Hoz, E. de la, Gimenez-Guzman, J. M., Marsa-Maestre, I., Cruz-Piris, L., Orden, D., 2017, "A distributed, multi-agent approach to reactive network resilience", Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)

A Distributed, Multi-Agent Approach to Reactive Network Resilience

Enrique de la Hoz,
Jose Manuel Gimenez-Guzman,
Ivan Marsa-Maestre and Luis Cruz-Piris
Computer Engineering Department
University of Alcala
enrique.delahoz@uah.es,
josem.gimenez@uah.es, ivan.marsa@uah.es,
luis.cruz@uah.es

David Orden
Department of Physics and Mathematics
University of Alcala
Alcala de Henares, Spain
david.orden@uah.es

ABSTRACT

Critical network infrastructures are communication networks whose disruption can create a severe impact on other systems. Multi-agent systems have been successfully used to protect critical network infrastructures, with approaches ranging from reasoning about secure design and policies to multi-agent intrusion detection systems (IDS). However, there is little research on the possibilities for multi-agent systems to react to known intrusions. In this paper we propose a multi-agent framework for *reactive network resilience*, that is, to allow a network to reconfigure itself in the event of a security incident so that the risk of further damage is mitigated. The proposed framework takes advantage of a risk model based on multilayer networks and of distributed belief propagation techniques to agree on a new, more resilient configuration of the network in the event of an attack. We compare our proposal with a number of centralized optimization and multi-agent negotiation techniques. Experiments show that our proposal outperforms the reference approaches both in terms of risk mitigation and performance

Keywords

Critical network infrastructures, reactive resilience, multi-agent belief propagation

1. INTRODUCTION

Critical network infrastructures (CNIs) are those communication network infrastructures whose disruption, intentional or accidental, may have a high impact because of either its massive, even global, deployment (e.g., Internet, cell provider networks) or its supporting role for other critical infrastructures (e.g., the network for the internal communications of a control system in a nuclear facility, or the communication network for the electrical grid). The protection of critical network infrastructures is a top priority in the agendas of our governments and critical service operators.

In the last years, multi-agent systems have emerged as a successful approach for addressing complex security problems [1][2][3]. Most approaches deal with secure design and policing (e.g. how to design a system which is as secure/resilient as possible with the available resources, or how to allocate resources for surveillance maximizing effectiveness). In the particular case of critical network infrastructures, one of the most promising areas of research are intrusion detection systems (IDS). However, the use of multi-agent systems to react to such intrusions is still an emergent field.

In this work, we address reactive network resilience, which extends the concept of network resilience [4] to cover the ability of a network to reconfigure itself in the event of a security incident so that the risk of further damage is mitigated. In particular, we propose a distributed, multi-agent framework for reactive network reconfiguration in (semi-)virtualized CNIs over a zero-day attack threat model. This assumes that the specific vulnerabilities in the attack are unknown to the network operators, and therefore can only be detected by using an anomaly-based IDS. The framework will use the alert report from such an IDS and an *a priori* risk analysis model of the network to propose a new network configuration (that is, an alternative redeployment of the different components in the network) to minimize the impact of the incident. A more in-depth discussion of the most relevant related works, with an emphasis on zero-day resiliency, can be found in Section 2.

Our goal is, therefore, to build a multi-agent reactive risk-mitigation framework for critical network infrastructures. The paper contributes to this goal in the following ways:

- We propose a risk model based on multi-layer networks, which takes into account both conventional risk analysis metrics and novel zero-day resilience metrics (Sections 3.1 and 3.2).
- We propose a novel perspective of reactive redeployment management as a multi-agent negotiation, with a strategy based on a variation of the well-known graph coloring problem (Section 3.3).
- We propose an agreement determination approach based on distributed belief-propagation to collectively find a redeployment of the network upon a security incident alert (Section 3.4).

To validate our approach and evaluate the impact of our contributions, we have conducted a set of experiments over a range of randomly-generated scenarios (Section 4). Results show our approach outperforms the reference techniques both in terms of risk mitigation and performance. The last section summarizes our contributions and sheds light on future lines of research.

2. RELATED WORK

In this section we briefly review the most relevant works related to our proposal. We first discuss our threat scenario, resilience

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

against zero-day attacks, and then we discuss some of the most relevant and closely related multi-agent approximations to security and network resilience in the literature. Finally, we frame the application context of our approach within (semi-)virtualized network infrastructures

2.1 Network resilience under zero-day attacks

Zero-day attacks are cyber-attacks which are performed by using a vulnerability or attack vector which is not publicly-known (and generally not known by the software/hardware vendor) [5]. While for already known attacks the solution will be probably found in the use of firewalls, antivirus or software patching and upgrading, the mitigation of zero-day attacks is a harder task due to the lack of information. In the literature, the work related to zero-day attacks comprises two different fields. The first one is the detection of zero-day attacks [6][7]. The second one, more related to this work, is devoted to the development of metrics for evaluating the resilience of a network against zero-day attacks. The proposal of metrics is of vital importance to network security, as "you cannot improve what you cannot measure" [8]. Network security metrics usually assign scores to vulnerabilities according to their probability and/or impact. A review and performance comparison of such metrics can be found in [9]. However, this type of metrics is not valid for zero-day attacks, due to the lack of knowledge and information about them. For that reason, there are some works [8][10] that propose metrics specifically suited for zero-day attacks. In [10], authors consider network diversity as a metric for evaluating resilience. Similarly,in [8] authors propose a metric based on counting how many distinct zero-day vulnerabilities are required to compromise a network asset.

Another interesting set of related works, but not specially devoted to zero-day attacks, are the pioneering works [11] and [12], where authors analyze the robustness of complex networks under localized attacks for determining how much damage a network can sustain before it collapses. The differences between these works and our work are clear, as these papers are more theoretical and we consider more realistic computer network models. This realistic model takes advantage of the fact that we model the problem by means of a multi-layer graph, to capture the features of real-world networks. As research on complex networks has evolved, it has been clearer the need of going further than monolayer graph modelling and exploring more realistic and complex models. Multiplex or multi-relational networks connect nodes using links that can express different kinds of relationships [13]. Multilevel networks and meta-networks enable also hierarchical structures and node and links of different types [14]. Recently, Kivela [15] has presented a unified modelling for multi-layer networks that include these concepts in a unified manner that takes advantage of the different mathematical tools available in the state of the art.

In addition to all the aforementioned and directly related fields, there are some important works in different areas that can be related to the study of resilience in complex networks. First, mobile ad-hoc networks and vehicular ad-hoc networks (MANET and VANET). In these networks, nodes are constantly moving, and, accordingly, the connectivity must evolve to deal with this. For these scenarios, it is critical to guarantee service continuity under this dynamicity [16]. Second, delay tolerant networks [17], where service interruption after power failures, attacks or node dispersion are taken into account, has attracted much interest and its study has been promoted by DARPA. And last, wireless sensor networks, where it is necessary to create resilient network topologies able to provide connectivity even after some sensors stop contributing as a result of a battery outage [18].

2.2 Multi-agent approximations to security and critical infrastructures

As inherently connected and interdependent complex systems, critical infrastructures are not alien to multi-agent modelling [19]. In fact, a number of successful approaches to security in these environments have emerged in the last years. Of particular interest are the approximations based on the idea of "security games" [3][1]. These works use computational game theory to build decision support systems for efficient security resource allocation in surveillance scenarios (e.g. airport security), by modelling the allocation process as a Bayesian Stackelberg game [1]. In domains more related to communication networks, we can find game-theoretic works regarding intrusion detection, especially for distributed systems [20] and cyber-physical systems [21]. However, there is little body of work about how to react to such intrusions once they are detected. This is a particularly critical matter, since intrusions in CNIs can easily lead to cascading failures. The most closely related work to our own is probably the one in [22], although with some key differences. First, authors address a different critical infrastructure domain (the power grid), but with the same rapid cascading failure scenario we study. Second, they use a game-theoretic approach which has serious complexity and scalability issues, as the authors acknowledge. On the contrary, in this work we present an approach based on heuristic, multi-agent negotiation. Negotiation-based approaches have been proven successful for many scenarios where complexity and scalability issues made game-theoretic approaches not suitable [23][24][25].

2.3 (Semi-)virtualized networking

The approach described in this paper takes into account the ongoing transformation of datacenters and datacenter networks, driven by virtualization, automation, and containers. Virtualization has transformed the way in which we design and manage servers and datacenters. This has led to increasing efficiencies and a trend towards automation. Virtualization along with configuration management tools, such as Puppet, Chef or Ansible [16], has led to a new approach known as programmable infrastructure or Infrastructure as Code (IaC) [26]. Infrastructure as Code is an approach to infrastructure automation, based on practices from software development, that emphasizes consistent, repeatable routines for provisioning and changing systems and their configuration. IaC can be defined as "the process of managing and provisioning computing infrastructure (processes, bare-metal servers, virtual servers, etc.) and their configuration through machine-processable definition files, rather than physical hardware configuration or the use of interactive configuration tools". This way, IT infrastructure supports and enables change, and users are able to define, provision, and manage the resources they need. Changes to the running systems become routine and improvements can be made continuously: there is a document describing the deployment and introducing changes to the deployment is just a matter of modifying that document and applying the modification by using the proper provisioning and configuration management tools. It is important to notice that this is not only limited to server management but to network creation, configuration and management, thanks to approaches and technologies such as Network Function Virtualization (NFV) and SDN [27]. Finally, for working with large distributed infrastructures, the use of container technologies, such as Docker [28], is also increasing. Containers are isolated places where an application can run without affecting the rest of the system, and without the system affecting the application [29]. By using containers, it is possible to isolate pieces of your system into separate containers. For instance, you can have a container for Apache, a container

for MySQL, and one for MongoDB. The underlying idea is being able to create your distributed system by combining containers that provide the functionality that you need. This allows to isolate individual elements of the application into independent units that can be managed in a flexible manner. Containers can be created and distributed by means of public or private repositories [30], just like code is distributed in places such as Github or Bitbucket. Container technology represents the technology foundation for the component layer that we propose in this work.

3. A MULTI-AGENT FRAMEWORK FOR REACTIVE RESILIENCE IN CRITICAL INFRASTRUCTURES

In this section we describe our proposal for reactive resilience in (semi-)virtualized critical network infrastructures. The framework starts from the following environment assumptions:

- There is a weighted risk assessment of the system prior to the operation of our framework. This risk assessment comprises an inventory of system assets, along with their value and interdependencies.
- Assets rely on components to function. These components can be instantiated according to different configurations.
- The main threat considered are zero-day attacks, which are normally configuration-specific, that is, they affect a single configuration or a set of similar ones.
- Components are virtualized and can be deployed in different physical hosts along the physical network infrastructure, using a container paradigm as discussed above.
- There is a control plane where the deployment decisions are made. This control plane is in an overlay, isolated network, and thus it is assumed not to be under the discussed threats.
- There is an intrusion detection system (IDS) able to identify compromised components. We assume it to be an anomaly-detection based IDS, since zero-day attacks are by definition unknown, and therefore no additional information about the incident will be provided.

In the following, we describe our proposal, which is based on four key elements: a multilayer network model, a reactive resilience risk model, a multi-agent negotiation process for reactive redeployment, and a belief-propagation agreement approach.

3.1 Multilayer network model

As stated above, our framework will take risk assessment and alert reports and derive a new network configuration intended to maximize resilience. To achieve this goal, we will use a multilayer network model to capture the situational awareness input (i.e. actual risks due to a security incident) for the critical network infrastructure. Figure 1 shows an outline of the model.

The model defines three layers. The asset layer, on top, is extracted from risk analysis, and captures the relative importance of the assets and their interdependencies. The bottom layer is the infrastructure layer, which represents the actual infrastructure elements (hosts, links and network appliances) upon which the assets are deployed within the network. Between these two end layers, there is a component layer, which represent the components (e.g. databases, frontend elements) upon which assets depend on to provide their functionality. In the following, we describe in more detail the different layers and the information they provide to the model.

3.1.1 *Asset Layer*

As stated above, the asset layer is extracted from risk analysis, and is intended to capture the value of their different assets and the dependency relationships between them. This is represented as a weighted directed graph (Figure 2), where nodes represent assets and directed edges represent dependencies of a node (head) on another (tail). For instance, in the figure, asset a_2 depends on assets a_1 and a_3, asset a_4 depends on a_3, and assets a_5 and a_6 are mutually interdependent.

Node weights (e.g. w_{a_1}) are mandatory and represent the relative importance of a given asset for the organization from the point of view of risk analysis. Edge weights (e.g.$w_{a_1 \to a_2}$) are optional and represent the criticality of the dependency between two assets. There may be different ways to estimate the value of these weights, but in general they will represent the relative impact of the failure of one asset on the other. For instance, a weight $w_{a_1 \to a_2} = 0.5$ may represent that a failure of asset a_1 will reduce the value/performance of asset a_2 to 50%, or that if asset a_1 fails there is a probability of 50% of a cascading failure propagation to a_2.

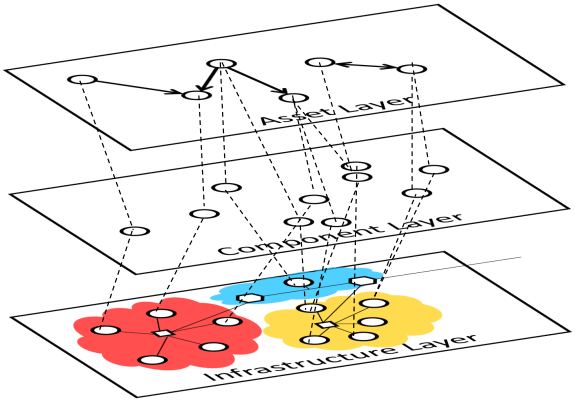

Figure 1: Multilayer network model.

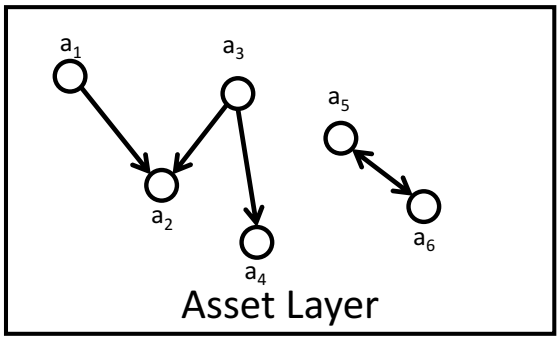

Figure 2: Asset Layer.

3.1.2 *Component Layer*

This layer captures the components (generally software) each asset relies on to provide its functionality and value to the organization. For instance, a given control application within the system may require to use a Web Service, or a database. Each component c_i is instantiated with a specific configuration γ_i (see Figure 3. That is, for a given component c_i (e.g., a web application), the configu-

ration γ_i represents the actual instantiation of that component (e.g., a PHP web application using a MySQL database).

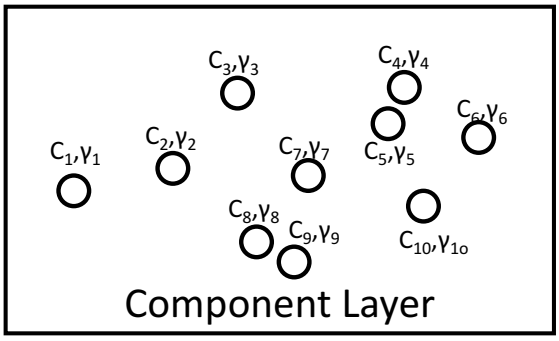

Figure 3: Component Layer.

Additional component layers can be included in the model to provide finer granularity in the description of the component configurations. For instance, the configuration for a given web application can be provided in a single layer, with possible values being e.g. LAMP (Linux, Apache, MySQL and PHP) or WAMP (Windows, instead of Linux). Another option could be to split the configuration description into four layers: operating system (Linux, Windows), database (MySQL, PostgreSQL), Web Server (Apache, Nginx) and Web Application framework (PHP, JSP). This allows a great expressiveness when describing the system.

3.1.3 Infrastructure layer

This layer represents the actual physical/logical network infrastructure in the system, that is, the actual hosts and network elements in the infrastructure. We assume a (partially) virtualized network, so that the topology of this layer may not represent an actual physical topology. This layer could be split into virtual and physical network layers as necessary, though. For the purpose of our resilience analysis, we will assume that the network infrastructure is divided into segments (e.g. DMZs, VLANs), and we will give them colors for convenience (Figure 4).

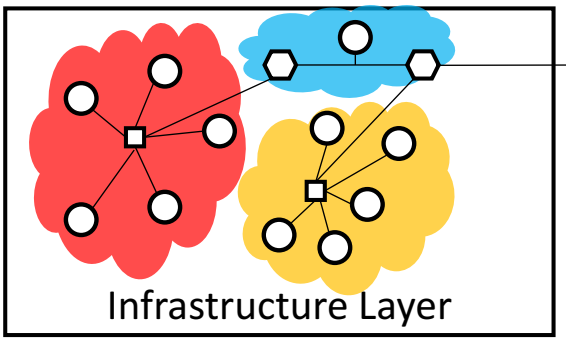

Figure 4: Infrastructure Layer.

3.2 Reactive resilience model

As stated above, we will deal mainly with zero-day resilience. Zero-days are normally configuration-specific (e.g. a zero-day exploit for Mac OS X 10.10.1), and they may lead to cascading attacks (e.g. when having adjacent firewalls which are all vulnerable to the same zero-day). Therefore, we need to take into account in the model the effect of potential vulnerabilities and attack chains, as a result of the interactions between configurations and between network segments.

To account for this, we will include two parameters:

- Cross-configuration zero-day vulnerability (ν_{ij}): this vulnerability value ν_{ij} accounts for the probability that a zero-day exploit over configuration γ_i would allow also to compromise configuration γ_j. Usually, this is directly related to configuration diversity [8]. For instance, given that LAMP and WAMP configurations are similar, they could be assigned a cross-configuration zero-day vulnerability value of 0.8.

- Cross-segment vulnerability (μ_{ij}): this vulnerability value μ_{ij} represents how easy would be for an attacker to send packets from segment s_i to segment s_j, provided that a machine in segment s_i has been compromised. This largely depends on firewall access control rules, though it can be generally assumed that adjacent segments will have higher cross-segment vulnerability values than distant ones.

Our framework is intended to act in response to a security incident alert. Such alert will correspond to anomalous behavior in a given component, host or network segment. Both machine-wide or segment-wide alerts can be generalized to component-wise alerts, by assuming a worst-case scenario where all components in the affected machine or network segment may have been compromised. Therefore, without loss of generality, we can assume that our framework reacts to compromises in components. If a component has been compromised, *attack paths* can be traced from this component to other components in the system, provided that there is a way an attacker could progress from one component to another, throughout the network and by compromising successive machines. Under the assumption of a zero-day attack, an attacker can progress through the network (pivot) if two conditions hold. First, a *network path* must exist between the compromised component and the component the attacker wants to pivot to. Second, enough similarity between source component and target component configurations must exist so that the zero-day attack is still effective. The extent to which these two conditions hold is given, respectively, by the appropriate vulnerability values μ_{ij} and ν_{ij}. To take this into account, we can "flatten" the multilayer network model into a *configuration-spread projection* from a given compromised component. Figure 5 shows an example of such a projection. In this example, the leftmost component has been compromised by a given zero-day, and the projection shows all components to which the zero-day attack could spread from that node. Two components c_i and c_j are connected if it is possible for the zero-day to spread between them (that is, there is enough similarity between configurations for the zero-day to be potentially effective in both components). The weight of the edge connecting these two nodes would be the corresponding cross-configuration vulnerability value ν_{ij}. In the figure, node color represents the network segment where a particular component instance has been deployed, according to the infrastructure layer.

With this projection in mind, an *attack path* from any node to another node is a loopless path between these two nodes. Any attack path P will have an associated *attack probability* or risk $\propto_P$, computed as the product of the edge weights in the path (given by ν_{ij}) and the *node traversal costs* given by cross-segment vulnerability value μ_{ij}, to account for the fact that components in "close" network segments will be easier to compromise for an attacker. That is, for a given attack path $P = (c_1, c_2, \ldots, c_N)$, the associated

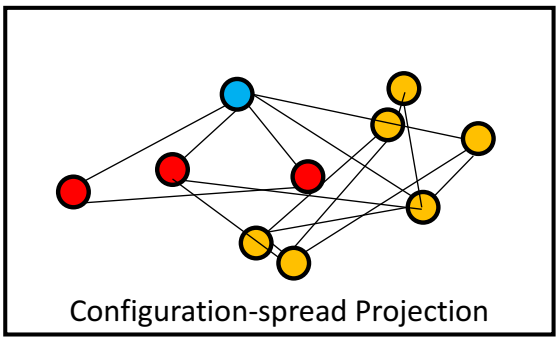

Figure 5: Configuration-spread projection of the model.

probability $\propto_P$ will be:

$$\propto_P = \prod_{i=1}^{N-1} \mu_{i,i+1} \cdot \nu_{i,i+1}$$

defined as a product in order to reflect the conditioned probabilities.

This allows us to estimate the worst-case overall risk over a given component c_i when we know c_j have been compromised as the risk of the highest probability attack path:

$$\rho_{c_i|c_j} = \max_{\forall P_{c_j \to c_i}} \propto_P$$

Given that every component c_i is associated to an asset a_i (we denote it as $c_i \to a_i$) through the links between the asset layer and the component layer, we can compute the inherent risk over a given asset after the compromise of c_j as

$$\rho_{a_i|c_j} = \sum_{k|c_k \to a_i} \rho_{c_k|c_j}$$

Taking into account the dependencies between assets $w_{a_k \to a_i}$, extracted from the asset layer, we can iteratively compute the aggregated risk over all assets , as follows:

- At iteration 0: $\overline{\rho_{a_i|c_j}}^0 = \rho_{a_i|c_j}$
- At iteration t: $\overline{\rho_{a_i|c_j}}^t = \rho_{a_i|c_j} + \sum_{k|a_k \to a_i} w_{a_k \to a_i} \overline{\rho_{a_k|c_j}}^{t-1}$

This computation is guaranteed to converge in at most d iterations, where d is the diameter of the graph induced by the asset layer, yielding a final value for the risk over each asset. From these risks and the asset weights w_{a_i}, we can finally derive the overall risk over the system as a result of the compromise of component c_j:

$$\overline{\rho_{c_j}} = \sum_{a_i} w_{a_i} \overline{\rho_{a_i|c_j}}$$

Therefore, the goal of our framework will be to produce, in the event of the compromise of a component c_j, an alternative configuration of the system which minimizes this risk $\overline{\rho_{c_j}}$.

It is important to note that this is not the only possible projection (or "flattening") of the multilayer network model. We have chosen it because it presents some interesting advantages for our purpose. First, the computation of $\rho_{c_i|c_j}$ can be thought as an analogy of the shortest path problem with non-negative real weights (taking $\frac{1}{v_{ij}}$ and $\frac{1}{\mu_{ij}}$ as weights), which is roughly solvable in *O(E+V)* time, which *E* and *V* being, respectively, the number edges and vertices of the configuration-spread projection. On the other hand, for a given graph structure of the projection, network reconfiguration after an attack reduces to $re-coloring$ the graph -that is, redeploying components in alternative infrastructure segments- so that the overall risk induced by the security incident is minimized. This will allow us to address the problem by distributed agent negotiation with local information only, as we will see in the following section.

3.3 Redeployment as multi-agent negotiation

From the projection discussed above, we can see that redeployment in reaction to an incident alert is a nonlinear optimization process involving finding the configuration which minimizes the overall risk $\overline{\rho_{c_j}}$ caused by the incident. Formally, we can define the reconfiguration problem as a tuple $\langle S^0, c_j \rangle$ where:

- $S^0 = \{s_i^0 \mid i = 1, \ldots, N\}$ is the initial configuration of the system, where each one of the N components c_i is deployed in segment s_i^0 in the infrastructure layer.
- c_j is the component which has been compromised, which immediately induces an initial risk $\overline{\rho_{c_j}}^0$.

A solution S to the problem $\langle S^0, c_j \rangle$ will be of the form $S = \{s_i \mid i = 1, \ldots, N\}$, giving each component c_i an alternate deployment segment s_i, and therefore inducing a new risk value $\overline{\rho_{c_j}}$, which we call *residual risk*.

General optimization of $\overline{\rho_{c_j}}$, as we will see in the evaluation, is a costly process in terms of computation time, which would not suit the real-time performance requirements of such a system. Therefore, it will be necessary to find approaches which can yield satisfactory solutions (even if sub-optimal) in a reasonable time. To accomplish this, we will adopt a conversion of the problem into multi-agent negotiation, with the following rationale:

- We assume a negotiating agent A_{a_i} per asset a_i. This agent will have the goal to minimize $\rho_{a_i|c_j}$, given the information it receives from the neighboring nodes in the multilayer network model (components and assets it depends on).
- We assume a negotiating agent A_{c_i} per component c_i. This agent will be entitled with the goal to minimize $\rho_{c_i|c_j}$, given the information it receives from the neighboring nodes in the configuration-spread projection (that is, the components with configurations similar enough to propagate zero-days to it).

With this conversion, we greatly reduce the computational complexity of the problem by distributing the computation among agents and by making them react only to local information. We need, however, to establish agent communication and decision making mechanisms for the negotiation to progress. We propose these mechanisms in the following section.

3.4 Distributed redeployment agreement based on belief propagation

Given that $\overline{\rho_{c_j}}$ depends on the maximum probability $\propto_P$ for all attack paths, and that the factors contributing to that probability are the cross-segment and cross-configuration vulnerability values of adjacent nodes in the configuration-spread projection, intuitively we need to minimize these vulnerability values in adjacent nodes. Roughly speaking, we need to avoid having highly similar configuration in highly reachable network segments, to difficult tracing successful attack paths through the network.

This is quite similar to the *Threshold Coloring Problem* (TSC) [31], which is depicted in Figure 6. In this problem, we have an undirected graph and a set of available colors (in the example, red,

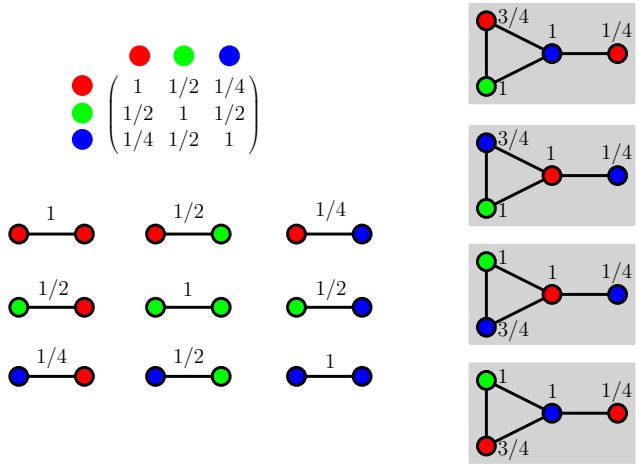

Figure 6: Example of the Threshold Spectrum Coloring Problem (TSC).

green and blue), with an associated interference matrix, which assigns an interference value for the occurrence of any pair of colors in any edge of the graph. The goal of the TSC problem is to find a coloring which minimizes the maximum interference per node (the optimal solutions for the example problem can be seen shadowed in the figure). Our hypothesis is that, by translating the problem of recoloring the configuration-spread projection (as we stated in the previous section) to this problem, we will find suitable resilient alternative topologies in a reasonable time. We will have as the available color set the different network segments s_i where we can deploy the different components, and as the color interference matrix the cross-segment vulnerability values μ_{ij}. We also have to augment the model to introduce edge weights, which will correspond to the cross-configuration vulnerability values ν_{ij} of the components represented by adjacent nodes.

Nonlinear optimization and negotiation techniques have been successfully applied to the TSC problem in other contexts [31], but they were centralized and used graph-wide information rather than local information, and therefore they are not applicable to the multi-agent setting described in the previous section (though they will provide useful benchmarks for evaluation). We need, therefore, a new technique, and here we propose a negotiation method based on distributed belief propagation.

Belief Propagation (BP) is a message-passing heuristic for solving optimization and inference problems in the context of a graphical model [32]. Under certain conditions, BP is able to find optimal solutions to factorized optimization problems, that is, optimization problems of the form

$$\text{minimize} \sum_{i \in V} \Phi_i(x_i) + \sum_{c \in C} \Psi_c(x_c)$$

$$\text{subject to } x_i \in \mathbb{R}, \forall i \in V,$$

where V is a finite set of variables and C is a finite collection of subsets of V representing constraints. Φ_i functions are called variable functions (they depend on the value of a single variable), and Ψ_c functions are called factor functions (they depend on specific combinations of variables called factors). All these functions are real-valued.

For the purposes of this paper, we need to translate our augmented Threshold Spectrum Coloring to a factorized optimization problem. To do this, we use our components as variables (which

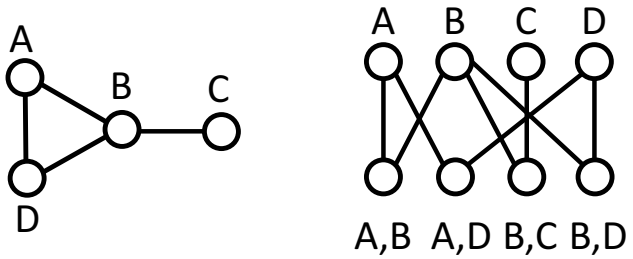

Figure 7: Factor graph F_P (right) for our example TSC problem.

can take different values depending on where they are redeployed) and the links between pairs of nodes (which represent similar nodes through which attack spreads may occur) as constraints. According to this, we define the corresponding functions as follows:

$$\Phi_i(s_i) = 0$$

$$\Psi_C(s_i, s_j) = \nu_{ij}\mu_{ij}, \forall C \equiv (i, j)$$

That is, we use a constant zero value for each variable function, and we multiply the cross-segment and cross-configuration vulnerability values for each factor function. With this formulation, we try to mitigate the impact of putting close very similar components. This is coherent with the risk model described in Section 3.2, where risks are restricted to those caused by attack paths from the compromised node. It is worth noting that this formulation differs from the TSC problem, given that here we try to minimize the sum of the contributions for all nodes in the graph, while pure TSC aims to minimize the maximum contribution for any single node in the graph. However, as shown in [31], sum minimization is a good heuristic to minimize the maximum in this context, and therefore successful techniques proposed for TSC can be used here as benchmarks.

Once the formulation has been established, we need to build the factor graph F of our problem, which is a bipartite graph with *variable nodes* in one side of the partition and *factor nodes* corresponding to the constraints in the other side of the partition. Links between both partitions occur between a constraint and the variable nodes it refers to. For instance, in the graph example given in Figure 6, the resulting factor graph F would be as shown in Figure 7.

Finally, we would have to apply the min-sum algorithm for BP [32], which is reproduced in Algorithm 1 for convenience.

The problem with applying directly the min-sum BP algorithm to our problem is that the algorithm only has correctness and convergence guarantees when the solution is unique and the factor graph is a tree. Although solution uniqueness can be achieved with randomized weights as suggested in [32], most of our scenarios do not create tree factor graphs. The usual junction tree technique used in machine learning to address this problem [33] is not applicable here, because we need computation to be distributed and use only local information.

Taking this into account, to ensure convergence and correctness of the algorithm, we propose to divide the factor graph into trees using a gossip-inspired technique [34]. The technique we propose works as follows:

- All component agents in the configuration-spread projection are initialized to the *unassigned* state, which means they do not belong to any tree.
- Nodes in *unassigned* state respond to the behaviour:

Algorithm 1: min-sum BP

Input : F: bipartite factor graph with edges (i, f) between variable nodes and factor nodes representing constraints
N: number of iterations
Z: $\{z_i\}$: available color set
Output: S: estimated optimal assignment
Initialize $t = 0$
foreach *edge* (i, f) *in* F **do**
  initialize $m^0_{f\to i}(z) \forall z \in Z$
end
for $t = 1, 2, \ldots, N$ **do**
  foreach *edge* (i, f) *in* F **do**
    update $m^t_{i\to f}(z) = \Phi_i(z) + \sum_{k\in f_i \setminus f} m^{t-1}_{k\to i}(z)$
    update $m^t_{f\to i}(z) =$
    $\min_{y\in C^{|f|}, y_i = z} \Psi_f(y) + \sum_{j\in f\setminus i} m^{t-1}_{j\to f}(y_j)$
  end
  $t = t + 1$
end
Set the belief function as
$b^N_i = \Phi_i(z) + \sum_{k\in f_i} m^N_{k\to i}(z)$ for each variable node i
Estimate the optimal assignment S as
$\hat{s}^{N(z)}_i$ for each variable node i

  - Decide with probability p whether to start a new tree (therefore changing their status to *assigned*) or to wait a random time
  - Upon receiving a message from an *assigned* neighbour (that is, a neighbour already belonging to a tree), switch to assigned status and acknowledge the membership to the tree.

- Nodes in *assigned* state respond to the behaviour:
  - Decide with probability p whether to invite a random subset of its (not already-invited) neighbors to its tree or to wait a random time.

This technique asynchronously divides the configuration-spread projection graph into a set of disjoint trees, from which tree factor graphs can be derived so that BP converges. Of course, when we work with the resulting set of trees, we lose the information about the influencing factors Ψ_{ij} corresponding to components c_i and c_j which are neighbors in the configuration-spread projection but have ended up in different trees. To minimize the impact of this simplification, we iteratively introduce this effect in the functions Φ_i of the frontier nodes (that is, the nodes in a tree which are neighbors of nodes in other trees). That is, the belief propagation process is repeated several times in an iterative manner, and at each iteration K the frontier nodes are assigned a variable function $\Phi^K_i(s_i)$ which is computed as follows:

$$\Phi^K_i(s_i) = \sum_{j\in\aleph(i)} \Psi_{ij}(s_i, \hat{s}^{K-1}_j)$$

Where $\aleph(i)$ is the set of neighbors of component c_i in the configuration-spread projection graph and $\hat{s}^{K-1}_j$ is the optimal assignment for neighbor c_j for the previous execution of the BP algorithm.

Computation of the Φ^K_i functions is performed at each corresponding component agent C_i. Computation of the Ψ_{ij} functions is randomly assigned to agents C_i and C_j to avoid agent manipulation of the belief propagation process. Asset agents A_i make a final vote after convergence based on their perceived risk mitigation, with each agent's vote being weighted by the asset weigh w_{a_i}. A configuration is accepted if it receives more favorable than negative votes (after weighting). If a configuration is rejected, the whole process starts again.

4. EVALUATION

In this section we describe the experiments we have conducted to validate our contributions and we discuss the results obtained.

4.1 Experimental settings

To test the performance of our multi-agent framework for reactive risk mitigation, we have generated an extensive set of scenarios representing different instances of the multilayer network model described in Section 3.1. In particular, we have generated 60 different scenarios, as follows:

- We have generated six different Asset Layer graphs, two for each of the following numbers of assets: {60, 70, 80}. Each graph was generated as an Erdös-Renyi random acyclic graph, with $p = 0.05$. This creates an average degree of dependencies between assets in the range of 3-4. This means that, on average, each asset depends on 3-4 other assets. Each asset was assigned a random weight, drawn from a uniform distribution between 1 and 10. For the scope of this work, dependencies between assets are assumed to be absolute. That is if asset a_i depends on asset a_j and asset a_j fails, a_i fails too and all its value is lost.

- For each asset graph of N assets, two component layer categories have been generated, having $2N$ components each. Each component layer category was generated as a random undirected Erdös-Renyi graph with p values in the set {0.1,0.3}. This generates component/instance layer categories where a given 0-day is supposed to affect, on average, 10% and 30% of the components (with varying degree of similarity among instances). This models a realistic situation and a worst-case scenario. For each category, 10 different component layers were generated corresponding to different configuration choices. For each instance layer, cross-configuration vulnerability values ν_{ij} between configurations were randomly assigned, drawn from a uniform distribution between 0.5 and 1. Finally, components were randomly linked to assets, ensuring that each asset depends on at least one component and that no two assets rely on the same component.

- We randomly deployed each instance in an infrastructure layer comprising six different network segments in a typical defense-in-depth scenario (i.e. concentric rings of security). For each pair of segments (s_i, s_j) , the cross-segment vulnerability value μ_{ij} depends on the distance between the segments in the topology, following an exponential decay, that is, $\mu_{ij} = \frac{1}{2^{|i-j|}}$.

For each one of the sixty scenarios, we ran simulations of the redeployment negotiation process in the event of the compromise of each single component, which accounts for 4200 different security breach scenarios. Along with our proposed approach, we ran simulations with four reference techniques, for comparison:

Augmented Lagrangian Particle Swarm Optimization (ALPSO): This technique has been used because particle swarm is a well-known optimizer that has been successfully applied to a number of problems [35][36] and can be considered a generic nonlinear optimizer that uses complete information. More specifically, we have chosen a parallel augmented Lagrange multiplier particle swarm

Table 1: Residual risk ratio results for $N = 60$ assets.

	$p = 0.1$			$p = 0.3$		
	Mean	95% CI	Time	Mean	95% CI	Time
ALHSO	0.2145	0.0057	0.9784	0.6174	0.0182	2.0667
ALPSO	0.4124	0.0197	3.1754	0.6963	0.0264	9.2430
BP	**0.1680**	0.0054	**0.5089**	**0.5326**	0.0154	**1.0918**
HC	0.2447	0.0088	1.0753	0.6522	0.0184	2.6617
SA	0.2097	0.0065	1.1552	**0.5578**	0.0205	2.7585

Table 2: Residual risk ratio results for $N = 70$ assets.

	$p = 0.1$			$p = 0.3$		
	Mean	95% CI	Time	Mean	95% CI	Time
ALHSO	0.3116	0.0092	1.1194	**0.6677**	0.0112	4.0253
ALPSO	0.3969	0.0192	6.6128	0.8024	0.0264	15.8024
BP	**0.2010**	0.0066	**0.6877**	**0.6566**	0.0138	**2.2234**
HC	0.2691	0.0063	1.5114	0.7909	0.0271	3.5227
SA	**0.2074**	0.0062	1.6023	0.7049	0.0121	3.6170

optimizer, which solves nonlinear non-smooth constrained problems using an augmented Lagrange multiplier approach to handle constraints [37].

Augmented Lagrangian Harmony Search Optimization (ALHSO): Another generic optimizer, this one based on harmony search, which is an evolutionary optimization algorithm inspired by musical composition [38]. As with ALPSO, this technique will be used as a reference of a nonlinear optimizer with complete information. As above, we have used augmented lagrangian multipliers to deal with constraints.

Hill-climber mediated negotiation (HC): This technique starts from a randomly-generated solution, that is, starts by assigning a random segment to each component in the graph. From this point, and at every iteration, the mediator proposes a new solution (contract) where a random graph node should change its value randomly. If that mutation produces a better solution, we use the new solution to generate the next mutation. This process goes on until a maximum number of iterations is reached.

Simulated annealing mediated negotiation (SA): This is a negotiation mechanism first proposed in [24] and further refined in [39] and [25], uses a widespread nonlinear optimization technique called simulated annealing [40]. The mechanism is similar to HC, but when a contract yields a utility loss (i.e. risk increase) against the previous mutually accepted contract, there will be a probability for the agent to accept it nonetheless. This probability P_a depends on the utility loss associated to the new contract Δu, and also depends on an *annealing temperature* parameter τ, so that $P_a = e^{-\Delta u/\tau}$. Annealing temperature begins at an initial value, and linearly decreases to zero as the protocol iterates.

4.2 Simulation results

For each of the experiments detailed in the previous section, we measured the *residual risk ratio* of the overall risk $\overline{\rho_{c_j}}$ between the original deployment and the reconfiguration yielded by our approach and each of the reference techniques. We also measured the amount of time taken by each mechanism to provide a solution.

In Tables 1 to 3 we summarize the results obtained for the different scenarios under study and for the different analyzed techniques. The columns labeled with "Mean" represent the mean value for the residual risk ratio for the 10 different instance layers generated corresponding to different configuration choices. Note that the residual risk ratio has been computed as the fraction of the residual risk $\overline{\rho_{c_j}}$ obtained by the optimization technique deployed and the initial risk $\overline{\rho_{c_j}}^0$ of the original deployment, so lower values are better. We highlight in bold the best result for each scenario. We check the validity of those results including the 95% confidence intervals (columns labeled with "95% CI"). Finally, in those tables we also include the time, expressed in seconds, required for each mechanism to operate, also highlighting in bold the best performing technique. Results show that the best residual risk ratio is obtained by BP in all cases, with some statistical ties. For the rest of the analyzed techniques, it is interesting to note that SA yields reasonably good results, and clearly outperforms HC, which suggests a highly non-monotonic scenario. Regarding the time required to compute their results and, therefore, the time required to respond to an incident, our approach also significantly outperforms the others.

Table 3: Residual risk ratio results for $N = 80$ assets.

	$p = 0.1$			$p = 0.3$		
	Mean	95% CI	Time	Mean	95% CI	Time
ALHSO	0.2320	0.0047	4.8296	0.7115	0.0234	12.9538
ALPSO	0.4094	0.0117	6.4584	0.8213	0.0252	11.4589
BP	**0.1862**	0.0056	**0.8190**	**0.6730**	0.0112	**2.6136**
HC	0.3036	0.0111	1.7981	0.7136	0.0167	4.5522
SA	0.2341	0.0057	1.8826	0.7246	0.0251	4.6598

5. CONCLUSIONS AND FUTURE WORK

Securing critical network infrastructures (CNIs) is a complex problem for today's society, one in which artificial intelligence in general and multi-agent systems in particular can play a big role. Multi-agent systems have been successfully used for threat prevention and detection in communication networks, but the challenge of real-time reaction to cyber-attacks is still largely unexplored by the multi-agent community. In this paper we have proposed a novel multi-agent framework for reactive resilience in critical network infrastructures. In particular, we use a multi-layer network risk model and a multi-agent negotiation approach to provide the network with the ability to reconfigure itself in the event of an attack. We also propose a negotiation process based on belief propagation (BP) and compare it to other optimization and negotiation techniques from the literature. Our experiments show that our BP negotiator clearly outperforms the other approaches both in terms of residual risk and performance. Although our experiments have yielded satisfactory results, there is still plenty of research to be done in this area. We have till now focused on reconfiguring the network via component redeployment throughout the available physical infrastructure, but now we want to explore the dual approach of changing component configuration instead (i.e. reinstantiating components with different configurations in the same deployment point) and see how these two approaches combine. We are also working on reactive resilience against multiple simultaneous compromises. Finally, we would like to study the influence of the underlying physical infrastructure on the effectiveness of the system.

Acknowledgments

This work has been supported by the Spanish Ministry of Economy, Industry and Competitiveness grants TIN2016-80622-P (AEI/FEDER, UE), TIN2014-61627-EXP, and MTM2014-54207, and by the University of Alcala through CCG2016/EXP-048.

REFERENCES

[1] Bo An, Milind Tambe, Fernando Ordonez, Eric Shieh, and Christopher Kiekintveld. Refinement of Strong Stackelberg Equilibria in Security Games. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 587–593, San Francisco, California, 2011. AAAI Press.

[2] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóòez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshal Service. *Interfaces*, 40(4):267–290, July 2010.

[3] James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. GUARDS: Game Theoretic Security Allocation on a National Scale. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 37–44, Taipei, Taiwan, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[4] James P. G. Sterbenz, Egemen K. Çetinkaya, Mahmood A. Hameed, Abdul Jabbar, Shi Qian, and Justin P. Rohrer. Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation. *Telecommunication Systems*, 52(2):705–736, 2013.

[5] Mingyu Guo, Hideaki Hata, and Ali Babar. Revenue Maximizing Markets for Zero-Day Exploits. In Matteo Baldoni, Amit K. Chopra, Tran Cao Son, Katsutoshi Hirayama, and Paolo Torroni, editors, *PRIMA 2016: Princiles and Practice of Multi-Agent Systems: 19th International Conference, Phuket, Thailand, August 22-26, 2016, Proceedings*, pages 247–260. Springer International Publishing, Cham, 2016.

[6] Leyla Bilge and Tudor Dumitras. Before We Knew It: An Empirical Study of Zero-day Attacks in the Real World. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 833–844, New York, NY, USA, 2012. ACM.

[7] Ghassan Ahmed Ali, Aman Jantan, and Abdulghani Ali. Honeybee-based model to detect intrusion. In *International Conference on Information Security and Assurance*, pages 598–607. Springer, 2009.

[8] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel. K-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 11(1):30–44, January 2014.

[9] M. J. F. Alenazi and J. P. G. Sterbenz. Comprehensive comparison and accuracy of graph metrics in predicting network resilience. In *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*, pages 157–164, March 2015.

[10] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese. Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks. *IEEE Transactions on Information Forensics and Security*, 11(5):1071–1086, May 2016.

[11] Yehiel Berezin, Amir Bashan, Michael M. Danziger, Daqing Li, and Shlomo Havlin. Localized attacks on spatially embedded networks with dependencies. *Scientific Reports*, 5:8934, March 2015.

[12] Shuai Shao and Xuqing Huang and H Eugene Stanley and Shlomo Havlin. Percolation of localized attack on complex networks. *New Journal of Physics*, 17(2):023049, 2015.

[13] Osman Yaıfmmode \breveg\else ğ\fian and Virgil Gligor. Analysis of complex contagions in random multiplex networks. *Phys. Rev. E*, 86(3):036103, September 2012.

[14] Kathleen M. Carley, Jana Diesner, Jeffrey Reminga, and Maksim Tsvetovat. Toward an interoperable dynamic network analysis toolkit. *Decision Support Systems*, 43(4):1324 – 1347, 2007. Special Issue Clusters.

[15] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer networks. *Journal of Complex Networks*, July 2014.

[16] Stefan Dietzel, Julian Gürtler, and Frank Kargl. A resilient in-network aggregation mechanism for {VANETs} based on dissemination redundancy. *Ad Hoc Networks*, 37, Part 1:101 – 109, 2016. Special Issue on Advances in Vehicular Networks.

[17] R. Fan, Y. T. Yu, and M. Gerla. RobustGeo: A Disruption-Tolerant Geo-Routing Protocol. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8, August 2015.

[18] Y. Yao, Q. Cao, and A. V. Vasilakos. EDAL: An Energy-Efficient, Delay-Aware, and Lifetime-Balancing Data Collection Protocol for Heterogeneous Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 23(3):810–823, June 2015.

[19] E. Casalicchio, E. Galli, and S. Tucci. Agent-based modelling of interdependent critical infrastructures. *International Journal of System of Systems Engineering*, 2(1):60–75, January 2010.

[20] I. Butun, S. D. Morgera, and R. Sankar. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Communications Surveys Tutorials*, 16(1):266–282, First 2014.

[21] Kun Wang, Miao Du, Dejun Yang, Chunsheng Zhu, Jian Shen, and Yan Zhang. Game-Theory-Based Active Defense for Intrusion Detection in Cyber-Physical Embedded Systems. *ACM Trans. Embed. Comput. Syst.*, 16(1):18:1–18:21, October 2016.

[22] Paulo Shakarian, Hansheng Lei, and Roy Lindelauf. Power Grid Defense Against Malicious Cascading Failure. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '14, pages 813–820, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[23] Enrique de la Hoz, M. Jose Gimenez-Guzman, Ivan Marsa-Maestre, and David Orden. Automated Negotiation for Resource Assignment in Wireless Surveillance Sensor Networks. *Sensors*, 15(11), 2015.

[24] Mark Klein, Peyman Faratin, Hiroki Sayama, and Yaneer Bar-Yam. Negotiating Complex Contracts. *Group Decision and Negotiation*, 12(2):111–125, 2003.

[25] Fabian Lang and Andreas Fink. Learning from the Metaheuristics: Protocols for Automated Negotiations. *Group Decision and Negotiation*, 24(2):299–332, 2015.

[26] Kief Morris. *Infrastructure as Code. Managing Servers in the Cloud*. O'Reilly Media, 2016.

[27] Ken Gray and Thomas D Nadeau. *Network Function Virtualization*. Morgan Kaufmann, 2016.

[28] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.*,

2014(239), March 2014.

[29] D. Bernstein. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, 1(3):81–84, September 2014.

[30] James Turnbull. *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014.

[31] David Orden, Ivan Marsa-Maestre, Jose Manuel Gimenez-Guzman, and Enrique de la Hoz. Spectrum graph coloring and applications to wifi channel assignment. *arXiv preprint arXiv:1602.05038*, 2016.

[32] David Gamarnik, Devavrat Shah, and Yehua Wei. Belief Propagation for Min-Cost Network Flow: Convergence and Correctness. *Operations Research*, 60(2):410–428, April 2012.

[33] Lu Zheng and Ole Mengshoel. Optimizing Parallel Belief Propagation in Junction Treesusing Regression. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 757–765, Chicago, Illinois, USA, 2013. ACM.

[34] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pages 1–12, New York, NY, USA, 1987. ACM.

[35] K. Ishaque and Z. Salam. A Deterministic Particle Swarm Optimization Maximum Power Point Tracker for Photovoltaic System Under Partial Shading Condition. *IEEE Transactions on Industrial Electronics*, 60(8):3195–3206, August 2013.

[36] Pratyay Kuila and Prasanta K. Jana. Energy efficient clustering and routing algorithms for wireless sensor networks: Particle swarm optimization approach. *Engineering Applications of Artificial Intelligence*, 33:127 – 140, 2014.

[37] P. W. Jansen and R. E. Perez. Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach. *Computers & Structures*, 89(13–14):1352 – 1366, 2011.

[38] Zong Woo Geem, Joong Hoon Kim, and G.V. Loganathan. A New Heuristic Optimization Algorithm: Harmony Search. *SIMULATION*, 76(2):60–68, 2001.

[39] Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Juan R. Velasco, and Enrique de la Hoz. Avoiding the Prisoner's Dilemma in Auction-based Negotiations for Highly Rugged Utility Spaces. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 425–432, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671, May 1983.