Learning in Cross-layer Wireless Network Optimization

Elena Meshkova, Janne Riihijärvi and Petri Mähönen*

Department of Wireless Networks, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email: {eme, jar, pma}@mobnets.rwth-aachen.de

ABSTRACT

We study the use of learning for cross-layer optimization of wireless networks. In particular, we incorporate learning in the form of *graphical models* into our *cognitive engine* performing network utility maximization task using simulated annealing. Our results show that this learning approach can significantly accelerate the convergence rate of the optimizer, and help in adjusting to changes in network conditions. However, we also observed significant differences in the behavior and performance between the various types of graphical models studied. We discuss these results at length, and identify some of the key challenges faced when incorporating learning into cross-layer optimization design.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design– *Wireless communication*

General Terms

Optimization, network and resource management

Keywords

Wireless networks, cross-layer optimization, simulated annealing, approximate graphical models

1. INTRODUCTION

Cross-layer optimization is a promising approach for improving network performance, especially in heterogeneous and wireless environments [6]. However, despite a large amount of work, general purpose algorithmic solutions for solving cross-layer optimization tasks have not emerged. Indeed, most of the approaches documented in the literature focus on dealing with the interactions of particular protocol combinations in relatively static environments. We study the application of *learning* in conjunction with black-box optimizers as a basis for *generic* cross-layer optimization tool, called the *cognitive engine* (CE). Our focus is on small and medium-sized edge networks, especially ones containing wireless links, since in such networks the potential advantages of cross-layer optimization are the highest. We show that our approach can lead to significant improvements in the quality of experience for different applications in a variety of scenarios. However, more work is needed to better understand the limitations of the methodology as well.

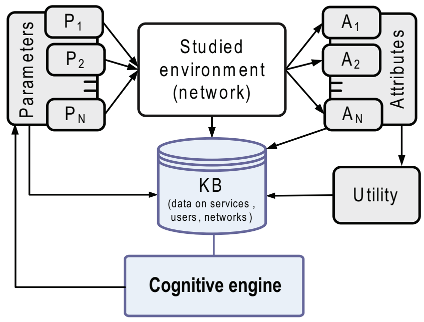


Figure 1: The system architecture considered.

The setting of our work is illustrated in Figure 1. The cognitive engine is a part of a autonomic feedback control loop [2], and acts as an intelligent agent [4]. We assume that the quality of the network connection for a particular application is measured by a *utility function* $U(a_1, \ldots, a_n)$, which depends on a number of measurable *network attributes* $\{a_i\}$. Most of the networking objectives today such as QoS classes or elastic bandwidth-sharing typical to file-transfer applications can be thought of as utilities (see Figure 2). Examples of common attributes include throughput, delay, jitter and packet error rate. The attributes are in turn influenced by a number of *configurable parameters* $\{p_i\}$, ranging from a choice of protocols used on different layers to individual settings of those protocols. The possible values of the parameters depend on the used protocols and technologies, as well as a number of user-defined *policies*. However, the influence of parameter settings to attribute values is in general unknown, and depends on various network externalities such as the topology and interference levels. Therefore, even if utilities as functions of attributes are convex, their mapping to the parameter space leads in general to non-convex search landscape. This is precisely where the application of

*Authors would like to thank RWTH Aachen University and EU (Aragorn-project) for financial support. We would also like to thank Andreas Achtzehn for his help.

Learning for Networking Workshop (in conjunction with SIGMETRICS/Performance) June 15, 2009, Seattle, USA.

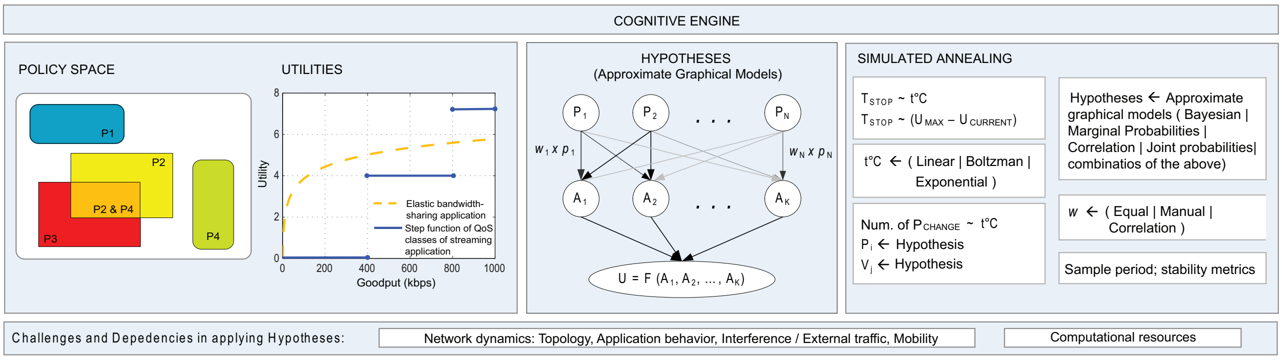


Figure 2: The CE design estimates the network performance using user-defined utility functions and policies, exploits the prior knowledge stored in the form of hypotheses, and runs the adjusted version of the SA to conduct optimization.

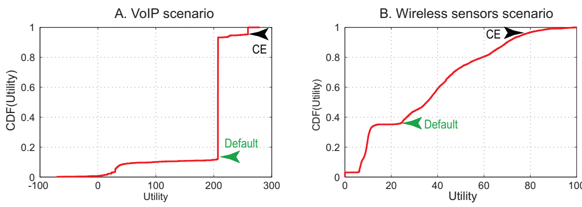


Figure 3: Cumulative distribution function of utilities.

learning is done in our system.

In order to obtain the solution to the optimization problem at hand, namely the maximization of the overall utility of the network as a function of the parameters, we apply *simulated annealing*, a well-known metaheuristic optimization method. Naive application of basic simulated annealing results in slow convergence to a reasonable solution, or even in no convergence at all provided the network conditions are dynamic enough. Accordingly, we use knowledge accumulated by various learning techniques to predict the expected impact of parameter changes to attributes and, therefore utilities, to increase the convergence rate of the optimizer. This information is stored in the *knowledge base* of our system architecture depicted in Figure 1, and used by the cognitive engine during the optimization process. An example of the potential of this approach is shown in Figure 2, contrasting the utilities obtained in two different scenarios (VoIP call over Wi-Fi links and a wireless sensing application) using the default protocol settings against ones obtained using the CE. Clearly the improvement is significant.

The rest of the paper is structured as follows. In Section 2 we describe the introduction of learning into the optimizer. We study difference in the performance of the alternative versions of the CE in Section 3, and state the challenges related to the learning approaches. Finally, we draw extensive conclusions and outline the future work in Section 4.

2. LEARNING AND OPTIMIZATION

Network utility maximization approach [3] allows to formulate a uniform optimization task that can be achieved using simulated annealing (SA) search [5]. We define the search space for SA as a multi-dimensional landscape where each dimension corresponds to one adjustable parameter. The basic simulated annealing explores the parameter space essentially randomly, with the change of the parameter values between time steps proportional to a *temperature parameter*. Usually temperature is reduced monotonically, resulting in the convergence of the algorithm. The additional condition for stopping the search in the CE is the absence of significant utility improvement for a certain number of iterations. The *reheating* of the system occurs either in the case of performance degradation, or periodically to better explore the search space or detect improved network conditions.

In order to further accelerate the convergence of simulated annealing, our design incorporates learning via use of approximate *graphical models* estimating the likely impact of a parameter change to an attribute and thereby utility [1]. These likelihoods, a.k.a *hypotheses*, are encoded as weights on edges of a bipartite graph, with the vertices of two types being precisely the parameters and attributes of the system (see Figure 2). Changes that are likely to result in large increase of utility are favored over others. We explore different flavors of graphical models in terms of their performance in accelerating the cross-layer optimization process. Simplest of these use marginal probabilities and statistical correlations as approximations of the likelihoods of positive attribute/utility responses to a parameter change. We also study the performance of *Bayesian networks*, sophisticated graphical models estimating the full joint probabilities of the changes in attributes as parameters are being tuned by the simulated annealing. As a final design parameter, our system allows for the introduction of *prior* weights on the model, which can be used to manually or automatically incorporate long-term knowledge on expected system behavior. One way to estimate weights is to use parameter-utility correlation coefficients that, according to our experiments, tend to hold for a wide range of network conditions.

3. EVALUATION AND DISCUSSION

In this section we study the performance of the CE, and challenges faced when applying various graphical models.

3.1 Scenarios and Assessment of the CE

We evaluate the performance of the cognitive engine in two different scenarios. The focus of the first scenario is

[1]Since we employ the optimizer even in the case no prior training data is available, the use of graphical models only is not feasible.

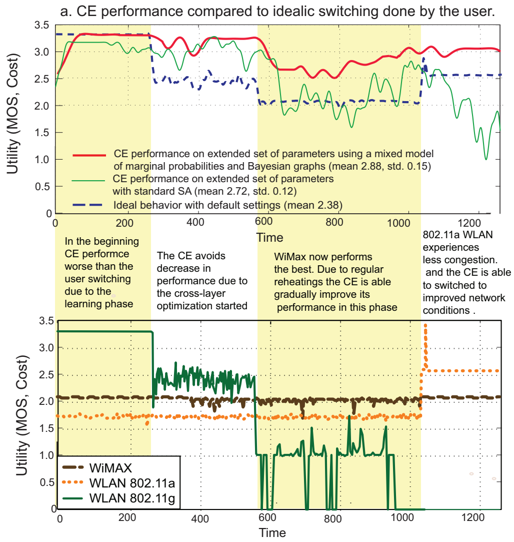


Figure 4: The performance in the VoIP scenario showing learning and optimization capabilities of the system.

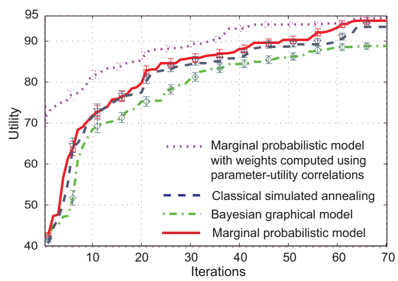


Figure 5: Typical performance of SA with different graphical models in the WSN scenario.

an access selection and configuration with multiple wireless interfaces available to a terminal. This scenario is specified in the Qualnet network simulator [1]. The second scenario incorporates a range of Wireless Sensor Networks (WSNs). These networks allow for extensive cross-layer optimization, due to the absence of established protocol stacks and the usage of easily disrupted low-power radios.

The cognitive engine is realized in MATLAB. It periodically queries the studied environment to get information on the performance of the nodes of interest, and adjusts their parameters based on the results of the decision process.

3.1.1 VoIP Scenario

The scenario, where the user initiates a VoIP call, illustrates the behavior of the cognitive engine in a dynamic environment. The utility function is a combination of the Mean Opinion Score (MOS) and a cost function. The user has access to free WLAN 802.11g and 802.11a access points that experience increasing congestion (see Figure 4.b). The user has to pay for WiMAX usage, but the capacity of this network satisfies all her requirements. We assume seamless vertical handovers in this scenario. The "ideal" device behavior in terms of interface switching, with protocol parameters kept at default levels, are shown by the dashed line in Figure 4.a. The parameters and their values adjustable by the CE are given in the Table 1.

Table 1: The optimized parameters and their values. Boldface indicates the default parameter values.

Protocol	Parameter	Values
VoIP scenario		
IP	Max. packet size	500, 1000 **1500**, 2000, 3000
MAC	cwMin	2, **5**, 20
	cwMax	300, 1023, **2100**, 4000
	Short retry limit	1, **3**, 5
	RTS/CTS thres.	50, 600, 1600, **10000**
Access Method		802.11a, 802.11b, WiMax
WSN scenario		
Application	Sensing interval	200, **800**, 1400
Routing	Beaconing interval	5, **20**, 40
	Beaconing timeout	200, **800**, 1500
MAC	Max. retransmissions	1, **3**, 9
	Duty cycle	10, 50, 80, **100**
	Acknowledgements	"on", **"off"**
	Backoff symbol length	5, **30**, 40

Example results for the scenario are shown in Figure 4. Even the basic version of the CE on average performs better by 14% than the "ideal" device behavior describe above. However, occasionally the engine with the trivial version of SA acts notably worse than the default functioning (light solid line in Figure 4.a). If we enhance the SA with learning capabilities, the average performance is improved, achieving the gain of 21%, with almost constant run-time win in the utility of the optimizer over the "ideal" switching behavior (dark solid line in Figure 4.a). This scenario, besides demonstrating the overall validity of our approach to the network optimization, also illustrates the feasibility of learning in the networks with cross-layer optimization capabilities.

3.1.2 Wireless Sensor Network Scenario

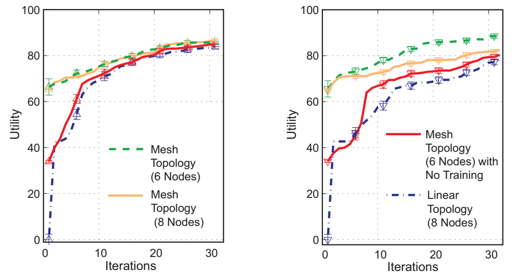
We also evaluate the performance of various flavors of the cognitive engine in several wireless sensor networks. Topologies are six- and eight-node meshes, and an eight-node linear chain. The utility function used aims at minimizing power consumption of the network while fulfilling basic constraints for packet delay and losses. Table 1 reflects the possible parameter value combinations for the CE.

The performance of the CE in the WSN scenario is shown in Figure 5. The marginal probabilistic graphical model performs fairly well in comparison to a standard implementation of SA with improvement in convergence rate by 15 iterations. This is roughly 25% reduction in required iterations considering that the basic algorithm converges to 90% of maximum utility in 60 iterations. The SA flavor based on Bayesian graphs performs the worst due to the lack of training data, converging to 90% utility only after 80 iterations.

The application of the weights computed using parameter-utility correlation coefficients estimated after a single run of the CE on the same topology leads to additional performance increase. The 90% of the maximum utility is achieved at the 35^{th} iteration, resulting in 58% gain.

3.2 Challenges in Applying Learning

Care should be taken when applying hypotheses in an op-



a. Hypothesis after marginal probabilities. b. Hypothesis after a bayesian network.

Figure 6: Training hypotheses on different topologies. Applying them on the six node mesh network.

timizer design. Our experiments confirm the intuitive assumption that the more specific the hypothesis, the fewer network conditions it is suitable for. We trained hypotheses on three different WSN topologies (the eight node linear network, the six node and the eight node meshes). We used these hypotheses to optimize of the network exhibiting one of these topologies (see Figure 6). As expected the CE with the hypothesis trained on the most different topology (the mesh vs. the linear network) performs even worse than version of the system with no training at all. However, results obtained using the hypothesis from a similar network (another mesh) are notably better leading to higher utility gains in the beginning of the optimization.

The simpler the graphical model used for learning the smaller the difference in the results obtained are after a large number of search iterations, as these hypotheses are capable of faster re-learning on new data. More sophisticated models converge faster than the simple ones (by approximately 15 iterations) provided that that the sufficient amount of the appropriate training data is available. Otherwise, they achieve worse results than the simple models.

We observed similar behavior in the VoIP scenario as well. By varying the network load, instead of the topology, we determined that the CE performs better with untrained hypotheses rather than the trained ones in case of drastic and frequent changes in network conditions.

Another major challenge in the cross-layer optimization is the choice of appropriate parameters and their value sets. There is a performance tradeoff between the fast convergence of the process and the quality of the results achieved. The variables to be accounted for are the number of parameters tweaked, the length of the search phase and the diversity of the parameter values tried. It is evident that the higher the number of possible permutations the harder it is to locate a near-optimum value and that more search iterations are needed. However, a large number of permutations is also likely to lead to the high absolute utility. This tradeoff is illustrated in Figure 7 for two different graphical models.

4. CONCLUSIONS

In this article we studied the application of learning techniques for the problem of autonomous cross-layer optimization. The designed intelligent agent relies on a modified version of simulated annealing. The learning features of SA are based on use of hypotheses, which employ approximate graphical models to give additional "direction" to the search or change the search landscape. We considered models based on trivial marginal probabilities, correlation coefficients, and Bayesian networks. We discussed their performance in a scenario portraying a terminal with multiple wireless interfaces, as well as in a wireless sensor networks scenario. We studied the applicability of the various hypotheses to different network dynamics, their convergence rates, as well as the portability of trained models to other networks.

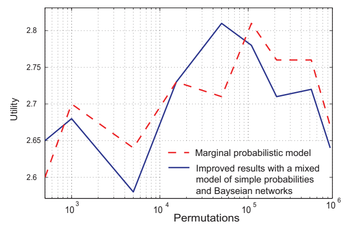


Figure 7: Scalability and effectiveness of the CE depending on total number of parameter value permutations.

The graphical model based on marginal probabilities provides the highest performance when minimal or no prior to execution training is available. A single drawback of this model is its inability to respond to non-linearity of the search space, for example to effectively find the best congestion control flavor of TCP, as no logical ordering of these flavors is possible. Sophisticated graphical models do not have this drawback, and if sufficient amount of training is conducted, they offer higher convergence rate and more stability. However, they still can not effectively explore unknown search regions and therefore rarely deliver better results that the trivial models. The mixture of the models is desirable.

Complex hypotheses, though attractive, should be applied with care, as in contrast to simple models they grow to be more specific and, therefore, less applicable in case of changes in network conditions. The challenge is to determine the state when a specific trained hypothesis can be applied, and when it should be switched for the other one or completely disregarded and the learning must begin anew.

One could argue that scenarios with wireless sensor networks or simulations of wireless networks might not be representative for the real world. However, our early work on cross-layer optimization in a wireless testbed shows the same tendencies as stated in this article.

5. REFERENCES

[1] QualNet. Scalable Network Technologies, http://www.scalable-networks.com [23.04.2009].

[2] J. Kephart and *et al*. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[3] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.

[4] S. Russel and P. Norvig. *Artificial Intelligence A Modern Approach*. Pearson Education, Upper Saddle river, New Jersey 07458, USA, 2 edition, 2003.

[5] S. Kirkpatrick and *et. al*. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[6] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *IEEE Communications Magazine*, 43(12):112–119, 2005.