WAITS 2012

Workshop on Artificial Intelligence in Telecommunications and Sensor Networks

ECAI 2012

Montpellier, France

August 28th, 2012

WAITS 2012

Workshop on AI in Telecommunications and Sensor Networks

Traffic on our telecommunications networks has grown exponentially in the last decade, and is continuing to grow. Many of the services and applications that exist now were not even imagined a few years ago. The development of machine-to-machine traffic is expected to place serious demands on network infrastructure, as more and more sensors, actuators and data sources come online. Although we can extrapolate some trends and make educated guesses, we cannot predict exactly what lies ahead. Networks that are designed with change in mind will be more robust to disruption caused by growing demands and changing user patterns and yet-unimagined applications. The risks associated with investment in these kinds of networks will be lower as they will be more durable and scalable. Networks that are designed with change in mind will make effective use of resources (e.g. spectrum, bandwidth, power, processing capabilities, cooling capabilities etc.) and ensure a sustainable future.

The objective of this workshop is to bring together researchers and technologists from academia and industry to explore the applications of artificial intelligence to the most pertinent technical challenges in telecommunications and sensor networks. All papers were reviewed by at least two international reviewers.

This workshop is being run in association with CTVR, the telecommunications research centre, funded by Science Foundation Ireland (Grant Number 10/CE/I1853). We are grateful for their support.

Ken Brown, Barry O'Sullivan, Cormac Sreenan

Workshop Chairs

August 2012

Organization

Workshop Chairs

Ken Brown[1], Barry O'Sullivan[1], Cormac Sreenan[2]

CTVR

and

(1) Cork Constraint Computation Centre (2) Mobile and Internet Systems Laboratory

Department of Computer Science

University College Cork

Ireland

Program Committee

Gianluca Bontempi	Universite Libre de Bruxelles, Belgium
Luiz DaSilva	Trinity College Dublin, Ireland
Pierre Flener	Uppsala University, Sweden
Anna Förster	University of Applied Sciences and Arts of Southern Switzerland
Justin Pearson	Uppsala University, Sweden
Dirk Pesch	Cork Institute of Technology, Ireland
Helmut Simonis	University College Cork, Ireland

Table of Contents

A Complete Data Mining process to Manage the QoS of ADSL Services

Françoise Fessant and **Vincent Lemaire**[1]

Abstract. In this paper we explore the interest of computational intelligence tools in the management of the Quality of Service (QoS) for ADSL lines. The paper presents the platform and the mechanisms used for the monitoring of the quality of service of the Orange ADSL network in France. The context is the availability of the VoIP services. In particular, this platform allows the detection and the classification of the unstable lines of the network. The interpretation of the classification results allows the discovery of some new knowledge used to improve the ADSL lines labeling and to prevent inefficient supervision of the network.

1 INTRODUCTION

Internet is now the common platform for voice and video services. The services are delivered through ADSL lines. At the end of the line, at customer's home, the connection point between the customer and the internet is in most cases a box. The box is also the mean by which an internet access provider can deliver its services (telephony, internet, television, video over demand).

The quality of the multimedia services given on internet can be affected by various factors dependant of what occurs in the network like congested links, latency, data loss, or dependant of the good working of the platforms delivering the services.

Quality of Service (QoS) is crucial to guaranty that the services will be delivered with good quality to the customers. QoS refers to a broad collection of networking technologies and techniques [12]. The monitoring of the QoS is a way to detect for instance when a process or an element of network are working outside of their working area or are not working properly. The data collection and the observation of some end to end QoS measures is a way to determine the origin of the trouble: element of network, physical line or box. The precise knowledge of the source of the trouble is a key point to an appropriate and rapid reaction of the supervision service to assure a good quality of service.

Orange, the French telecommunication company has implemented a complex chain dedicated to the monitoring of the QoS for its ADSL services in order to increase the satisfaction of its customers. This chain is based on the collection of a large number of end to end measures and on the creation of indicators.

This paper focuses on one specific part of this chain and on the detection of one type of problem that is the identification of unstable ADSL lines. The context of the QoS is restricted to the availability of the telephony on IP (VoIP) service from the Orange box (the live box). We show how the detection of unstable ADSL lines helps to improve the QoS and allows the extraction of additional knowledge to reinforce the global supervision chain.

[1] Orange labs, Lannion, France, email: francoise.fessant@orange.com

We will successively present the different parts of the chain from the point of view of a data mining process (from objective and data understanding to modeling and results exploitation). Two steps of the process will be more specifically described: the modeling step and the result interpretation step. The detection of the unstable ADSL lines is performed with a dedicated classifier based on naïve Bayes. Then a deep analysis of the classification results has been performed to better understand which explanatory variables explain the classification results and what are the actions that can be applied to improve the ADSL lines stability but also the labeling process.

2 MANAGEMENT OF THE QoS WITH A COMPLETE DATAMINING PROCESS

Data mining can be defined "the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [7]. Several industrial partners have proposed to formalize this process using a methodological guide named CRISP-DM, for CRoss Industry Standard Process for Data Mining [5]. The CRISP-DM model provides an overview of the life cycle of a data mining project, which consists in the following phases: business understanding, data understanding, data preparation, modeling, evaluation and deployment. The CRISP-DM model is mainly a process guide for data mining project. The presentation of the QoS processing chain is based on this structure. In our discussion we are specifically interested in the modeling and evaluation phases. The whole process in the QoS context is given on Figure 1.

2.1 Business understanding

This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition, and a preliminary plan designed to achieve the objectives. In our context the objective is clearly: how to increase the customer satisfaction? In this study, the detection of the unavailability of the services delivered by the live box (VoIP or internet) is the mean chosen to contribute to this objective.

The Orange's service in charge of the QoS management has proposed two kinds of answers. The first one consists in the creation and publication of weekly dashboards to follow some QoS metrics and to be able to react quickly when a degradation is detected. The second one is the automatic classification of ADSL lines (in two main classes: stable or unstable).

2.2 Data understanding

The data understanding phase starts with an initial data collection and proceeds to an explanatory analysis to get familiar with the data and to identify data quality problems.

The availability of the IP telephony service is watching by a functionality included in the live box. The functionality works when the live box is on. It is able to detect the unavailability of the service but also to raise the reason of this unavailability. It can also perform some measures of vocal quality during calls. The main advantage of this system is that it is located as close as possible of the customer and gives an end to end vision of the service as it is perceived by the customer. This paper is mainly concerned by the VoIP service availability.

Each time an unavailability event is detected the mentioned functionality produces a ticket which is send to a platform collecting and centralizing the whole tickets. To be more precise, the availability ticket is produced twice: the first time when the service is lost and the second time when the service comes back. This last ticket gives the nature of the event that has caused the lost of the service. It is also possible to assess to the duration of unavailability. The content of a ticket concerns only anonymous information about the state of the live box according to the services it has to deliver. In no way private information about the network activity of the customer is seen.

2.3 Data preparation

This phase covers all activities to construct the dataset that will be fed into the modeling tool, from the initial raw or relational data [10, 5].

In the QoS processing chain, the tickets are centralized in an analysis and treatment platform where they are enriched with additional information like the network characteristics of the line and several key point indicators specifying the types of problems and the length of unavailability. The tickets are then fed into a database thanks to a specific application that takes the tickets log and transforms it into a flat table (a table composed of K instances and J variables). A view of the live boxes events for the last 35 days is kept in this database by the mean of all the produced tickets.

2.4 Label of ADSL lines

Orange uses a specific network application to qualify the quality of its ADSL lines. This application is based on the interrogation of the DSLAMs (for Digital Subscriber Line Access Multiplexer), which return the count of the number of resynchronizations for each line and the length of these resynchronizations. This number is strongly correlated to the stability of the line which is, it, directly linked to the availability notion. An inspection period of at least 6 hours is needed before the qualification of the line. The tool is able to label each ADSL line to one of the third following categories: stable, risky or instable.

2.5 Modeling

In this phase, various modeling techniques can be selected and applied, and their parameters have to be calibrated to optimal values according to a success criterion. In the scope of this paper, the detection of the ADSL lines stability is the task to perform. This classification problem will be defined and discussed in-depth in section 3.1.

2.6 Evaluation and result interpretation

Before proceeding to the final deployment of a model, it is important to thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. At the end of this phase, a decision on the use of the data mining results should be reached. The evaluation method and the generalization performance of the classification model are discussed section 3.2.

We propose an interpretation phase in section 4 to better explain the classification results (for example, which explanatory variables explain that an ADSL line has been classified as unstable and how this line can be made less unstable or even become stable). This allows us to extract additional knowledge to reinforce the QoS chain.

2.7 Deployment

The deployment phase can be as simple as generating a report or as complex as implementing a repeatable data mining process. The model will be applied to a larger database (the target population) than the training database. In this phase it is important for the user to understand up front what actions will need to be carried out in order to actually make use of the created models. For the QoS application the data collection and preparation phases as well as the modeling phase are currently industrialized. The industrialization of the use of prediction information is still in progress (fine analysis of results, labeling improvement of line state, automatic labeling of tickets for filtering).

3 MODELING: CLASSIFICATION OF ADSL LINES

In this section the modeling step of the data mining process is presented. The model is a classifier designed for the classification of ADSL lines stability. The following notations are introduced: we call T the data table with K instances and J explanatory variables. C is the number of classes. An instance x_k is represented with a vector of J dimensions, f is the probabilistic classifier learned from a modeling table.

3.1 Data and experimental protocol

Five days of tickets have been extracted from the data platform storage for the analysis. These data make the learning database used for the construction of the classification model. The unit we want to analyze is the ADSL line.

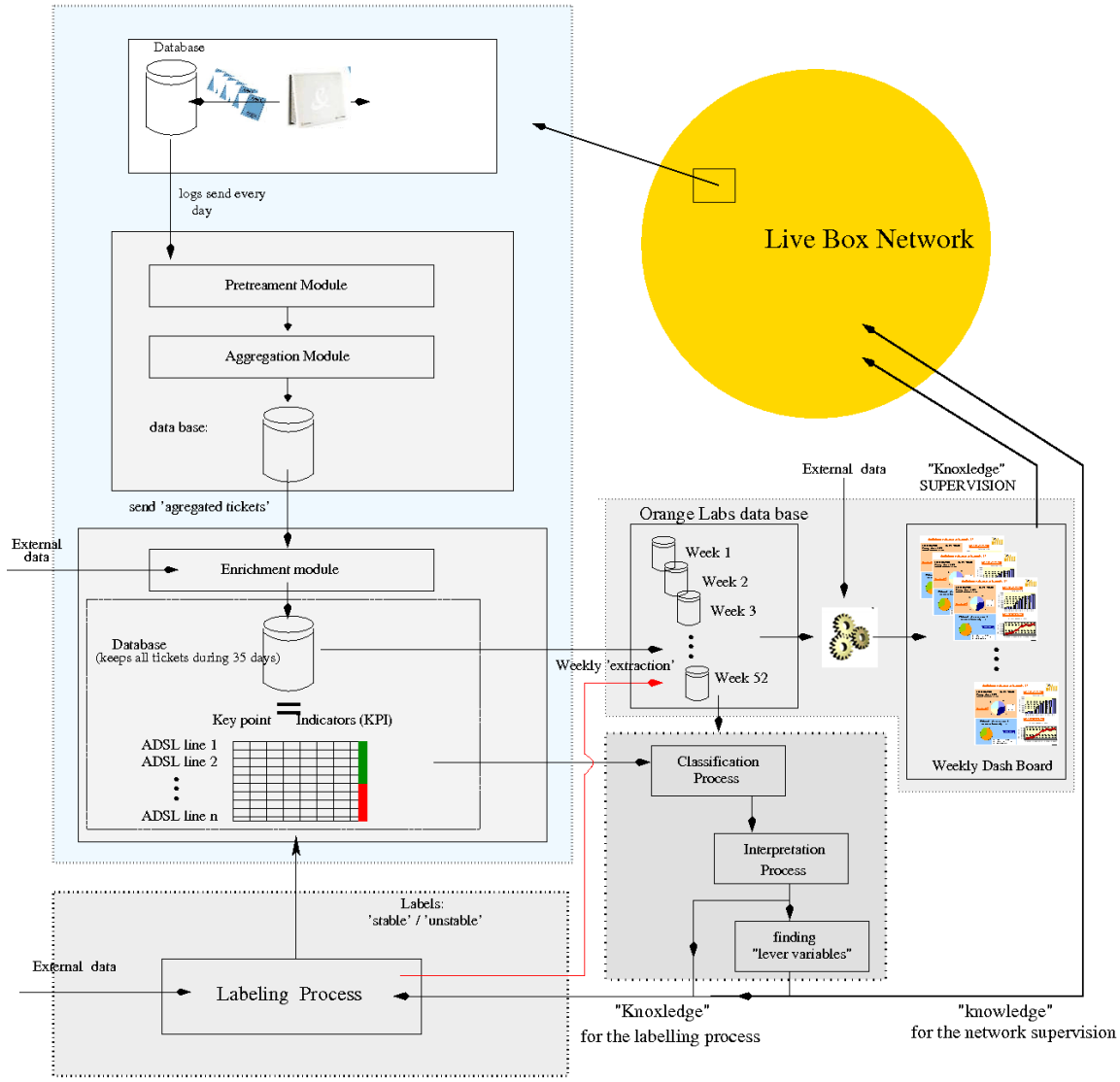


Figure 1. The data mining process: from tickets to QoS indicators

A line is characterized by a set of indicators gathered from the tickets. The indicators are of different sorts. Some of them identify the live box (MAC address of the box, serial number, model and firmware for example). Others concern the line and the network elements the line is connected up (DSLAM model and name, BAS name, etc.). The last group is for the events that have led to the ticket production (total number of tickets, number of tickets of service lost and service return during the period, total duration of events, duration of events of each type, etc.). Five types of unavailability events are detected by the live box: reboots, desynchronizations of the ADSL line, lost of session PPP (Point to Point Protocol), ToIP connection problem (Telephony over IP), ToIP configuration lost at the live box level.

Finally a line is represented by a vector of 39 explanatory variables that are categorical or continuous. We add to these descriptors the information about the state of the line coming from the network application as described above. The modeling database holds 71164 instances. The priors on the classes stable, risky and unstable are respectively of 0.881, 0.054 and 0.064.

Experiments have been made with the Khiops[2] software (developed by Orange Labs). Khiops is a data mining tool allowing to automatically building successful classification models. Khiops offers an optimal preprocessing of the input data, efficient selection of the variables and averaging the models. It is a parameter-free classification method that exploits the naive Bayes assumption. Khiops operates in two steps. In the first step consisting in preparing data it estimates the univariate conditional probabilities using the MODL (Minimum Optimized Description Length) method, with Bayes optimal discretizations and value groupings for numerical and categorical variables [1, 2]. In the second step, the modeling step, it searches for a subset of variables consistent with the naive Bayes assumption, using an evaluation based on a Bayesian model selection approach and efficient add-drop greedy heuristics [3]. Finally, it combines all the evaluated models using a compression-based averaging schema [4].

This approach also quantitatively evaluates the predictive importance of each variable for the target. At the end of the preparation step (preprocessing of the input data) Khiops returns a value (the level) that is directly the predictive importance of the

[2] www.khiops.com

explicative variable for the target. A level of zero means that the variable has been discretized in one single interval (for a numerical variable) or the modes are in one single group (for a symbolic variable). The variable is not informative for the modelisation and thus can be reliably discarded.

3.2 Classification results

A k-fold cross validation procedure (with *k=10*) has been used. The performance of every model is computed on the fold which has not been used to train the classifier. The ten 'test' results are then combined to give an estimation of the generalization error of the model. The folds used to do the training of the initial classifier do not cross the test set. Note that the classification model Khiops is robust to unbalanced classes. There is no need to sample the data to balance the classes.

The predictive model is evaluated using the accuracy on classification (ACC) and the area under the ROC curve (AUC) [6] (the higher the criteria, the better, with 1.00 indicating perfect performance). The numerical results for the two criteria are respectively *0.8924 +/- 0.0017 and 0.8185 +/- 0.0078* that are very good results.

Another way to present the performance of a classification model is the cumulative gain curve. It is a graphical representation of the advantage of using a predictive model to choose which unstable lines to select. The x-axis gives the proportion of the lines with the best probability to correspond to the target (unstable class), according to the model. The y-axis gives the percentage of the targeted lines reached. The curves are plotted in Figure 2. The diagonal represents the performance of a random model. If we target 20% of the lines with the random model, we are able to reach 20% of the unstable lines. With the current model, when 20% of the lines are observed, 77% of the unstable lines are reached.

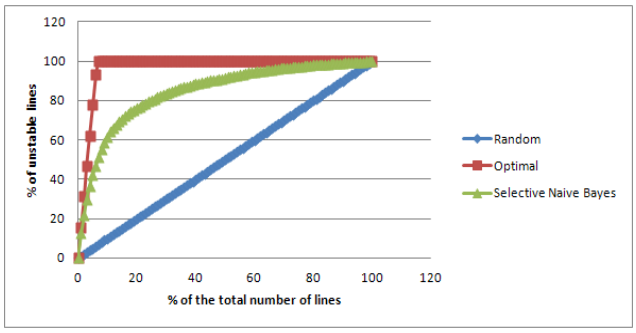


Figure 2. The lift curve

A deep analysis of the classification errors has shown that main classification errors come from the wrongly assignment of the label stable to a line that had been found unstable by the network application. This difference is coherent with the observation made by the network service in charge of the labeling of ADSL lines that has detected a problem in the labeling of some lines (with the label unstable while they are not). This happens in some DSLAMs with specific cards that are unable to distinguish modem extinctions from desynchronizations due to ADSL transmission. The consequence is the increase of the count of desynchronizations used by the network application to determine the state of a line and so to wrongly label a line as unstable.

Khiops allows further interpretation of the numerical results. As said earlier, the tool analyses each variable independently for the target in the preparation step and return a value that is directly its predictive importance (the level) for the classification model. The table 1 contains the 5 most informative variables found by Khiops. 31 variables have been found useful for the model (with a level >0).

Table 1. The variables ranked by level of importance.

rank	Variable name	level
1	nb_tickets_306_1	0.172868
2	nb_tickets_308_1	0.172453
3	nb_tickets_C	0.151913
4	nb_tickets	0.150877
5	nb_tickets_308	0.14135

306_1 is the code for a lost of service for an event of type desynchronization, 308_1 is the code for a return of service for the same type of event.

The observation of the more informative variables confirms that the stability notion is strongly correlated to the information of ADSL line desynchronization. This is perfectly consistent with the lines labeling method used by the network application whereas the view given by the software agent in the live box is only a service view completely independent of the network view.

4 EVALUATION AND RESULT INTERPRETATION

This section is dedicated to the evaluation and interpretation steps. We propose a method to *(i)* identify the importance of the explanatory variables for every ADSL line in the database (and not "in average" for all the examples) and *(ii)* propose an action in order to change the probability of the desired class. The method is completely automatic. It is based on the analysis of the link between the probabilities at the output of the classifier and the values of the explanatory variables at the input.

4.1 Individual importance of explanatory variables

For a classifier as the naive Bayes we choose the method called Weight of Evidence (WoE) described in [9] and [11] as the importance measure. This indicator measures the log of the odds ratio. It is computed for all the explanatory variables at the input of the classifier and for one specific class. The class of interest (*q)* is generally the predicted class for the instance x_k. A variable with a positive importance (WoE) contributes positively to define the predicted class. At the opposite, a variable with a negative importance (WoE) contributes negatively to define the predicted class (and so positively contributes to define another class of the classification problem).

This point is illustrated in Figure 3 with 3 ADSL lines. The first line is classified as stable, the second is classified as risky and the third is classified by the model as unstable. The x-axis represents simply the index of the 31 input variables (with level >0). The figure (which can be seen as a nomogram [9]) indicates that:

- the line predicted as stable is characterized by number of little positive contributions (positive importances) and some negative contributions,

- the line predicted as risky is characterized by number of little positive contributions (positive importances) and hardly any negative contributions
- the line predicted as unstable is characterized by some heavy positive contributions.

So we have a completely individual interpretation for each ADSL line allowing a precise diagnosis. For each line we can have:

- the class predicted by the classifier,
- a confidence score on this prediction,
- the importance value for each explanatory variable.

Arguments for the choice of the method and details about the algorithm are given in [8].

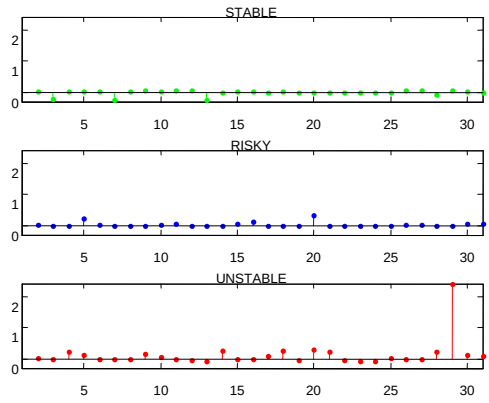


Figure 3. Examples of explanatory variables importances for 3 ADSL lines (from top to bottom) for a line predicted as stable, a line predicted as risky and a line predicted as unstable.

4.2 Indexes for the improvement of the line stability

In this section a method that study the influence of the input values on the output scores of the naïve based classifier is exploited. The goal is to increase the predictive probability of a given class by exploring the possible values of the input variables taken independently. Then it became possible to act on some variables, the lever variables that are defined as the explanatory variables for which it is conceivable to modify their value to induce changes of occurrence of the desired class.

We use the methodology described in [8]. Given C_z the target class among the C target classes (for example here the class unstable line). Given f_z the function which models the predicted probability of the target class $f_z(X = x) = P(C_z|X = x)$. Given the equality of the vector X of the J explanatory variables to a given vector x of J values. The method proposed here tries to increase the value of $P(C_z|X = x_k)$ successively for each of the K examples of the considered data base. $P(C_z|X = x_k)$ is the natural value of the model output. The idea is to modify the values of some explanatory variables (the lever variables) in order to study the variation of the probabilist classifier output for the considered example. The values of the lever variables are modified to see the variations in the posterior probabilities. This is done by covering some interval for continuous variables and by trying the possible values for discrete ones (after the preparation step where variables have been discretized or grouped).

The method is illustrated with the explanatory variable "live box firmware" as lever variable. This symbolic variable can take 4 modalities that will be called *A*, *B*, *C* and *D* for confidential reasons. These four groups are preserved after the first step of pretreatment for the classifier construction.

We focus on the ADSL lines labeled unstable and effectively predicted unstable by the classifier (1611 lines). For 1321 of these lines there exists a value that can increase the probability of the class stable. To do that the variable "live box firmware" has to take the value *D*. For the other 380 ADSL lines the probability of stability cannot be improved (the value of the variable is already *D*).

The improvements of the probability are presented Figure 4. We have, distributed on the x-axis, the 3 modalities of the variable "live box firmware" that can be changed. Then within each modality the values of $(PCa(x_k)-PCi(x_k))$ are arranged by ascending order (with (Ca) the value which leads to the improvement, (PCa) the associated improved probability and (PCi) the initial probability). To be more precise we have for x_k in [1:407] LBF='A', for x_k in [408:778] LBF='B' and for x_k in [779:1231] LBF='C'. The blue points coincide to an improvement without class change. The red squares coincide to an improvement with class change. We can conclude that the variable "live box firmware" is really a lever variable. When its value is changed to *D*, it allows obtaining more stable lines (from 1231 cases on 1611) or even obtaining stable lines (from 51 lines, Figure 4).

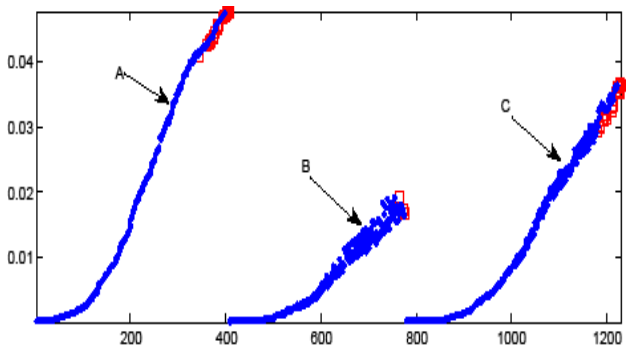


Figure 4. Possible improvement of $(PCa(x_k)-PCi(x_k))$ of the stability for the 1280 ADSL lines x_k .

The exploration of correlations has been made ADSL line by ADSL line. This exploration offers some means to improve their stability. The knowledge extracted at the end of the exploration step can be listed with the four following points for each ADSL line labeled as instable:

- the initial probability of instability *PCi(.)*
- the improved probability of instability (reduced) *PCa(.)*
- the explanatory variable that allows the gain
- the value that has to take this explorative variable to reach the gain.

5 DISCUSSION

We have presented a complete process for the supervision of VoIP services. Into this process, we have made a focus on a specific point that is the detection of the stability of the ADSL lines. A dedicated classifier based on naïve Bayes has been defined. The

model used to classify the ADSL lines according to their stability led to very good classification performances validating the data extraction and data creation steps. The analysis of the most important variables has shown a strong correlation between the stability state of the ADSL line and the information about the desynchronization of the line (the two most informative variables for the target are the number of tickets of lost of service and of return of service for an event of type desynchronization). This result is totally coherent with the network process that is currently used to label the state of ADSL lines.

It appears from this study two ways of using the classification model in an operational way:

It can be used as a filter of the unstable lines, to keep only the stable lines. In this way a large number of tickets, produced mainly because of the line instability are discarded.

Another use could be the reinforcement of the knowledge used by the network tool to label the ADSL lines and in the end the improvement of this labeling. Indeed, we know that the labeling process is imperfect for some DSLAM models that are unable to distinguish electric start/stop from desynchronizations. As the stability of the ADSL lines is directly correlated to the number of desynchronizations, this involves wrongly unstable labels for some lines. On the other side, the live box counts the number of desynchronizations in a way that is totally independent of those of the DSLAM. So, the exploitation of the number of tickets of this type could help to make more precise the counters from which is based the stability decision.

The various steps of the data mining process are for the most part industrialized (around 75%): from the data acquisition to the stability ADSL lines prediction. The exploitation of the prediction information is under industrialization.

The interpretation step of the classification results produces an interpretation totally individual of each ADSL line allowing a very precise diagnostic. The exploration of correlation step can give some means to improve the stability of the lines and so intervention plans can be designed.

REFERENCES

[1] M. Boullé, 'A bayes optimal approach for partitioning the values of categorical attributes', *Journal of Machine Learning Research*, **6**, 1431–1452, (2005).

[2] M. Boullé, MODL: 'a Bayes optimal discretization method for continuous attributes'. Machine Learning, *Machine Learning*, **65**, (1) 131–165, (2006).

[3] M. Boullé, 'Compression based averaging of selective naïve bayes classifiers', *Journal of Machine Learning Research*, **8**, 1659–1685, (2007).

[4] M. Boullé. 'Regularization and Averaging of the Selective Naive Bayes Classifier'. In *IJCNN*, 2989-2997, (2006).

[5] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C Shearer and R. Wirth. *CRISP-DM 1.0: step-by-step data mining guide*, 2000.

[6] T. Fawcett, *ROC graphs: Notes and practical considerations for researchers*, Technical Report HPL-2003-4, HP Laboratories, 2003.

[7] U.M. Fayad, G. Piatetsky-Shapiro and P. Smyth, *Advances in Knowledge Discovery and data Mining*, 1–34, Chapter From Data Mining to Knowledge Discovery: An Overview, AAAI/MIT Press, 1996.

[8] V. Lemaire, C. Hue and O. Bernier, *Correlation Analysis in Classifiers*, 204–218, Chapter Data Mining in Public and Private Sectors: Organizational and Government Applications, IGI Global, 2010.

[9] M. Možina, J. Demšar, M. Kattan and B. Zupan, B. (2004). 'Nomograms for visualization of naïve Bayesian classifier'. In Proceedings of *the 8th european conference on principles and practice of knowledge discovery in databases (PAKDD)*. 337–348. New York, USA: Springer-Verlag New York, Inc, 2004.

[10] D. Pyle, *Data Preparation for Data Mining*, Morgan Kaufman, 1999.

[11] M. Robnik-Sikonja and I. Kononenko, 'Explaining classifications for individual instances'. *IEEE TKDE*, **20** (5), 589–600, (2008).

[12] Z. Wang, *Internet QoS: Architectures and Mechanisms for Quality of Service*, The Morgan Kaufman Series in Networking, San Francisco, 2001.

Learning Nash Equilibria in Distributed Channel Selection for Frequency-agile Radios

Irene Macaluso and **Luiz DaSilva** and **Linda Doyle**[1]

Abstract.

Wireless communication networks are evolving towards self-configuring, autonomous and distributed multiagent systems in which nodes are deployed randomly and have to adapt to the environment in which they operate. A cognitive network is a self-organising system that relies on the ability of its autonomous nodes to support communication in an adaptive and distributed manner. In this paper we address the distributed channel selection problem, which is a crucial component of many cognitive networks scenario, in the context of frequency-agile radios that are able to operate in multiple frequency bands simultaneously. We formulate the problem as an N-player stochastic game with incomplete information. We prove that by adopting a simple reinforcement scheme, namely learning automata, nodes will converge to a Nash equilibrium, under the assumption of symmetric interference between the players.

1 Introduction

Recent years have seen the evolution of traditional wireless networks towards self-configuring, autonomous and distributed multi-agent systems. This trend opens the doors to a new and exciting research field involving both the specialization of state-of-the-art general purpose multiagent algorithms and the development of new techniques to solve issues which are specific of the wireless communication domain.

A crucial factor in the evolution of wireless communication systems is the management of radio spectrum. In today's static communication systems, regulators decide both the use of a certain frequency band as well as the users that are allowed to transmit on that band (see Figure 1(a)). On the one hand this model, by rigidly planning the allocation of frequency bands, ensures that the interference between neighboring systems is limited. On the other hand, as the allocation of spectrum bands is implemented on a long term assignment basis, it is widely recognised that such an approach leads to an inefficient usage of the spectrum resources. A first consequence of this quasi-static paradigm is the presence of idle capacity within the system: numerous studies and measurement campaigns have shown that spectrum resources are often underutilized.

As spectrum becomes a more and more valuable resource, it is imperative to address the inherent inefficiency that characterizes the current spectrum management mechanism. Previous work focused on mechanisms that assign spectrum more dynamically over shorter time frames on an as-needed basis (see Figure 1(b)). The opening up of the TV white spaces [2] has made the concept of dynamic access to spectrum a reality. Networks using the TV white spaces can avail of any unsed spectrum in the TV bands as distinct from being assigned a static allowance. Frequency-agile radios make it possible to use whatever spectrum is available. In addition further advancements mean that radios can use non-contiguous as well as contiguous blocks of spectrum.

The use of non-contiguous blocks of spectrum for communications is increasingly of interest. LTE-Advanced [10] introduces the concept of carrier aggregation which facilitates the combining of different blocks of LTE spectrum to form larger transmission bandwidths. This will be crucial in supporting the ever-increasing data demands on mobile networks. In LTE-Advanced carrier aggregation can be performed using contiguous or non-contiguous blocks of spectrum and radio receiver architectures are being designed with this in mind. Though carrier aggregation, as currently envisaged, will involve the aggregation of static assignments of spectrum, it is not unrealistic to envisge this becoming more dynamic in the future.

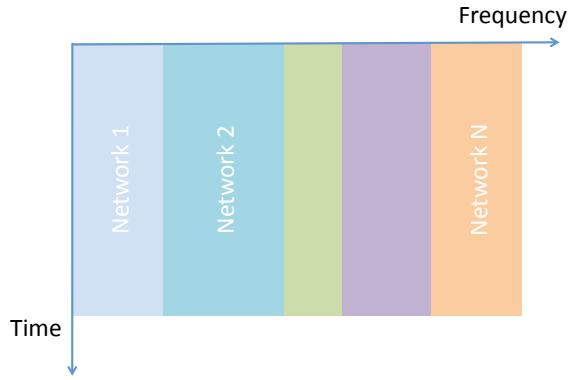
In this paper we look at scenarios in which a group of networks dynamically access non-contiguous blocks of spectrum and do so without the need for central coordination. We address the problem of distributed spectrum resources allocation in the context of frequency-agile radios that are able to operate in multiple frequency bands simultaneously. A number of frequency bands, herein named channels, have to be assigned to a number of wireless networks in a distributed manner so that the interference between adjacent systems is minimized. As each radio is able to operate simultaneously on multiple non-contiguous frequency bands, we extend the traditional distributed channel selection problem so that each network has to decide how many and which channels it should access. The problem of distributed channel selection has been addressed in the literature from a game theory point of view [7], from a distributed multi-agent learning perspective [1, 3] as well as a combined approach [8]. For special types of games, e.g. potential games, a proof of convergence of a reinforcement learning procedure has been provided [6]. We have recently applied learning automata to the problem of distributed channel selection for radios that have to decide whether it is advantageous to select an additional channel as opposed to keep using only its current transmission channel [4].

The problem we address in this paper is an extension of the traditional channel selection problem in that we allow each network to take advantage of of non-contiguous frequency bands. This problem can be formulated as an N-player stochastic game with incomplete information, i.e. the distribution of each player's payoff is unknown to other players. All the wireless networks have to reach, in an autonomous and distributed manner, a stable allocation of the available channels, corresponding to a Nash equilibrium (NE). We model each

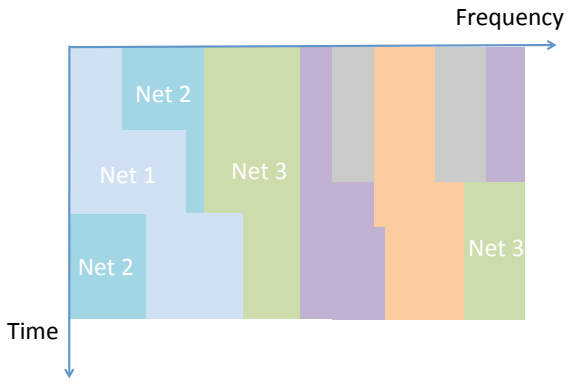
[1] CTVR Telecommunications Research Centre, Trinity College Dublin, Ireland

player as a learning automaton [5], which is a reinforcement learning scheme where each agent is a policy iterator. We prove that by adopting this simple reinforcement scheme players will converge to a Nash equilibrium, under the assumption of symmetric interference between the players.

The remainder of the paper is organized as follows. Section 2 details the system model and some of the key properties of learning automata. Section 3 provides the proof of convergence to a NE of the proposed learning procedure. Section 4 presents the numerical results. We summarize our conclusions in section 5.



(a) Fixed band planning.



(b) Dynamic channel allocation.

Figure 1. (a) A central entity divides the spectrum into frequency channels which are then assigned to various technology/users. (b) Spectrum resource assignment changes in time according to the requirements of the various wireless networks.

2 Problem Description

Let us assume that N wireless networks are operating in the same geographical area where M frequency channels are available. Two or more networks can share the same channel only if the interference they cause to each other is below a certain threshold; otherwise they have to choose different channels. The goal of a distributed channel allocation algorithm is to compute a stable channel assignment, either through negotiation or blind interaction between the networks, so as to minimize interference between the systems.

In this paper we adopt the blind interaction approach. The players indirectly interact by causing interference to each other. No explicit communication is allowed. This tight restriction is a desideratum when one is designing a communication system in which cooperation between the different players cannot be assumed. Moreover it simplifies the system design: in the case of cooperation between networks a common control channel to exchange information has to be established and maintained; also different networks need to agree on a common protocol. Thus these requirements might hinder the deployment of heterogenous wireless networks.

We model each player i as a learning automaton: each player i keeps a probability distribution $\mathbf{p}_i(k)$ over the set of actions and, at each stage k, it chooses an action according to $\mathbf{p}_i(k)$. Then each player updates its action probabilities based on the response $r_i(k)$ from the environment, ignoring the presence of the other agents in the same environment.

In this paper we adopt the Linear Reward-Inaction scheme (L_{R-I}). The automaton increments the probability of the action that resulted in a favorable response from the environment, while accordingly decreasing the probabilities of all the other actions. In case of an unfavorable environment response, the action probabilities are left unaltered. The L_{R-I} updating rule is:

$$\mathbf{p}_i(k+1) = \mathbf{p}_i(k) + br_i(k)(\mathbf{e}_i - \mathbf{p}_i(k)) \tag{1}$$

where $b \in (0,1)$ is the reward parameter and $\mathbf{e}_i$ is the unit vector, where the i^{th} element is unity.

Each network i wants to get assigned a number of channels $c_i \leq M_i$, where $M_i \leq M \ \forall i$. Thus each agent has to decide how many and which channels it should access. By exploiting the development of frequency-agile radios, which are capable of transmitting on non-contiguous frequency bands, the cardinality of each agent's action space is:

$$|A_i| = \sum_{j=1}^{M_i} \binom{M}{j}. \tag{2}$$

In other words, each agent can decide to transmit on any of the possible combinations of the desired number of channels.

Let r_i be the payoff of player i, modeled as a random variable, with $r_i \in [0,1]$. Let us define the utility function of player i:

$$d^i(a_1, a_2, ..., a_N) = E[r_i | a_1, a_2, ..., a_N] \tag{3}$$

where a_j is the action chosen by player j. If we denote by p_{ij} the probability that player i chooses action j, a strategy for player i is defined as $\mathbf{p}_i = [p_{i1}, p_{i2}, ..., p_{i|A_i|}]$. As in [9], we define the functions f_{is} as the expected payoff of player i, given that player i selects action s and player l adopts strategy $\mathbf{p}_l$:

$$f_{is}(Q) = \sum_{\substack{j_1,\ldots,j_{i-1},\\ j_{i+1},\ldots,j_N}} d^i(j_1, \ldots, j_{i-1}, s, j_{i+1}, \ldots, j_N) \prod_{l \neq i} p_{lj_l} \tag{4}$$

where $Q = (\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N)$ and j_l is the action selected by player l with probability p_{lj_l}.

If the learning factor b in (1) is small the following statements hold [9]:

1. If the learning algorithm converges, it always converges to a NE.
2. For any N-player game, all strict Nash equilibria in pure strategies are asymptotically stable.
3. If there exists a bounded differentiable function $F : R^{M_1+M2+...+M_N} \rightarrow R$, such that for some constant $v > 0$

$$\frac{\partial F}{\partial p_{is}}(Q) = v f_{is}(Q), \forall i, s \text{ and } \forall Q \in I^{M_1+M2+...+M_N} \tag{5}$$

where $I = [0,1]$, then the learning algorithm always converges to an NE, for any initial condition.

3 Learning Nash Equilibria

We model the problem of multiple channel selection as an N-player game. Each player has to decide how many and which channel it should transmit on while minimizing the interference caused to adjacent systems. Thus a single action a_i denotes the set of channels selected by player i.

In our model the players' payoff is a function of each node's SINR. The SINR for player i transmitting on channel c_i is defined as:

$$\gamma_{ic_i} = \gamma_i(a_1, ..., c_i, ..., a_N) = \frac{|h_i(c_i)|^2 P}{\sigma_i^2(c_i) + \sum_{j|c_i \in a_j} |h_{ji}(c_i)|^2 P} \quad (6)$$

where a_j is the action selected by player j, P is the transmit power, $|h_i(c_i)|$ is the channel gain between transmitter i and receiver i in channel c_i, $|h_{ji}(c_i)|$ is the channel gain between transmitter j and receiver i in channel c_i, and $\sigma_i^2(c_i)$ is the noise power at the receiver i in channel c_i. In our study we assume that the channel gains are time-invariant realizations of a circular symmetric complex normal distribution with zero mean and unit variance. The noise vector is also modeled as a circular symmetric complex Gaussian random variable.

We define the payoff of player i as:

$$r_i(a_1, ..., a_i, ..., a_N) = \frac{1}{M_i} \sum_{c_i \in a_i} \left(1 - \frac{\bar{\gamma}}{\gamma_{ic_i}}\right) + \left(1 - \frac{n_{a_i}}{M_i}\right) \quad (7)$$

where $\bar{\gamma}$ is the minimum SINR that receiver i can support, n_{a_i} is the number of channels that node i transmits on when selecting action a_i (i.e. $n_{a_i} = |a_i|$) and M_i is the maximum number of channels that node i is interested in.

Theorem 3.1. *A pure strategy NE $(a_1^*, .., a_i^*, ..., a_N^*)$, according to the payoff given in (7), corresponds to the players choosing the maximum number of channels characterized by an SINR which is greater than $\bar{\gamma}$.*

Proof. If action a_i^* includes a channel k for which the receiver i experiences an SINR $\gamma_{ik} \leq \bar{\gamma}$, then the payoff of player i is:

$$\begin{aligned} r_i(a_1^*, .., a_i^*, ..., a_N^*) &= \frac{1}{M_i} \sum_{c_i \in a_i} \left(1 - \frac{\bar{\gamma}}{\gamma_{ic_i}}\right) + \left(1 - \frac{n_{a_i}}{M_i}\right) \\ &\leq \frac{1}{M_i} \sum_{c_i \in \bar{a_i}} \left(1 - \frac{\bar{\gamma}}{\gamma_{ic_i}}\right) + \left(1 - \frac{n_{a_i} - 1}{M_i}\right) \\ &= r_i(a_1^*, .., \bar{a_i}, ..., a_N^*) \end{aligned} \quad (8)$$

where $\bar{a_i} = a_i \setminus \{k\}$.

If there exists a channel $k \notin a_i^*$ such that $\gamma_{ik} > \bar{\gamma}$, then:

$$\begin{aligned} r_i(a_1^*, .., \hat{a_i}, ..., a_N^*) &= \frac{1}{M_i} \sum_{c_i \in \hat{a_i}} \left(1 - \frac{\bar{\gamma}}{\gamma_{ic_i}}\right) + \left(1 - \frac{n_{\hat{a_i}}}{M_i}\right) \\ &> r_i(a_1^*, .., a_i^*, ..., a_N^*) \end{aligned} \quad (9)$$

where $\hat{a_i} = a_i \cup \{k\}$ and $n_{\hat{a_i}} = n_{a_i} + 1$.

□

To simplify the notation, we denote by $g_{ji}^{(s)}$ and $g_i^{(s)}$ the square of the expected value of the channel gain between player j and player i corresponding to channel s and the square of the expected value of the channel gain between the i^{th} transmitter-receiver pair in channel s. Let us define the *interference functions* as:

$$r_{ij}(c_i, c_j) = \begin{cases} \frac{g_{ji}^{(c_i)}}{g_i^{(c_i)}} & \text{if } c_i = c_j \text{ and } j \in N_i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where N_i denotes the set of players that can interfere with player i. The interference functions are said to be symmetric if $r_{ij}(c_i, c_j) = r_{ji}(c_j, c_i), \forall i, j, a_i, a_j$.

Theorem 3.2. *The learning algorithm specified by (1) with payoff function given in (7) always converges to a NE if the interference functions are symmetric and the parameter b in (1) is small.*

Proof. Accordingly to the payoff function in (7), (4) can be rewritten as:

$$\begin{aligned} f_{is}(Q) &= \frac{1}{M_i} \sum_{c \in s} \left(1 - \frac{\bar{\gamma}}{g_i^{(c)} P} \left(\sigma^2(c) + \sum_{j \in N_i} g_{ji}^{(c)} P \sum_{a_j | c \in a_j} p_{ja_j}\right)\right) \\ &+ \left(1 - \frac{n_s}{M_i}\right) \end{aligned} \quad (11)$$

Let us define the function:

$$\begin{aligned} F(Q) &= \frac{2}{M_i} \sum_i \sum_k n_k p_{ik} - \frac{2\bar{\gamma}}{M_i P} \sum_i \sum_k \sum_{c_k \in k} \frac{\sigma^2(c_k)}{g_i^{(c_k)}} p_{ik} \\ &- \frac{\bar{\gamma}}{M_i} \sum_i \sum_j \sum_k \sum_t \sum_{c_k \in k} \sum_{c_t \in t} r_{ij}(c_k, c_t) p_{ik} p_{jt} \\ &+ 2 \sum_i \sum_k (1 - \frac{n_k}{M_i}) p_{ik} \end{aligned} \quad (12)$$

From (11) and (12), by exploiting the symmetry of the interference functions, we get:

$$\frac{\partial F}{\partial p_{is}}(Q) = \frac{2n_s}{M_i} - \frac{2\bar{\gamma}}{M_i} \sum_{c \in s} \left(\frac{\sigma^2(c)}{g_i^{(c)} P} + \frac{\sum_{j \in N_i} g_{ji}^{(c)} \sum_{a_j | c \in a_j} p_{ja_j}}{g_i^{(c)}} \right) + 2(1 - \frac{n_s}{M_i})$$

Therefore, according to the results reported in the previous section (theorems 3.2 and 3.3 in [9]), the learning algorithm always converges to a NE. □

4 Simulation results

In this section we discuss the effect the number of networks N, number of channels M, and noise power σ^2 has on the convergence time of the proposed learning algorithm. The payoff defined in (7) has a direct implementation and it allows a fully distributed solution to the problem of channel selection. The receiver estimates the SINR on the set of channels selected by its transmitter and sends back this information. The transmitter then updates its policy accordingly. If the transmitter does not receive an ACK from its receiver(s), an unfavorable response is assumed. For each scenario we run 10^5 independent simulations. We assumed that two or more networks cannot operate on the same channels.

Figure 2 shows the average number of iterations required to converge to a NE with respect to the cardinality of each player action space. We assumed that each network tries to access at most $M_i = 2$ channels. Thus the cardinality of the action space is $A_i = M + \binom{M}{2}$. We can observe that the convergence time increases with the number of players N. The impact of the cardinality of the action space A_i on the convergence time is an interesting aspect of these results. As expected, until a certain point the convergence time of the algorithm

increases with A_i. Then, increasing the number of available actions does not affect the convergence time.

In Figure 3 we analyzed the combined effect of the number of available channels M and the noise power on the average number of iterations to converge to a NE. In particular, we assumed that $N = 4$ networks operate on the same geographical area and cannot utilize the same channels. Accordingly to what we discussed above, the impact of increasing the number of channels becomes less significant after a certain point, independently on the noise power. For each value of A_i, the convergence time linearly increases with the noise power.

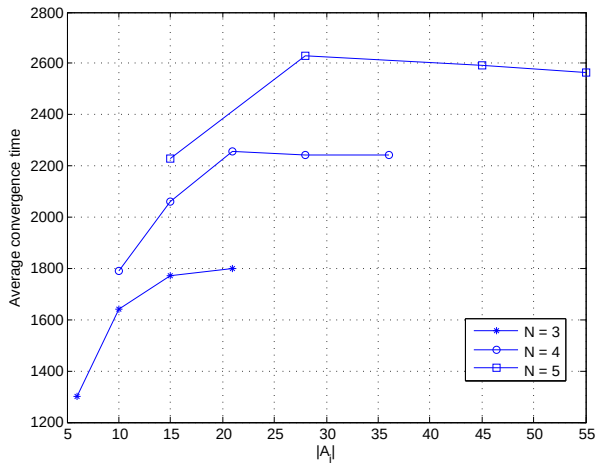


Figure 2. The average number of iterations to converge to a NE versus cardinality of each player action space ($\sigma^2 = 0.01$). For each scenario, i.e. number of players and number of available channels, we run 10^5 independent simulations ($b = 0.05$).

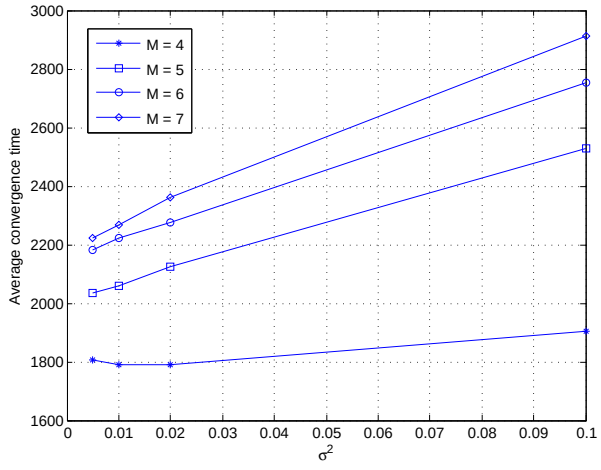


Figure 3. The average number of iterations to converge to a NE versus noise power at the receiver with $N = 4$ players. For each scenario, i.e. noise power and number of available channels M, we run 10^5 independent simulations ($b = 0.05$).

5 Conclusions

In this paper, we studied the problem of distributed channel allocation by exploiting a radio's ability to operate in multiple channels simultaneously. In particular, we extended the traditional channel selection problem to allow a radio to decide how many and which channels it should use, while minimizing the interference suffered from nearby systems. By modeling each player as a learning automaton and using the Linear Reward Inaction scheme, we formally proved convergence to a pure Nash equilibrium, under the assumption of symmetric interference between the players.

REFERENCES

[1] A. Anandkumar, N. Michael, A.K. Tang, and A. Swami, 'Distributed Algorithms for Learning and Cognitive Medium Access with Logarithmic Regret', *Selected Areas in Communications, IEEE Journal on*, **29**(4), 731–745, (2011).

[2] Federal Communications Commission. Second report and order and memorandum opinion and order in the matter of unlicensed operation in the tv broadcast bands and additional spectrum for unlicensed devices below 900 MHz and in the 3 GHz band, 14th November 2008.

[3] H. Li, 'Multiagent Q-learning for Aloha-like Spectrum Access in Cognitive Radio Systems', *EURASIP Journal on Wireless Communications and Networking*, **2010**, 56, (2010).

[4] I. Macaluso and L.A. DaSilva, 'Learning Automata and Distributed Channel Selection in Cognitive Networks', (under review, 2012).

[5] K.S. Narendra and M.A.L. Thathachar, *Learning automata: an introduction*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989.

[6] S.M. Perlaza, H. Tembine, and S. Lasaulce, 'How can Ignorant but Patient Cognitive Terminals Learn their Strategy and Utility?', in *Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on*, pp. 1–5. IEEE, (2010).

[7] L. Rose, S.M. Perlaza, and M. Debbah, 'On the Nash Equilibria in Decentralized Parallel Interference Channels', in *IEEE Worshop Game Theory and Resource Allocation for 4G*, (2011).

[8] L. Rose, S.M. Perlaza, S. Lasaulce, and M. Debbah, 'Learning Equilibria with Partial Information in Decentralized Wireless Networks', *IEEE communications Magazine*, **49**(8), (2011).

[9] PS Sastry, VV Phansalkar, and MAL Thathachar, 'Decentralized Learning of Nash Equilibria in Multi-Person Stochastic Games with Incomplete Information', *Systems, Man and Cybernetics, IEEE Transactions on*, **24**(5), 769–777, (1994).

[10] Guangxiang Yuan, Xiang Zhang, Wenbo Wang, and Yang Yang, 'Carrier aggregation for lte-advanced mobile communication systems', **48**, 88–93, (2010).

Social Choice in Sensor Networks

Conor Muldoon[1] and **Gregory M. P. O'Hare**[1]

Abstract. In this position statement, we argue for the use of online algorithms for social choice and group decision making in sensor networks whereby self-interested agents socially maximize their utility and preferences, which are based on variable network state. Specifically, we consider the nondictatorship principle of Arrow's Impossibility Theorem and discuss this in the context of the Schulze voting method.

1 POSITION STATEMENT

In Arrow's seminal paper on Social Choice Theory [1], it is shown that it is not possible to construct a voting procedure that is fair in the sense of Pareto optimality, independence of irrelevant alternatives, and universality save a dictatorship. This position statement argues for the application of Social Choice Theory and, in particular, the Schulze voting method [2] to the problem of making group decisions with regard to a discrete set of Medium Access Control (MAC) layer operational parameters.

In sensors networks, determining the MAC layer operational parameters is an important task in ensuring the network meets given Quality of Service requirements subject to constraints, such as network longevity and available bandwidth. Typically, the operational parameters are optimized for peak load requirements, leading to wastage at times of low usage.

Recent work in sensor networks [3] has considered the problem of gathering global network state at the base station to optimise the MAC layer parameters in terms of bandwidth, energy efficiency, and end-to-end reliability so as to enable the network to adapt to changes in the topology and dynamic workload requirements. Determining the optimal choice over the preferences of individual agents, or nodes within the network, in relation to social choice criteria, such as the Condorcet principle, is not equivalent to aggregating the network state at the base station and then solving a global optimization problem. As such, the approach discussed here from prior work [3][4] in MAC layer parameter optimisation[2].

In contrast to prior work, we argue that mechanisms, such as the Schulze voting method, should be incorporated into sensor network applications whereby social choice properties, such as the Condorcet criterion, Pareto optimality, the resolvability criterion, and clone independence are important; that is, in applications where group decisions are made through the voting of different nodes in the network. The Schulze method operates in polynomial time and can be easily implemented with a runtime complexity of $O(n^3 + n^2v)$, where n is the number of candidate options and v is the number of voters, through the incorporation of a modified version of the Floyd-Warshall algorithm. Due to its low complexity, this approach could potentially be used in the development of an in network voting procedure whereby agents, in the networks of moderate size, initiate ballots at different points throughout execution. Alternatively, it could be used at the base station for large scale networks of hundreds of nodes.

In short, the Schulze voting method offers a practical approach to developing sensor network applications whereby group decision making at runtime, over a discrete set of actions or values for global network parameters, is necessary so as the ensure that the network operates efficiently. Incorporating social choice theory into sensor applications and protocols will enable the developer to ask not only does the application operate correctly, but what social choice characteristics it exhibits.

Future work will investigate the role of social choice properties in the development of a Time Division Multiple Access MAC layer. Specifically, the Schulze method will be used in conjunction with the Agent Factory Micro Edition agent programming framework [5][6] and tested on a 100 node deployment of Java-enabled motes developed at the Tyndall National Institute, University College Cork.

[1] CLARITY: Centre for Sensor Web Technologies, School of Computer Science and Informatics, University College Dublin, Belfield, Dublin 4, Ireland, email: {conor.muldoon,gregory.ohare}@ucd.ie

[2] It would, of course, be possible to determine individual agent preferences and then use a voting procedure at the base station provided the global state of the network is known, but prior work in sensor networks does not consider this.

ACKNOWLEDGEMENTS

This work was supported by IRCSET, the European Commission Marie Curie Actions under FP7, and SFI under Grant No. 07/CE/I1147.

REFERENCES

[1] Arrow, K.J., "A Difficulty in the Concept of Social Welfare", Journal of Political Economy 58(4) (August, 1950), pp. 328–346.

[2] Schulze, M., "A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method", Social Choice and Welfare, 2011.

[3] Zimmerling, M. and Ferrari, F. and Mottola, L. and Voigt, T. and Thiele, L., "pTunes: Runtime Parameter Adaptation for Low-power MAC Protocols", Information Processing in Sensor Networks, 2012.

[4] Raman, B. and Chebrolu, K. and Bijwe, S. and Gabale, V., "PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer", In ACM SenSys, 2010.

[5] Muldoon, C., O'Hare, G. M. P., O'Grady, M. J., and Tynan, R. "Agent Migration and Communication in WSNs", In Sensor Networks and Ambient Intelligence, IEEE Computer Society Press, 2008.

[6] Muldoon, C. and O'Hare, G. M. P. and Bradley, J. F. "Towards reflective mobile agents for resource-constrained mobile devices", In AAMAS, 2007.

Model-based simulation and configuration of mobile phone networks – The SIMOA approach

Iulia Nica and **Franz Wotawa**[1]
and **Roland Ochenbauer** and **Christian Schober**[2]
and **Harald Hofbauer** and **Sanja Boltek**[3]

Abstract. Machine to machine communication via mobile phone networks is of increasing importance for mobile phone network suppliers as well as for companies offering applications in the domain. For example the question whether a distributed application that communicates within such a network can meet certain quality of services has to be answered before deployment. In this paper we present a tool that uses a model of the network and the intended network use in order to answer such questions. Beside simulation of the network the proposed approach is also capable of providing configurations, i.e., changes in certain parameters in order to meet given specifications. The underlying tool makes use of a modeling language that allows to specify the behavior of components over time and their interconnections. In the paper we discuss the approach, the currently deployed tool, the used modeling language, and some empirical results obtained that indicate feasibility of the approach for the indicated application domain.

1 INTRODUCTION

There is a growing interest in machine to machine (M2M) communication. This is due to the fact of the currently available infrastructure like the TCP/IP based internet or mobile phone networks. Because of the still increasing amount of data that is transported using the infrastructure, there is a growing need for actively controlling the infrastructure in order to prevent delays or breakdowns in the worst case. In this paper we report on a tool, which has been developed as part of the project *Simulation and configuration of mobile networks for M2M applications (SIMOA)* that deals with the study of principles for the simulation, configuration and optimization of mobile networks. The focus of the project is to provide a tool for investigating and solving problems in wireless networks under the assumption of a future user behavior and growing M2M applications. In particular, we are interested in reconfiguring a mobile network in case of bottlenecks caused by requirements coming from future M2M applications. The result of the suggestions for reconfiguration coming from the tool might be changes in the structure or the parameters of the mobile network.

It is worth noting that the importance of the application area of the introduced tool increases. The M2M market is growing in the coming years to the largest customer of the mobile operators. The application scenarios provide a large variety, with an emphasis in communication with energy suppliers of electricity meters, the automatic meter reading (smart metering), and the tracking of vehicles and goods. These application scenarios provide additional benefits for the customers. For example,smart metering solutions can save utilities from the problems of a short blackout, when the power is limited before a blackout occurs. This requires that thousands of GPRS terminals can communicate simultaneously. Unfortunately, the current mobile networks are not designed to handle these requirements. For example, at New Year's Day handling all calls at midnight, is still almost impossible. In the future, because of the growing number of M2M applications, handling this vast amount of simultaneous calls is the requirement throughout the whole year all time.

From the application domain we are able to derive the requirements for a tool that allows for providing change information for the given mobile network considering future usage scenarios. The future usage scenarios are determined by the communication needs of the intended M2M applications.

- *Language.* The application domain requires information on the structure and behavior of the network as well as on usage scenarios. Because of the huge variety of networks and usage scenarios, a modeling language has to be provided for formalizing the information. This language has to allow for stating the structure of a system and the behavior of its components. Temporal aspects like changes in network access over time has to be captured as well. Moreover, in case of optimization the language has to provide means for stating optimality criteria.
- *Simulation.* The first step to evaluate whether an available mobile network is capable of handling all requests imposed by a future M2M application is to simulate the network behavior using the communication requirements of the M2M application. Communication requirements might be maximum delays of replies of remote machines on messages send or the required amount of data to be send at the same time. If the simulation returns that the network cannot handle certain requirements of the M2M application, someone is interested in finding a re-configuration of the network that allows for fulfilling the requirements if possible. The simulation itself should be based on the introduced modeling language.
- *Configuration.* Changing an available network for fulfilling communication scenarios imposed by M2M applications can be done on several levels. Certain parameters of mobile phone cells can be changed in order to provide more channels. New cells can be added to the network. Thus changing the network topology. In our application all such possible changes should be reported. Again the same modeling language that is used for simulation should also be used for providing configuration information.
- *Optimization.* Since there are very likely more than one reconfiguration for handling future communication requirements

[1] Technische Universität Graz, Austria, email: {inica,wotawa}@ist.tugraz.at
[2] Kapsch Business Com, Austria
[3] Kapsch Carrier Com, Austria

someone is interested in the optimal solution. This might be the one with least costs or one that fulfills another optimality criteria. Therefore, a requirement is that the optimality criteria can be specified and that it is possible to search for configuration fulfilling the optimality criteria.

In order to handle simulation, configuration, and optimization based on the same underlying information a model-based approach has to be adopted. In particular, a modeling language that allows for specifying structure and behavior of a systems as well as configuration information is necessary. Not mentioned in the above requirements it is also necessary that the language can be used by ordinary technicians and not only AI experts. This is due to the fact that both the considered network as well as the M2M application requirements change over time and has to be adapted. In this paper we present the SIMOA approach that implements a model-based configuration engine, which is capable of fulfilling the mentioned requirements. The SIMOA approach is based on the SIMOL programming language, which is object-oriented and from the syntax close to the Java programming language. The SIMOL language is converted into a set of constraints that can be used directly to compute configurations. In this paper we also briefly discuss this conversion. The currently deployed SIMOA tool simulates a mobile network and also allows for deriving configurations.

The paper is organized as follows. In the next section we give an overview of the SIMOA system's architecture. Afterwards, we introduce a small example we are going to use throughout the paper. Further on we briefly introduce the modeling language SIMOL. A discussion of obtained empirical results showing the scalability of the approach and a conclusion finalize the paper.

2 SIMOA ARCHITECTURE

The requirements of the application domain indicate a general language that allows for modeling technical systems comprising components which are interconnected. Moreover, the language should be able to formalize the behavior for simulating the system and the knowledge necessary for carrying out configuration and later on optimization. A language that is restricted to one M2M scenario is likely not capable of handling different new scenarios at the same time. In order to save investments we decided not only to come up with the general modeling language SIMOL but also to develop a tool and its underlying framework that is as general and thus flexible as possible.

The resulting SIMOL tool architecture comprises three modules that are connected and interchange information. Figure 1 depicts the SIMOA architecture. The central input to the tool is beside observations, i.e., case specific knowledge, the domain knowledge. Both case specific knowledge and domain knowledge can be formalized using the SIMOL language. We are discussing the language in the next chapters of this paper and therefore omit any information regarding the language. The program written in SIMOL is taken as input of the first module, i.e., the SIMOL compiler, which maps the program to a set of constraints. Moreover, the compiler generates a symbol table that allows for mapping the variables used in the SIMOL program to the variables used in the constraint representation. This is necessary later on to map back the computed solutions.

The second module is the constraint solver. For more information on constraints and constraint solving we refer the interested reader to Rina Dechter's book [2]. In our tool we are using the MINION constraint solver [4]. Therefore, the SIMOL compiler generates a MINION program, which is used by the CSP solver for computing solutions. A solution to a CSP problem is an assignment of variables

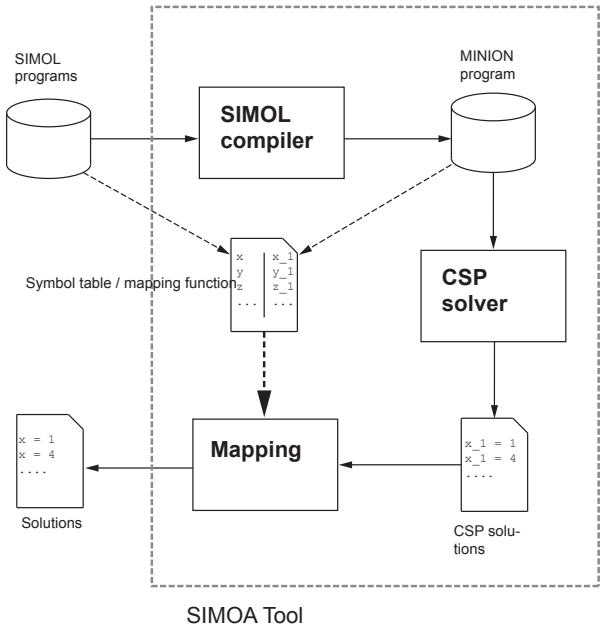


Figure 1. The SIMOA tool architecture

to values such that no constraint is violated. Hence, the result of the second module is basically nothing else than a list of variable assignment. This list of variable assignments is taken by the third module the Mapping module. This module takes the solutions coming from the constraint solver and changes the variable names to the names used in the original SIMOL program. For this purpose the generated symbol table is used.

Because of this architecture the obtained tool is general and can be used to solve various tasks. The only requirement is that the domain and the case specific knowledge are specified using SIMOL. In the next chapters we are going to introduce the language using a small running example from the M2M domain where we smart meters communicate via a mobile phone network to a server. It is worth noting that the application we are going to deploy falls into this domain. The specified tool is specially tailored and hides the domain knowledge from the user. The purpose of the tool is to show that a given set of smart meters can be used in a region while still fulfilling all requirements. Such requirements include for example, shutting down power lines in a region within a certain amount of time. The adapted SIMOA tool has been implemented in Java using the Google Web Toolkit[4]. The tool is available for all partners of the project via the web interface (see Figure 2).

3 RUNNING EXAMPLE

Complex systems like mobile networks are generally designed with structural decomposition in mind in order to solve independently different subproblems, which may refer to various parts of the network such as a mobile network cell or the required data storage.

For the sake of clarity, we choose to present here the simulation process for a simplified cell with M2M communication capability, comprising one base transceiver station and five smart meters. The base transceiver station (BTS) facilitates the wireless communication

[4] see `code.google.com/webtoolkit/`

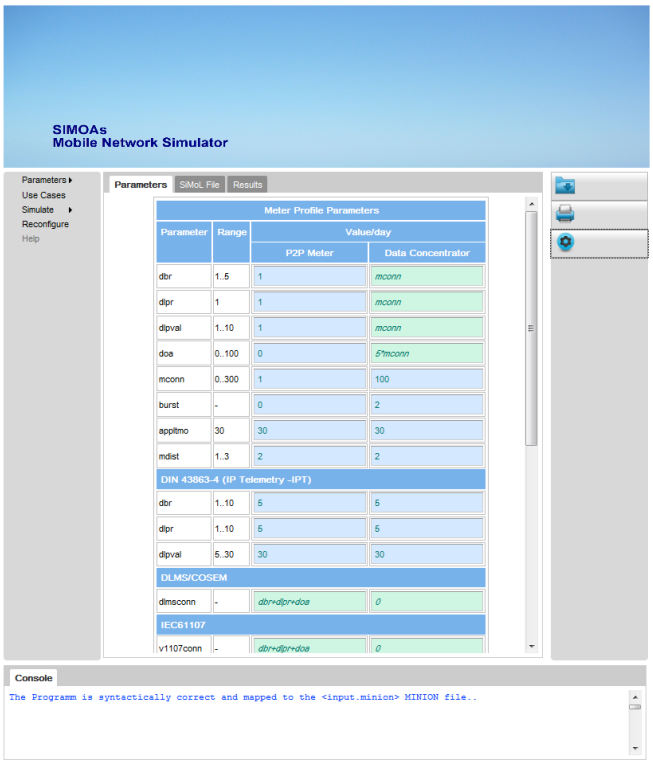


Figure 2. SIMOA Web GUI

between the smart meters and the network. For the running example, we will further consider individually addressable smart meters, called Point-to-Point (P2P) meters. As illustrated in the web interface (see Figure 2), both P2P meters and BTS present specific configurable parameters, based on which the simulator must generate the appropriate output, corresponding to the desired use case. In the following, we present one of the simulated use cases.

A typical scenario is the transfer of the daily load profiles from the meters to the central meter-management office. That means only the up-link traffic of the cell is investigated. Assuming that the meters can only send this information from 12 a.m. to 7 a.m., the required transfer time is seven hours that is 25,200 seconds. We also assume that, during this time period, the data transfer is uniformly distributed.

Considering the input parameters depicted in Table 1. The SIMOA simulator computes as outcome the resource utilization, i.e., the number of the used time slots, and the average, minimum and maximum data transfer speed in $kbit/s$ per cell (see Table 3).

Table 1. Input Parameters

Parameter Name	Variable/Value
Total number of P2P meters	$z = 5$
Number of P2P meters with CS1	v
Number of P2P meters with CS2	w
Number of P2P meters with CS3	x
Number of P2P meters with CS4	y
Data Amount per P2P Meter	$D_{Meter} = 20kB$
Required Transfer Time T	$00:00 - 07:00$

The SIMOA tool simulates the cell over a number of states. Each state corresponds to a different codeset distribution where a code set is channel encoding that influences the amount of data to be transferred within a second. See Table 2 for possible channel encodings. In order to simulate the dynamics of a real mobile network, we allow the cell passing from one state to another. We fix the total number of states during the simulation. Note that the model and the transition function between the states is defined in a SIMOL program. We present the program in the next section. In this section we now focus on available constraints for representing the model of the system.

For every state, the tool computes the real transfer rate per cell R_{real} according to the following formula

$$R_{real} = (v * CS1 + w * CS2 + x * CS3 + y * CS4)/z$$

,

where *CS1..CS4* refer to the channel encodings. In the simulation given in Table 2. Consequently we obtain the maximum data transfer speed in state $S0$ and the minimum data transfer speed in state $S1$ (see Table 3) .

Table 2. Channel Encoding

GPRS Coding Scheme	kbit/s
CS1	8 kbit/s
CS2	12 kbit/s
CS3	14,4 kbit/s
CS4	20 kbit/s

Table 3. Real Transfer Rate per Cell

State: (v,w,x,y)	Real Transfer Rate per Cell R_{real}
S0:(0,5,0,0)	12 kbit/s
S1:(2,3,0,0)	10,44 kbit/s
S2:(2,2,1,0)	10,88 kbit/s
S3:(2,1,2,0)	11,36 kbit/s

Note that the needed data transfer speed R_{needed} differs from the upper mentioned real transfer rate and is computed using the following formula:

$$R_{needed} = z * D_{Meter}/T$$

In our example, the average (needed) data transfer speed R_{needed} becomes:

$$R_{needed} = 5 * 163840Bit/25200s = 32,51Bit/s$$

It is easy to observe that, in each of the generated states from Table 3, R_{real} is much greater than the upper computed R_{needed}, which means that the requested data transfer is possible for all the codeset distributions.

But let us consider now a scenario in which the transfer would not be possible.

If we assume that the time duration is T=1 minute, then the needed data transfer speed R_{needed} becomes:

$$R_{needed} = 5 * 163840Bit/60s = 13,33kBit/s$$

Another important parameter that one must take into consideration at this point is the number of available GPRS time slots in the cell $TS_{available}$.

We should also mention that the number of the used time slots, depicted by TS_{needed} is derived from the needed data transfer speed R_{needed} divided by the real transfer rate R_{real}, i.e.:

$$TS_{needed} = R_{needed}/R_{real}$$

Assuming that our BTS allocates $TS_{available}$=1 time slots for the M2M communication in the cell, the constraint that must be satisfied is:

$$TS_{needed} <= TS_{available}$$

In other words, if $R_{needed} > R_{real}$ and we have just one time slot allocated for the data transfer, the simulation will fail. This is the case for all the states presented and takes us to the next step, i.e., the identification of a new parameter that will change its value from one state to the other. We emphasize the fact that we currently restrict ourself to reconfigurations changing parameters and other changeable functions of the system only. With this restriction, reconfiguration can be seen as a method for identifying changeable parts that enable a system transition such that the resulting system fulfills the stated requirements. Some aspects of this idea are discussed in [10].

4 SIMOL LANGUAGE

System knowledge modeling represents a significant part both in the simulation and configuration process. In order to fit the different necessities and goals in each practical area, various languages have been developed over time.

For the first category of languages, we mention Matlab/Simulink[4] and Modelica[5] as they are the most famous ones in the area of dynamic systems modeling and simulation. Although both of them are complex languages, capable of modeling a great variety of components, they are less appropriate for re-configuration purposes.

Further on, if we restrict ourselves to the domain of mobile networks applications, ns-2[9] and OPNET[1] are two of the most popular network simulation packages on the market. The ns-2 simulator uses OTcl(an object oriented version of Tcl) as command and configuration interface, while in OPNET the user creates networking models by using the drag and drop approach. One drawback of using these tools in the domain of M2M applications is the difficulty of simulating new platforms and protocols. In [6], the author mentions that implementing new protocols in the existing tool packages is rather difficult and time consuming. Further more, both ns-2 and OPNET operate at the packet level, which is not the case in our approach. For instance, in case of smart metering requirements, all the meter profiles parameters are given per day and cell.

Regarding the languages used in the configuration domain, they are usually not easy to learn and prevent the systems based on such languages to be used in practice.

Hence, there is a strong need for an easy to learn and use modeling language that is expressive enough to state configuration problems. The modeling language we describe in this paper serves this purpose. The SIMOL language is a declarative object-oriented language with multiple inheritance, which adopts a clear Java-like syntax. Beside the basic data types like integer and boolean, SIMOL makes use of component instances and allows the modeler to state constraints between variables. For more details about the implemented language, we refer the reader to [11].

In order to get a clear picture of the language without going into details, we will further focus on a concrete example. Have a look at

[4] www.mathworks.com
[5] www.modelica.com

```
1.  kbase CellSimulation;
2.  component P2PMeter {
3.    attribute int dbr, dlpr, dlpval, mdist,
      doa, iptlogin, iptconn, iptwtg, dlmsconn,
      ulgmsc, dlgmsc, gcspc;
4.    constraints {
5.      dbr = 1;
6.      dlpr = 1;
7.      dlpval = 1;
8.      doa = 0;
9.      mdist = {1..3};
10.     iptlogin = 5;
11.     iptconn = 5;
12.     iptwtg = 30;
13.     gspc = {1..4};
14.     ulgmsc = 2;
15.     dlgmsc = 4;
16.   }
17. }
19. component BTS {
20.   attribute int fpc, gtpc;
21.   constraints{
22.     fpc = 0..7;
23.     gtpc = {1..4}
24.   }
25. }
26. component Cell {
27.   constraints {
28.     BTS b1;
29.     P2PMeter s[5];
30.     exist (1, P2PMeter, midst=2);
31.     exist (2, P2PMeter, midst=3);
32.     exist (2, P2PMeter, midst=1);
33.   }
34.   transition {
35.     forall ( P2PMeter ) {
36.       if (midst=2 or gcspc != 2 )
37.         gcspc.next = gcspc;
38.       if (midst=1)
39.         gcspc.next = gcspc + 1;
40.       if (midst=3)
41.         gcspc.next = gcspc - 1;
42.     }
43.   }
44. }
```

Figure 3. A SIMOL Program. The upper implemented SIMOL model is a simplified version of the running example from section 3.

the SIMOL program depicted in Figure 3. Every SIMOL program comprises two main parts:

- *the knowledge base declaration*, which is optional and usually used to organize the components which are specific to a certain domain or problem (similar to a Java package),
- *the component definition*, that is the main constructing unit of a SIMOL program and is mandatory. All the defined components posses a set of attributes and introduce constraints in the system. Furthermore, for example the `Cell` component has a `transition{ ... }` block, by means of which the modeler defines the transition function between two consecutive temporal states of the model. The attributes declaration is marked with

the `attribute` keyword. Whereas the relations stated between the component attributes and new component instance-declaration statements appear enclosed in a `constraints{ ... }` block. In our approach, the constraints can be of three kinds:

- *general constraints*, which are satisfied in all the states. The general constraints appear in the `constraints{ ... }` block;
- *state-specific constraints*, which are satisfied only in a certain state. The state-specific constraints are given in an additional observations file, that contains at least the initial constraints (for the cell in the initial state). In our experiments, we considered that the observations file contains the default values for the codesets used for each P2P meter, i.e., the codesets in state $S0$.
- *transition constraints* between two consecutive states. The transition constraints appear in the `transition{ ... }` block and are implemented by means of the *next* operator.

In our running example, one can adapt the transfer speed automatically depending on the meter location relative to the BTS location. Thus, depending on the meter profile parameter $mdist$, defined in Line 9 (see Figure 3), which represents the distance between the BTS and the current meter, the codeset value, defined in Line 13, can be increased or decreased by following the constraints from the `transition{ ... }` block of the `Cell` component. Based on these transition constraints, the simulator will generate for every state the new corresponding constraints.
Thus, in case of the defined `Cell` (Lines 26- 44), we have the constraints defined within the `constraints { ... }` block plus the constraints generated in the `transition{ ... }` . Moreover, the component `Cell` also receives constraints from the instances used in the component definition. For example, when specifying in Line 28: `BTS b1;`, a new instance of `BTS` is generated and all the constraints of `BTS` are added to the constraints of `Cell`.

Regarding the types of statements illustrated in the given SIMOL program, we briefly mention:

- the *component instance declaration*, in Line 28, 29;
  Using this kind of statements, we define the subcomponent hierarchy in our model. The cardinality of these relations (i.e., the number of subcomponents, which can be connected to a certain component) is always finite.
- the *arithmetic or/and boolean expression*, illustrated in Line 36, 39 etc.;
- the *initialization of attributes with integer valued ranges*, in Line 9, that provides direct control over the possible values of a component attribute;
- the *forall block*, in Line 35, and the *conditional block*, in Line 36;
- the *exist statement*, in Line 30, meant to ease the manipulation of large sets of component instances.

5 EXPERIMENTS

In order to prove feasibility of the proposed approach, we conducted some experiments in the application domain using the first model of a mobile network cell that communicates with a varying number of P2P meters. The number of P2P meters varied from 5 to 1,000 leading to quite large constraint systems comprising up to 36,000 constraints and 16,000 variables. Table 4 shows the obtained running times when restricting to searching for 10 solutions at maximum and

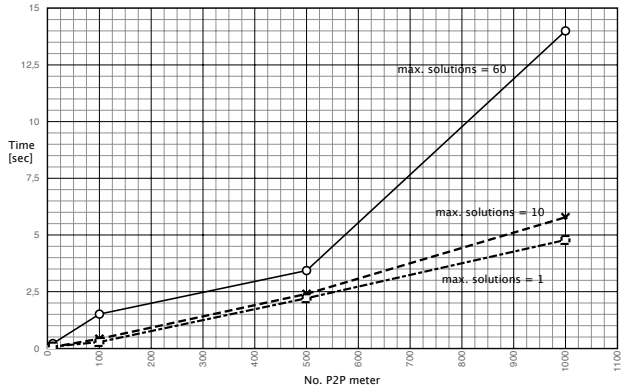


Figure 4. Comparing running times for constraint solving for a specific maximum number of solutions when the variable domain is fixed to [0..1,000]

4 temporal states. For the experiments we made use of a notebook with Intel(R) Core(TM) i7 CPU 1.73 GHz and 4 GB of RAM running under Windows 7.

The obtained results indicate that the approach is feasible for the application domain. Especially when considering that in most cases not more than 1000 P2P meters should be connected to one cell. Moreover, usually the simulation should only clarify whether there is a result for the intended configuration. Hence, finding a small number of solutions is acceptable in the M2M domain. In order to give an estimate on the impact of the searching for more solutions we also conducted an experiment where we restricted the variable domain to [0..1,000] and used Minion to search for a maximum of 60 solutions. Figure 4 shows the measured running times in comparison to searching for 10 solutions. From this figure we see an increase of running time. However, the results are still in an acceptable range and we are able to proof feasibility with respect to our application domain.

In future experiments we will use more sophisticated models from other domains in order to explore the influence of the models for solving constraints using SIMOL as modeling language. Moreover, there are many ways of representing SIMOL as set of constraints, which might also influence the overall running time. Exploring different mappings from SIMOL to Minion has to be done in future research.

6 RELATED RESEARCH AND CONCLUSION

In our research, we intend to combine techniques from both simulation and configuration domains, such that the implemented tool could meet the demands imposed by different problem types.

Beside the already mentioned available network simulators, ns-2 and OPNET, there is a multitude of commercial software planning tools which combine the environment specific constraints with signal propagation. In [5], the implemented WLAN modeling tool defines the environment by describing the floor plan structure and the wall types. This definition will further serve as input for the electromagnetic propagation model, which eventually leads to an estimation of the maximum achievable throughput in specific sites within the considered building. The optimization module is used to automatically optimize the number of access points and their position to meet site specific demands.

SIMOA is, on the contrary, no design tool, it has no drawing tools and the results vary depending on the desired use case. They may in-

Table 4. Simulation results obtained for different integer variable domains. The solution number is limited to 10, by making use of the MINION command-line switch $-sollimit10$ when running the MINION Solver. The given running time is a rounded average value over 3 runs.

SIMOL Program		MINION File		Simulation Time [sec] for different domains				
P2P Meters No.	**LOC**	Constraints No.	Variables No.	[0..100]	[0..200]	[0..500]	[0..1,000]	[0..10,000]
5	46	215	97	0.015	0.031	0.046	0.047	0.078
10	46	406	178	0.031	0.046	0.046	0.062	0.078
100	46	3,507	1,620	0.320	0.320	0.350	0.420	0.700
500	46	19,015	8,020	1.960	2.070	2.260	2.400	na
1,000	46	36,010	16,011	4.570	4.630	4.800	5.780	na

clude the graphical evolution of the data transfer rate over the time, tables with the constant and variable simulation parameters, etc. Regarding the optimization, we intend to introduce different optimality criteria, based on our industrial partners requirements. We could have for instance: minimality of necessary changes in a cell/ network, minimal duration for a specific data transfer etc.

In the end, we will shortly recall a couple of configuration systems, which illustrate some important knowledge-based configuration specific notions.

A powerful configuration system that makes use of constraint programming(CP) together with a description logic(DL) is the ILOG (J)Configurator [8]. The combined CP-DL language provides, on the one hand, the constraints, needed in the decision process specific to configuration, and on the other hand, the constructs of the description logic, able to deal with unknown universes. When solving the problem, the constructs of description logic, which are well-suited to model the taxonomic and partonomic relations, are mapped on constraints and hereby the wide range of constraint solving algorithms can be used.

Other configuration systems include ConBaCon (Constraint-Based Configuration) [7] and CAWICOMS (Customer-Adaptive Web Interface for the Configuration Of products and services with Multiple Suppliers) [3]. ConBaCon treats the special case of reconfiguration, using the conditional propagation of constraint networks and has its own input language - ConBaConL. In [7], the authors present ConBAConL, a "largely declarative specification language", by means of which one can specify the object hierarchy, the context-independent constraints and the context constraints. Furthermore, the constraints are divided into Simple Constraints, Compositional Constraints and Conditional Constraints.

In [3], an application scenario for semantic Web services is presented, choosing as example the domain of telecommunication services. In order to define a common language for representing the properties of configurable products and services, the authors use a hierarchical approach of related ontologies. By means of DAML+OIL language, a flexible product ontology for complex, customizable products can be modeled, the domain knowledge being consistently represented in XML. For the configuration task, the ILOGs domain-independent and Java-based JConfigurator was adopted. JConfigurator implements Generative Constraints Satisfaction for solving complex configuration problems. Actually, the configuration task is executed on a distributed architecture, i.e., each involved configurator has only a partial view on the product model. The communication between the configurators occurs by means of XML-based SOAP messaging and Web Services.

In contrast to previous work in configuration, we have models that capture the temporal aspects of the real system. Moreover, the approach is a model-based approach where simulation and configuration results can be directly obtained from the underlying models. The currently deployed system uses a model of the communication within a mobile phone network written using the SIMOL modeling language. SIMOL is an object-oriented language that allows for specifying constraints at a higher level. In the current implementation SIMOL programs are mapped to Minion constraints and Minion is used to provide solutions. The configuration capabilities are currently limited to parameter configurations. An extension to allow for changing structures and components as well as an module for computing optimal solutions is left for future research. Moreover, in the future we also want to provide different models from different domains.

REFERENCES

[1] Min Chen. OPNET Network Simulation. Press of Tsinghua University, 2004.

[2] Rina Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.

[3] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker, 'Semantic Configuration Web Services in the CAWICOMS Project', in *ISWC '02 Proceedings of the First International Semantic Web Conference on The Semantic Web*, pp. 192–205, (2002).

[4] I. P. Gent, C. Jefferson, and I. Miguel, 'Minion: A fast, scalable, constraint solver', *17th European Conference on Artificial Intelligence*, **ECAI-06**, (2006).

[5] Alan Mc Gibney, Martin Klepal, and Dirk Pesch, 'A wireless local area network modeling tool for scalable indoor access point placement optimization', in *Proceedings of the 2010 Spring Simulation Multiconference*, SpringSim '10, pp. 163:1–163:8, San Diego, CA, USA, (2010). Society for Computer Simulation International.

[6] Imad Jawhar, 'A flexible object-oriented design of an event-driven wireless network simulator', in *Proceedings of the 2009 conference on Wireless Telecommunications Symposium*, WTS'09, pp. 80–86, Piscataway, NJ, USA, (2009). IEEE Press.

[7] Ulrich John and Ulrich Geske, 'Reconfiguration of Technical Products Using ConBaCon', in *Proceedings of WS on Configuration at AAAI99*, Orlando, (1999).

[8] Ulrich Junker and Daniel Mailharro, 'The logic of ILOG (J)Configurator: Combining Constraint Programming with a Description Logic', in *Proceedings of IJCAI-03 Configuration WS*, pp. 13–20, (2003).

[9] Steven McCanne and Sally Floyd. ns Network Simulator. http://www.isi.edu/nsnam/ns/.

[10] Iulia Nica and Franz Wotawa, 'Diagnosis-based reconfiguration using the MINION constraint solver', in *Proceedings of the 22nd International Workshop on Principles of Diagnosis (DX 2011)*, pp. 225–232, (2011).

[11] Iulia Nica and Franz Wotawa, 'The SiMoL: Modeling Language for Simulation and (Re-)Configurationr', in *SOFSEM 2012: Theory and Practice of Computer Science 38th Conference on Current Trends in Theory and Practice of Computer Science*, pp. 661–672. Springer-Verlag, (2012).

Multiple Sink and Relay Placement in Wireless Sensor Networks

Lanny Sitanayah [1] and **Kenneth N. Brown** [2] and **Cormac J. Sreenan** [3]

Abstract. Wireless sensor networks are subject to failures. Deployment planning should ensure that when a sink or sensor node fails, the remaining network can still be connected, and so may require placing multiple sinks and relay nodes in addition to sensors. For network performance requirements, there may also be path-length constraints for each sensor node. We propose two local search algorithms, GRASP-MSP and GRASP-MSRP, to solve the problem of multiple sink placement and the problem of multiple sink and relay placement, respectively. GRASP-MSP minimises the deployment cost, while ensuring that each sensor node in the network is double-covered, i.e. it has two length-constrained paths to two sinks. GRASP-MSRP deploys sinks and relays to minimise the deployment cost and to guarantee that all sensor nodes in the network are double-covered and noncritical. A sensor node is noncritical if upon its removal, all remaining sensor nodes still have length-constrained paths to sinks. We evaluate the algorithms empirically and show that both GRASP-MSP and GRASP-MSRP outperform the closely-related algorithms from the literature for the lowest total deployment cost.

1 INTRODUCTION

A wireless sensor network (WSN) is composed of a large number of sensor nodes [1]. Each sensor node is a battery-powered device with limited storage, processing and communication capability. It is able to sense a close-by physical phenomenon, perform a simple computation and send its data wirelessly over a multi-hop network to a special node called a data sink. This network is subject to failure as the wireless devices and the communication links are unreliable. A sensor node may fail due to limited battery life or hardware malfunction, or may be damaged by weather or human intervention. When some sensor nodes fail, the network may be disconnected and thus it cannot gather information from the isolated area. Although a sink has more resources than a sensor node, this electronic device may fail too.

To protect against network failure, it is important to plan the topology deployment. In this paper, we study WSN deployment planning, with the aim of protecting the network against a single failure. That is, after a failure of a sink or a sensor node, each remaining sensor node can deliver its data to a sink through a multi-hop path with an acceptable length. We consider the path length restriction as data latency requirements may be important in WSN applications. To be robust to sink failure, it is necessary to deploy multiple sinks in the network such that each sensor node is *double-covered*, i.e. it has length-bound paths to two sinks. While we restrict our assumption to k-covered, where $k = 2$ in this paper, our solution is also applicable to any integer $k \geq 1$. To protect against sensor node failure, it is necessary to place some relay nodes, which do not sense, but only forward data from other nodes. So when a sensor node fails, each remaining sensor node still has at least one length-bound path to a sink. Installing sinks and relays comes at a cost that includes not only the hardware purchases, but also the installation and maintenance cost, thus motivating our solution to minimise the total deployment cost.

Our main contribution is a solution that minimises the total cost of sink and relay deployment but ensures the network robustness against a single device failure, either a sink or a sensor node. Our solution uses local search algorithms based on GRASP [4]. Firstly, we look at the multiple sink placement (MSP) problem to ensure that each sensor node is double-covered. We propose GRASP-MSP and show that it can find the same cost as the optimal solution with shorter runtime. Even though finding the optimal solution is sufficient for the multiple sink placement problem, the GRASP-MSP performance gives us confidence to use the same local search technique for the more complex multiple sink and relay placement (MSRP) problem. GRASP-MSRP employs the concept of length-constrained connectivity and rerouting centrality (l-CRC) introduced in [10] to identify every sensor node which if failed can cause other nodes to lose their length-bound paths to sinks. We demonstrate empirically that the solutions produced by GRASP-MSRP are over 30% less costly than those of a greedy-based approach. Although GRASP-MSRP's execution time is longer than that for the greedy approach, the runtime issue is not a problem since the deployment planning is an offline process that is executed before the actual deployment.

The remainder of this paper is organised as follows. In Section 2, we briefly describe the background and review the related work on sink and relay deployment algorithms. We present and evaluate GRASP-MSP and GRASP-MSRP in Section 3 and 4, respectively. Finally, Section 5 concludes the paper.

2 BACKGROUND AND RELATED WORK

A WSN can be modeled as a graph $G = (V, E)$, where V is a set of nodes and E is a set of edges. Each edge connects two nodes that are within transmission range of each other[4], and the two nodes are said to be *adjacent*. A *path* of length t between two nodes v and w is a sequence of nodes $v = v_0, v_1, \ldots, v_t = w$, such that v_i and v_{i+1} are adjacent for each i. A path from a node v to a set of nodes W is simply

[1] Mobile & Internet Systems Laboratory (MISL) and Cork Constraint Computation Centre (4C), Department of Computer Science, University College Cork, Ireland, email: ls3@cs.ucc.ie

[2] Cork Constraint Computation Centre (4C), Department of Computer Science, University College Cork, Ireland, email: k.brown@cs.ucc.ie

[3] Mobile & Internet Systems Laboratory (MISL), Department of Computer Science, University College Cork, Ireland, email: cjs@cs.ucc.ie

[4] For simplicity we assume bi-directional links, but this could be easily relaxed by specifying a more complex connectivity graph.

a path from v to any node $w \in W$. Two nodes are *connected* if there is a path between them. A graph is connected if every pair of nodes is connected. Sensor network topology is an undirected graph and for simplicity, we assume that the graph is connected. $H = (W, E{\downarrow_W})$ is an induced subgraph of $G = (V, E)$ if $W \subset V$ and $E{\downarrow_W}$ has exactly the edges that appear in G over the same vertex set (where $E{\downarrow_X}$ means a set of edges restricted to those that connect nodes in X).

In the literature, the problems of deploying sinks and relays are solved separately. Some sink deployment algorithms have been proposed with objectives to minimise and balance the energy consumption across networks [11, 6], reduce packet delivery latency [12], meet the required lifetime [8] and make the network double-covered for fault-tolerance [3]. Even though the algorithm in [3] is not designed for WSNs, the problem of finding the optimal positions for core nodes in passive optical networks [3] is similar to the problem of finding optimal positions for sinks in WSNs. Similar to our objective, i.e. to minimise the deployment cost, the algorithms proposed in [11, 8] try to minimise the number of sinks deployed, while the number of sinks is given in [6, 12, 3]. In [11], a sink is chosen greedily from a set of candidate locations such that it can cover as many sensor nodes, which are within the hop count bound from the sink, as possible. However, this algorithm does not consider double-covered networks. In [8], the algorithm deploys sinks one by one until the desired network lifetime is reached. It does not make the network double-covered and uses the well-known k-means clustering algorithm to identify the positions of sinks, which are assumed can be placed anywhere in the region. The algorithm in [3] also uses the k-means clustering algorithm to place a given number of sinks to make the network double-covered.

The problem of deploying relay nodes for increased reliability has long been acknowledged as a significant problem [2, 9, 5, 7, 10]. The relay placement algorithm proposed in [10], GRASP-ABP, is also a GRASP-based local search algorithm. It uses length-constrained connectivity and rerouting centrality (l-CRC) to identify critical nodes. Centrality indices is a core concept in social network analysis, used to determine the importance of a node in a network. Originally, it is measured by counting the number of the shortest paths passing through a certain node. A sensor node is identified as a critical node if upon its removal, more sensor nodes will have no path of length $\leq l_{\max}$ to a sink, where $l_{\max}$ is the maximum acceptable path length. Using l-CRC, a sensor node is critical if its centrality index is above a threshold. l-CRC is formulated as

$$l\text{-CRC}(v) = < l\text{-CC}(v), l\text{-RC}(v) > \quad (1)$$

l-CRC has two values: length-constrained connectivity centrality (l-CC) and length-constrained rerouting centrality (l-RC), which are formulated below.

$$l\text{-CC}(v) = |\{w \in D(v);\ d(w,S) \leq l_{\max}, d_v(w,S) > l_{\max}\}| \quad (2)$$

$$l\text{-RC}(v) = \sum_{w \in D(v)} \left(\frac{\max\{d_v(w,S), l_{\max}\}}{\max\{d(w,S), l_{\max}\}} - 1 \right);\ d_v(w,S) \neq \infty \quad (3)$$

$D(v)$ is the set of node v's descendants in the routing tree, S is the set of sinks, $d(u,w)$ denotes the shortest path length between nodes u and w, while $d_v(u,w)$ represents the length of the shortest path from u to w which does not visit v. After identifying the critical nodes, relays are deployed around those nodes to preserve backup paths if they die. Using l-CRC allows us to trade off the deployment cost against the robustness of the network. GRASP-ABP has been shown to deploy fewer relays compared to the algorithm in [2, 9].

The greedy randomized adaptive search procedure (GRASP) [4] is a metaheuristic intended to capture the good features of pure greedy algorithms and of random construction procedures. It is an iterative process, which consists of two phases: a construction phase and a local search phase. The construction phase builds a feasible solution as a good starting solution for the local search. The probabilistic component of a GRASP is characterised by randomly choosing one of the best initial solution. Since the solution produced by the construction phase is not necessarily the local optimum, the local search works iteratively to replace the current solution by a better one.

3 MULTIPLE SINK PLACEMENT (MSP)

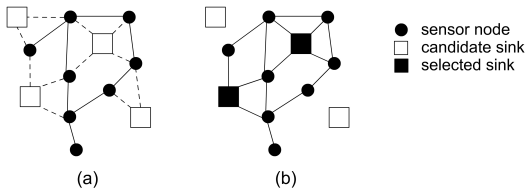


Figure 1. (a) A WSN with 4 candidate sinks and (b) the double-covered WSN where $l_{\max} = 3$

We partition nodes into a set of sensors T and sinks S. In the graph representation, $V = T \cup S$. Note that at this stage we do not use relay nodes yet. A sensor node is *double-covered* if and only if it has at least two paths of length $\leq l_{\max}$ to two sinks in S. If a sensor node is not double-covered, it is *uncovered*. We define a WSN to be double-covered if each sensor node $v \in T$ is double-covered. In the *multiple sink placement* problem, given a graph $G = (T \cup A_{\mathrm{S}}, E)$, where A_{S} is a set of candidate locations for sinks with a non-negative cost function $c : A_{\mathrm{S}} \rightarrow \mathbb{R}$, we find a minimum cost subset $S \subseteq A_{\mathrm{S}}$ such that $H = (T \cup S, E{\downarrow_{T \cup S}})$ is double-covered. We illustrate this problem in Figure 1.

We propose GRASP-MSP to solve the multiple sink placement problem. As with other GRASP-based algorithms, GRASP-MSP consists of two steps: *construction phase* to construct an initial feasible solution and *local search phase* to explore the neighbourhood of the initial solution, looking for lower cost solutions. To speed-up our GRASP algorithm's processing time, we compute the shortest path from all sensor nodes to all candidate sinks once in the beginning and store the length of the shortest path in *Distance* table, while the parent of each sensor node in the shortest path to a candidate sink is stored in *Parent* table. For example, for $G = (T \cup A_{\mathrm{S}}, E)$, $\mathit{Distance}_G(v,w)$ shows the length of the shortest path from a sensor node $v \in T$ to a candidate sink $w \in A_{\mathrm{S}}$, while $\mathit{Parent}_G(v,w)$ shows the parent of a sensor node v on the shortest path to a candidate sink $w \in A_{\mathrm{S}}$.

3.1 GRASP-MSP: Construction Phase

In the construction phase, we find S, an initial set of sinks. Instead of selecting the best candidate sink from A_{S} to be put in S, which can minimise the number of uncovered nodes, we add randomisation to the initial solution by choosing a sink from A_{S} randomly. This random selection is repeated until the network is double-covered or all candidate sinks have been chosen.

3.2 GRASP-MSP: Node-based Local Search

Let S be the set of sinks. We explore the neighbourhood of the current solution by adding a new sink $s \in A_{\mathrm{S}} \backslash S$ into S that can eliminate some existing sinks from S to minimise the total cost as possible. This move must always ensure that the network is double-covered.

3.3 GRASP-MSP: Algorithm Description

The GRASP-MSP pseudocode is given in Algorithm 1. It takes as input the original graph $G=(T\cup A_{\mathrm{S}}, E)$, the set T of sensor nodes, the set A_{S} of candidate sinks, the cost function c, the pre-computed $Distance_G$ table, the maximum acceptable path length $l_{\max}$, and the number of iterations (*max_iterations*). In each iteration, the construction phase to find the initial set of sinks S is executed in line 3. The local search starts with the initialisation of the best set and the best cost in line 6. The loop from line 7 to 22 searches for the best move, i.e. finding a new sink $r\in A_{\mathrm{S}}\setminus S$ that can eliminate as many existing sinks from S as possible. The loop from line 9 to 15 tries to find the set $Z\subseteq S$ of existing sinks that are safe to be removed after the insertion of r. The sinks in Z are safe to be removed if all sensor nodes in $H=(T\cup S\cup\{r\}\setminus Z, E{\downarrow}_{T\cup S\cup\{r\}\setminus Z})$ are double-covered. In line 16, we check if the new solution reduces the total cost of the current best solution. If the total cost can be reduced, we reset the set of the best set in line 17. If the total cost is the same, we keep this new solution in the set of the best set as shown from line 19 to 21.

When all moves have been evaluated, we check in line 23 if an improving solution has been found. If the moves produce a better solution, the set of sinks S is updated in line 24 by selecting one best set randomly from the set of the best set. Then, the local search continues. If at the end of the local search we find a better solution compared to the best solution found so far, we update in line 29 the set of sinks and the lowest total cost found. The best sink set S^* is returned in line 32.

Algorithm 1. GRASP-MSP

Input : $G, T, A_{\mathrm{S}}, c, Distance_G, l_{\max}, max\_iterations$
Output: S^*

1: $best\_value\leftarrow\infty$
2: **for** $i\leftarrow 1$ to *max_iterations* **do**
/* Construction phase */
3: Find S by choosing sinks from A_{S} randomly
4: **do**
5: $solution\_updated\leftarrow$ **false**
/* Local search phase */
6: $best\_set_0\leftarrow S$, $best\_cost\leftarrow\sum_{v\in S} c_v$, $best\_num\_set\leftarrow 1$
7: **for** all $r\in A_{\mathrm{S}}\setminus S$ **do**
8: $Z\leftarrow\emptyset$
9: **for** all $t\in S$ **do**
10: $Z\leftarrow Z\cup\{t\}$, $H=(T\cup S\cup\{r\}\setminus Z, E{\downarrow}_{T\cup S\cup\{r\}\setminus Z})$
11: Find *num_uncovered* in H using $Distance_G$ and $l_{\max}$
12: **if** $num\_uncovered>0$ **then**
13: $Z\leftarrow Z\setminus\{t\}$
14: **end if**
15: **end for**
16: **if** $\sum_{v\in S\cup\{r\}\setminus Z} c_v < best\_cost$ **then**
17: $best\_num\_set\leftarrow 0$
18: **end if**
19: **if** $\sum_{v\in S\cup\{r\}\setminus Z} c_v \le best\_cost$ **and** $S\cup\{r\}\setminus Z\notin best\_set$ **then**
20: $best\_set_{best\_num\_set}\leftarrow S\cup\{r\}\setminus Z$, $best\_cost\leftarrow\sum_{v\in S\cup\{r\}\setminus Z} c_v$, $best\_num\_set$++
21: **end if**
22: **end for**
23: **if** $best\_cost<\sum_{v\in S} c_v$ **then**
24: $S\leftarrow$ select a set randomly from *best_set*
25: $solution\_updated\leftarrow$ **true**
26: **end if**
27: **while** *solution_updated*
/* Best solution update */
28: **if** $\sum_{v\in S} c_v < best\_value$ **then**
29: $S^*\leftarrow S$, $best\_value\leftarrow\sum_{v\in S} c_v$
30: **end if**
31: **end for**
32: **return** S^*

3.4 Evaluation of GRASP-MSP

In the evaluation, we want to show that GRASP-MSP is effective and efficient in finding the lowest cost sink deployment compared to other closely-related algorithms and the optimal solution. To measure the performance of the algorithms, we use cost and runtime metrics. Cost measures the total cost of sinks, while runtime is the algorithm's total execution time.

The results are based on the mean value of 20 randomly generated network deployments. The network consists of 100 sensor nodes deployed within randomly perturbed grids, where a sensor node is placed in a unit grid of 8m $\times$ 8m and the coordinates are perturbed. To get sparse networks (average degree 2-3), we generate more grid points than the number of nodes. We use 11×11 grids to randomly deploy 100 sensor nodes. 25 candidate sinks are also distributed in a grid area, where a candidate occupies a unit grid of 18m $\times$ 18m. Both sensor nodes and sinks use 10 metres transmission range.

We compare GRASP-MSP against *minimise the number of sinks for fault-tolerance* (MSFT) algorithm, *cluster-based sampling for multiple sink placement* (CBS-MSP), the greedy version of multiple sink placement (Greedy-MSP) and the optimal solution. MSFT and CBS-MSP are algorithms based on the well-known k-means clustering algorithm. MSFT is similar to [8]. In [8], sinks can be deployed anywhere and they are added one by one in the network until a required lifetime is met. Unlike [8], MSFT has candidate locations and keeps adding sinks until the network is double-covered. CBS-MSP is similar to *cluster-based sampling* (CBS) algorithm proposed in [3], but with some modification. In CBS, the number of sinks is given and the objective is to minimise the total road distance from all nodes to the sinks where each node is required to be double-covered. Unlike CBS, CBS-MSP tries to reduce the number of sinks and thus the deployment cost. We implement CBS-MSP using path length to represent distance between two nodes and also we have path-length restrictions. In each iteration, both CBS and CBS-MSP try to find the best sink locations to ensure the network is double-covered. k-means clustering algorithm is used in these algorithms to divide the network into clusters and to find the position of each sink, which is in the centre of a cluster. The performances of MSFT and CBS-MSP depend on the randomly selected sink locations. Therefore, we use the maximum iteration (*MaxIter*) to limit the number of iterations. Greedy-MSP is similar to [11], but it considers double-covered networks. Greedy-MSP deploys sinks one by one until the network is double-covered. In each iteration, the greedy move picks the best sink that can minimise the number of uncovered nodes as possible. In Greedy-MSP, if two or more moves offer the same solution, we select one arbitrarily. We also try to evaluate them all, which we call Greedy-MSP-All.

The optimal solution is modeled using binary linear programming with the objective is to minimise the total sink cost, i.e.

$$\min \sum_{j\in S} c_j x_j \tag{4}$$

subject to the following constraints

$$\sum_{j\in S} l_{ij} x_j \ge 2;\ \forall i\in T \tag{5}$$

$$d_{ij}\le l_{\max}\Rightarrow l_{ij}=1,\ d_{ij}>l_{\max}\Rightarrow l_{ij}=0;\ i\in T,\ j\in S \tag{6}$$

$$x_j\in\{0,1\};\ j\in S \tag{7}$$

c is the cost of a candidate sink x. The first constraint guarantees each sensor node has at least two paths to two sinks. The paths are length-bounded, which are shown in the second constraint using a binary

function l_{ij}. It has value equal to one if the shortest path length from a sensor node i to a sink $j \leq l_{\max}$, otherwise its value is zero. In the third constraint, a candidate sink is either selected to be deployed or not. The binary linear programming for the optimal solution is implemented in Matlab, while the other algorithms are written in C++. Tests are carried out in 2.40 GHz Intel Core2 Duo CPU with 4 GB of RAM.

In the simulation, we find the best locations to deploy the least number of sinks to make the networks double-covered. We consider the cases where the maximum acceptable path length from each sensor node to a sink is 6 and 10. The number of sinks deployed by each algorithm is shown in Figure 2 and the runtime is in Table 1. The simulation results show that GRASP-MSP with *max_iterations* $= 10$ requires the same number of sinks with shorter runtime compared to the optimal solution. It also outperforms MSFT, CBS-MSP and Greedy-MSP. Greedy-MSP has the shortest runtime, but it places more sinks compared to GRASP-MSP. At this stage, finding the optimal solution is sufficient for the multiple sink placement problem since the runtime issue is not a problem in offline algorithms. However, using local search technique to later solve the multiple sink and relay placement problem is fully justified by GRASP-MSP as it provides the same result as the optimal solution and even faster.

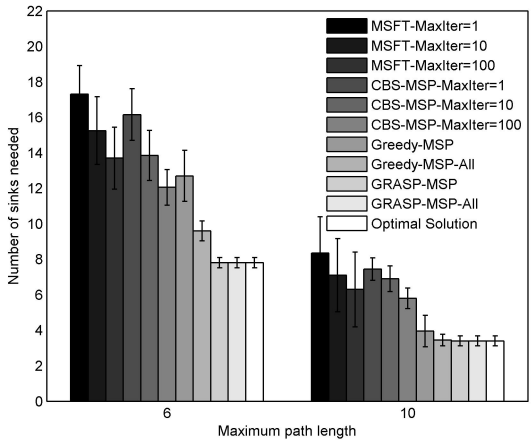


Figure 2. Number of sinks needed for multiple sink placement algorithms versus maximum path length

Table 1. Multiple sink placement algorithms' runtime

Algorithms	Runtime (sec)	
	$l_{\max}=6$	$l_{\max}=10$
MSFT-MaxIter=1	0.4188	0.0288
MSFT-MaxIter=10	4.0429	0.3102
MSFT-MaxIter=100	40.8313	3.1647
CBS-MSP-MaxIter=1	1.0297	0.0235
CBS-MSP-MaxIter=10	10.1797	0.2069
CBS-MSP-MaxIter=100	97.0899	2.0844
Greedy-MSP	0.0024	0.0040
Greedy-MSP-All	7.9664	0.0071
GRASP-MSP	0.0688	0.0452
GRASP-MSP-All	0.1305	0.0750
Optimal Solution	0.0727	0.0867

4 MULTIPLE SINK AND RELAY PLACEMENT (MSRP)

For the sink and relay placement, nodes are partitioned into a set of sensors T, relays R and sinks S. In the graph representation, $V = T \cup R \cup S$. We identify a sensor node to be *critical* if and only if upon its removal, more sensor nodes will have no path of length $\leq$

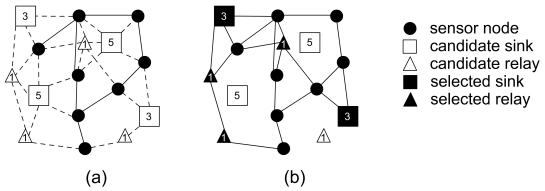


Figure 3. (a) A WSN with 4 candidate sinks and 4 candidate relays, and (b) the double-covered and noncritical WSN where $l_{\max} = 3$

$l_{\max}$ to a sink. Otherwise, it is *noncritical*. We define a WSN to be noncritical if each sensor node $v \in T$ is noncritical. In the *multiple sink and relay placement* problem, given a graph $G = (T \cup A_{\mathrm{R}} \cup A_{\mathrm{S}}, E)$, where A_{R} and A_{S} are sets of candidate locations for relays and sinks, respectively, we find minimum cost subsets $R \subseteq A_{\mathrm{R}}$ and $S \subseteq A_{\mathrm{S}}$ such that $H = (T \cup R \cup S, E \downarrow_{T \cup R \cup S})$ is double-covered and noncritical. The relay and sink candidate locations are associated with a non-negative cost function $c : A_{\mathrm{R}} \cup A_{\mathrm{S}} \rightarrow \mathbb{R}$. We assume that a relay is cheaper than a sink because sinks usually are assumed to be powered and have WiFi/ethernet backhaul. The multiple sink and relay placement problem is illustrated in Figure 3, where the numbers represent the devices' costs.

We develop GRASP-MSRP to solve the multiple sink and relay placement problem. To identify critical nodes, GRASP-MSRP uses length-constrained connectivity and rerouting centrality (l-CRC) proposed in [10]. A node is critical if its centrality index is above a threshold. We can raise the threshold to trade off the deployment cost against the robustness of the network. However, in this paper, we only assume zero threshold for full reliability.

4.1 GRASP-MSRP: Construction Phase

In the construction phase, we find R and S as our initial sets of relays and sinks, respectively. We need at least two sinks for a double-covered WSN, so we firstly choose two sinks randomly from A_{S}. We then deploy relays from A_{R} to minimise the number of uncovered and critical nodes. If a sensor node $v \in T$ is uncovered, we try to place some relays to construct a path to a sink $w \in S$ if *Distance*$_H(v, w) > l_{\max}$ but *Distance*$_G(v, w) \leq l_{\max}$. We choose the relays that appear on the shortest path from v to w by tracing the path in *Parent*$_G(v, w)$. If the sensor node needs two paths to make it double-covered, this step is repeated twice. If a sensor node $v \in T$ is critical, we deploy relays that appear on the shortest path from each descendant of v to a sink $w \in S$ bypassing v, as long as the shortest path length is $\leq l_{\max}$. We basically alternate the sink and relay addition during this process. However, we do not add more sinks if at some points the network is already double-covered. If the problem has a feasible solution, the network is double-covered and noncritical at the end of the construction phase.

4.2 GRASP-MSRP: Node-based Local Search

Let R be the set of relays and S be the set of sinks. We look for a lower cost solution by adding either a new relay $r \in A_{\mathrm{R}} \setminus R$ into R or a new sink $s \in A_{\mathrm{S}} \setminus S$ into S that can eliminate some existing relays from R and sinks from S to minimise the total cost as possible. Given that the cost of a sink is higher than the cost of a relay, we also try to minimise the total cost by adding some relays into R when we eliminate an existing sink from S. The local search moves are performed to reduce the total cost, but must ensure that the network is still double-covered and noncritical in each iteration.

4.3 GRASP-MSRP: Algorithm Description

The GRASP-MSRP pseudocode is given in Algorithm 2. Its concept is similar to GRASP-MSP in Algorithm 1 with some differences. We will only describe its differences. Firstly, GRASP-MSRP takes the set of candidate relays A_{R} as one of its input. Secondly, the shortest paths from all sensor nodes to all sinks are computed several times in line 12, 18 and 24 due to the addition and elimination of relays.

Algorithm 2. GRASP-MSRP

```
Input  : G, T, A_R, A_S, c, l_max, max_iterations
Output: R*, S*
 1:  best_value ← ∞
 2:  for i ← 1 to max_iterations do
                                              /* Construction phase */
 3:      Find initial R and S,  W ← R∪S
 4:      do
 5:          solution_updated ← false
                                              /* Local search phase */
 6:          best_set_0 ← W,  best_cost ← Σ_{v∈W} c_v,  best_num_set ← 1
 7:          for all r ∈ A_R ∪ A_S \ W do
 8:              Y ← {r},  Z ← ∅,  X ← ∅
 9:              for all t ∈ W ∪ X do
10:                  Z ← Z ∪ {t},  X ← ∅
11:                  H = (T ∪ W ∪ Y \ Z, E↓_{T∪W∪Y\Z})
12:                  Calculate Distance_H
13:                  Find uncovered_set in H using Distance_H and l_max
14:                  if |uncovered_set| > 0 then
15:                      X ← X ∪ {Find relays to minimise |uncovered_set|}
16:                  end if
17:                  H = (T ∪ W ∪ X ∪ Y \ Z, E↓_{T∪W∪X∪Y\Z})
18:                  Calculate Distance_H
19:                  Find critical_set in H using Distance_H and l_max
20:                  if |critical_set| > 0 then
21:                      X ← X ∪ {Find relays to minimise |critical_set|}
22:                  end if
23:                  H = (T ∪ W ∪ X ∪ Y \ Z, E↓_{T∪W∪X∪Y\Z})
24:                  Calculate Distance_H
25:                  Calculate num_uncovered and num_critical in H
                     using Distance_H and l_max
26:                  if num_uncovered = 0 and num_critical = 0 then
27:                      Y ← Y ∪ X,  Z ← Z \ X
28:                  end if
29:                  if num_uncovered > 0 or num_critical > 0 then
30:                      Z ← Z \ {t}
31:                  end if
32:              end for
33:              if Σ_{v∈W∪Y\Z} c_v < best_cost then
34:                  best_num_set ← 0
35:              end if
36:              if Σ_{v∈W∪Y\Z} c_v ≤ best_cost and W ∪ Y \ Z ∉ best_set then
37:                  best_set_{best_num_set} ← W ∪ Y \ Z,
                     best_cost ← Σ_{v∈W∪Y\Z} c_v,
                     best_num_set++
38:              end if
39:          end for
40:          if best_cost < Σ_{v∈W} c_v then
41:              W ← select a set randomly from best_set
42:              solution_updated ← true
43:          end if
44:      while solution_updated
                                              /* Best solution update */
45:      if Σ_{v∈W} c_v < best_value then
46:          R* ← ∅,  S* ← ∅
47:          for all v ∈ W do
48:              if v ∈ A_R then
49:                  R* ← R* ∪ {v}
50:              else
51:                  S* ← S* ∪ {v}
52:              end if
53:          end for
54:          best_value ← Σ_{v∈W} c_v
55:      end if
56:  end for
57:  return R*, S*
```

In line 13, the algorithm checks for uncovered nodes. If some exist, it tries to deploy relays in line 15. The identification of critical nodes is performed in line 19. If some exist, relays are added in line 21. Note that we try to minimise the total cost by adding some relays when we eliminate a sink. These relays are saved in X as shown in line 15 and 21, which later will be included in Y, the set of new relays and sinks to be inserted, if X helps the network become double-covered and noncritical. The network is checked if it is double-covered and noncritical in line 25. The rest of the pseudocode has similar role to GRASP-MSP's.

4.4 Evaluation of GRASP-MSRP

We evaluate the total deployment cost and the runtime of GRASP-MSRP against *minimise the number of sinks and relays for fault-tolerance* (MSRFT) algorithm, *cluster-based sampling for multiple sink and relay placement* (CBS-MSRP) and the greedy version of multiple sink and relay placement (Greedy-MSRP). MSRFT, CBS-MSRP and Greedy-MSRP use MSFT, CBS-MSP and Greedy-MSP, respectively, to find the best locations to deploy sinks and GRASP-ABP [10] to deploy relays. These three algorithms start by finding the best locations for two sinks before utilising GRASP-ABP [10] to deploy relays. The number of sinks is gradually increased and GRASP-ABP is used to deploy relays until the network becomes double-covered and noncritical.

We follow the same simulation setting as for the multiple sink placement problem in Section 3.4. In addition, we have 81 candidate relays distributed evenly in a grid area. A candidate relay occupies a unit grid of 10m $\times$ 10m. In the simulation, we only use 100 as the maximum iteration (*MaxIter*) for MSRFT and CBS-MSRP. We also use different sink costs (c_{S}), i.e. 3, 6, randomly between 3 and 6, and 10 units, while relay cost is 1 unit. The total sink and relay deployment cost suggested by each algorithm with $l_{\max} = 6$ is shown in Figure 4 and the runtime is in Table 2. The results show that GRASP-MSRP with *max_iterations*$= 10$ has the lowest total cost compared to other algorithms. Although GRASP-MSRP's runtime is the longest, this is acceptable since the deployment planning is an offline process, which is carried out during the initial design phase.

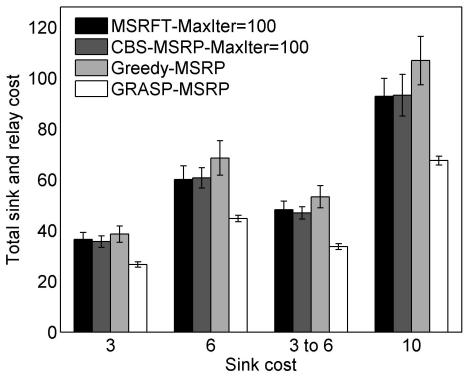


Figure 4. Total cost for multiple sink and relay placement algorithms versus sink cost

Table 2. Multiple sink and relay placement algorithms' runtime

Algorithms	Runtime (sec)			
	$c_{\mathrm{S}} = 3$	$c_{\mathrm{S}} = 6$	$c_{\mathrm{S}} = 3-6$	$c_{\mathrm{S}} = 10$
MSRFT-MaxIter=100	61.9235	60.3157	60.1228	59.2399
CBS-MSRP-MaxIter=100	61.2929	64.0727	62.0789	61.9352
Greedy-MSRP	153.7541	146.8796	130.1220	141.8679
GRASP-MSRP	196.5039	216.4508	251.2635	228.3422

The numbers of deployed sinks and relays for GRASP-MSRP are depicted in Figure 5. The results show that more sinks are exchanged with relays when the sink cost increases to reduce the total deployment cost. We also simulate GRASP-MSRP with $l_{\max} = 10$ as shown in Figure 6. When we increase $l_{\max}$, the number of sinks decreases significantly but the number of relays does not increase much. The simulation runtime for $l_{\max} = 6$ is 196.5039 seconds and for $l_{\max} = 10$ is 348.6041 seconds.

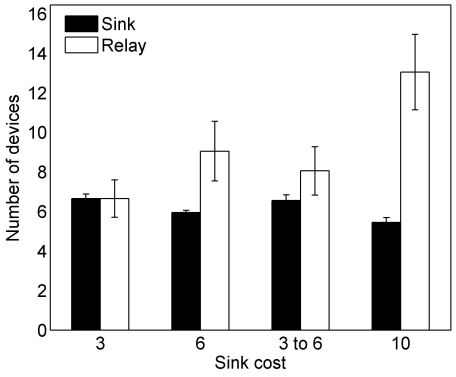


Figure 5. Total numbers of sinks and relays for GRASP-MSRP versus sink cost

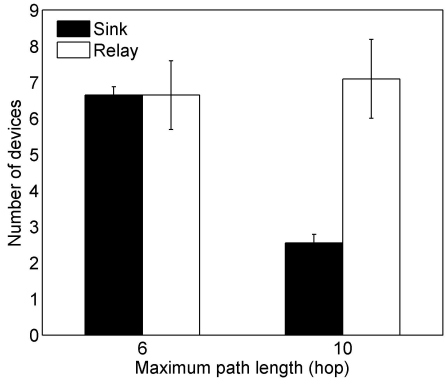


Figure 6. Total numbers of sinks and relays for GRASP-MSRP versus maximum path length

5 CONCLUSION

We have studied the WSN deployment planning problem where the aim is to protect the network against one single failure, of either a sink or a sensor node. We design a network to be double-covered and noncritical. Double-covered means each sensor node must have at least two paths with acceptable length to two sinks. Noncritical means all sensor nodes must have length-bound paths to sinks when an arbitrary sensor node fails. We propose GRASP-MSP and GRASP-MSRP, local search algorithms based on the GRASP technique to minimise the total cost of deployment. GRASP-MSP solves the multiple sink placement problem by ensuring that each sensor node in the network is double-covered. We demonstrate empirically that it achieves the same deployment cost as the optimal solution with shorter runtime. GRASP-MSP's simulation results justify the use of local search to solve the multiple sink and relay placement problem, where linear solution is not available. GRASP-MSRP tries to optimise the multiple sink and relay placement problem. Compared to the most closely-related algorithms, GRASP-MSRP sacrifices runtime to achieve the lowest total cost. This is not a significant problem in the network deployment planning because it is an offline process. We are currently investigating the performance of the topologies generated by our proposed algorithms using a network simulator.

ACKNOWLEDGEMENTS

This research is fully funded by the NEMBES project, supported by the Irish Higher Education Authority PRTLI-IV research program.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, 'Wireless Sensor Networks: A Survey', *Computer Networks*, **38**(4), 393–422, (Mar. 2002).

[2] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, 'Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance', in *Proc. 6th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, pp. 309–319, (May 2005).

[3] H. Cambazard, D. Mehta, B. O'Sullivan, L. Quesada, M. Ruffini, D. Payne, and L. Doyle, 'A Combinatorial Optimisation Approach to the Design of Dual-Parented Long-Reach Passive Optical Networks', in *Proc. 23rd IEEE Int'l Conf. Tools with Artificial Intelligence (ICTAI'11)*, pp. 785–792, (Nov. 2011).

[4] T. A. Feo and M. G. C. Resende, 'Greedy Randomized Adaptive Search Procedures', *Journal of Global Optimization*, **6**, 109–133, (1995).

[5] X. Han, X. Cao, E. L. Lloyd, and C. C. Shen, 'Fault-tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks', *IEEE Transactions on Mobile Computing*, **9**(5), 643–656, (May 2010).

[6] S. Mahmud, H. Wu, and J. Xue, 'Efficient Energy Balancing Aware Multiple Base Station Deployment for WSNs', in *Proc. 8th European Conf. Wireless Sensor Networks (EWSN'11)*, pp. 179–194, (Feb. 2011).

[7] S. Misra, S. D. Hong, G. Xue, and J. Tang, 'Constrained Relay Node Placement in Wireless Sensor Networks to Meet Connectivity and Survivability Requirements', in *Proc. 27th Ann. IEEE Conf. Computer Communications (INFOCOM'08)*, pp. 281–285, (Apr. 2008).

[8] E. I. Oyman and C. Ersoy, 'Multiple Sink Network Design Problem in Large Scale Wireless Sensor Networks', in *Proc. IEEE Int'l Conf. Communications (ICC'04)*, pp. 3663–3667, (Jun. 2004).

[9] J. Pu, Z. Xiong, and X. Lu, 'Fault-Tolerant Deployment with *k*-connectivity and Partial *k*-connectivity in Sensor Networks', *Wireless Communications and Mobile Computing*, **9**(7), 909–919, (May 2008).

[10] L. Sitanayah, K. N. Brown, and C. J. Sreenan, 'Fault-Tolerant Relay Deployment Based on Length-Constrained Connectivity and Rerouting Centrality in Wireless Sensor Networks', in *Proc. 9th European Conference on Wireless Sensor Networks (EWSN'12)*, pp. 115–130, (Feb. 2012).

[11] X. Xu and W. Liang, 'Placing Optimal Number of Sinks in Sensor Networks for Network Lifetime Maximization', in *Proc. IEEE Int'l Conf. Communications (ICC'11)*, (Jun. 2011).

[12] W. Youssef and M. Younis, 'Intelligent Gateway Placement for Reduced Data Latency in Wireless Sensor Networks', in *Proc. IEEE Int'l Conf. Communications (ICC'07)*, pp. 3805–3810, (Jun. 2007).