ANT ROUTING VS. Q-ROUTING IN TELECOMMUNICATION NETWORKS

Andrzej Pacut, Małgorzata Gadomska, Andrzej Igielski
Institute of Control and Computation Engineering
Warsaw University of Technology
00-665 Warsaw, Poland

Abstract—**In our simulations we analyzed two classes of adaptive routing algorithms: Q-routing algorithms based on reinforcement learning techniques, and ant-routing based on swarm intelligence. We use a simulation environment to compare the performance of adaptive routing algorithms. Performance of adaptive algorithms was tested under a constant load, and also under various non-stationary loads, to find out how load time dynamics might interfere with learning dynamics. We tested step load changes and periodic load changes. We also tested an influence of non-homogeneous loads on the behavior of routing algorithms, motivated by DoS and DDoS network attacks. We investigate robustness to parameter changes, and adaptiveness to dynamically changing load levels and traffic patterns. We demonstrated that both classes of adaptive routing algorithms adapt well to occasional changes of the load and perform very well under periodic load level changes.**

Keywords— **Ant routing, Q-routing, Reinforcement learning, Adaptive routing**

I. INTRODUCTION

Expansion of the Internet and other communication networks increased the requirement for an efficient traffic organization. Apart from a constant call to increase the link bandwidths, there is a need to improve packet routing algorithms, which play a critical role in the network performance, especially in terms of throughput and transmission delay.

The load of telecommunication networks is neither constant in time (homogeneous) nor uniformly distributed over the whole network (uniform). It is then very important to develop effective routing algorithms which work well under constant conditions yet may adapt to dynamical and unpredictable changes, such as

- changes in the network load distribution, which may occur in time as well as in space (non-homogeneous and non-uniform loads)
- changes in the network topology, which may be caused by router or link failures, or by movements of some routers which may occur in mobile or ad-hoc networks.

There is thus a strong need to construct adaptive algorithms which could change their routing politics according to the information available in the network.

In recent years, many such algorithms have been proposed. We concentrate on two classes of distributed learning procedures, namely, the *ant routing* algorithms based on swarm intelligence and *Q-routing* algorithms based on reinforcement learning techniques. Both approaches have already proved their efficiency in various simulated environments. Both do not require a supervision, and their distributed form makes them well applicable to the routing problem.

The aim of this paper is to compare the performance of ant-like agents and reinforcement learning agents in adaptive routing problems. We investigate their robustness to parameter changes, and their adaptiveness to dynamically changing load levels and traffic patterns. We show that both classes not only adapt well to momentary changes of load level, but also perform very well under continuous load level changes.

The paper is organized as follows. In Sec. II we shortly introduce the swarm intelligence and reinforcement learning approaches to adaptive routing, and describe Q-routing in Sec. III and ant-routing in Sec. IV. The simulation environment is described in Sec. V. Simulation results for constant loads are summarized in Sec. VI, which are followed by non-stationary load simulations in Sec. VII and non-uniform load simulations in Sec. VIII. Section IX summarizes the paper.

II. ADAPTIVE ROUTING

A. Q-routing

Reinforcement learning techniques are based on interactions between an agent(s) and the - (unknown or known) environment. In every state, the agent chooses an *action* according to his *policy* and receives a *reinforcement* from the environment evaluating his actions. On the basis of the reinforcement received and the state reached after taking the chosen action, the agent modifies his policy in order to attain his goal, which is typically a discounted sum of all rewards. There are various learning algorithms which govern these modifications.

The first application of reinforcement learning to the routing problem was proposed by Boyan and Littman in 1994 [1]. The proposed solution was an adaptation of the Q-learning algorithm of Watkins and Dayan [8]. In the following years, many modifications of the Q-routing have been proposed, modeling queue lengths influence [4], using Boltzman's routing policy [9], or applying the policy gradient methods ([5]). .

B. Ant routing

Ant algorithms were inspired by observations of real ant colonies and first proposed by Dorigo, Maniezzo, and Colorni in 1991 [2] as a multi-agent approach to the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). The ants are capable of finding the shortest path from the nest to food sources. While traveling, they deposit a chemical substance (a pheromone), and its high concentration guides the other ants to form the shortest paths.

The first ant routing algorithm was proposed by Schoonderwoerd *et al.* in 1996 [6]. Their ABC algorithm could be applied only to symmetric circuit-switched networks. The AntNet algorithm proposed first by Dorigo and di Caro in 1993 [3] is the first ant-routing algorithm investigated in this paper. Many modifications of AntNet have been developed in the following years, like limiting the number of ants in the network [7], or non-greedy policy determination. The Adaptive Swarm-based Routing (ASR) proposed in 2004 for packet-switched networks by Yong, Guang-zhou, and Fan-jun [10] is the second ant routing algorithm simulated in our studies.

III. Q-ROUTING ALGORITHMS

In Q-routing [1], each (say, k-th) router (agent) learns an estimate $Q_k(d, n)$ of packet's trip time to a destination node d, via its neighbor $n \in \mathbf{N}_k$. The *greedy routing policy* consists in choosing the route to d through a neighbor node n to minimize the total trip time, $n = \arg\min_{n' \in \mathbf{N}_k} Q_k(d, n')$. This policy enables no *exploration*, since it bases the future estimates only on the extremal routes chosen by using earlier (uncertain) estimates. This 'closed loop of uncertainty' can be cut by using *ϵ-greedy policy* which with probability α chooses a random neighbor, otherwise being greedy. Another form of a non-greedy policy is the *Boltzmann policy*, which chooses the next node with probability

$$P_k(d, f) = \frac{e^{-\lambda Q_k(d,f)}}{\sum_{f' \in \mathbf{N}_f} e^{-\lambda Q_k(d,f')}} \tag{1}$$

where λ determines the level of exploration. We use the name BQ-routing for Q-routing using the Boltzmann policy.

When the data packet reaches the next node n, an *information packet* is sent back to node k, which enables to update Q_k. The update procedure has a form

$$Q_k^+(d, f) = o_k(f) + q_k + \min_{z' \in \mathbf{N}_z} Q_f(d, z') \tag{2}$$

$$Q_k(d, f) \leftarrow Q_k(d, f) + \mu \left(Q_k^+(d, f) - Q_k(d, f)\right) \tag{3}$$

where $o_k(f)$ is the trip time from k to f, q_k is the time spent in the queue of router k (or, in the modification proposed in [4], of router f). The *reinforcement* $o_k(f)$ and $Q_f(d, z')$ are included in the information packet.

In the original algorithm [1], the information packet was assumed to immediately arrive at its destination node. Since this is unrealistic, we assume in our simulations that it travels in the network as the data packets do.

IV. ANT ROUTING ALGORITHMS

A. AntNet

In AntNet [3] there are two types of simple agents (ants): the *forward ants* that explore the network in order to find paths, and the *backward ants* that use information collected by the forward ants to improve the routing policies. Every network node k is assigned a routing table P_k that stores the routing policy, and a statistical model M_k including some local traffic statistics. The routing table T_k stores the probabilities $t_k(d, n)$ for each neighbor node n and each destination node d, used to determine the probabilistic routing policy. Both the routing tables and the statistical model are calculated iteratively during normal operation of the network.

The *forward ants* $F_{s \to d}$ are launched at regular intervals from randomly selected *source nodes* s to randomly selected destination nodes d. For each node visited, the forward ant stores its age, i.e. the time elapsed from its launch, and chooses the next node to be visited n according to a probability $p_k(d, n)$. This probability is calculated by modifying the probability $t_k(d, n)$ stored in the routing table T_k by a relative length $q_k(n)$ of the output queue in node k, namely

$$p_k(d, n) = \frac{t_k(d, n) + \alpha\,(1 - \ell_k(n))}{1 + \alpha\,(|\mathbf{N}_k| - 1)}, \quad n \in \mathbf{N}_k \tag{4}$$

where $\mathbf{N}_k$ denote the set of neighbors of the node k, $|\mathbf{N}|$ denotes the number of elements of a set $\mathbf{N}$, α is a parameter and

$$\ell_k(n) = \frac{q_k(n)}{\sum_{n' \in \mathbf{N}_k} q_k(n')} \tag{5}$$

where $q_k(n)$ denote the queue lengths. If a cycle is detected, namely, the ant visits a node repeatedly, the nodes in the cycle are removed from the ant's memory, and if the cycle length is greater than half the ant's age, the ant is destroyed.

After reaching the destination node, the forward ant $F_{s \to d}$ creates the *backward ant* $B_{d \to s}$, transfers all the collected knowledge to the backward ant and is removed from the system (dies). The backward ant travels back the same path as its parental forward ant. In every visited node k it updates the values of the traffic model and the routing probabilities for all the entries corresponding to every node i on the path.

The traffic model M_k at node k consists of the estimates $\mu_k(d)$ and $\sigma_k^2(d)$ of the expected value and the variance of the trip time from k to the destination node d, and are updated according to

$$\mu_k(d) \leftarrow \mu_k(d) + \eta\,(o_k(d) - \mu_k(d)) \tag{6}$$

$$\sigma_k^2(d) \leftarrow \sigma_k^2(d) + \eta\Big(\big(o_k(d) - \mu_k(d)\big)^2 - \sigma_k^2(d)\Big) \tag{7}$$

where $o_k(d)$ is the trip time from k to the destination node d as observed by the forward ant $F_{s \to d}$, and η is a parameter. Note that $\eta = 1/m$ in (6), where m is the update number, corresponds to the usual average value.

In the routing table T_k, the probability corresponding to the neighbor f chosen at node k by the forward ant $F_{s \to d}$ is increased

$$t_k(f, d) \leftarrow t_k(f, d) + r\,(1 - t_k(f, d)) \tag{8}$$

with the corresponding decrease of the remaining neighbor probabilities

$$t_k(d, n) \leftarrow t_k(d, n) - r\, t_k(d, n), \quad n \in \mathbf{N}_k, n \neq f \tag{9}$$

to make their sum equal to one. The *reinforcement parameter* r depends on the ant trip time and on the local statistical model M_k, namely

$$r = c_1 \frac{W_k(d)}{o_k(d)} + c_2 \frac{I_k^{\text{hi}}(d) - I_k^{\text{lo}}(d)}{\big(I_k^{\text{hi}}(d) - I_k^{\text{lo}}(d)\big) + \big(o_k(d) - I_k^{\text{lo}}(d)\big)} \tag{10}$$

where $o_k(d)$ is the observed trip time, constants c_1 and c_2 control the effect of the last $o_k(d)$, and $W_k(d)$ is the ant's shortest trip time for the given destination and last observation period. $I_k^{\text{lo}}(d)$ and $I_k^{\text{hi}}(d)$ are the lower and upper bounds, respectively, of a confidence interval of the mean trip time. All the details and justifications can be found in [3].

B. Ant Swarm-based Routing

The ASR algorithm [10] is very similar to AntNet, so only the main differences between the algorithms will be described.

A different model M_k of the network's traffic is used. For every destination node d and every neighbor node n two last estimates, $D_k(d, n)$ and $D_k^-(d, n)$, of the trip time from k via

n to the destination node d are stored. The backward ant $B_{d\to s}$ updates the values of the model

$$\begin{aligned} D_k(d,n) &\leftarrow D_k(d,n) \\ &+ \eta\left((1-\alpha)\Delta D_k(d,n) + \alpha\Delta D_k^-(d,n)\right) \end{aligned} \qquad (11)\ (12)$$

where $d_k(d,n)$ is the forward ants trip time from node k to node d via node n, η is a learning rate, α is a momentum parameter, and

$$\Delta D_k(d,n) = d_k(d,n) - D_k(d,n) \qquad (13)$$

$$\Delta D_k^-(d,n) = D_k(d,n) - D_k^-(d,n) \qquad (14)$$

The second significant difference is the way of updating the routing probabilities stored in the routing table T_k at every node k. After updating the network's traffic model M_k, the routing probabilities are computed as

$$t_k(d,n) = \frac{(D_k(d,n))^{-\beta}}{\sum_{n'\in\mathbf{N}_k}(D_k(d,n'))^{-\beta}} \qquad (15)$$

where $\beta \geq 1$ is a non-linearity parameter. Note that this method corresponds to Boltzmann exploration since we can write

$$t_k(d,n) = \frac{e^{-\beta\ln D_k(d,n)}}{\sum_{n'\in\mathbf{N}_k} e^{-\beta\ln D_k(d,n')}} \qquad (16)$$

Finally, the ant agent realizes an ϵ-greedy policy, namely, it chooses a random neighbor node with a probability ϵ and with the probability $1-\epsilon$ chooses a node according to the routing probabilities $t_k(d,n)$. This approach prevents the probabilities of the shortest paths from becoming very close to 1, which would reduce the exploration, i.e. the capability of the algorithm to adapt to the environmental changes.

In order to speed up the learning process, the routing probabilities are initialized in such a way that a higher initial probability is given to a neighbor when it happens to be a destination node.

V. SIMULATION ENVIRONMENT

The experiments were performed using a simple event-driven network simulator implemented in Java. Every router independently generated the packets according to a Poisson process. We tested several scenarios of load level changes, such as a sudden jump/drop of the load level, periodical changes of the load level (sine wave, rectangular wave). Moreover, we performed non-homogeneous load testing, simulating the DoS (Denial of Service) and DDoS (Distributed Denial of Service) attacks. The DoS attack occurs when an extra traffic load is being sent between two nodes and DDoS occurs when a large group of nodes produce extra traffic to a single network node.

The experiments presented in this paper were performed using the network structure proposed by Boyan and Littman (Fig. 1). To check the performance for more real-like structures, we also run our experiments on NASK network (Scientific and Academic Network, Poland (Fig. 2).

We used two closely related indicators of the learned policy quality. The first is the average packet delay, and the second

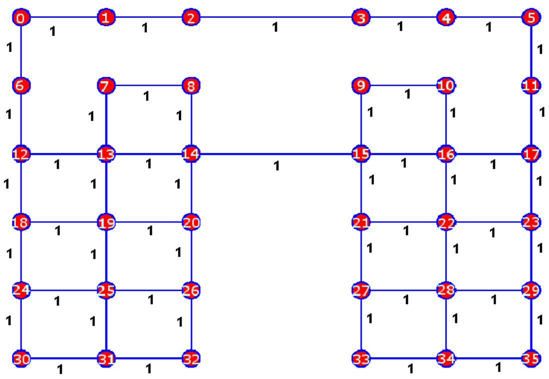
Fig. 1. Littman's grid, [1]

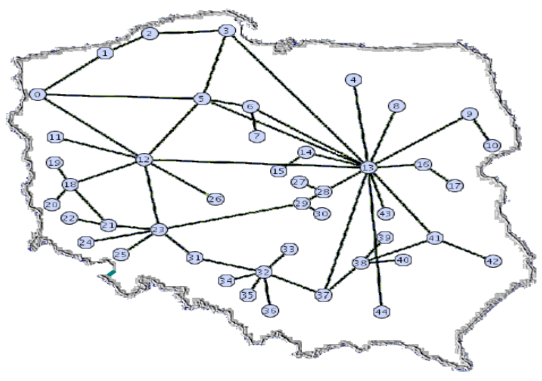
Fig. 2. The NASK network

is the throughput (the number of packets reaching their destination per simulation step). To measure the learning time we used the time necessary for the average packet delay to stabilize. To express the network load we used relative units, with the unit defined as the maximum load level under which the Dijkstra Shortest Path (SP) algorithm still works. For short, we called such the relative unit the Dijkstra (D). In other words, the network load is 1.5 D if it is 50% higher than the maximum load under which the given network still works with the SP algorithm policy. Therefore, all loads above 1D cannot be served by the SP policy. We want to investigate how ant-routing and Q-routing serve above-1D loads. On the other hand, we will make sure that these adaptive algorithms work as well as the SP policy for below-1D loads. We will talk about *high loads* if they are above 1D. Otherwise, we will talk about *low loads*.

VI. CONSTANT LOADS

Under a low load level (lower than 1 D), the Shortest Path routing is certainly optimal. All investigated adaptive algorithms learned the policies similar to the shortest path (Fig. 3). The average packet delay in the network stabilized at a level slightly higher than the one for the SP algorithm. Both ant algorithms we simulated not only adapt much faster than the Q-Routing based algorithms, but also the maximal average packet delay during the learning process is lower. It can be seen that ASR provides a shorter learning time.

The advantages of adaptive policy algorithms can be seen under constant load levels higher than 1D. While the Shortest Path algorithm does not work efficiently, both the Q-routing

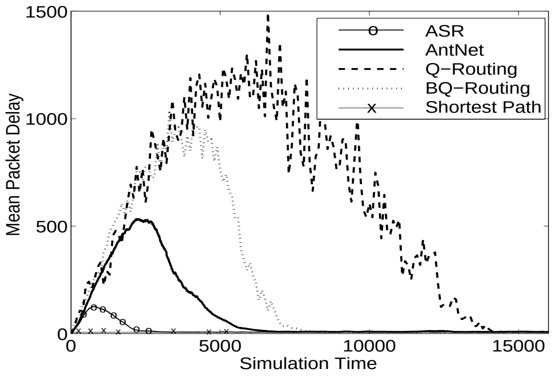

Fig. 3. Performance comparison of SP, Q-routing and ant-routing under low load level (0.9 D), Grid network.

and ant-routing algorithms manage to learn an efficient policy and converge to steady states (Fig. 4). The use of Boltzmann policy in Q-routing instead of the ϵ-greedy policy markedly improves the learning time and the average packet delay. Both ant algorithms, AntNet and ASR, are able to adapt much faster than the Q-Routing based algorithms and the maximum value of the average packet delay during the learning process is lower. Moreover, the average packet delay of the learned process is also lower for ant algorithms.

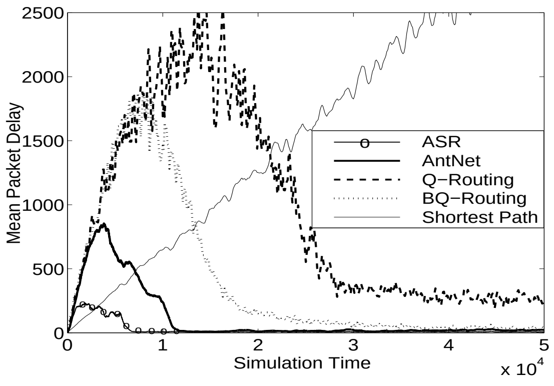

Fig. 4. Performance comparison of SP, Q-routing and ant-routing under high load level (1.2 D), Grid network.

In comparison to the Shortest Path algorithm, the adaptive algorithms in most cases extend the range of load levels under which the algorithms succeed in finding the efficient routing policies. This is demonstrated on the NASK network (Fig. 5), where the adaptive algorithms (ASR and BQ-Routing) work efficiently under much higher load levels than the SP algorithm. The ASR and BQ-routing algorithms widen the range of load levels at a similar factor.

A. Robustness against parameter changes

For each algorithm we determined the range of parameter that assure a stable work of tested networks. For ant algorithms, for every load, there exists a minimal number of ants that assures convergence of learning, and the optimal number of ants that minimizes the learning time, Fig. 6.

With an increase of the ant proportion, defined as a fraction of ants in all packets, the learning time decreases, along with the average number of packets traveling in the network,

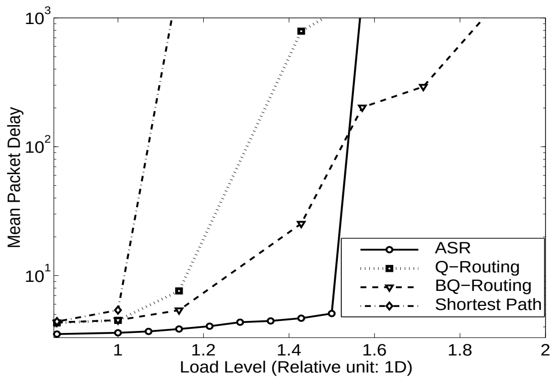

Fig. 5. Comparison of the convergence ranges of SP, ASR, and BQ-routing, NASK Network.

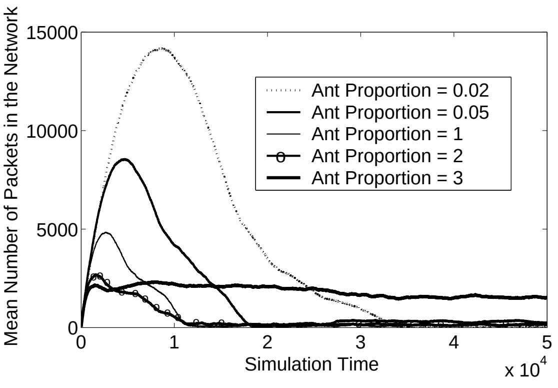

Fig. 6. The influence of the ant proportion on the learning process. AntNet algorithm, the Grid network, load level of $1.2D$.

since the information about the efficient routes propagates faster through the network. There is yet a certain critical ant proportion level, whose overpass increases the total load so much that the learning time increases again.

In Q-Routing, the exploration level is determined by ϵ. Increase of the exploration level increases the number of packets being sent purely randomly. This results in a longer learning time and a much higher value of the average number of packets in the network. Change of ϵ doesn't influence the maximum value of the average number of packets during learning. The minimum learning time and the lowest average packet delay is obtained for the null exploration level (Fig. 7). Like for the ant routing, too high exploration makes it impossible for the learning process to converge.

VII. NON-STATIONARY LOADS

Performance of adaptive algorithms was tested also under various non-stationary loads, to find out how load time dynamics might interfere with learning dynamics. We tested step load changes and periodic load changes. Both scenarios have important origins in network exploitation.

In step load simulations, the algorithm first learned a policy under a constant load level, and then the load level jumped up rapidly. For both the ant routing and the Q-Routing algorithms the adaptation process in this situation was similar: immediately after the load level jump, the average packet delay started increasing and the adaptation process begun. All the algorithms under testing managed to find a new efficient routing policy.

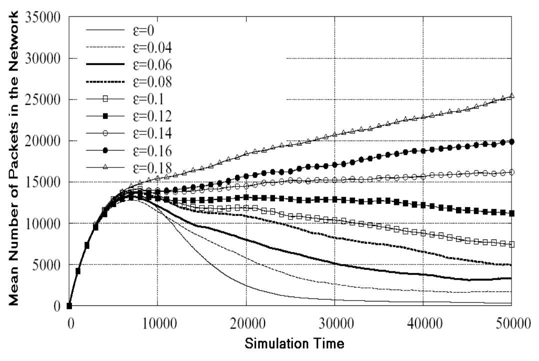

Fig. 7. Influence of the exploration level ϵ on the learning process. Modified version of Q-Routing ([4]), the Grid network, load level: *1.2 D*.

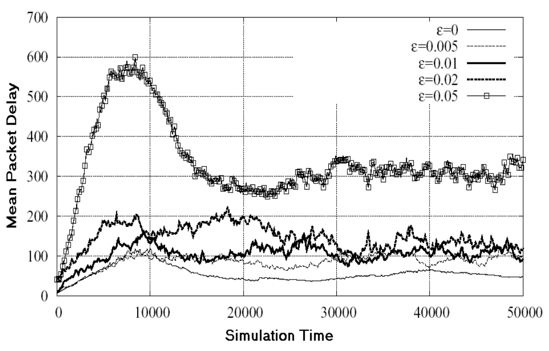

Fig. 9. Dependence of the adaptation process on the exploration level, load jump from 0.9 D to 1.2 D. Grid network, modified Q-Routing ([4]).

If the load level increase was small, the adaptation was very quick. The higher were the final load level and the size of the jump, the longer lasted the adaptation process and higher was the maximum average packet delay during the learning process (Fig. 8). This behavior is related to strong differences between the efficient routing policies for the initial (low) and subsequent (higher) loads.

As for constant load levels, the speed and the result of the learning process intensely depends on the exploration level (Fig. 9). Increase of the exploration factor results in a longer adaptation time and a higher average packet delay. The adaptation is fastest when there is no exploration at all.

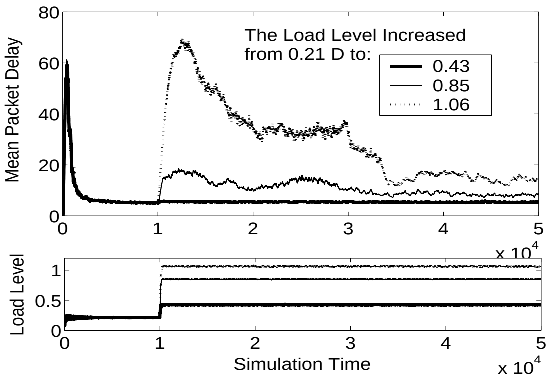

Fig. 8. Adaptation to jump increase of the load level for various jump sizes. The initial load level is $0.21D$. Grid Network, AntNet.

The second load patterns tested were periodic. We applied a sine wave load with values at maximums exceeding the 1 D level so the shortest path was not efficient. All the studied learning algorithms very quickly adapted to periodic load changes (Fig. 10). Under the sinewave load, the average packet delay and the average number of packets in the network also change periodically. The amplitude of the changes of the average number of packets in the network is lowest (the best) for ASR and highest (the worst) for the Q-Routing algorithm.

A delay can be observed between the initial time of the load increase and the starting time of the average packet delay increase (Fig. 10). The delay is highest for Q-Routing. If the period of the sinewave is long enough, the adaptation process takes place during the first increase phase of the load level (ASR and BQ-Routing). However, if the algorithm does not

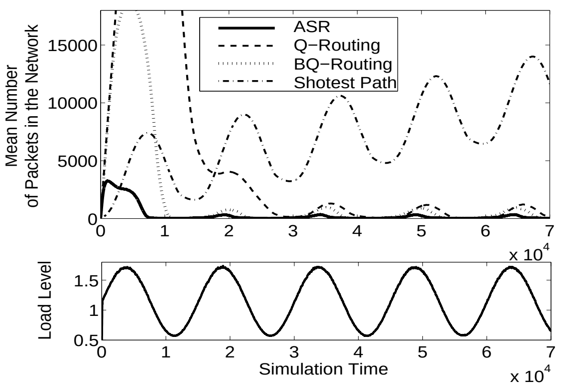

Fig. 10. Sinewave load level changes. The average load level varies between 0.6 D and 1.7 D. NASK network.

manage to adapt that fast, the process continues in successive periods of the load increase phases (Q-Routing).

We finally applied rectangular wave load changes, which blend both periodic and jump non-stationarity. As for the sinewave loads, the learning algorithms adapt very quickly (Fig. 11). Under the Shortest Path algorithm, if the maximum load is higher than 1D, the average packet delay grows rapidly. The learning algorithms, however, manage to find efficient routing policies and, after the first increase of the average packet delay, it stabilizes. The amplitude of the changes of the average packet delay and the maximal average packet delay during adaptation are the lowest for the ASR.

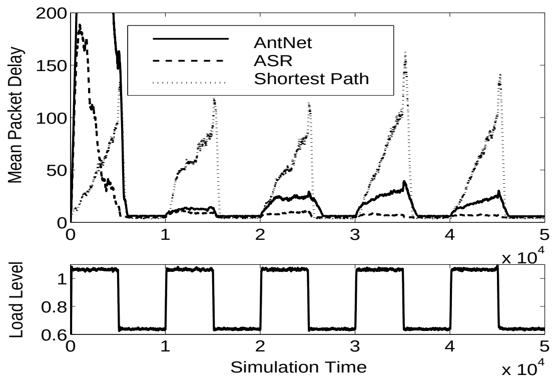

Fig. 11. Comparison of algorithms under rectangle wave load level changes. The average load level varies between 0.64 D and 1.06 D. Grid network.

VIII. NON-HOMOGENEOUS LOADS

We tested an influence of non homogeneous load on the behavior of non-adaptive and adaptive routing algorithms. This is, among other, motivated by DoS and DDoS network attacks. When using the Shortest Path algorithm, both DoS or DDoS attack result in a huge increase of the average packet delay, because the queues for the attacked router get filled. The adaptive algorithms, however, manage to cope with these situations very well and find efficient routing policies (Figs. 12 and 13);

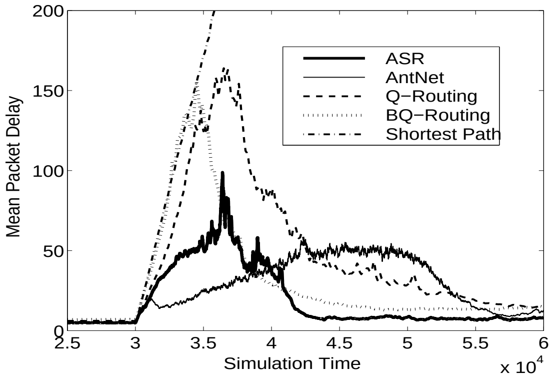

Fig. 12. DoS attack. A homogeneous network load is initially 0.66 D; the DoS attack after 30000 simulation steps generates extra 0.1 D. Grid network.

For Q-Routing algorithms, the average packet delay grew first very rapidly after the DoS attack start and then decreased exponentially (Fig. 12). For ASR, the growth was not so rapid, but the decrease was very fast, what resulted in the shortest learning time. For AntNet, the packet delay increased slowly and started to decrease much later, so the adaptation took longer.

A rapid increase of the average packet delay immediately after the DoS attack starts can be observed for all algorithms. For ASR, the adaptation is much faster than for AntNet, but AntNet provides a lower average packet delay after the adaptation (Fig. 13, left). The BQ-Routing adapted much faster than Q-Routing, yet the average packet delay was similar (Fig. 13, right).

IX. CONCLUSIONS

We compared the ant routing algorithms and Q-Routing algorithms as applied to routing problems. We performed our simulations on a simple Grid network, as well as on various actual structures, like Polish Scientific and Academic Network NASK. Both types of the adaptive algorithms we investigated perform very well in dynamic networks, under high and/or time varying load levels. Adaptive algorithms, as compared to the Shortest Path algorithm, assure an increased load range under which the network is stable and works properly. Moreover, the learning algorithms easily adapt to periodical load level changes.

This would be interesting to include in simulations some restrictions introduced by the TCP protocol at the transport layer, and some modifications of the TCP protocol that would enable multi-path routing. We plan also to introduce adaptive modifications of parameters due to simulation conditions, such as the load level. This would pave the way to testing in real networks.

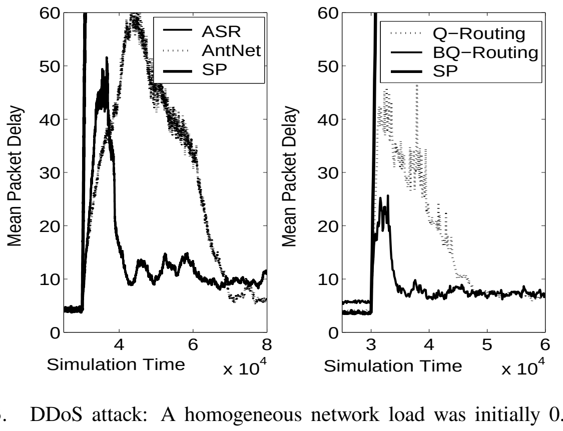

Fig. 13. DDoS attack: A homogeneous network load was initially 0.66 D, then DDoS attack was performed after 30000 simulation steps. Grid network (left), NASK network (right).

References

[1] J. A. Boyan and M. L. Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach", *Advances in Neural Information Processing Systems*, volume 6, pages 671ℓ678, Morgan Kaufmann Publishers, Inc., 1994.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy", *Technical Report 91-016*, Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica, 1991.

[3] M. Dorigo and G. di Caro, "AntNet: Distributed Stigmergetic Control for Communications Networks", *Journal of Artirtificial Intelligence Research 9*, 317-365, 1998.

[4] S. Kumar and M. Risto, "Dual reinforcement Q-Routing: An on-line adaptive routing algorithm", 1997.

[5] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing", *In Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2002

[6] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based load balancing in telecommunications networks", *Adaptive Behavior, 5(2)*, 169ℓ207, 1996.

[7] F. Tekiner, F. Z. Ghassemlooya, and S. Al-khayattb, "The Antnet Routing Algorithm - A Modified Version", http://soe.unn.ac.uk/ocr/papers/2004/ Firat_Tekiner_CSNDSP2004_2.pdf

[8] C. J. C. H. Watkins, and P. Dayan, "Q-Learning", *Machine Learning*, 8:279ℓ292, 1992.

[9] K. Wawrzyniak and A. Pacut, "Reinforcement learning based methods for communications networks", Report 05-04 of the Institute of Control and Computation Engineering, Warsaw University of Technology, 2005.

[10] Lu Yong, Zhao Guang-zhou, and Su Fan-jun, "Adaptive swarm-based routing in communication networks", *Journal of Zhejiang University SCIENCE, 5(7)*, 867-872, 2004.

AUTHOR BIOGRAPHIES

Andrzej Pacut, Ph.D., D.Sc. received his M.Sc. in Control and Computer Engineering in 1969, Ph.D. in Electronics in 1975, and D.Sc. in Control and Robotics in 2000. Since 1969 he is with Warsaw University of Technology, presently being a professor at the Institute of Control and Computation Engineering. Since 2001 he is also the head of Biometric Laboratory of Research and Academic Computer Network NASK. He was Visiting Prof. in the Lefschetz Center for Dynamic Systems at Brown University, Providence, Rhode Island 1980–1981, and Visiting Prof. in the Department of Electrical and Computer Engineering of Oregon State University, Corvallis, Oregon, 1986-1991. He serves as the president of the Poland Section of the IEEE. He is interested in learning systems, neural networks, biometrics, identification, stochastic modeling, and related areas.

Małgorzata Gadomska received her M.Sc. in 2005 from the Faculty of Electronics and Information Technology of the Warsaw University of Technology. She is presently a Ph.D. student at the Institute of Control and Computation Engineering at the Faculty of Electronics and Information Technology of the Warsaw University of Technology. She is interested in artificial intelligence, ant algorithms, and adaptive learning

Andrzej Igielski received his M.Sc. in 2005 from the Faculty of Electronics and Information Technology of the Warsaw University of Technology.