Exploring Simulated Annealing and Graphical Models for Optimization in Cognitive Wireless Networks

Elena Meshkova, Janne Riihijärvi, Andreas Achtzehn, Petri Mähönen
Department of Wireless Networks, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email: {eme, jar, aac, pma}@mobnets.rwth-aachen.de

Abstract— **In this paper we discuss the design of optimization algorithms for cognitive wireless networks (CWNs). Maximizing the perceived network performance towards applications by selecting appropriate protocols and carrying out cross-layer optimization on the resulting stack is a key functionality of any CWN. We take a "black box" approach to the problem and study the use of simulated annealing for solving it. To improve the convergence rate of the basic algorithm we apply machine learning techniques to construct graphical models on the perceived relations between network stack parameters and application-specific network utilities. We test our optimizer design both in a simulation environment as well as on a network testbed with low-power radios. Our results show that even basic simulated annealing works well, but simple graphical models can further increase the convergence rate. However, use of sophisticated models such as Bayesian networks does not always lead to substantially better performance. The results indicate that enhanced simulated annealing indeed appears to be a promising foundation for future cognitive engine designs.**

I. Introduction

The complexity of protocol stacks used in wireless networks is constantly increasing. Some of the emerging wireless technologies requuire adjustments of low-level protocols. Traditionally more stable higher layer protocols are also both increasing in numbers and tend to offer more and more tunable parameters to control their behavior. Good example of the latter trend is the proliferation of different TCP variants over the past years, with highly different performance characteristics over wireless and fixed network types. Cognitive Wireless Networks (CWNs) [1], [2] have emerged as a promising paradigm for managing this increasing complexity and choosing for each application or device the most appropriate protocol and parameter settings. However, most of the work on CWNs has focused on optimization applied at a single protocol or stack level. In this paper we focus on the optimization of the composition and configuration of the entire protocol by developing appropriate metaheuristics for solving the resulting complex combinatorial optimization problem.

We extend our previously reported work in [3], [4] by both giving the experimental evaluations and the theoretical foundation. We also specify how the optimization task can be stated using utility functions and provide the corresponding practical example. We focus on the use of *simulated annealing* (SA), a powerful metaheuristic method originally developed in the physics community [5], as a basic method for protocol stack optimization. In particular, we study the enhancement of basic simulated annealing by introduction of *approximate probabilistic graphical models* encoding prior knowledge on the performance and impact of changes in the protocol stack. Nodes in a CWN may gather such knowledge themselves as part of the optimization process, or it could be made available as a service through the network. We show that application of graphical models does indeed improve the performance of simulated annealing, and quantify this improvement for different types of models and scenarios. Specifically, we show that inclusion of a relatively simple graphical model based on elementary statistics already provides significant improvement over basic simulated annealing, and that adoption of more sophisticated models, such as Bayesian networks, often does not lead to major improvements. Our results indicate that simulated annealing combined with simple graphical models presented here indeed provides a promising foundation for the future work in holistic protocol stack optimization in cognitive wireless networks.

The rest of the paper is structured as follows. In Section II we formulate protocol stack composition and configuration as an optimization problem and describe the overall architecture of a cognitive engine, designed for this problem space. The design of the optimization algorithm used by the engine, simulated annealing, combined with graphical models are discussed in Section III. We describe two diverse scenarios where our approach is evaluated in Section IV. We summarize basic results of the cognitive engine performance, as well as evaluation of SA with a number of graphical models in Section V. We conclude in Section VI.

II. Problem Formulation and Overall Design

We take a service-oriented and user-centric view of the network, where each node, cluster or a whole network can be modeled as a collection of services that interact with each other as well as with external entities to provide end services to the user. In our interpretation services are network protocols with tunable parameters that affect application-level tasks executed by the user. Their quality is expressed using an application-specific *utility function* [6] that can be further constraint by a set of policies. Utility functions are the most flexible way to

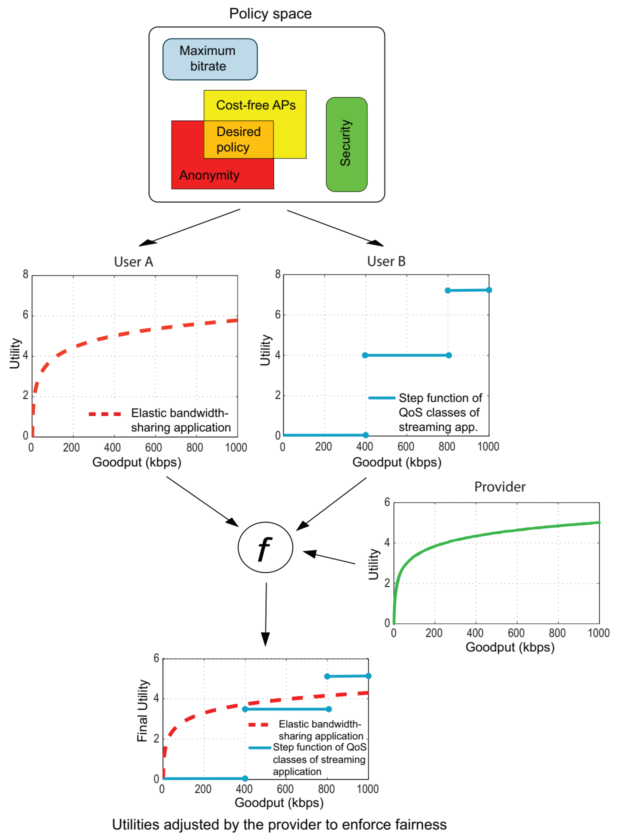


Fig. 1. Possible definition of the networks optimization goal though constraints specified by the policy space and utilities define by the end-users further modified by the intermediate parties (for e.g. providers).

define satisfaction, as they allow for a quantifiable expression of user needs [7]. They are known to be difficult to design [8], but embody most of the objective functions in networking used today such as QoS classes, for example in DiffServ [9], or logarithmic utility for TCP [10]. The common forms of utilities are linear, logarithmic and step-wise functions, with components that can be additive, weighted additive, or raised to power of a weight. However, other forms of objective functions can be used as well. We expect that in future utilities will be first defined by individual applications with relative weights that will be later adjusted by users. For example, a user might stress that the streaming application is N times more important than the downloading application. The utilities of separate nodes can be further modified by network providers to ensure, e.g., fairness (see Figure 1).

We treat an examined network as a "black box" that is influenced by tunable *parameters* (often referred to as "knobs" or "actuators") and performance of which can be judged according to the obtained utility as a function of several *attributes* ("dials" or "sensors"). Examples of parameters are the maximum number of retransmissions in MAC protocols, contention window sizes or the flavor of TCP used. Common examples of attributes are network delay, throughput, power consumption or reliability.

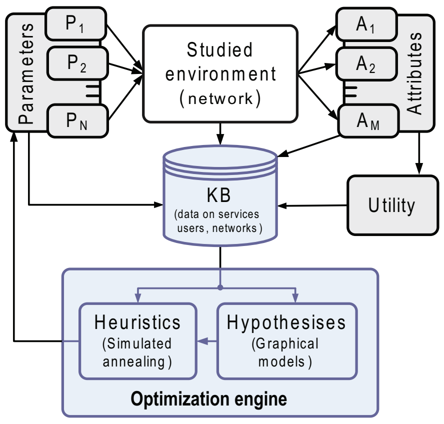

Parameters
P_1
P_2
P_N
Studied environment (network)
A_1
A_2
A_M
Attributes
KB (data on services users, networks)
Utility
Heuristics (Simulated annealing)
Hypothesises (Graphical models)
Optimization engine

Fig. 2. Architecture of the cognitive engine for network autoconfiguration.

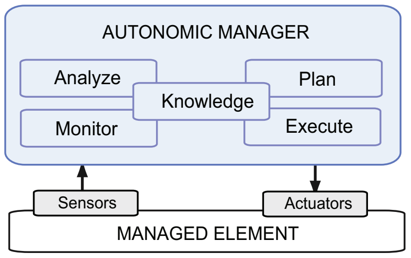


Fig. 3. Autonomic element. IBMs MAPE-K (Monitor, Analyse, Plan, Execute, Knowledge) reference model for autonomic control loops (adapted from [11]).

Such representations allows for a simple control loop in which parameters are tuned in accordance with attribute readings to improve the performance of the system. The parameter adjustments are conducted by an intelligent agent called a *cognitive engine*. The overall system model is depicted in Figure 2. It is based on the seminal introduction of the autonomic control loop by the research group of IBM [11] (see Figure 3) and it resembles the definition of intelligent agents commonly used in the AI community [12].

A. Assumptions

The studied system, in our case a network, a cluster or a node, can be characterized by several objectives its users want to achieve. For simplicity we take a centralized view on the problem, i.e., there exists a single utility function that all participants of the optimization contribute to and strive to maximize, and all relevant nodes use the same parameter values. Therefore, we concentrate only on "vertical" or "cross-layer" optimization. However, the described methods are general and can be easily extended to include "horizontal" or inter-node optimization as well. In this work we also do not consider the fact that network parameters are likely to be optimized at different time scales. For example, switching between different access points or networks is likely to take longer than adjustment of a single MAC parameter. Such considerations will lead to even more effective optimization than the results shown below and will be implemented as part of our future work.

B. Formal description

We can interpret the protocol stack optimization problem as *utility maximization task over all permutations of parameter values of network protocols*. The formal description of the problem is as follows. We denote by S the collection of all network protocol stacks. Each stack is subject to several constrains $c : S \rightarrow \{0,1\}$, such as protocols that are mutually exclusive, or protocols which are a subset of each other and do not add to the overall functionality, or user constrains. The set of stacks satisfying the constraints is simply defined by

$$S_c = \{x \in S \mid c(x) = 1\}. \qquad (1)$$

By $P(s)$ we further denote the list of tunable parameters, that each particular stack $s \in S$ provides. Parameters themselves may also be subject to further constrains, ${c'}_s : P(s) \rightarrow \{0,1\}$, which are either globally delimited (such as the maximum power a wireless interface can be run at) or are a result of a combination of several protocols in a stack. We can therefore derive a set of valid parameter value vectors that are applicable to the stack, which is given by

$$P_{c'_s} = \{x \in P(s) \mid {c'}_s(x) = 1, s \in S\}. \qquad (2)$$

The performance of a stack and its characteristics is described by a set of attributes A. This set is not necessarily defined only from a technical point of view. Economic considerations such as costs of deploying a certain technology, bandwidth-based fees and other application-specific attributes may also be taken into account. The sets of attribute values need to be weighted against each other. This is usually achieved by applying an utility function $U(A(s,p,E))$, that maps the attribute set to a single numerical value, taking into consideration the deployed stack ($s \in S, p \in P$) and the operational environment E. Higher values of utility determine a preference of the users of one set of parameter values over the other.

Finally, with the help of utility functions, the optimization problem can be described as

$$(s,p) = \underset{(s,p) \in (S_c \times P_{{c'}_{S_c}})}{\arg\max} U(A(s,p,E)). \qquad (3)$$

The maximization task is to find the stack with a valid combination of parameters that yields the highest overall utility value. It is obvious that an exhaustive search is only feasible up to a very limited amount of parameters and their values. Therefore, we have been evaluating search methods that are capable of coping with this highly complex problem.

C. Simulated Annealing

The adoption of a generic network model allows the use of the cognitive engine for a wide range of tasks. However, lack of an exact analytical model of attribute responses to changes in parameters keeps us from using the classical optimization techniques such as convex optimization [13][1]. Therefore, the logical choice are *metaheuristic* search techniques [14].

Simulated annealing (SA) is a popular metaheuristic algorithm that is known to be effective for "black box"-type of problems [5]. We have chosen this algorithm as the basic search method for our implementation as it potentially allows the use of a smaller number of parameter sets, compared to, for example, genetic algorithms [15] that employ parallel searches to fill in the candidate population. Simulated annealing has also been shown to have good asymptotic convergence properties. The basic algorithm is very straightforward (see Algorithm 1). Starting from a random position in the search space the next point is chosen at an arbitrary location within distance of a jump proportional to a *temperature* parameter. Initially, the temperature is set to maximum. In our implementation, the probability of accepting the new location as a new starting point is proportional to the improvement in utility. Each iteration of the search also results in a temperature decrease. The algorithm stops when the temperature drops below a certain threshold. The process of "heating" of the search space and its controlled cooling can be performed iteratively and is called "reheating".

We define the search space for simulated annealing as a multi-dimensional landscape where each dimension corresponds to one adjustable parameter. Each parameter might take a number of predefined states or values. In this paper we regard our optimization problem as utility maximization task in this search space. We also discuss several adjustment to the classical simulated annealing that were introduced to enhance the system's performance.

```
begin
    $s \longleftarrow s_0$; $u \longleftarrow U(s)$ // Initialize: state, utility.
    $s_b \longleftarrow s$; $u_b \longleftarrow t$ // Initialize "best" solution.
    $t \longleftarrow t_0$ // Set initial temperature.
    while ($t > t_{min}$ and $u < u_{good}$) // While temperature is above threshold and utility is not good enough: do
        $s_n \longleftarrow neighbor(s)$ // Pick some neighbor.
        $u_n \longleftarrow U(s_n)$ // Evaluate its utility.
        if $u_n > u_b$ // Is the new set better? then
            $s_b \longleftarrow s_n; u_b \longleftarrow u_n$
        end
        if $P(u, u_n, temp(t/t_{max})) > random()$ // Should we move to it? then
            $s \longleftarrow s_n; u \longleftarrow u_n$ // Yes, change state.
        end
        $t \longleftarrow (t-1)$ // One more iteration done.
        $return\, s_b$ // Return the best solution found.
    end
end
```

Algorithm 1: General overview of simulated annealing.

[1]As far as we act on assumption of unexplored network environment, it is difficult to use analytical models for the optimization as the variables of there models have to be estimated first. Besides the utility response to changes in some parameters like TCP congestion control algorithm, access point (AP) association or individual node/AP frequency assignment are difficult to model.

D. *Properties of Network Protocol Optimization*

The task of network optimization from a protocol stack configuration perspective has several properties that distinguish it from classical optimization. First, there usually exists a *definite starting point of search* – the *default settings* of the parameters of the protocols that form the stack. This point in the search space usually provides utility above average. However, in absence of bottlenecks it is usually very close to the global optimum. The default settings result in very low utility only in cases of abnormal network conditions, such as drastic increases in network load. Therefore, the performance of the system is often significantly decreased during the search phase without any guarantee that it will ultimately lead to performance improvements.

The second peculiarity of the network optimization task is a distinction of *two search phases*: *static and runtime*. The static configuration stage can be regarded as a long-term learning, i.e., the engine can observe the network for a long time without the need to keep high average utility during the training period. This stage can take place at times of low network load, e.g., during night times or at the stages of pre-deployment network testing. It allows for the best network optimization, however this period is not available to all systems. Some traffic patterns that a real-working network can display might be missed as well.

The runtime optimization has more constraints. First, it must have shorter learning periods not to disturb the user. Second, the average utility values must be kept high for user satisfaction. For the same reason the system must react fast to changing network conditions (utility drops), which again results in short learning phases. If a network allows for utility increase then appropriate parameter value sets should be detected as well. Ideally the optimization process incorporates both of these search phases: first conducting a static training as a preliminary learning, for example limiting the number of possible parameter set permutations, and then continuously executing runtime optimization as a background process on the working system.

An additional property of the optimization task was encountered during preliminary trials. There exists a strong dependency of parameter-utility correlation depending on the current network environment. The correlation of parameters and application-specific utilities tend to stay the same for a wide range of network scenarios (including different topologies and application traffic patterns). The exception is a drastic change in the network's performance, when contribution of parameters via attributes to the final utility changes. As an example there might be a sudden increase in the node failure rate when the utility focus shifts from low-power message delivery to just message delivery.

III. Modification to the Optimization Algorithm

In this section we describe several modifications to classical simulated annealing that allow for improved network stack optimization. We distinguish between basic enhancements of SA that improve performance of the system and "intelligent" ones that introduce learning features in to the algorithm.

A. *Common modifications*

During our test runs we have been observing that the faster a stable near-optimum solution is achieved, the more likely the SA tries for other, significantly worse parameter combinations. This is due to the impact of the temperature algorithm that chooses highly varying parameter sets at early stages. Therefore, we introduced utility-awareness into the temperature algorithms, and thereby limited variation at near-optimum solution points. We added a maximum search time as an additional limitation to the search. If the algorithm is unable to find better results within a certain number of steps, it stops the search and returns to the parameter values that yielded highest utility. So far, this scheme has been used in other implementations of SA as well, but we added a component that makes the maximum search time inverse proportional to the so-far encountered highest utility.

In order to overcome short-time variations in utility (basically background noise due to the changing utilization of the network), we extended the SA algorithm by a system of comparing the values of two sliding windows, with window sizes adjustable to the variance in the utility.

Since our cognitive engine is to be applied at runtime, we have been further trying to limit the negative impact of the exploratory search to the users. As the algorithm (especially during early stages) tries for parameter combinations that yield comparably low utility, we added several conditional barriers to the algorithms. This way we were able to cope with high fluctuations in utility. For example, once the SA algorithm has converged to a good stable solution, the search is stopped. The utility is continuously monitored over time. If the system notices a sudden drop in utility that cannot be traced back to short-time variances but indicates a change in network usage, SA is restarted with an agility, that is the likeliness of parameter variation, which is proportional to the utility drop. A static memory of the best parameter values discovered over time is used to boost up the algorithms performance by letting it first try those combinations. Additional temperature thresholds can be defined for the regular reheating phase, which aims for a limited additional exploration of the search space and should not be very disruptive for the user.

We are also able to make a trade-off between the duration of the search period and the average utility achieved during this time. The search iterations may be conducted not every cycle that the cognitive engine accesses the optimized system, but only every N^{th} iteration, for all other returning to the best utility state. This allows to limit the negative experience for the user at expense of a prolonged convergence time.

Another modification to SA concerns the cooling scheduler. There exist several methods of decreasing temperatures, e.g., Boltzmann, exponential and linear. We evaluated these functions in both scenarios and determined that in a majority of cases the linear temperature decrease leads to the best performance (see Table I). For convenience and computational

TABLE I

PERFORMANCE OF DIFFERENT COOLING SCHEDULERS IN SA.

Dynamic case: VoIP over wireless networks		
Temperature	Mean Utility	Std.
Boltzmann	2.78	0.15
Exponential	2.71	0.15
Step-wise	2.78	0.15
Linear	2.79	0.15
Static case: wireless sensor networks		
Temperature	Mean Utility	Std.
Boltzmann	72.0	4.0
Exponential	67.0	3.6
Step-wise	64.0	3.7
Linear	55.0	4.0

performance reasons we additionally realized a step-wise linear decrease function where temperature is decreased by N degrees at every iteration. Its performance is very close to the classical linear behavior. The variation of the step size directly affects the maximum number of iterations during one search cycle.

B. *Approximate Probabilistic Graphical Models*

Simulated annealing is a metaheuristic search method that does not usually include learning. It can store the states with best utilities, but cannot deduce from them the most promising directions of search using long-term knowledge. The primary reason is a limited applicability of such knowledge, as we are dealing with a changing network environment, which can change both at runtime as well as from scenario to scenario. Therefore, only a limited amount of long-term learning experience is applicable. Additionally, as we aim to achieve *the maximum utility of the network in a minimal number of iterations* the accumulated information need to be used more aggressively.

We have chosen *approximate probabilistic graphical models* as a representation of dependencies learned by the optimizer [16]. This information is later fed to the simulated annealing algorithm to give its search a "direction". Basically, we assign probabilities to parameters to be chosen for alteration, as well as probabilities of parameters to take certain values[2]. The probabilities are altered during the work of the optimizer depending on the explored search space. We have examined a number of graphical models ranging from simple ones that consider only basic probabilities or correlation coefficients to more sophisticated like Bayesian networks [17]. The discussion on performance of graphical models is continued in Section V.

IV. SCENARIOS AND IMPLEMENTATION

We have evaluated the cognitive engine for two scenarios with significantly different characteristics. The first scenario deals with runtime optimization, where a user initiates VoIP calls that can be served by a number of wireless networks: WLAN (both IEEE 802.11a, 802.11g) and WiMAX. Here we consider one node optimization. The number of permutations for each access technology is approximately 2000 (see Table II). The WLAN access points are free of charge, but their performance varies over time as they might become saturated. The user has to pay for WiMAX usage, but the capacity of this network satisfies all her requirements. The utility here is a function of the Mean Opinion Score (MOS) and wireless access cost (see Figure 4) with maximum achievable value of 3.3. Mean opinion score specification is taken directly from the Qualnet simulator, as a function of delay, packet loss and datarate. For simplicity we assume seamless vertical handover [18]. Due to the changing network environment the runtime optimization can evaluated in this scenario.

As we want to study the applicability of the optimization engine to a wide range of cognitive wireless networks, we have constructed a second scenario featuring low-power radios as their performance characteristics are highly different from broadband wireless networks of the first scenario. For prototyping we have chosen wireless sensor networks (WSNs). The performance of such networks is known to be extremely sensitive to the deployment environment, and therefore, it is feasible to conduct extensive static, pre-deployment tests of these networks. This is due to the fact that signals of low-power radios are easily disrupted. Once the network has been deployed at the location, the task of automatic node configuration with the purpose of its fine-tuning to the surrounding environment is therefore highly relevant. Additionally, there

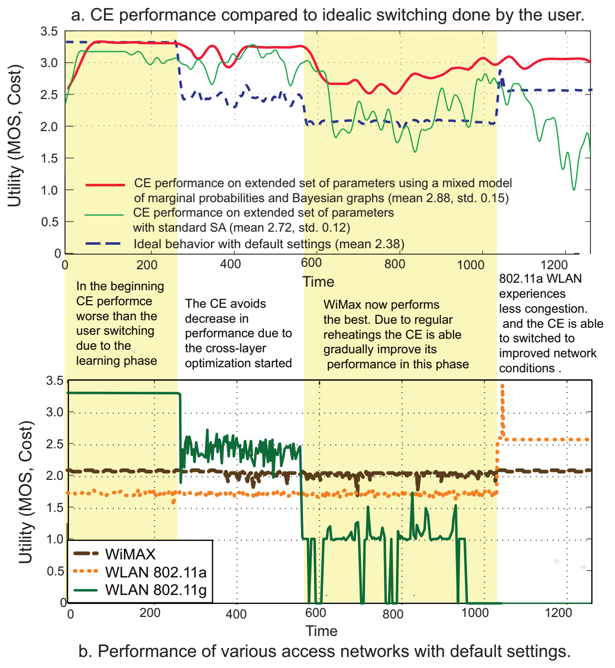


Fig. 4. Performance of the cognitive engine with basic simulated annealing in the VoIP scenario.

[2]We use the term "approximate graphical models" not just "graphical models" to stress the fact that we consider not only Bayesian networks, but other probabilistic models as well.

TABLE II
THE OPTIMIZED PARAMETERS AND THEIR VALUES.
(BOLDFACE INDICATES THE DEFAULT SETTINGS.)

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. retransmit.	1, **3**, 9
	Duty cycle	10, 50, 80, **100**
	Acknowledgments	"on", **"off"**
	Backoff length	5, **30**, 40

exist no widely accepted well-performing protocol stacks for these networks, and multiple protocols might be considered for deployment. For this case a testing tool for evaluation of combinations of these protocols, which scales well needs to be developed.

We have evaluated the performance of the cognitive engine in a WSN testbed over a wide range of scenarios using different utility functions and network topologies. Network sizes varied from four to twelve nodes with linear, mesh and random topologies. Utilities were functions of power, reliability, goodput and delay. As a representative example we use averaged results of these tests with the utility as a function of power consumption with a constraint on the reliability (maximum number of packets lost). The cumulative distribution function (CDF) of this utility is shown in Figure 5a. As can be seen, high utility values were only obtained in 3-5% of the cases. We limited total number of parameter value permutations to 421, since we wanted to evalute performance of the algorithm on scenarios where the exhaustive search is possible (see Table II). Our experiments show that the sigmoid shape of the CDFs as depicted in Figure 5 can be considered representative for many network stack optimization problems.

The cognitive engine is realized in MATLAB with interfaces to a mySQL database that stores the performance of the WSN testbed and Qualnet simulator [19], where the VoIP scenario is run. The engine queries the systems every N milliseconds and gets information on node performance. Using the same interface, parameters of the nodes are adjusted as a result of the decision process. In the static scenario each trial was conducted at least 100 times and there were at least 20 repetitions for each runtime simulation.

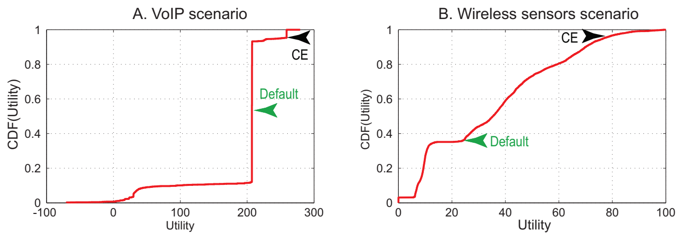


Fig. 5. Cumulative distribution functions of utilities.

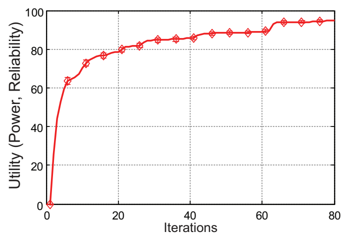


Fig. 6. Performance of the cognitive engine with basic simulated annealing in the WSN scenario. (The graph is obtained by conducting multiple experiments at random starting parameter configurations and then averaging over these results is done.)

V. BASIC RESULTS AND FURTHER EXPERIMENTS WITH GRAPHICAL MODELS

In Figure 4 the performance of our cognitive engine with the implementation of simulated annealing minimally adapted to suit the framework is presented[3]. The runtime optimization conducted at VoIP scenario shows good performance. The results obtained with the cognitive engine that tunes parameters of the whole protocol stack are better than the ideal ones obtained when only access technology is switched[4]. We achieve on average utilities of 2.72 compared to 2.37 with default settings, which is almost a 15% improvement.

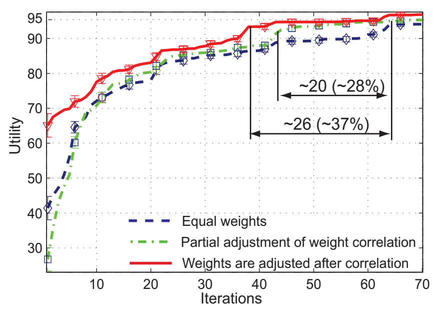
The static optimization of the WSN scenario provides even better results (see Figure 6). It converges to 95% from maximum after approximately 65-70 iterations. We reach 80% of maximum within 20 iterations. Further, we provide evaluation of a hybrid version of simulated annealing that is modified with several probabilistic graphical models.

A. Simple Probabilistic Search

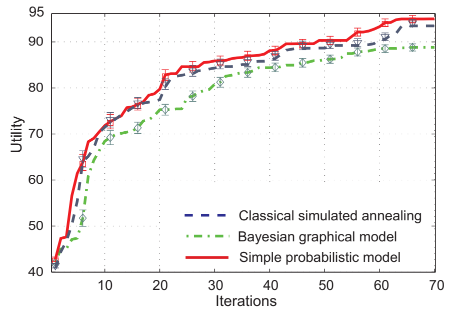
The first graphical model we consider incorporates a *simple probabilistic approach*. Here, we sort parameter values in ascending order and record the probabilities if an increase or decrease of value of a certain parameter has resulted in an increase or decrease of the utility in the past. The most influential parameters have higher chances to be altered in the next search iteration. This metric is applicable only in cases where parameter values can be put into a sorted order

[3]The plotted graphs are a smoothed and averaged version of the results received, post-processed for visualization clarity. We have also plotted the actual measurements obtained as points at the background.

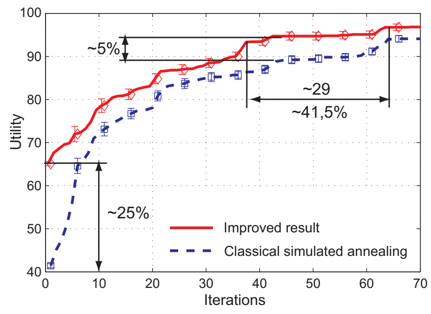
[4]We found through exhaustive search the ideal behavior access method selection, which is not entirely realistic due to fast switching.



a. Adjustment of weights using correlation between parameters and utility

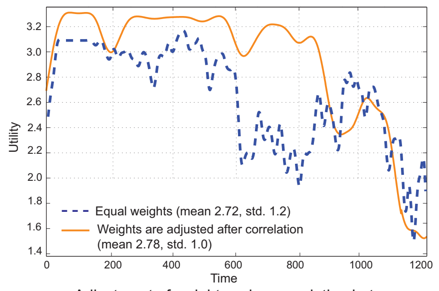


b. Typical performance of SA with different graphical models

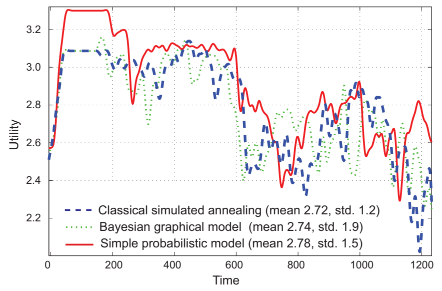


c. Average performance increase reached using correlation and simple graphical models

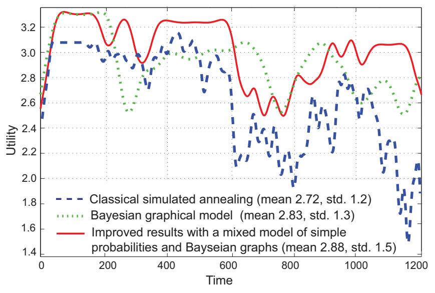
Fig. 7. Static optimization of wireless sensor network scenario.



a. Adjustment of weights using correlation between parameters and utility



b. Typical performance of SA with different graphical models



c. Average performance increase reached using correlation and simple graphical models

Fig. 8. Dynamic optimization of VoIP scenario in wireless networks.

that resembles the stack's behavior upon change. Examples of those are congestion window sizes, counter-examples are TCP variants. This simple metric is computationally effective and usually performs better than the basic SA (see Figures 7b and 8b). The average convergence rate improvement in the static scenarios is around 7% with a slight increase in the overall utility value achieved. For the runtime scenarios we achieved an overall gain of only 3%, however the local gains are higher and reach up to 18%.

This is the most stable learning model for our optimization problem, as it shows almost constant improvement over the standard SA, while performance of other graphical models depends heavily on the search landscape. This is due to the fact that this approach allows for considerable randomness in the search and tends not to get stuck at a local maxima with a tradeoff of less thorough, compared to other methods, exploration of the local neighborhood.

B. *Correlation-based Search*

The second graphical model considered is based on a correlation coefficient between each parameter and the utility. Parameters to change are chosen as inverse of the correlation $(1 - |\mathrm{corr}(P, U)|)$, as we want to examine the most influential parameters first and later converge to a local search in the space of less influential parameters. By changing the initial temperature, as well as steepness of the temperature curve, we can explore both far and local search states. This rather trivial graphical model works best for the long-term training of the system. Basically it allows to explore parameter-utility dependencies that do not correlate strongly on environment conditions as discussed earlier in Section II. We calculated correlation coefficients on the data of several optimization runs and incorporated inverse of these values as weights which restrict the maximum probability for a parameter to be chosen in the search space. Therefore, we heavily alter the parameter search landscape. Though correlation coefficients obtained are not gathered over the whole parameter space, i.e., no exhaustive search is done, we still obtain notably better results than in case when all the parameters are treated equally (see Figures 7a and 8a). In the wireless sensor network scenario we achieve a gain of 37%. In the VoIP scenario the improvement is quite small from 2.72 to 2.78. However, it allows for bigger performance gains in combination with more sophisticated graphical models, due to the restrictions on the search space.

Long-term learning with correlation coefficients is also computational effective, as it allows for the re-use of previously gathered data. Provided that the data set has records on all necessary attributes, it is easy to calculate different correlation weights depending on the specific form of the utility function.

C. *Sophisticated Graphical Models*

We have also considered a number of other graphical models based either on Bayesian networks [17], [20], correlation or joint probabilities for the on-line learning of the cognitive engine. Surprisingly, these models or their combinations do not perform notably better than the simple probabilistic search. As the representative example we have chosen the trivial Bayesian graphical model with connection from parameters to attributes and no hidden nodes, which performance is shown in Figures 7 and 8. We discovered that the complex models start to work well only when either a sufficiently large set of samples is gathered or the search neighborhood is limited. The situation improves by mixing these model with the simple probabilistic one (see Figure 8c). In this case a considerable improvement can be achieved. In our example the average gain compared to the basic SA was around 6% with local improvement over 20%. Unfortunately it is difficult to predict when the Bayesian-enabled search is efficient and, therefore, we quite often end up with a performance lower than that for the simple probabilistic search. The computational costs for such complex graphical models are also quite high and these methods can be used only restrictively in resource-constrained environments. In future work we want to study this problem further to hopefully find effective combinations of simple and complex graphical models to balance local-neighborhood and exploratory searches.

VI. Conclusions

In this paper we have considered the network stack optimization problem and discussed its properties. Our cognitive engine has successfully applied on two diverse scenarios to perform static and runtime optimization tasks. The main focus of the paper is on possible modifications to the basic simulated annealing that might lead to improved performance of the network in terms of convergence rate and achieved utility. We tested a number of temperature algorithms, as well as modified learning schemes using several probabilistic graphical models. We estimated that the linear temperature algorithm with adjustable step size performs best in both static and runtime scenarios, allowing for additional flexibility in the reheating phase. The long term correlation between parameters and a particular utility function boosted the convergence rate of the system up to 35% for the static optimization, increasing the utmost utility value reached for the run-time scenario as well. The simple probabilistic graphical model almost always increases the convergence rate for another 5%-10%. Advanced probabilistic graphical models, such as Bayesian networks, proved to be effective only for the search on a limited "local" space, after an extensive training period. The mixture of the learning approaches showed more stable, but still not universally applicable results.

In our current work we focus on studying the cognitive engine performance on a wider range of utilities and parameter permutations (up to 10^{15}). To reduce the number of parameters specific to the optimization algorithm a user needs to tune, we are working to make the algorithm more dynamic and self-adjustable. We also plan to create a distributed version of the system by adopting, for example, some hybrid genetic algorithm.

Acknowledgment

The authors would like to thank the RWTH Aachen University and the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) for providing financial support through the UMIC research center. We would also like to thank the European Union for providing partial funding of this work through the ARAGORN project.

References

[1] P. Mähönen, J. Riihijärvi, M. Petrova, and Z. Shelby, "Hop-by-hop toward future mobile broadband IP," *IEEE Communications Magazine*, vol. 42, no. 3, pp. 138–146, 2004.

[2] G. Minden, J. Evans, L. Searl, D. DePardo, V. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. Wyglinski, and A. Agah, "Kuar: A flexible software-defined radio development platform," *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, pp. 428–439, April 2007.

[3] E. Meshkova, A. Achtezehn, R. J., and P. Mähönen, "Towards User-centric Network Optimization Engine," in *Proc. of IEEE SECON*, Rome, Italy, June 2009.

[4] E. Meshkova, R. J., and P. Mähönen, "Towards User-centric Network Optimization Engine," in *Proc. of ACM Workshop Learning for Network (in conjunction with SIGMETRICS/PERFORMANCE)*, Seattle, USA, June 2009.

[5] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[6] S. Shenker, "Fundamental design issues for the future Internet," *IEEE JSAC*, vol. 13, no. 7, pp. 1176–1188, 1995.

[7] D. Raymer, S. Meer, and J. Strassner, "From Autonomic Computing to Autonomic Networking: an Architectural Perspective," in *Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems, 2008. EASE 2008*, 2008, pp. 174–183.

[8] D. Goodman and R. Nash, "Subjective quality of the same speech transmission conditions in seven different countries," *Communications, IEEE Transactions on [legacy, pre-1988]*, vol. 30, no. 4, pp. 642–654, 1982.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," pp. 1–32, December 1998, iEFT RFC 2475.

[10] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 5, pp. 556–567, 2000.

[11] J. Kephart, D. Chess, I. Center, and N. Hawthorne, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[12] S. Russell, P. Norvig, J. Canny, J. Malik, and D. Edwards, *Artificial intelligence: a modern approach*. Prentice Hall Englewood Cliffs, NJ, 1995.

[13] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[14] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[15] M. Mitchell, *An introduction to genetic algorithms*. Bradford Books, 1996.

[16] C. Borgelt and R. Kruse, *Graphical Models: Methods for Data Analysis and Mining*. John Wiley and Sons, 2002.

[17] I. Ben-Gal, *Bayesian Networks, Encyclopedia of Statistics in Quality and Reliability*. John Wiley and Sons, 2007.

[18] M. Ylianttila, J. Makela, and P. Mahonen, "Supporting resource allocation with vertical handoffs in multiple radio network environment," in *Proceedings of The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, vol. 1, Lisbon, Portugal, September 2002, pp. 64–68.

[19] "QualNet," scalable Network Technologies, http://www.scalable-networks.com [last visited on 23.11.2008].

[20] K. Murphy *et al.*, "The Bayes Net Toolbox for Matlab," *Computing Science and Statistics*, vol. 33, no. 2, pp. 1024–1034, 2001.