arc capacity and the flow.

`LengthOfFeasibleShortestPath (i,k)`: It returns the length (given in the number of arcs) of a feasible path from the node k to the destination node of the ant i. The feasibility of the route means that in order to calculate the shortest path, we consider only those arcs in which the residual capacity is bigger than the bandwidth requirement of the ant i. If such a path does not exist, the procedure returns the value $2n$—it is a kind of penalty function.

`PheromoneUpdate` (τ): Ants update pheromone. When all ants die, i.e., when ants reach their destination nodes, we update the pheromone value. We use an approach related to the ant-cycle algorithm proposed in (Dorigo *et al.*, 1991) and the global update proposed in (Varela and Sinclair, 1999). Pheromone information is updated after all ants have completed their routes. If we assume that in time cycle t the ant i uses the route π_i^t, the variable L_i^t denoting the length of the ant's route is calculated according to the formula

$$L_i^t = \sum_{k \in A} a_{ik}^t \left(f_k + Pn\big(\varpi(f_k - c_k)\big)^2 \right). \tag{16}$$

We use the following pheromone updating rule:

$$\tau_{ij}^{t+1} = \tau_{ij}^t + \frac{R}{L_i^t}, \tag{17}$$

where R is one of the parameters in the algorithm.

Pheromone evaporation reduces the level of the pheromone on all links by a factor ρ (the evaporation coefficient) at the end of each cycle t (Dorigo *et al.*, 1991):

$$\tau_{ij}^{t+1} = \rho\, \tau_{ij}^t. \tag{18}$$

`DaemonActions` (): When an ant reaches the destination node, it dies, i.e., it is removed from the system, but the memory of the ant's route it transferred to the global daemon. Using this information, the value of the obtained solution can be calculated. A global daemon can compute some other information, e.g., network statistics.

The ANB algorithm has a large set of parameters that has to be tuned in order to provide the best possible feasible solutions of the NBFA problem. A detailed study of the influence of parameter settings on the quality of solutions is made in the next section.

An analysis of ANB algorithms shows that many ideas are related to the algorithm presented in (Varela and Sinclair, 1999). The most important similarities are as follows: one ant using its own type of pheromone for each commodity, backtracking, and a global update of the pheromone. However, due to the specific capacity constraint, which is not considered in (Varela and Sinclair, 1999), we have to modify Varela and Sinclair's algorithm and use local heuristic information in the ant decision probability formula. Consequently, the role of pheromone information is less important in our algorithm.

It should be mentioned that many various versions of ant algorithm were evaluated by the author, and the ANB algorithm presented above is the best one in terms of the obtained results. Many ant algorithms developed for network optimization problems could not be applied directly to the NBFA problem due to the specific features of that problem: static optimization, the non-bifurcated flow (only one route for commodity) and capacity constraints.

5. Numerical Results

In this section, we present an analysis of the ANB and ANBIS performances. Both the algorithms were implemented in C++. Since we had not performed a mathematical analysis of the algorithm, which would have helped to obtain an optimal parameter setting in each situation, we ran many simulations to assemble statistical data for this purpose. The results presented in this section are obtained from simulations on 10 sample networks (Fig. 2). The name of each network indicates the number of nodes (two first digits) and the number of links (two last digits). Table 1 summarizes the parameters of all sample networks. The heading of each column specifies the name of the parameter, and various tested values are listed in the respective column. In an experiment, it is assumed that there is a requirement to establish a connection for each direction of every node pair. Consequently, the total number of ants in the network is equal to $n(n-1)$. For a given experiment, the value of the bandwidth requirement is the same for each commodity. We study the performance of the algorithm for an increasing traffic load, examining the evolution of the network status toward a saturation condition. This means that experiments for a particular topology have different traffic demands. The last column of Table 1 shows the number of bandwidth requirements patterns for each network. For instance, on the network 1034 we perform 4 experiments: 10340, 10345, 1034a, 1034e with the bandwidth requirement of each commodity 40, 35, 30 and 26, respectively.

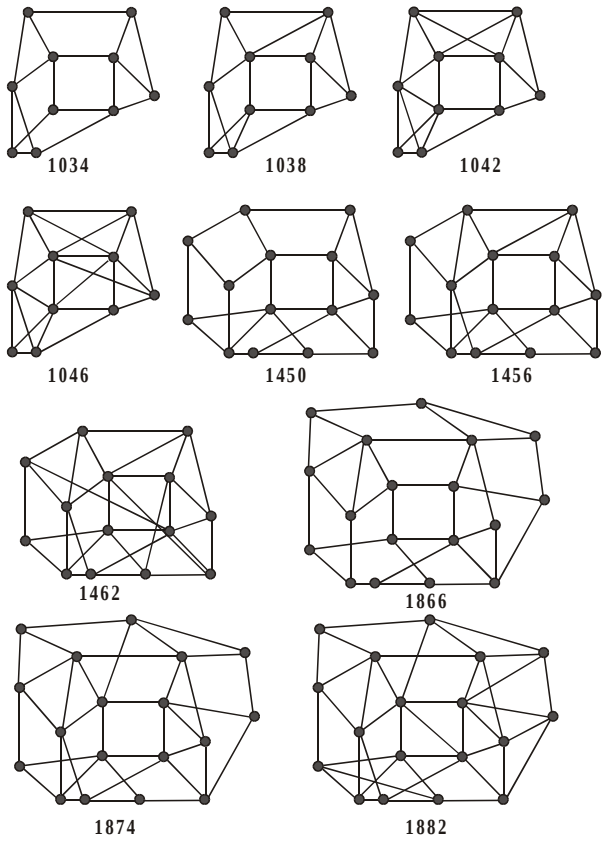

Fig. 2. Sample networks.

Table 1. Parameters of sample networks.

Name of network	Number of nodes	Number of links	Average node degree (avnd)	Number of ants	Number of bandwidth requirements
1034	10	34	3.40	90	4
1038	10	38	3.80	90	3
1042	10	42	4.20	90	5
1046	10	46	4.60	90	5
1450	14	50	3.57	182	7
1456	14	56	4.00	182	7
1462	14	62	4.43	182	7
1866	18	66	3.66	306	4
1874	18	74	4.11	306	4
1882	18	82	4.56	306	4

5.1. ANB Parameters Setting

In this subsection we consider parameters that affect directly or indirectly the ANB algorithm. As was mentioned above, the number of ants has always been set as the number of commodities in the network to be established. We apply the same approach like in (Dorigo *et al.*, 1991), i.e., for each simulation, we vary only the value of one parameter while keeping all other parameters constant. Even though NBFA and TSP are quite different problems, we decided to use as a starting point values similar to those in (Dorigo *et al.*, 1991), used in the ant algorithm for the TSP problem: $Pn \in \{0, 1, 2, 5, 10, 20\}$, $R \in \{1, 100, 10000\}$, $\rho \in \{0.3, 0.5, 0.7, 0.9, 0.999\}$, $\alpha \in \{0, 0.5, 1, 5\}$, $\beta \in \{0, 1, 2, 5, 10, 20\}$. We use the following notation: Each *experiment* consists of 2160 *simulations* of the ANB algorithm for different values of parameters. An experiment differs from the others in terms of the topology of the tested network and the traffic demand pattern. A simulation differs from other simulations in terms of the values of the parameters Pn, R, ρ, α and β. Due to tested values of parameters values, the overall number of simulations for one experiment is $6 \times 3 \times 5 \times 4 \times 6 = 2160$. In all simulations, the number of iterations is set to 50. A result obtained for a particular simulation is the best result among all 50 results obtained for each iteration of the algorithm.

Analyzing the obtained results, we have noticed that parameters Pn, R and ρ generally do not have strong influence on the value of the objective flow function given by (13). Figures 3–5 show the results of running the same set of simulations on the topology 1874 with values of the parameters α and β fixed as 0.5 and 20, respectively. The other three parameters were changed according to the default values given above. It yields 90 simulations for each set of the bandwidth requirement on the topology 1874. The y-axis represents the aggregate flow summed over all tests. The x-axis represents the bandwidth requirements. For the experiments 18744, 18746, 18748 and 1874a the value of the bandwidth requirement of each commodity is 20, 19, 18 and 17, respectively.

The general trend shown in Figs. 3–5 is that changing the value of any three parameters considered does not influence the flow in the network. Only for the experiment 18744 the difference is slightly more significant than for other cases. For example, in the experiment 18744, the biggest gap between the values of the flow function is 4.96%, while in experiments 18746, 18748 and 1874a, it decreases to 0.34%. Analyzing the results of all 10800 simulations for various networks and different values of parameters, we conclude that the default values of the parameters Pn, R and ρ should be fixed as 2, 100 and 0.9, respectively. However, the importance of these parameters was uninfluential in most of the experiments.

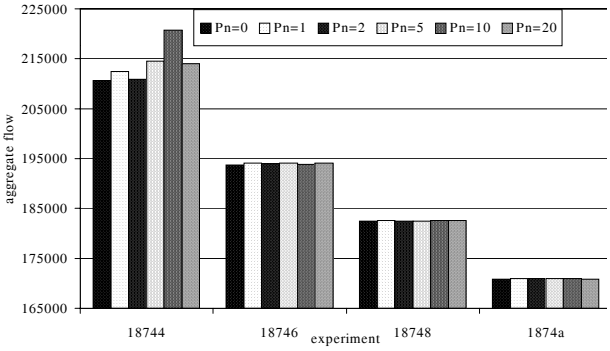

Fig. 3. Aggregate results for the network 1874 showing the influence of the Pn parameter.

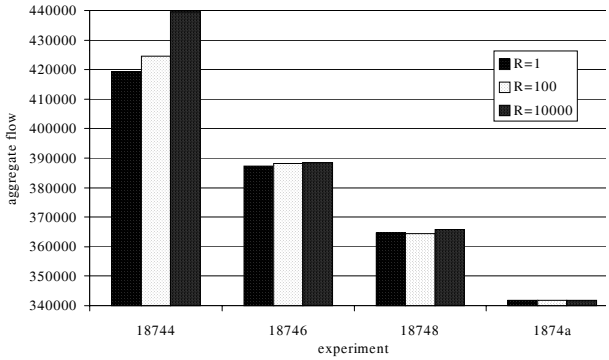

Fig. 4. Aggregate results for the network 1874 showing the influence of the R parameter.

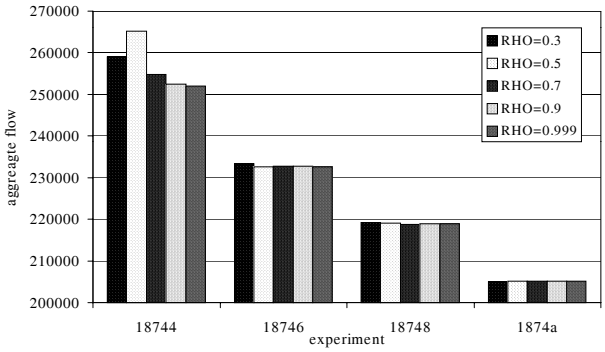

Fig. 5. Aggregate results for the network 1874 showing the influence of the ρ parameter.

Recall that the flow function is calculated according to (13), which takes into account the penalty function. If a solution is not feasible (the capacity constraint does not hold), the value of the flow function may be very large. If we calculate the aggregate flow function as a sum over many experiments, the presentation of results could be difficult. Moreover, for various experiments different values of bandwidth requirements are used, which results in huge deviations of the flow function value. Therefore, to make the performance evaluation easier, we introduce the concept of the *competitive ration*. The competitive

ration, which indicates how well an algorithm performs for a given set of the parameters Pn, R, ρ, α and β, is defined as the difference between the result obtained for a particular simulation and the minimum value of the flow function obtained in the experiment considered. For instance, if for the experiment consisting of 2160 simulations the minimum value of the flow function is 10000 and the flow function of the simulation considered is 15000, the competitive ration of this simulation is calculated as $(15000 - 10000)/10000 = 0.5$. The competitive ration indicates the quality of the obtained result of a given simulation compared with the results of other simulations for a particular experiment. A low value of the competitive ration means that the simulation result is very close to the best results in a given experiment. For the presentation of aggregate results, we apply the *aggregate competitive ration*, which is the sum of competitive rations over all experiments considered.

We now describe the influence of two parameters α and β on the value of the objective function using the competitive ration as the performance index. To examine the impact of these parameters, we fix three other parameters at default values and change the values using $\alpha \in \{0, 0.5, 1, 5\}$ and $\beta \in \{0, 1, 2, 5, 10, 20\}$. For clarity, we show aggregate results for all networks and for the network 1874 discussed above concerning the tuning of the parameters Pn, R and ρ. Figure 6 shows the aggregate competitive ration for all experiments. Figure 7 depicts the same function only for the topology 1874. In both figures, the z-axis uses the logarithmical scale. The general trend in both figures is that the best results are obtained for $\alpha \in \{0, 0.5, 1\}$ and $\beta \in \{10, 20\}$. Due to the large difference in the tested topologies in terms of the number of nodes and links, we have noticed that for networks having 10 nodes the best results are obtained for $\beta = 10$, while for larger networks the parameter should be fixed at 20.

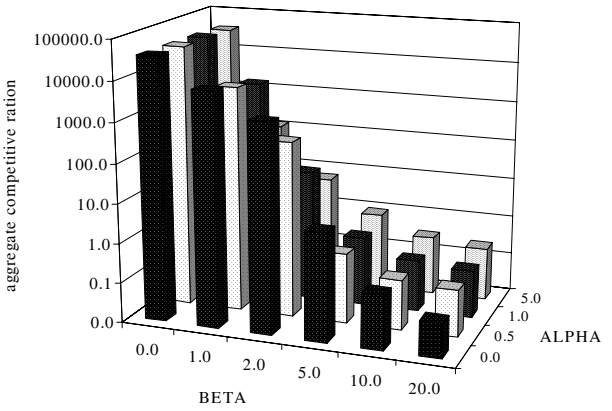

Fig. 6. Aggregate results for all tested networks showing the influence of the α and β parameters.

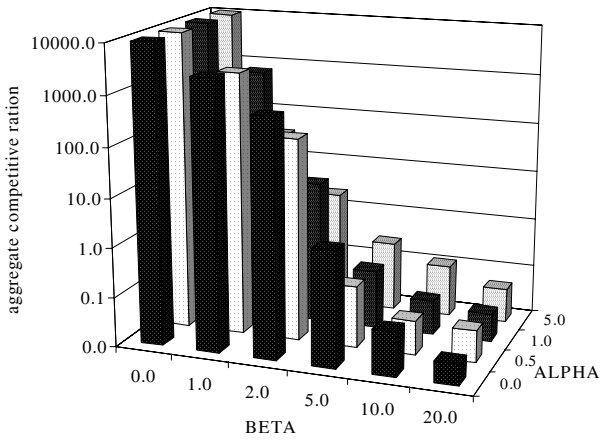

Fig. 7. Aggregate results for the network 1874 showing the influence of the α and β parameters.

An important observation compared with previous works in the field of ant algorithms is that good results are obtained for $\alpha = 0$. According to the description of the algorithm presented in the previous section, setting α as 0 means that the pheromone information is not applied and only local heuristic data are used to determine the routes. This can be explained by the fact that local heuristic information is calculated according to a complicated formula using a lot of data on the current state of the network. The second potential explanation of the fact is that relatively many commodities are established between neighbor nodes of networks, especially for smaller networks. For instance, in the 1042 topology 42 of 90 demands, i.e., 47% of all demands connect adjacent nodes. However, for the network 1874, the ratio of such connections is 24%. Since the distance between the source and destination nodes is only one hop, the importance of pheromone information is reduced. Consequently, local heuristic information gains much more significance. To verify this hypothesis, we repeat the same simulations as presented above. However, we limit the number of commodities in the network according to the minimum distance between the source and destination nodes. This means that we ignore those demands for which end nodes are too close in terms of the number of hops. Table 2 summarizes the parameters of these new simulations. As in Table 1, the heading of each column specifies the name of the parameter, and the various values tried are listed in the respective column. In this experiment, it is assumed that there is a requirement to establish a connection for each direction of every node pair for which the distance is at least 3 hops or 4 hops. For a particular test, the value of the bandwidth requirement is the same for all commodities. The last column indicates the number of various traffic demand patterns for each network. As above, the parameters Pn, R and ρ are fixed to default values 2, 100 and 0.9, respectively.

Table 2. Parameters of experiments.

Name of network	Number of nodes	Number of links	Minimum distance in hops	Number of ants	Number of bandwidth requirements
1866	18	66	3	126	6
1874	18	74	3	106	5
1882	18	82	3	94	6
1866	18	66	4	34	6
1874	18	74	4	20	6
1882	18	82	4	10	6

Figures 8 and 9 depict the impact of the α and β parameters on the networks 1866, 1874 and 1882 for the minimum distance between the source and destination nodes of commodities fixed to 3 and 4 hops. The general trend in these figures is similar to that in previous experiments. However, comparing Figs. 8 and 9 against Figs. 6 and 7, we see that increasing the minimum distance between the end nodes of the commodities makes the results for $\alpha = 0$ relatively worse. For instance, in the experiment with the minimum distance fixed at 4 hops, the best results are obtained for $\alpha = 0.5$, $\beta = 20$ and $\alpha = 1$, $\beta = 10$ (Tables 3 and 4). This proves that the good results obtained for $\alpha = 0$ and presented above can be explained partially by the tested traffic demand pattern.

The results obtained for the ANB algorithm are not in agreement with the results of the algorithm presented in (Dorigo *et al.*, 1991). However, since in this work we do not consider the traveling salesman problem but the non-bifurcated multicommodity flow problem, a direct comparison is very difficult. A major difference is the capacity constraint of the flow problem. In our opinion, the capacity constraint strongly influences the algorithm, which results in great importance of local heuristic information and a lower influence of pheromone information. It must be noted that using $\alpha = 0$ leads to an instability of the algorithm, i.e., the algorithm cannot converge to one solution.

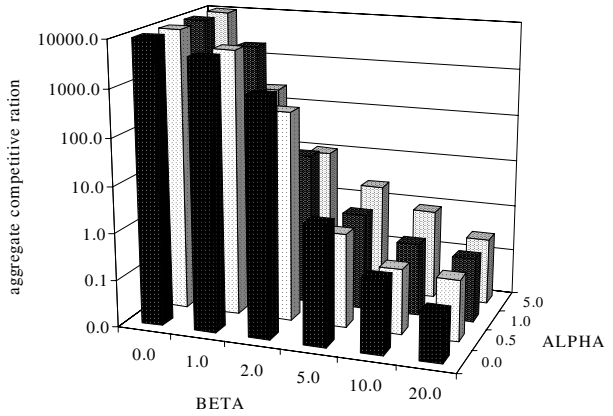

Fig. 8. Aggregate results for the topologies 1866, 1874 and 1882 with the minimum hoop distance set as 3 showing the influence of the α and β parameters.

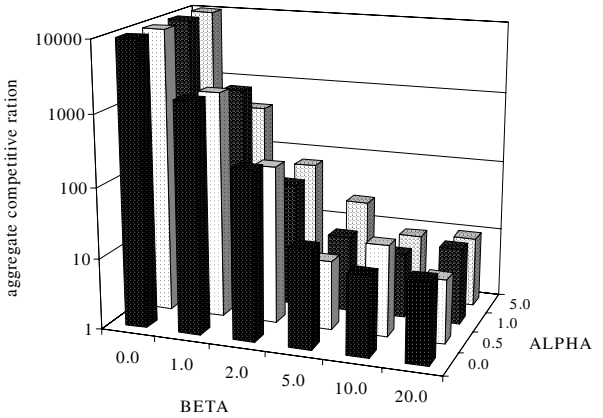

Fig. 9. Aggregate results for the topologies 1866, 1874 and 1882 with the minimum hoop distance set as 4 showing the influence of the α and β parameters.

Table 3. Aggregate competitive ration of various simulation scenarios for $\beta = 10$.

Networks	All			1866, 1874, 1882			1866, 1874, 1882			1866, 1874, 1882		
Minimum distance	1			1			3			4		
α	0	0.5	1	0	0.5	1	0	0.5	1	0	0.5	1
Aggregate competitive ration	0.7	0.8	1.8	0.23	0.18	0.20	0.36	0.25	0.35	13.2	19.4	8.1

Table 4. Aggregate competitive ration of various simulation scenarios for $\beta = 20$.

Networks	All			1866, 1874, 1882			1866, 1874, 1882			1866, 1874, 1882		
Minimum distance	1			1			3			4		
α	0	0.5	1	0	0.5	1	0	0.5	1	0	0.5	1
Aggregate competitive ration	0.5	0.9	1.8	0.07	0.15	0.16	0.11	0.20	0.23	13.9	8.0	12.0

Summarizing the process of parameter setting, we can list the main results:

1. The influence of the parameters Pn, R and ρ on the performance of the ANB algorithm is relatively small. Only for strongly saturated networks the difference between the results obtained for various values of these parameters is larger than 1%.
2. The default values of the parameters Pn, R and ρ should be fixed as 2, 100 and 0.9, respectively.
3. The influence of the parameters α and β on the performance of the ANB is strong for all tested networks.
4. The default values of the parameters α and β should be $\alpha \in \{0, 0.5, 1\}$ and $\beta \in \{10, 20\}$.
5. Good results obtained for $\alpha = 0$ suggest the significance of local heuristic information.
6. The traffic demand pattern may change the parameters setting. If we consider only commodities between remote (in the number of hops) nodes, the pheromone trail becomes more significant.

5.2. ANBIS Parameter Setting

In this subsection we find the best parameter setting for ANBIS. As in the previous subsection, we consider the following values of the parameters: $Pn \in \{0, 1, 2, 5, 10, 20\}$, $R \in \{1, 100, 10000\}$, $\rho \in \{0.3, 0.5, 0.7, 0.9, 0.999\}$, $\alpha \in \{0, 0.5, 1, 5\}$, $\beta \in \{0, 1, 2, 5, 10, 20\}$. The methodology of tests is the same as for ANB. However, the parameter setting for ANIBS differs from that for ANB. The best results are obtained for the following values: $Pn = 0$, $R = 10000$, $\alpha \in \{1, 5\}$ and $\beta = 20$. ANBIS is not strongly influenced by ρ. For instance, for the topology 1874 modifying ρ changes the result by no more than 1%. Recall that in ANBIS we start with a feasible solution, for which the capacity constraint is satisfied. Therefore, the penalty factor Pn, which is significant in searching for a feasible solution, is set to 0. The importance of R grows because the algorithm should stay in feasible regions of the solution space. Recall that the parameter R is used in the pheromone updating rule (17). Therefore, R affects the reinforcement of the pheromone according to the route selected by ants in the current iteration. If R is relatively small, the route selected by an ant has also little influence on the pheromone lying and it is more probable that the ant will select in the next iteration another route, much different from the previous one. Consequently, in next iterations the solution may become infeasible. On the other hand, when R is relatively large, it is more probable that ants will use routes similar to previous ones and the obtained solution will stay in a feasible region.

Figure 10 shows the aggregate competitive ration for all experiments. Figure 11 depicts the same function only for topology 1874. In both the figures the default values of parameters Pn, R and ρ are used. The general trend in both the figures is that the best results are obtained for $\alpha \in \{1, 5\}$ and $\beta = 20$. It is consistent with the philosophy of ant algorithms—the pheromone information plays an important role in the algorithm.

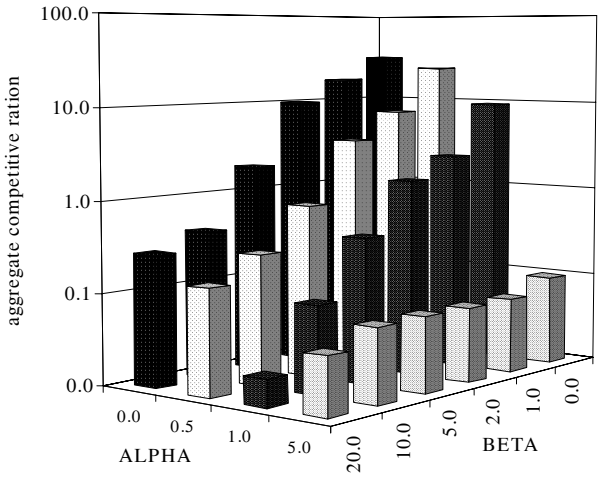

Fig. 10. Aggregate results of ANBIS for the networks 1866, 1874, 1882 showing the influence of the α and β parameters.

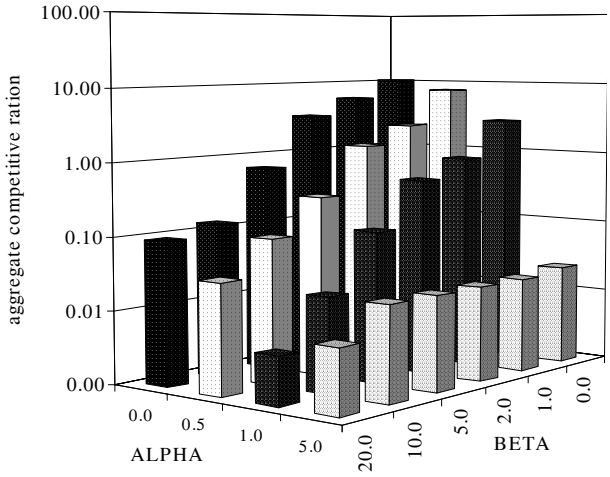

Fig. 11. Aggregate results of ANBIS for the network 1874 showing the influence of the α and β parameters.

The comparison of ANB and ANBIS parameter tuning indicates that the major issue in the problem considered is the feasibility of results. The ANB algorithm must be forced to search for the feasible solution. Therefore, local heuristic information enforced by the β parameter is more significant, while α is not so important. Since ANBIS starts with a feasible solution, the major effort of

the algorithm is to improve the result. Thus, the parameter α becomes much more influential.

5.3. Comparison with Other Heuristics

In this subsection we compare the performance of the ANB and ANBIS algorithms with that of other algorithms applied to the assignment of non-bifurcated flows. To assess the value of our approach applied to the non-bifurcated multicommodity flow problem, we selected for comparison two algorithms: a modification of the Flow Deviation (FD) algorithm presented in (Fratta *et al.*, 1973), and a heuristic algorithm called AlgNB and developed by Walkowiak (2003b). Both algorithms require an initial feasible solution that is provided by the initial phase of the FD algorithm (FDInit) from (Fratta *et al.*, 1973). It must be noted that the Flow Deviation algorithm is one of the most robust methods developed for the assignment of multicommodity flows. The FD method and its modifications are widely used for optimization problems related to NBFA (Bienstock, 2002; Burns *et al.*, 2003; Kasprzak, 2001; Murakami and Kim, 1996; Walkowiak, 2002; 2003a; 2003b). More details on both algorithms and results can be found in (Walkowiak, 2003b).

ANBIS, FD and AlgNB use the output of FDInit as the starting feasible solution. In Table 5 we report the results for all tested algorithms. We show the aggregate flow obtained for all 18-nodes networks (the second row of Table 2) and detailed results for the networks 1866, 1874 and 1882 (rows 3–5). We find the ANBIS algorithm to be superior to other algorithms. ANBIS improves the result of FDInit from 0.09% to 1.03% for particular networks. The second algorithm is AlgNB, which is only 0.10% worse than ANBIS. The worst performance is revealed by ANB. However, the maximum difference between the best and worst results reported in Table 5 is less than 2.5%.

Table 5. Aggregate flow obtained for various algorithms.

Network	ANB	ANBIS	FDInit	FD	AlgNB
18	135711	132974	133796	133106	133080
1866	32403	31770	31800	31770	31770
1874	50309	49216	49470	49176	49193
1882	52999	51988	52526	52160	52117

The results presented in Table 5 confirm that the main problem of ANB is the feasibility of the result. The major effort of ANB is to find a feasible solution—the parameters are set and fixed according to this issue. The tuning of ANBIS shows that if the algorithm finds a feasible solution, different values of parameters are needed in order to improve the solution. Therefore, the results obtained for ANB are not satisfactory.

Another important issue is the execution time of the compared heuristics. The comparison shows that the computational time of ANB and ANBIS is 15–45 times longer than that of the other heuristics. However, our focus in developing the implementation was on correctness, rather than on program speed, and it is anticipated that this figure could be significantly reduced. For instance, reducing the number of iterations could decrease the decision time of the ant algorithm.

6. Concluding Remarks

In this paper we have described how to apply an ant algorithm to a non-bifurcated multicommodity flow assignment problem. We have proposed and discussed comprehensively a novel ant algorithm. Since many static optimization problems encountered in real connection-oriented networks can be modeled as non-bifurcated multicommodity flow problems, the significance of this work is substantial. Because of the different optimization problem considered, the algorithm and implementation details tightly bound to the specific capacity constraint. Therefore, it was impossible for us to re-implement the existing ant algorithms approaches proposed in previous works. Is should be underlined that our focus in developing and testing ANB and ANBIS was on the correctness and evaluation of the algorithm, rather than issues of its effectiveness. By the correctness we mean that the algorithm can find a feasible solution, i.e., a solution in which the flow of each link does not exceed the link capacity.

The results of many simulations run on various network topologies suggest that ANB can be used for capacitated optimization of static flows. Nevertheless, there is still some room for further improvements. We have studied several issues that arise in tackling the NBFA problem by the ant algorithm. The major problem is the tuning of the algorithm. We observe that some parameters have an effect on the performance of ANB and ANBIS much more substantially than others. We proposed the best values of parameters and noticed that wrong selection may lead to completely incorrect results. A relatively strange, comparing with other ant algorithms, parameter setting obtained for ANB can be explained mainly by the specific capacity constraint that influences robustly the algorithm, especially for congested networks in which only a very small part of the solution space is feasible.

We also compared ANB and ANBIS with other heuristics. During numerical experiments we noticed several shortcomings of our approach which are listed as follows:

- for highly loaded networks the ANB algorithm cannot find a feasible solution;

- premature convergence—both algorithms stop too early further exploration of the search space;
- ANB performs worse than the known algorithms, and the difference in results is relatively small;
- running time of ANB and ANBIS is much longer compared with other algorithms.

Many questions remain and many problems are open: as a final remark we propose a brief catalogue of them. In further work, it will be possible to carry out a more detailed investigation of appropriate parameter settings for our algorithm according to various network topologies and traffic demands patterns. Another issue is the role of experience: how to operate in parameter tuning, for balancing convergence speed and good results. Adopting a feasible initial solution to pheromone initialization rather than setting the pheromone to the same value may prove beneficial, as ANB has sometimes trouble finding a feasible solution that conforms to the capacity constraint.

In future work we plan to make improvements to ANB. The first proposal is to change parameter values when the algorithm runs in the following way: We start with values providing a feasible solution. Next, we change the parameters to values promising convergence to better results. The second suggestion is to modify the ANB algorithm according to the initial phase of the Flow Deviation algorithm as follows: ANB finds a solution. If the capacity constraint is violated, the flows on all arcs are reduced proportionally, until a feasible flow is obtained. Next, ANB is run again and the flows are increased to a level very close to saturation. The process terminates when one of two cases occurs: either flows have starting values, or the network is saturated. In the former case ANB yields a feasible solution. In the latter case the problem is claimed to be infeasible. Furthermore, we plan to examine the possibility of applying ANB to the optimization of static flows according to other functions, e.g., the lost flow function introduced in (Walkowiak, 2002; 2003a) or functions using the bottleneck metric. An analytic proof and models of ANB algorithm performance are also open problems.

Finally, we must mention that most of previous studies concerning applications of ant algorithms focus on dynamic routing problems. This paper shows that an ant algorithm can be applied also to static network problems with capacity constraints. The paper (Varela and Sinclair, 1999) includes an ant algorithm for a static routing problem without capacity constraints. However, according to (Gendron *et al.*, 1998) and the author's experience, the capacity constraint makes the flow allocation problem very difficult and the algorithm by Varela and Sinclair could not be applied directly to the NBFA problem. Due to the problem-dependent constraints, the original ant algorithm has to be modified strongly in order to guarantee the feasibility of the results. Therefore, two developed algorithms for ANB and ANBIS have some features of ant algorithms, but also many new elements are added. The role of local heuristics is stronger that in many other ant algorithms. However, when we tested some versions of the algorithm without such complicated local heuristics, the algorithm was unable to find a feasible solution. This conclusion is consistent with the opinion presented in (Dorigo *et al.*, 1999) that the use of local heuristics improves ant algorithm performance considerably. We believe that the results presented in this paper may provide additional support for the development of ant algorithms for static network problems with capacity constraints.

References

Assad A. (1978): *Multicommodity network flows – A survey*. — Network, Vol. 8, No. 1, pp. 37–91.

Bienstock D. (2002): *Potential Function Methods for Approximately Solving Linear Programming Problems, Theory and Practice*. — Boston: Kluwer.

Burns J., Ott T., Krzesinski A. and Muller K. (2003): *Path selection and bandwidth allocation in MPLS networks*. — Perform. Eval., Vol. 52, No. 2–3, pp. 133–152.

Cantor D. and Gerla M. (1974): *Optimal routing in a packet-switched computer network*. — IEEE Trans. Comm., Vol. 23, No. 10, pp. 1062–1069.

Colorni A., Dorigo M., Maffioli F., Maniezzo V., Righini G. and Trubian M. (1996): *Heuristics from nature for hard combinatorial problems*. — Int. Trans. Oper. Res., Vol. 3, No. 1, pp. 1–21.

Corne D., Oates M. and Smith D. (Eds.) (2000): *Telecommunications Optimization: Heuristic and Adaptive Techniques*. — New York: Wiley.

Di Caro G. and Dorigo M. (1998a): *Mobile agents for adaptive routing*. — Proc. 31st Hawaii Int. Conf. *System*, Kohala Coast, Hawaii, USA, pp. 74–83.

Di Caro G. and Dorigo M. (1998b): *AntNet: Distributed stigmergetic control for communications networks*. — J. Artif. Intell. Res., Vol. 9, pp. 317–365.

Dorigo M., Maniezzo V. and Colorni A. (1991): *Positive feedback as a search strategy*. — Techn. Rep. No. 91–016, Politechnico di Milano, Italy.

Dorigo M., Di Caro G. and Gambardella L. (1999): *Ant algorithms for discrete optimization*. — Artif. Life, Vol. 5, No. 2, pp. 137–172.

Elbaum R. and Sidi M. (1996): *Topological design of local-area networks using genetic algorithms*. — IEEE/ACM Trans. Networking, Vol. 4, No. 5, pp. 766–778.

Fratta L., Gerla M. and Kleinrock L. (1973): *The flow deviation method: An approach to store-and-forward communication network design*. — Networks, Vol. 3, pp. 97–133.

Garlick R. and Barr R. (2003): *Dynamic wavelength routing in WDM networks via ant colony optimization*. — Lect. Notes Comput. Sci., Vol. 2463, pp. 250–255.

Gendron B., Crainic T.G. and Frangioni A. (1998): *Multicommodity capacitated network design*, In: Telecommunications Network Planning (B. Sans'o and P. Soriano, Eds.). — Norwell, MA: Kluwer, pp. 1–19.

Girard A. and Sanso B. (1998): *Multicommodity flow models, failure propagation and reliable loss network design*. — IEEE/ACM Trans. Networking, Vol. 6, No. 1, pp. 82–93.

Grover W. (2004): *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. — Upper Saddle River, NJ: Prentice Hall.

Herzberg M., Bye S. and Utano A. (1995): *The hop-limit approach for spare-capacity assignment in survivable networks*. — IEEE/ACM Trans. Networking, No. 6 pp. 775–784.

Kar K., Kodialam M. and Lakshman, T.V. (2000): *Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications*. — IEEE J. Select. Areas Commun., Vol. 18, No. 12, pp. 2566–2579.

Karp R. (1975): *On the computational complexity of combinatorical problems*. — Networks, Vol. 5, No. 1, pp. 45–68.

Kasprzak A. (2001): *Design of Wide Area Networks*. — Wrocław: Wrocław University of Technology Press, (in Polish).

Kassabalidis I., El-Sharkawi M.A., Marks II R.J., Arabshahi P. and Gray A.A. (2001): *Swarm intelligence for routing in communication networks*. — Proc. IEEE Conf. *Globecom*, San Antonio, Texas, USA, pp. 541–546.

Murakami K. and Kim H. (1996): *Virtual path routing for survivable ATM networks*. — IEEE/ACM Trans. Networking, Vol. 4, No. 1, pp. 22–39.

Ott T., Bogovic T., Carpenter T., Krishnan K.R. and Shallcross D. (2001): *Algorithms for flow allocation for multi protocol label switching*. — Telcordia Techn. Memorandum TM-26027.

Pióro M. and Medhi D. (2004): *Routing, Flow, and Capacity Design in Communication and Computer Networks*. — Amsterdam: Morgan Kaufmann.

Pióro M., Srivastava S., Krithikaivasan B. and Medhi D. (2003): *Path generation technique for robust design of wide area communication networks*. — Proc. 10-th Polish Teletraffic Symp., Cracow, Poland, pp. 67–80.

Schoonderwoerd R., Holland O. and Bruten J. (1997): *Ant-like agents for load balancing in telecommunications networks*. — Proc. Conf. *Agents'97*, Marina del Rey, California, USA, pp. 209–216.

Schwartz M. and Cheung C. (1976): *The gradient projection algorithm for multpile-routing in message switched networks*. — IEEE Trans. Comm., Vol. 24, No. 4, pp. 449–456.

Subramanian D., Druschel P. and Chen J. (1997): *Ants and reinforcement learning: A case study in routing in dynamic networks*. — Proc. Int. Joint Conf. *Artificial Intelligence*, Nagoya, Aichi, Japan pp. 832–838.

Szeto W., Boutaba R. and Iraqi Y. (2002): *Dynamic online routing algorithm for MPLS traffic engineering*. — Lect. Notes Comput. Sci., Vol. 2345, pp. 936–946.

Varela N.G. and Sinclair M.C. (1999): *ant colony optimisation for virtual-wavelength-path routing and wavelength allocation*. — Proc. Congress *Evolutionary Computation*, Washington, USA, pp. 1809–1816.

Walkowiak K. (2001a): *Genetic approach to virtual paths assignment in survivable ATM networks*. — Proc. 7-th Int. Conf. *Soft Computing MENDEL*, Brno, Czech Republic, pp. 13–18.

Walkowiak K. (2001b): *Ant algorithms for design of computer networks*. — Proc. 7-th Int. Conf. *Soft Computing MENDEL*, Brno, Czech Republic, pp. 149–154.

Walkowiak K. (2002): *The branch and bound algorithm for backup virtual path assignment in survivable ATM networks*. — Appl. Math. Comput. Sci., Vol. 12, No. 2, pp. 257–267.

Walkowiak K. (2003a): *A new approach to survivability of connection oriented networks*. — Lect. Notes Comput. Sci., Vol. 2657, pp. 501–510.

Walkowiak K. (2003b): *Heuristic algorithms for assignment of non-bifurcated multicommodity flows*. — Proc. Conf. *Advanced Simulation of Systems ASIS*, Sv Hostyn, Czech Republic, pp. 243–248.

Walkowiak K. (2004): *A branch and bound algorithm for primary routes assignment in survivable connection oriented networks*. — Comput. Optim. Applic., Vol. 27, No. 2, pp. 149–171.

White T., Pagurek B. and Oppacher F. (1998): *Connection management using adaptive mobile agents*. — Proc. Int. Conf. *Parallel and Distributed Processing Techniques and Applications*, Las Vegas, USA, pp. 802–809.

Received: 4 March 2004
Revised: 24 August 2004