Analyzing the Implications of Network Structure in Global Optimization and Peer-to-Peer Trust

---

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Yuting Wang

May

2013

APPROVAL SHEET

The dissertation

is submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

AUTHOR

The dissertation has been read and approved by the examining committee:

Alfredo Garcia

Advisor

Peter Beling

Randy Cogill

Gerard Learmonth

Zongli Lin

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

May

2013

Contents

Abstract

In this dissertation, we study how network structure plays an important role in two separate fields: global optimization and trust dynamics in peer-to-peer networks. In global optimization, we consider the class of model-based search methods. Roughly speaking, these are methods in which random search is implemented according to a probability distribution or "model" which is updated at each iteration depending upon the quality of solutions identified. We show that search for the global minimum is equivalent to search in a designated network and that the complexity of the problem can be described in terms of network structure. By focusing on the best solutions identified, most model-based algorithms emphasize "exploitation" over "exploration". Depending upon the structure of the problem, this emphasis in "exploitation" may end up significantly slowing down the identification of the global minimum. We propose a new algorithmic design for model-based search based on multiple interacting threads. In this design, each thread implements a model-based search and they interact through a simple acceptance-rejection rule preventing duplication of search efforts. We show that the speed of convergence (both in the worst case and on average) increases exponentially in the number of threads. Thus, in the proposed algorithmic design, exploration is a complement rather than a substitute to exploitation. The second part of the dissertation aims to characterize the dynamics of trust in unreliable peer-to-peer networks. Distributed trust or reputation systems have naturally evolved as a potential scalable solution to ensuring data exchange in peer-to-peer networks is reliable. In a trust-based system each node maintains and updates the trust (or reputation) score of neighboring nodes. These trust coefficients are modified depending upon

locally verified outcomes. For example, in a peer-to-peer file sharing system a node's trust will be decreased if neighboring nodes verify that the last file uploaded or forwarded by that node was corrupted. In this dissertation we consider the implications on network structure in distributed trust dynamics of peer-to peer networks of agents. We characterize the performance of a general class of trust-based schemes as a function of the underlying network structure. We conclude with an application to cyber security of large scale wind power system.

Acknowledgments

I would like to express my deepest appreciation to the individuals who helped me in completing this dissertation. In particular, I would like to thank my advisor, Dr. Alfredo Garcia, for his valuable advice and support. This dissertation would not have been possible without his persistent guidance and help. I would also like to thank the rest of my committee members, Dr. Peter Beling, Dr. Randy Cogill, Dr. Gerard Learmonth and Dr. Zongli Lin for the insights they provided for my dissertation. In addition, I would like to express my gratitude for Nathan Trantham, the graduate student in the department of systems and information engineering. Mr. Nathan Trantham's work on algorithm implementation and simulation contributed tremendously to my dissertation. Finally, I would like to take the opportunity to thank my husband, Cheng Peng, for his support and encouragement.

Chapter 1

Introduction

The earliest work on network analysis can probably be traced back to 1736 when Leonhard Euler studied the seven bridges of Konigsberg. Since then, network analysis has found applications in diverse fields including sociology, electrical engineering, biology, computer science amongst others. In many applications, networks that have been studied exhibit intricate patterns of connection among nodes. For example, the shortest path between two randomly-selected nodes is shorter in a "small world" network structure than in a lattice network structure of the same size. The reason is that in a "small world" network most nodes can be reached from each other by a small number of hops or steps (though most nodes are not neighbors of one another).

We are interested in the role network structure plays in two separate fields of study: global optimization and trust dynamics in peer-to-peer networks. In this dissertation, we will demonstrate how network structure plays an important role in these research areas.

In global optimization, we consider the class of model-based search methods, where random search is implemented according to a probability distribution or "model". The model will be updated at each iteration based on the solutions identified so far. We show that search for a globally optimal solution is equivalent to search in a given network and that the complexity of the problem can be described in terms of network structure. The basic loop in a model-based search method is

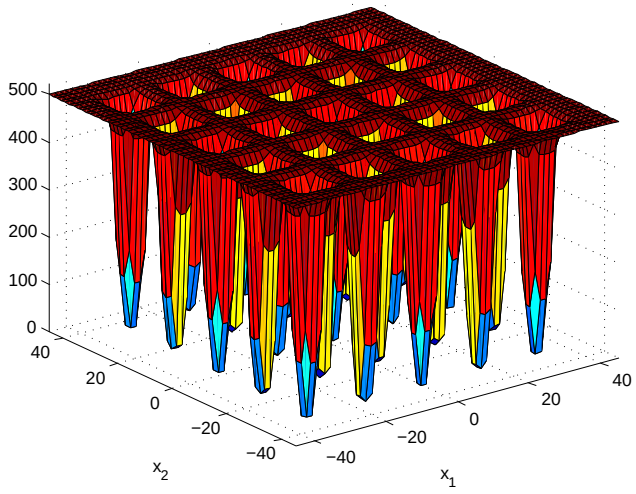

Figure 1.1: De Jong's 5th Function

as follows. First, a solution is randomly selected according to a given sampling distribution (or "model"). Second, a local search procedure is launched in order to identify a local minimum. Finally, the sampling distribution will be modified so that it will posit more mass on the best local solutions (lowest value) found so far. By focusing on the best solutions, this algorithm emphasizes "exploitation" over "exploration". However, depending on the structure of the problem, the emphasis in "exploitation" may cause the slow down of the identification of a globally optimal solution – provided such identification is feasible. To motivate our ideas, consider the problem of finding the minimum of De Jong's fifth function shown in Figure [1.1]. The function exhibits a "multi-funnel" structure with 25 local minima in total, as illustrated in Figure [1.2] and [1.3] – the shaded node being the globally optimal solution. Note that each locally-optimal solution has a relatively wide basin of attraction. Due to its emphasis on exploration, the algorithm is likely to be trapped within one of these basins. This can be seen as the algorithm positing small probabilities on the "links" or "edges" leading from one basin to another. As a result it may take a long time (and great computational effort) to identify a globally optimal solution. The structure of this particular case makes it obvious that exploration is of more importance than exploitation. However, in practice, the

underlying structure of the global optimization problem is not known in advance, making it hard to strike a balance between exploration and exploitation.

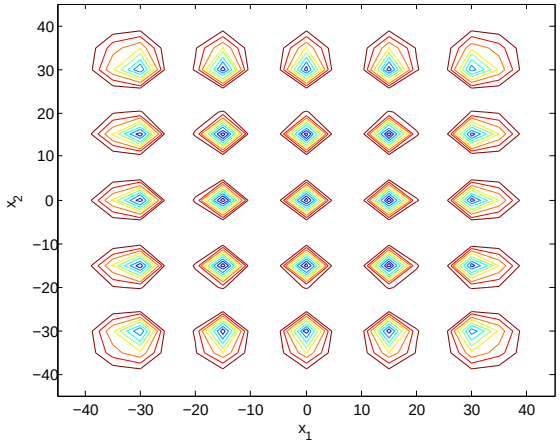

Figure 1.2: Contour of De Jong's 5th Function

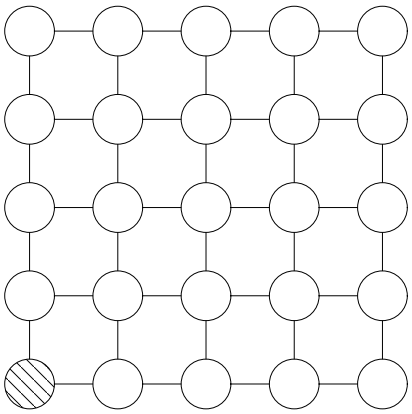

Figure 1.3: Network Representation of De Jong's 5th Function

Generally speaking, most single-thread algorithms for global optimization differ in the way computational effort between exploitation and exploration is allocated. The allocation used ultimately determines the overall performance of the algorithm. For example, if too little emphasis is put on exploration, the globally-optimal solution is likely to remain unidentified. On the other hand, more exploration boosts up the chances to identify the globally optimal solution but it also slows down convergence. In this dissertation, we propose a new algorithmic design for

global optimization based on multiple interacting threads. In this design, each thread implements a model-based search in which the allocation on exploration versus exploitation does not vary over time. Threads interact through a simple acceptance-rejection rule preventing duplication of search efforts. We show that the speed of convergence (both in the worst case and on average) increases exponentially in the number of threads. Thus, in the proposed algorithmic design, exploration is a complement rather than a substitute to exploitation.

The second part of the dissertation aims to characterize the dynamics of trust in unreliable peer-to-peer networks.

Advances in information and communications technologies have enabled the proliferation of significant amounts of data. In today's world, computer networks and cell phones pervade the daily activities of millions. As networked sensors are increasingly being embedded in day to day activities, the amount of data available is growing at a pace that tests the limits of state-of-the-art techniques for data analysis. Most problems posed by increasing amounts of data have been labeled in the media as the "big data" challenge. Decentralization is an important characteristic of many "big data" networks. While the lack of centralized authority allows for scalability, it also makes it hard to guarantee data is reliable. Distributed trust or reputation systems have naturally evolved as a potential scalable solution to ensuring reliable data. Typically, each node maintains and updates a trust or reputation score of neighboring nodes. These trust coefficients are updated depending on locally verified outcomes. For example, in a peer-to-peer file sharing system a node's trust score will be decreased if neighboring nodes verify that the last file uploaded or forwarded by that node was corrupted. A corrupted file may be the result of a malicious action of the node. However, it may also be due to a random event affecting a benevolent node. Hence, the dynamics of trust or reputation scores should be smooth enough in order to minimize false positives and false negatives.

In this dissertation we consider the implications on network structure in distributed trust dynamics of peer-to-peer network of agents that independently estimate a parameter. In the model, each agent in the network will periodically produce an independent estimate of an unobservable parameter and communicates that estimate to its neighbors. There are of two types of agents: "good" and "bad" estimators. When given enough data, a "good" agent will (asymptotically) produce the correct parameter estimation. On the contrary, a "bad" estimator is bound to produce an incorrect estimation no matter the amount of data consumed. Each agent is agnostic of its own nature. Hence, the dynamics of trust play an important role in enabling each agent to produce a correct *merged* estimate. This necessarily requires the correct identification of the nature of each agent. For this we examine the notion of "wisdom of crowds" as a basis for correct trust updates. Assuming the majority is indicative of good behavior, nodes will downgrade the trust scores of nodes in the minority. Our goal is to characterize the performance of this scheme as a function of the underlying network structure (as well as the number and distribution of "bad" nodes).

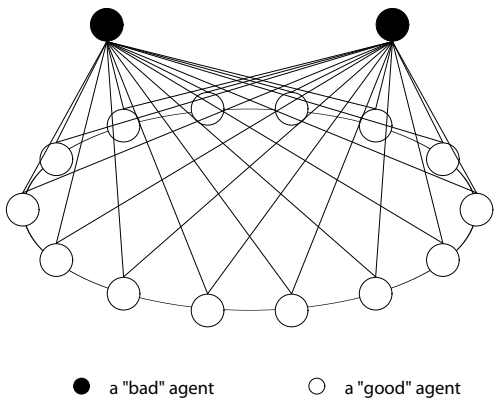

Figure 1.4: A Loosely-Connected Social Network with Two Highly Influential Agents

To illustrate the importance of network structure consider the network depicted in Figure 1.4. Note that all "good" agents form a cycle, with two "bad" agents connecting to every " good" agent. In this configuration, each "good" agent is

only able to communicate with two "good" neighbors, while both "bad" agents can communicate with any "good" agent directly. Hence, the degree of a "good" agent is 4, two of which belong to "good" neighbors, the other two of which are the two "bad" agents. The degree of a "bad" agent equals the number of total "good" agents. This type of structure may be considered as a loosely-connected social network with two highly influential agents. The application of trust dynamics based upon the notion of "wisdom of crowds" is likely to fail as each "good" agent may lower the weight accorded to other "good" agents within its neighborhood with some positive probability.

Chapter 2

Global Optimization

2.1 Introduction

Many heuristics used for global optimization can be described as *population-based* algorithms. Each iteration, a set of solutions is evaluated and a new set of solutions is randomly generated according to a given rule designed to strike a balance in the resource allocation between "exploration" and "exploitation". Examples include Genetic Algorithms (GA) (see [1]) and Particle Swarm Optimization (PSO) (see [2]). Despite their popularity, the theoretical ground of the speed of convergence of these algorithms remains limited as it is exceedingly difficult to characterize the relationship between the parameter settings and efficiency. Recently, a class of *model-based* algorithms has received increased attention (see [3]). In a model-based algorithm, the optimal solution is estimated at each iteration by sampling candidate solutions from a sequence of probability distributions (or "models") over the feasible region. Due to its successful applications to hard optimization problems (see [4] and [5] for a survey of these applications) the Cross-Entropy (CE) method is perhaps the most popular algorithm in this class.

In model-based algorithms, the trade-off between exploration and exploitation is reflected in the way the probability distribution (or the "model") is updated. The degree to which the new probability distribution (or "model") is concentrated around the best solutions identified so far reflects the relative emphasis on exploita-

tion versus exploration. The implication is clear: if too little emphasis is put on exploration, the globally optimal solution may not be identified. Increasing the allocation of computational effort to exploration increases the chances of identifying a globally optimal solution but it may also slow down convergence. In many model-based algorithms, the relative emphasis on exploitation versus exploration varies over time so that computational effort on exploitation *slowly* overtakes computational effort on exploration. As in most single-thread algorithmic designs for global optimization, convergence speed is sacrificed for the sake of guaranteeing sufficient exploration.

In this chapter, we propose a new algorithmic design for global optimization with multiple interacting threads. In this design, each thread implements a model-based search with an emphasis on exploitation that does not vary over time. Efficient exploration is achieved by means of a simple acceptance-rejection rule preventing duplication of search efforts amongst different threads. We show that the speed of convergence (both in the worst case and on average) increases exponentially in the number of threads. Thus, in the proposed algorithmic design, exploration does not need to be traded off over time with exploitation.

The structure of this chapter is as follows. In Section 2.2, we describe the single-thread model-based search and provide an illustration with the cross-entropy method. We also develop a Markov chain model where the state space is the set of all locally optimal solutions and characterize the speed of convergence of the single-thread model-based algorithm. The worst-case convergence rate is determined by the second largest eigenvalue associated with the transition matrix. Intuitively, the second largest eigenvalue is associated with the "worst" possible combination of locally optimal solutions that could be identified. The average speed of convergence is characterized in terms of clusters (an aggregation of states). A cluster posits a difficult computational challenge to the model-based search whenever new models generated by the algorithm (given a state in the cluster) do not vary significantly. Thus, the speed of convergence of the single-thread model-based search deteriorates

in problems with many difficult clusters whose basins of attraction are relatively large. In Section 2.3, we introduce a new interactive (multi-thread) version of the search method subject to an acceptance-rejection test aimed at preventing duplication on search effort. We show that the speed of convergence (both worst-case and average) increases exponentially in the number of threads. Finally,we offer concluding comments in Section 2.4. Numerical examples are provided in Chapter 3.

2.2 Single-Thread Model-Based Search

Consider a general optimization problem $\min\{f(x) : x \in \Omega\}$ where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuous and $\Omega \subset \mathbb{R}^n$ is such that $X^* = \arg\min_{x\in\Omega} f(x)$ is well-defined. Assume further f has N locally (non-globally) optimal solutions, say $X = \{x_1, x_2, \ldots, x_N\}$, that is,

$$x_i \in \arg \min_{x\in B(x_i)} f(x)$$

where $B(x_i) \subseteq \Omega$ will be defined below. The multi-start method that we shall describe later makes use of a local search algorithm which identifies $x_i \in B(x_i)$ as output if the initial input given to the algorithm is an element of the set $B(x_i) \subset \Omega$. Let $\ell : \Omega \to \Omega$ denote the map defined by $\ell(x) = x_i$ if $x \in B(x_i)$. We refer to $B(x_i)$ as the "basin of attraction" of solution x_i under the local search algorithm used. We make the following assumption on the quality of the local search algorithm:

Assumption 2.2.1. $\bigcup_{x_i\in X\cup X^*} B(x_i) = \Omega$

Let $J^t \in 2^{X\cup X^*}$ denote the "state" at iteration $t > 0$, i.e. the subset of the set of locally (and globally) optimal solutions identified in the first t local searches conducted. Let $\mathfrak{G}$ denote a class of probability distributions with support Ω. Let $g(J^t) \in \mathfrak{G}$ denote the probability distribution (or "model") associated with the state of information J^t. The basic iteration of the single-thread model-based search is as follows:

1. A sample, say y, from $g(J^t)$ is drawn and a local search algorithm is launched. The resulting state is

$$J^{t+1} = J^t \cup \ell(y)$$

2. A new model $g(J^{t+1}) \in \mathfrak{G}$ is selected by a designated rule.

2.2.1 Illustration: Cross-Entropy

One possible implementation of step (2) in the single-thread model-based search algorithm is as follows. Given state J^{t+1}:

1. A reference probability density function $h(x; J^{t+1})$ is computed. For example, such a reference probability density function can be defined as

$$h(x; J^{t+1}) = \frac{I(f(x), J^{t+1})U(x)}{E[I(f(x), J^{t+1})U(x)]}$$

where U is the uniform probability density function on Ω and

$$I(f(x), J^{t+1}) = \begin{cases} 1 & f(x) \leq \min\limits_{x \in J^{t+1}} f(x) + \epsilon \\ 0 & f(x) > \min\limits_{x \in J^{t+1}} f(x) + \epsilon \end{cases}$$

with $\epsilon > 0$. Note this reference density posits more mass around the best local solutions in J^{t+1}.

2. The model is updated by selecting a distribution that minimizes the cross-entropy with respect to the reference distribution:

$$g(J^{t+1}) = \arg\min_{g \in \mathfrak{G}} \int_\Omega \ln \frac{h(x; J^{t+1})}{g(x)} h(x; J^{t+1}) dx$$

2.2.2 Markov Chain Model

Abusing notation, we denote by J^* the class of states with a globally optimal solution, i.e. $J \in J^*$ if and only if $J \cap X^* \neq \varnothing$. The transition probability matrix

for the single-thread model-based search model can be partially specified as follows. For $J \in 2^X$,

$$P(J' \mid J) = \begin{cases} \int\limits_{B(x)} g(x; J)dx & J' = J \cup \{x\} \quad x \notin X^* \\ \int\limits_{B(X^*)} g(x; J)dx & J' \in J^* \\ 1 - \sum\limits_{x \in X \cup X^* \setminus J} \int\limits_{B(x)} g(x; J)dx & J' = J \end{cases} \tag{2.1}$$

Remark. The definition of the transition probability matrix in Equation (2.1) requires the assumption that all "models" $g \in \mathfrak{G}$ are integrable over the solution space Ω. This assumption is done for ease of exposition, i.e. when the underlying optimization problem is combinatorial in nature, a similar transition probability matrix can be obtained with a proper definition of "basins of attraction" associated to the given local search algorithm.

2.2.3 Convergence

Before we proceed with our analysis we make the following additional assumptions:

Assumption 2.2.2. *Given any $g \in \mathfrak{G}$, $g(x) \neq 0$ for $x \in \Omega$ almost surely.*

In other words, every basin of attraction has a chance to be visited when the algorithm generates samples at each iteration.

Assumption 2.2.3. $P(J^* | J^*) = 1$

Note this last assumption is satisfied when the new model is selected according to the minimization of cross entropy with respect to the reference distribution and

$$\min_{x \in X} f(x) - f(x^*) > \epsilon$$

We start our analysis of convergence with the following lemma.

Lemma 2.2.1. *(a)* The transition matrix P is upper triangular.

(b) The eigenvalues are $\lambda_i = P(J_i | J_i)$ for $J_i \in 2^X$ and $P(J^* \mid J^*) = 1$.

(c) The Markov chain $\{J^t : t > 0\}$ has only one absorbing state J^*.

Proof. Let $|J|$ denote the number of distinct locally optimal solutions for $J \in 2^X$. We can endow 2^X with the partial order "$\succeq$" defined as follows: $J \succeq J'$ if and only if $|J| > |J'|$. This partial ordering is extended as follows. For states with the same number of distinct locally optimal solutions, we select an arbitrary ordering. Finally, we posit $J^* \succeq J$ for any $J \in 2^X$ (recall that J^* denotes the class of states with a globally optimal solution, i.e. $J \in J^*$ if and only if $J \cap X^* \neq \varnothing$). Consider now transition matrix P on the partially ordered state space $2^X \cup J^*$. Based on Equation (2.1), $P_{J_2,J_1} = P(J_2 \mid J_1)$ might be non-zero only when either $J_2 \supseteq J_1$ or $J_2 = J^*$. Thus, P is upper-triangular and hence all the eigenvalues are entries along the diagonal. By construction, $P_{J^*,J^*} = 1$, i.e. J^* is an absorbing state and in light of Assumptions 2.2.1 and 2.2.2 all states $J \in 2^X$ are transient. □

It follows from Lemma 2.2.1 that the second largest eigenvalue is

$$\lambda_{[2]} = \max_{J \in 2^X} \sum_{x \in J} \int_{B(x)} g(x; J) dx$$

The second largest eigenvalue of the transition matrix has a close relationship with the convergence rate of the Markov chain represented by the transition matrix. Let $\pi^t(J) = \Pr(J^t = J | J^0)$ be denoted as the distribution at time t, and π as the stationary distribution. It follows from Lemma 1 that

$$\pi^t(J^*) \to 1$$

Moreover, it can be shown that the convergence rate of the Markov chain is upper-bounded by the second largest eigenvalue of its transition matrix.

Theorem 2.2.1. *Assume all the eigenvalues of the transition matrix P are distinct. There is a constant $C > 0$ such that*

$$\left|\pi^t(J^*) - 1\right| \leq C\lambda_{[2]}^t$$

Proof. The proof is similar to that of [6] for a simplified implementation of a genetic algorithm. Assume the Markov chain has K states, where the last state in order is J^*. By hypothesis, the transition matrix P is diagonalizable. In other words,

$$P = \sum_{i=1}^{K} \lambda_i v_i u_i^T$$

where v_i and u_i are the right and left eigenvectors of λ_i. Similarly, we will obtain

$$P^t = \sum_{i=1}^{K} \lambda_i^t Q^{(i,t)}$$

where $Q^{(i,t)}$ is a matrix completely determined by P. So at time t, the distribution $\pi^t(L)$ for state L is

$$\pi^t(L) = \sum_J P_{J,L}^t \pi_J^0 = \sum_J \sum_{i=1}^{K} \lambda_i^t Q_{J,L}^{(i,t)} \pi_J^0$$

Hence, it follows

$$\begin{aligned}
1 - \pi^t(J^*) &= \sum_{L \neq J^*} \pi^t(L) \\
&= \sum_{L \neq J^*} \sum_J \sum_{i=1}^{K} \lambda_i^t Q_{J,L}^{(i,t)} \pi_J^0 \\
&= \sum_{L \neq J^*} \sum_J 1 \cdot Q_{J,L}^{(K,t)} \pi_J^0 + \sum_{L \neq J^*} \sum_J \sum_{i=1}^{K-1} \lambda_i^t Q_{J,L}^{(i,t)} \pi_J^0 \\
&\leq C_1 + C_2 \left|\lambda_{[2]}\right|^t \\
&\leq C \left|\lambda_{[2]}\right|^t
\end{aligned}$$

□

2.2.4 Average Performance

Given a state J (i.e. a set of locally optimal solutions), the stopping time $\tau(J)$ is defined as follows:

$$\tau(J) = \inf\{\, t \geq 0 |\ \ J^t \in J^*, J^0 = J\}$$

Note that $E[\tau(J)]$ can be regarded as a measure of the average performance of the algorithm. In order to quantify the average performance, we first introduce the definition of "clusters" of states.

Definition 2.2.1. *Let $\eta > 0$. An η-cluster, say $\mathcal{J}$, is a set of states such that for every state $J \in \mathcal{J}$ and any $x \in B(\bar{J})$:*

$$|g(x; J) - g(x; \bar{J})| \leq \eta$$

where $\bar{J}$ which we shall refer to as the representative state of $\mathcal{J}$ is defined as:

$$\bar{J} = \bigcup_{x \in J, J \in \mathcal{J}} \{x\}$$

In words, a cluster is a set of states giving rise to similar "models" (e.g. sampling distributions over the domain Ω). The total information associated to locally optimal solutions in a cluster is encapsulated in the state $\bar{J}$. In other words, for any state $J \in \mathcal{J}$, by definition, we have $J \subseteq \bar{J}$.

Note also that the definition of a cluster trivially implies a single-solution state $\{x\} \in 2^X$ is a η-cluster. Hence, there exist a group of clusters whose basins of attraction partition the feasible region $\Omega \backslash\, B(X^*)$. Let $\mathcal{J}^1$,$\mathcal{J}^2$,...,$\mathcal{J}^{K(\eta)}$, $K(\eta) > 0$ denote the *coarsest* group of clusters whose basins of attraction partition the feasible region, i.e.

$$\Omega \backslash\, B(X^*) = \bigcup_{k=1}^{K(\eta)} B(\bar{J}^k)$$

where $\bar{J}^k$ is the representative state of cluster $\mathcal{J}^k$ respectively.

Definition 2.2.2. *For each cluster $\mathcal{J}^k$, $k \in \{1, \ldots, K(\eta)\}$, we define $\lambda_k = P(\bar{J}^k \mid \bar{J}^k)$ where $\bar{J}^k$ is the representative state of cluster $\mathcal{J}^k$.*

Recall that $P(J \mid J)$ is the probability that the model-based search will not yield a new locally optimal solution after one iteration given that the current state is J. In this sense, Definition (2.2.2) provides a measure of how likely it is a model-based search could "get stuck" in a cluster of states.

Definition 2.2.3. *Let $m(B)$ denote the Lebesgue measure of set $B \subseteq \Omega$. For each cluster $\mathcal{J}^k$, $k \in \{1, \ldots, K(\eta)\}$, we define*

$$\theta_k = \frac{1 - \lambda_k}{m(B(\bar{J}^k))}$$

A model-based search is likely to "get stuck" in a cluster $\bar{J}^k$ with a high value λ_k. However, getting stuck in such cluster is unlikely when it has a relatively small basin of attraction $m(B(\bar{J}^k))$. In this sense, θ_k provides a measure of the difficulty posed by different clusters for a single-thread model-based search algorithm. As we shall see in the following result, the average performance of a model-based search can be extremely poor when all clusters (in the coarsest partition) have low values of θ_k.

Theorem 2.2.2. *Let $\eta > 0$ and $\{\mathcal{J}^1, \mathcal{J}^2, \ldots, \mathcal{J}^{K(\eta)}\}$ the coarsest set of η-clusters. Assume the initial state J_0 is obtained by randomly choosing an initial point in Ω. The average performance (expected number of iterations needed to identify globally optimal solutions) is bounded below as follows:*

$$E[\tau(J_0)] \geq m(\Omega)^{-1} \sum_{k=1}^{K(\eta)} \frac{1}{\theta_k - \eta}$$

Proof. Let $J^k_{min} = \arg\min_{J \in \mathcal{J}^k} E[\tau(J)]$. It follows that

$$\begin{aligned} E[\tau(J^k_{min})] &= 1 + \sum_{J^k_{min} \subseteq J' \in \mathcal{J}^k} P(J' \mid J^k_{min}) E[\tau(J')] + \sum_{J^k_{min} \subseteq J' \notin \mathcal{J}^k} P(J' \mid J^k_{min}) E[\tau(J')] \\ &\geq 1 + \left(\sum_{J^k_{min} \subseteq J' \in \mathcal{J}^k} P(J' \mid J^k_{min}) \right) E[\tau(J^k_{min})] \\ &\geq 1 + \left(\lambda_k - \eta m(B(\bar{J}^k)) \right) E[\tau(J^k_{min})] \end{aligned}$$

Hence,

$$E[\tau(J^k_{min})] \geq \frac{1}{1 - \lambda_k + \eta m(B(\bar{J}^k))}$$

The expected stopping time from a uniformly randomly chosen initial (singleton) state J_0 is then lower bounded as follows:

$$\begin{aligned} E[\tau(J_0)] &\geq \sum_{k=1}^{K(\eta)} \frac{m(B(\bar{J}^k))}{m(\Omega)} \min_{\{x\} \in \mathcal{J}^k} E[\tau(\{x\})] \\ &\geq \sum_{k=1}^{K(\eta)} \frac{m(B(\bar{J}^k))}{m(\Omega)} \frac{1}{1 - \lambda_k + \eta m(B(\bar{J}^k))} \\ &= m(\Omega)^{-1} \sum_{k=1}^{K(\eta)} \frac{1}{\theta_k - \eta} \end{aligned}$$

□

Theorem (2.2.2) reveals that the average performance of the single-thread model-based search is closely linked to the numerical values of θ_k, $k \in \{1, \ldots, K(\eta)\}$. Note that each θ_k is decreasing in the eigenvalue λ_k and in the Lebesgue measure of the basin of attraction $m(B(\bar{J}^k))$. Hence, this result indicates that the single-thread model-based search will exhibit poor performance in problems in which *(i)* the coarsest group of clusters have large basins of attraction and *(ii)* the model-based search is likely to "get stuck" in each cluster (i.e. high values λ_k). This observation is similar to the characterization of "difficult" problems in the context of global optimization of energy structures (see [7]) in terms of "multiple basin structures" .

2.3 Interactive Model-based Search

Our analysis has identified conditions in which a single-thread model-based approach to global optimization may yield poor performance. In what follows we consider a multi-thread interactive implementation of model-based search that is shown to speed up convergence. To formalize, suppose that a total of M parallel threads implement the model-based search and their current state is denoted by J_i, $i \in \{1, \ldots, M\}$ and the corresponding model is $g_i(\boldsymbol{J})$ where $\boldsymbol{J} = (J_1, \ldots, J_M)$. Let us denote by $KL(g_i(\boldsymbol{J}), g_\ell(\boldsymbol{J}))$ the Kullback-Leibler divergence:

$$KL(g_i(\boldsymbol{J}), g_\ell(\boldsymbol{J})) = \int_\Omega \ln \frac{g_\ell(x; \boldsymbol{J})}{g_i(x; \boldsymbol{J})} g_\ell(x; \boldsymbol{J}) dx$$

2.3.1 Acceptance-Rejection Test

Let $\eta > 0$. The new model for the $i - th$ thread, say $g_i'(\boldsymbol{J})$, is defined as

$$g_i'(\boldsymbol{J}) = \begin{cases} g(x; J_i) & KL(g_i(\boldsymbol{J}), g_\ell(\boldsymbol{J})) > \eta \quad \forall \ell \neq i \\ \tilde{g}(x; J_i) & \text{Otherwise} \end{cases} \tag{2.2}$$

where $\tilde{g}(x; J_i)$ is defined as

$$\tilde{g}(J_i) \in \arg\min_{g \in \mathfrak{G}} \int_{B(J_i)} g(x) dx \tag{2.3}$$

We can now formally describe the interactive model-based search as follows. Let $\boldsymbol{J}^t = (J_1^t, J_2^t, \ldots, J_M^t)$ denote the joint state. To summarize, the basic iteration in the interactive model based search algorithm is described as follows:

1. A sample, say y_i, from $g(J_i^t)$ is drawn and a local search algorithm is launched for thread $i \in \{1, \ldots, M\}$. The resulting state is

$$J_i^{t+1} = J_i^t \cup \ell(y_i)$$

2. A temptative new model $g(J_i^{t+1}) \in \mathfrak{G}$ is selected. The acceptance-rejection test is implemented and the resulting model is given by $g_i'(\boldsymbol{J}^{t+1})$ as described in (2.2).

Note that threads are independent as long as they are far away from one another. However, if any two threads are too close to each other, they reject their own models derived from the current updating rule and turn to a model which in a sense guarantees improved exploration. The acceptance-rejection test formalizes the need to diversify search activities when current search activities by two or more threads are too similar. As we shall show, the interactive parallel implementation of the model-based search method is guaranteed to obtain a faster convergence rate, both in the worst case and on average.

Remark. In practice, we may randomly select a new model $\tilde{g}_i(\boldsymbol{J}^{t+1})$ from the family of parameterized probability functions to avoid using Equation (2.3) directly. This provides an easier way to expectedly allocate the search efforts for diversity.

2.3.2 Convergence Speed

Recall that the worst-case performance is characterized by the second largest eigenvalue of the transition matrix. In the analysis of average performance, we derived a lower bound *increasing* in the eigenvalues of the representative states for the coarsest group of clusters. These results suggest that in order to speed up the convergence of a model-based search the eigenvalues must be reduced. As we shall show, the interactive implementation of the model-based search has this property.

Theorem 2.3.1. *Let $\boldsymbol{J} = (J_1, \ldots, J_M)$ be a state for model-based search with M threads. If $\lambda_{\boldsymbol{J}}$ and $\lambda'_{\boldsymbol{J}}$ denote the corresponding eigenvalues of transition matrix under independent and interactive threads respectively, then*

$$\lambda_{\boldsymbol{J}} \geq \lambda'_{\boldsymbol{J}}$$

Moreover, assume $\lambda_{[2]}$ is the second largest eigenvalue of the single-thread model-based search.

$$(\lambda_{[2]})^M \geq \lambda'_{\boldsymbol{J}}$$

Proof. We consider the case when $M = 2$. For any state $\boldsymbol{J} = (J_1, J_2)$, eigenvalues of independent and interactive parallel implementation, λ and λ' respectively, are

$$\lambda_{\boldsymbol{J}} = \mathbb{P}\left((J_1, J_2) \mid (J_1, J_2)\right) = \left(\int_{B(J_1)} g_1(J_1)dx\right)\left(\int_{B(J_2)} g_2(J_2)dx\right)$$
$$\lambda'_{\boldsymbol{J}} = \mathbb{P}\left((J_1, J_2) \mid (J_1, J_2)\right) = \left(\int_{B(J_1)} g'_1(J_1)dx\right)\left(\int_{B(J_2)} g'_2(J_2)dx\right)$$

respectively. Because of the acceptance-rejection test, it follows that

$$\lambda'_{\boldsymbol{J}} = \begin{cases} \left[\int_{B(J_1)} g(x; J_1)dx\right]\left[\int_{B(J_2)} g(x; J_2)dx\right] & KL(g_1(\boldsymbol{J}), g_2(\boldsymbol{J})) > \eta, \\ & KL(g_2(\boldsymbol{J}), g_1(\boldsymbol{J})) > \eta \\ \left[\int_{B(J_1)} \tilde{g}(x; J_1)dx\right]\left[\int_{B(J_2)} g(x; J_2)dx\right] & KL(g_1(\boldsymbol{J}), g_2(\boldsymbol{J})) < \eta, \\ & KL(g_2(\boldsymbol{J}), g_1(\boldsymbol{J})) > \eta \\ \left[\int_{B(J_1)} g(x; J_1)\right]\left[\int_{B(J_2)} \tilde{g}(x; J_2)dx\right] & KL(g_1(\boldsymbol{J}), g_2(\boldsymbol{J})) > \eta, \\ & KL(g_2(\boldsymbol{J}), g_1(\boldsymbol{J})) < \eta \\ \left[\int_{B(J_1)} \tilde{g}(x; J_1)\right]\left[\int_{B(J_2)} \tilde{g}(x; J_2)dx\right] & KL(g_1(\boldsymbol{J}), g_2(\boldsymbol{J})) < \eta, \\ & KL(g_2(\boldsymbol{J}), g_1(\boldsymbol{J})) < \eta \end{cases}$$

It follows that

$$\begin{aligned}
\lambda'_{\boldsymbol{J}} &\leq \max\left\{\int_{B(J_1)} g(J_1)dx, \int_{B(J_1)} \tilde{g}(J_1)dx\right\} \\
&\quad \times \max\left\{\int_{B(J_2)} g(J_2)dx, \int_{B(J_2)} \tilde{g}(J_2)dx\right\} \\
&\leq \left(\int_{B(J_1)} g(J_1)dx\right)\left(\int_{B(J_2)} g(J_2)dx\right) \\
&= \lambda_{\boldsymbol{J}} \qquad (2.4)
\end{aligned}$$

where the second inequality comes from Equation (2.3). So the eigenvalue associated with an arbitrary state $\boldsymbol{J} = (J_1, J_2)$ is always no greater than that of the independent version. Let $\lambda_{[2]}$ denote the second largest eigenvalue of the single-thread model based search. With M threads, Inequality (2.4) implies that

$$\lambda'_{\boldsymbol{J}} \leq \lambda_{\boldsymbol{J}} = \prod_{k=1}^{M} \lambda_{J_k} \leq (\lambda_{[2]})^M$$

where $\boldsymbol{J}_k = (J_1, J_2 \ldots, J_M)$. Hence, *all* eigenvalues of the interactive model based search are bounded above by $(\lambda_{[2]})^M$. The speed of convergence is exponentially increasing in the number of threads. □

2.4 Conclusion

Model-based algorithms for global optimization have received increased attention recent years. In a model-based algorithm, an optimal solution is estimated at each iteration by sampling candidate solutions from a sequence of probability distributions (or "models") over the feasible region.

In this chapter, we have analyzed a general class of (single-thread) model-based search algorithms. We have showed the speed of convergence (worst-case) is associ-

ated with the "worst" possible combination of locally optimal solutions that could ever be identified. The average speed of convergence is characterized in terms of clusters (aggregation of states). A cluster posits a difficult computational challenge to the model-based search whenever new models generated by the algorithm (given a state in the cluster) do not vary significantly. Thus, the speed of convergence of the single-thread model-based search deteriorates in problems with many difficult clusters whose basins of attraction are relatively large.

We have introduced a new interactive (multi-thread) version of the model-based search method. Interaction takes place through a relatively simple acceptance-rejection. We show that the speed of convergence (both in the worst case and on average) is shown to be increasing exponentially in the number of threads. Numerical examples will be provided to illustrate these results in Chapter 3.

Chapter 3

Numerical Examples for Global Optimization

In this chapter, a number of numerical examples are performed to demonstrate the performance of the interactive model-based search method presented in Chapter 2 with regard to various global optimization problems. We start with a well-illustrated two-dimensional function who contains two separate valleys. Next we move forward to two high-dimensional global optimization problems. The objective functions used in this chapter are generally considered as hard global optimization problems in that they contain many local minima, usually preventing common search methods from discovering the global minimum efficiently. We will compare the algorithm's performance with regard to each of these classical global optimization problems by varying the quantity of computation budgets, i.e. threads.

3.1 Langermann's Function

We first consider a reversed two-dimensional Langermann's function on the two-dimensional solution space of $[0, 10] \times [0, 10]$ in the form of

$$f(\bar{x}) = -\sum_{i=1}^{m} c_i \left(e^{-\frac{\|\bar{x} - A(i)\|^2}{\pi}} \cos\left(\pi \|\bar{x} - A(i)\|^2\right) \right) \tag{3.1}$$

where

$$m = 5 \qquad A = \begin{pmatrix} 3 & 5 & 2 & 1 & 7 \\ 5 & 2 & 1 & 4 & 9 \end{pmatrix}^T \qquad c = \begin{pmatrix} 1 & 3 & 5 & 2 & 3 \end{pmatrix}^T$$

By saying "reversed" Langermann's function, we mean putting a negative sign in front of the original form of Langermann's function, as did in Equation (3.1). The reversion switches all the local maxima to local minima. It helps the simulation process much easier in that our algorithm description is based on minimization (though it also works in maximization).

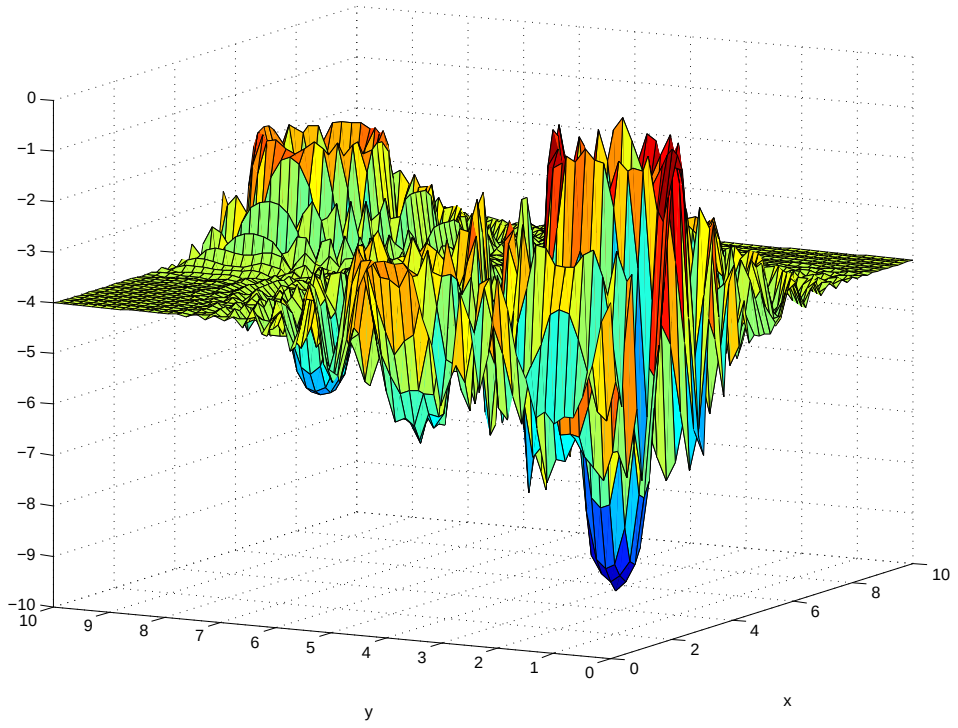

Figure 3.1: Langermann's Function (m=5)

As shown in Figure [3.1], Langermann's Function mainly contains two vast valleys separate from each other throughout the solution space. General model-based algorithms will be easily trapped into the valley that does not contain the globally-optimal solution. It happens mostly when the initial starting points are located in that valley. In fact, a model-based algorithm who concentrates its searching efforts heavily on exploitation is much more likely to wind up with

identifying as the globally-optimal solution the local minimum in the wrong valley. This makes it highly difficult for a model-based algorithm to efficiently discover the global minimum on average, and hence impairs the algorithm's performance. The reason behind is the landscape of Langermann's function, i.e., two separate but large valleys each of which contains some potential local minima.

We use our interactive model-based algorithm present in Chapter 2 in search of the global minimum of the reversed Langermann's function. The parameterized family of models used in the numerical example is the class of multivariate normal distributions with the fixed covariance matrix I. I is the identity matrix. So the varying parameter is the mean of a multivariate normal distribution. At each iteration the algorithm will select an appropriate mean and hence an appropriate multivariate normal distribution from the parameterized family of model. The specific rule to select the "optimal" mean in this numerical example is to minimize the cross-entropy with respect to the reference distribution presented in Section 2.2.1. To ease the computational burden, as we mentioned in Chapter 2, when the original model is rejected according the acceptance-rejection test, another model is randomly selected from the class of sampling distributions. Finally, the local search used in the simulation is BFGS Quasi-Newton method(details can be found in [8]).

Figure [3.2] compares the average performance (based on 1000 simulations) using the parallel implementation of acceptance-rejection test with regard to the reversed Langermann's function. The y-axis in Figure [3.2] represents the lowest value of the Langermann's function the algorithm identified until the current iteration. Note that the y-axis is moved up to make the global minimum equal to zero for a better and clearer visualization.

The figure depicts the average performance of the algorithm when the computational budget is 1 $2, 3, 5$ and 10 thread(s) respectively. When there is only one thread working on the global optimization problem, the convergence speed is shown to be relatively slow without any interaction. It actually fails to discover

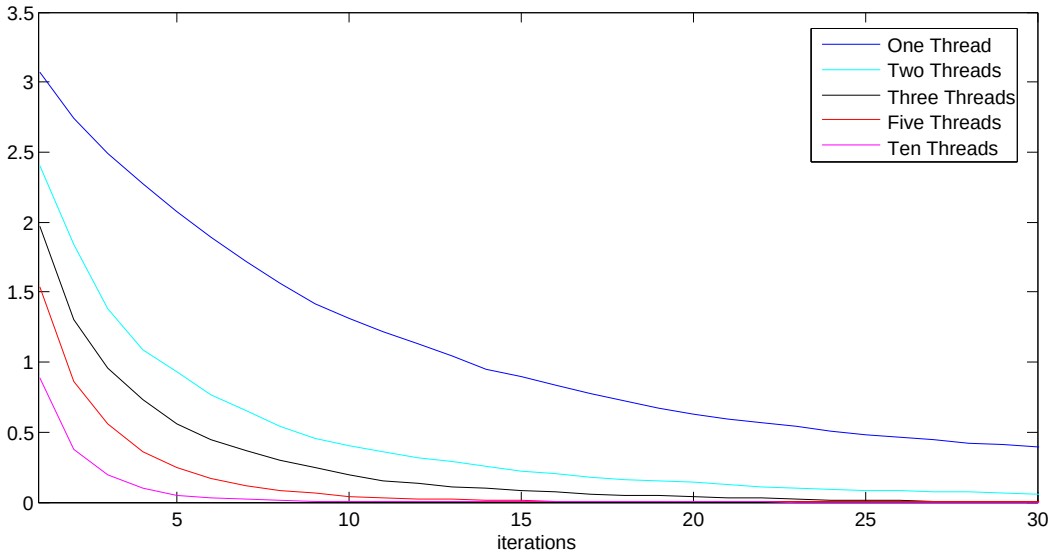

Figure 3.2: Performance with Several Threads

the correct globally-optimal solution within 30 iterations (It does converge and will discover the correct global minimum, expectedly, within 100 iterations. The result is not attached in the dissertation). The single thread was to search the best results using intensive local exploitation on the solution space. Because of the feature of landscape of Langermann's function, i.e., the existence of two distinguishing but large valleys, too much exploitation is likely to lead the thread into the wrong valley and is hard to jump out in shortage of sufficient effort to exploration.

It can be seen that the average performance is significantly improved when the algorithm used two threads instead of one single thread. This change allows the interactive parallel implementation present in Section 2.3 to play a vital role in the balance of exploration and exploitation. As a result of interaction between threads, the lowest value of the objective function identified by the algorithm using 2 threads is generally improved more than 100% compared to the one using a single thread at the same iteration.

With more computational budgets (such as 3, 5 and 10 threads), Figure [3.2] shows that the average speed of convergence is furthermore improved. When the number of threads increased to 10, the algorithm will discover the correct global

minimum within less than 10 iterations.

In the parallel implementation using the acceptance-rejection rule, different threads manage to avoid too much "exploitation", i.e. concentrating search activities in a promising cluster with a relative large basis of attraction. With the number of threads increase, the expected performance is also improved in that more potential bad clusters are avoided. Hence, the average performance of the algorithm performed in Langermann's function is improved with the number of threads increased.

The experiments also suggest that the marginal improvement decreases in the number of threads. There is a huge performance jump from a single-thread implementation to an interactive two-thread implementation. However, the improvement in performance using even more threads is not as dramatic as the case changing from one to two. It demonstrates how important to introduce the interaction implementation into a general model-based algorithm. The performance improvement does not merely comes from increase in the computational budgets, but is largely caused by the introduction of thread interaction in the purpose of making a balance between exploitation and exploration.

3.2 High-dimensional Global Optimization Problems

To see how the interactive model-based algorithm works on high-dimensional solution spaces, we launched our search method on a 10-dimensional solution space. The high-dimensional objective functions used for simulations are specified as follows.

3.2.1 Ackley's path function

The well-known Ackley problem is a minimization problem, originally defined for two dimensions ([9]). It has been generalized to high dimensions in [10]. The objective function, called Ackley's path function, is a highly multi-modal objective

function, containing many small valleys of local minima throughout its entire solution space.

The specific form of a 10-dimensional Ackley's path function is

$$f(\bar{x}) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{10} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{10}\cos(2\pi x_i)} \qquad \bar{x} \in [-10, 10]^{10} \subseteq \mathbb{R}^{10}$$

The function's global minimum is known to be $f(x^*) = 0$ with $x^* = (0, \ldots, 0)$.

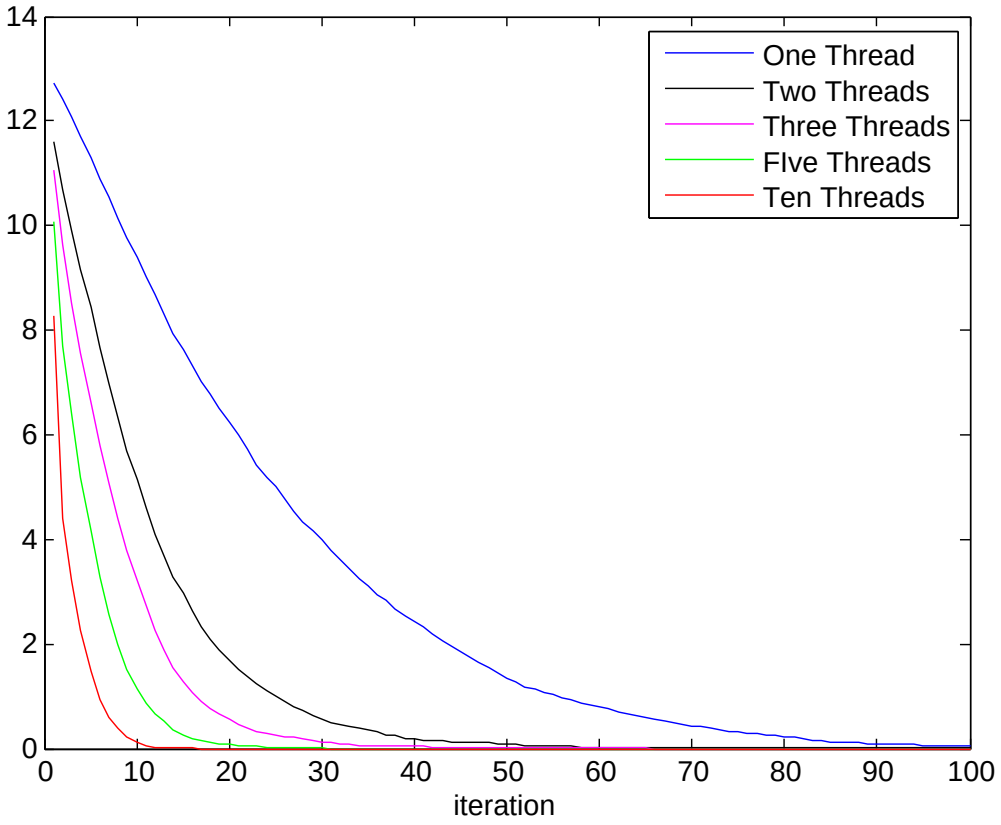

Figure 3.3: Average Performance with regard to 10-Dimensional Ackley's Function

In this numerical example, the parameterized family of models used is the class of multivariate normal distribution with fixed covariance matrix $\frac{1}{5}I$. I is the identity matrix. The local search used for each thread is BFGS Quasi-Newton method (details can be seen in [8]). The specific rule to select the "optimal" mean in this example is to minimize the cross-entropy with respect to the reference distribution presented in Section 2.2.1. Also, to ease the computational burden, as we mentioned in Chapter 2, when the original model is rejected according the

acceptance-rejection test, another model is randomly selected from the class of sampling distributions.

Figure [3.3] demonstrates the results based on 1000 independent simulations. As mentioned, the y-axis represents the lowest value of the objective function identified by the algorithm so far. The Figure basically depicts the same trends as in the two-dimensional Langermann's function in Section 3.1. First, the convergence rate is slow with only one single thread but boosts up radically when the number of threads increases to 2. Second, the performance of the algorithm continues to be improved with the increase in the number of threads.

We also include Figure [3.4] which adds error bars on each point in Figure [3.3]. Those vertically straight lines are the standard deviations of the average performance based on 1000 simulation results. It can be seen that the variances are reduced by increasing the computational budget, i.e., the number of threads.

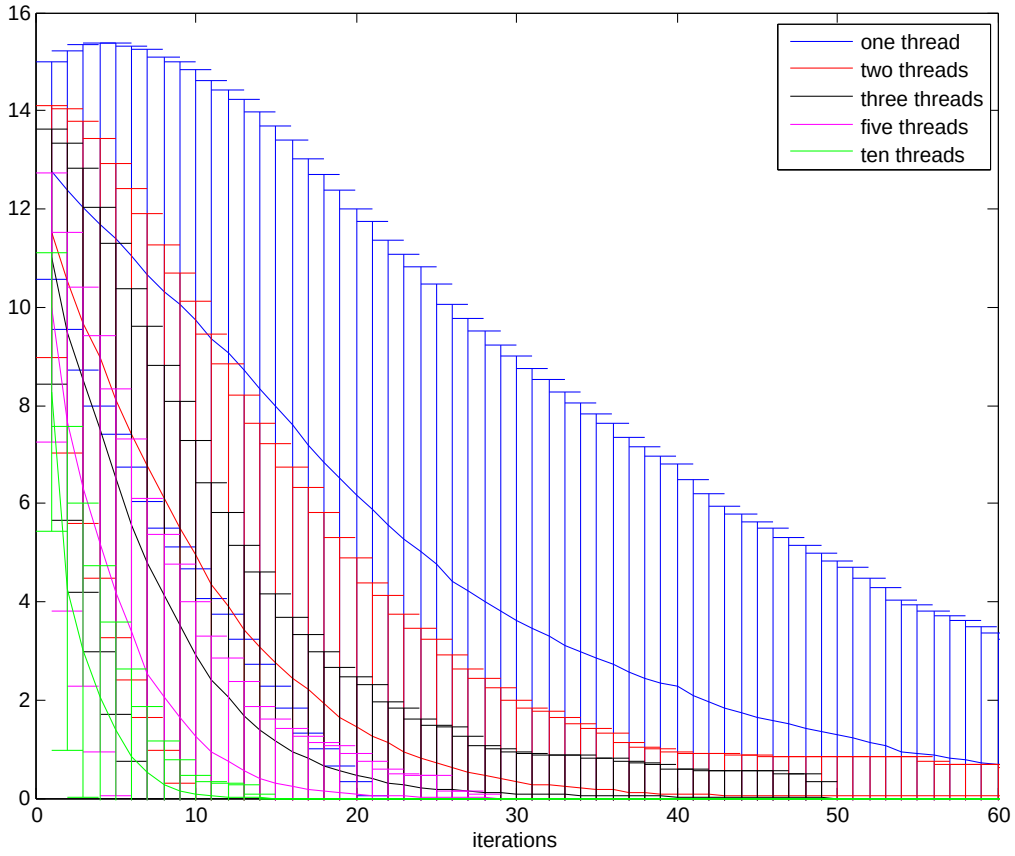

Figure 3.4: Performance Variance with regard to 10-Dimensional Ackley's Function

3.2.2 Griewank's function

The 10-dimensional Griewangk's function is also highly multi-modal. It contains many valleys of widespread local minima, because of the introduction of cosine modulation. The Griewank's function is first brought up by A. O. Griewank in [11].

The exact form of Griewangk's function is as follows.

$$f(\bar{x}) = \sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad \bar{x} \in [-10, 10]^{10} \subseteq \mathbb{R}^{10}$$

The function's global minimum is $f(x^*) = 0$ with $x^* = (0, \ldots, 0)$.

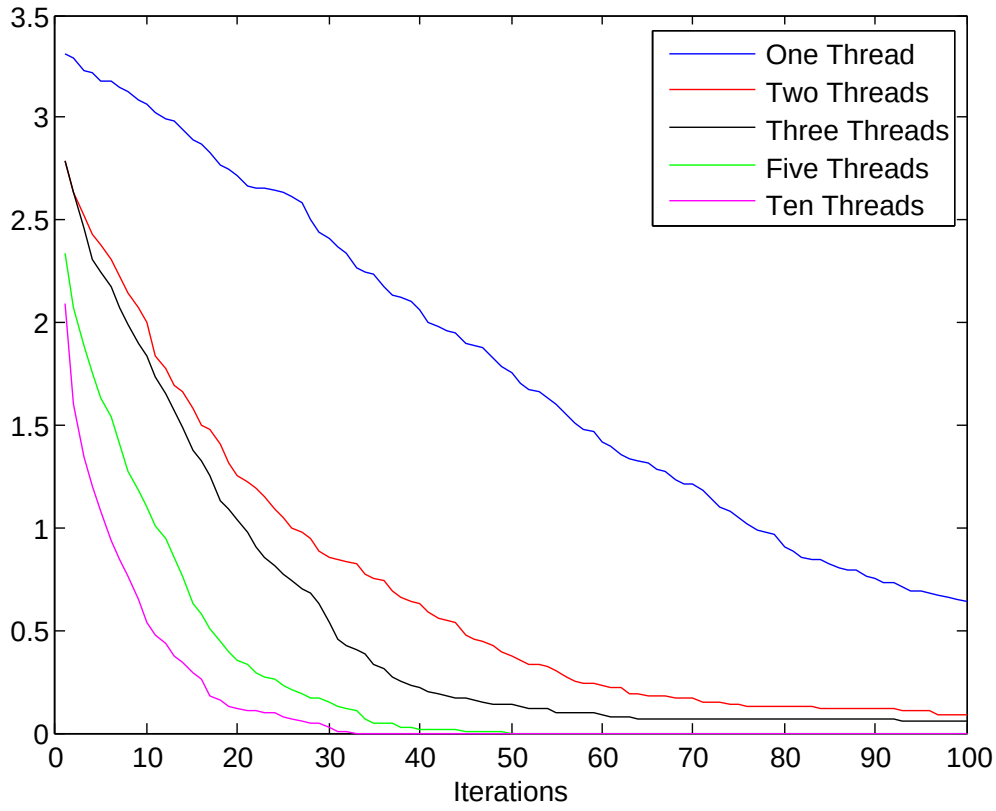

Figure 3.5: Performance with regard to 10-Dimensional Griewangk's Function

The parameterized family of models used in the simulation is the class of multivariate normal distribution with fixed covariance matrix $4I$, where I is the

identity matrix. The results are based on 200 independent simulations. All the other algorithm configuration is the same as that present in Section 3.2.1.

Figure [3.5] shows the trend of the average performance of the interactive model-based search method launched with regard to Griewank's function. It can be seen from the figure that it also follows the basic three trends (see generalization in Section 3.2.1) shown by the 2-dimensional reversed Langermann's function, though the speed of convergence is much slower compared to the low-dimensional case in all the quantities of threads.

3.2.3 Conclusion

Due to the "curse of dimensionality", high-dimensional global optimization problems turn out to be much harder to solve than those in low-dimensional solution spaces. However, it can be seen from our high-dimensional numerical examples of Ackley's path function as well as Griewank's function that, the parallel implementation with acceptance-rejection test of the model-based search method works smoothly and well on high-dimensional solution spaces. In both examples, the algorithm is able reach the true global minimum given sufficient computational budgets (the convergence of Griewangk's function is not completely attached). Especially working with each other interactively by information exchange, the search threads are likely to identify the correct global minimum within several iterations. Both examples suggest that the algorithm's performance is also improved dramatically by varying the quantity of threads. In particular, the speed of convergence of the interactive model-based search method raises intensively compared to the single-thread algorithms. It suggests that exploration on high-dimensional spaces is of great importance to help discover the globally-optimal solution in an efficient way. Also, the madrigal improvement in performance diminishes when the number of threads keep increasing.

Chapter 4

Trust Dynamics in Peer-to-Peer Networks

4.1 Introduction

Advances in information and communications technologies have enabled the proliferation of significant amounts of data in a large number of research areas (see for example [12] and [13]). In today's world, computer networks and cell phones pervade the daily activities of millions. As networked sensors are increasingly being embedded in day to day activities, the amount of data available is growing at a pace that tests the limits of state-of-the-art techniques for data analysis (see for example [14]). The many challenges posed by increasing amounts of data have been labeled in the media as "big data".

Decentralization is an important characteristic of a large number of "big data" networks (see [15]). While the lack of centralized authority allows for scalability, it also makes it hard to guarantee data is reliable (see [16]). For example, Ripeanu analyze performance of the Gnutella protocol, a typical P2P architecture, to demonstrate how the topology graph could impair its reliability (see [17]). Trust or reputation systems have naturally evolved as a potential scalable solution to ensuring reliable data (see for example, Resnick et. al. (2000) [18]). However, there is very little theoretical support for the effectiveness of trust or reputation

management systems (see for example, Sun et. al. (2006) [19] and Jiang and Baras (2006) [20] for notable exceptions).

In this chapter we analyze the dynamics of trust in a social network of data-analyzing agents. A number of algorithms have been proposed and studied to allocate trust amongst agents in the computer science community (see [21]). To facilitate the presentation of ideas we work in a stylized setting in which each agent is fitting a statistical model for an independent and identically distributed stream of data. There are two types of agents: a "good" agent is one that would correctly fit a model (asymptotically) while a "bad" agent is unable to do so. Agents do not know a priori whether they are "good" or "bad". In this framework, a "bad" agent can be interpreted as resulting from either a genuine, random fault or from deliberate malicious behavior.

Periodically, agents share their estimates with their neighbors in a network and individually evaluate the "trustworthiness" of each neighbor's estimate. The choice of trustworthiness has an impact on whether detection on malicious nodes will succeed or not (see [22] and [23]). In our work, we consider a simple rule in which every individual agent would downgrade an neighbor's trustworthiness if his/her estimate differs significantly from most other estimates available to him. We show that the resulting reputation system will converge given the condition that each neighbor's estimate is asymptotically consistent, i.e., either to agree with the majority in a long run, or to constantly differ from most other estimates. We then implement the rule to a more specific form feasible for practical purposes. We show that, using this implementation of "wisdom of crowds", each individual agent will successfully identify all the malicious nodes in his/her neighborhood with probability one when the majority of his neighbors are well-behaved.

Besides analysis of each agent's individual behavior, we also consider how the entire network structure could play a role in network's vulnerability. To achieve this goal, we analyze a widely-observed category of network structure: scale-free networks. We show the conditions when a scale-free network will survive completely,

together with when it will encounter an entire breakdown, on average. In addition, the number of agents that survived after a random attack is also approximately provided.

Our work is related to a recent literature in consensus with faulty interactions (see [24]). However our work differs mainly in how to identify malicious agents present in a network. Pasqualetti's work is to manipulate inputs of agents in order to discover inconsistent behavior of faulty nodes. In our work, the malicious intention will be found through information exchange and trust allocation. Moreover, knowledge of the entire network is necessary for the algorithms used for local detection in Pasqualetti's paper,; while the conditions we propose to ensure successful detection only requires each agent to observe his/her neighbors, regardless of the entire network structure. In a sense, detection and identification characterized in this chapter is more suitable for decentralized networks, where the knowledge of the entire network's connectivity is not completely available.

Our work also differ from the paper by Sundaram et. al. (2008) ([25]). Sundaram et. al. propose parity space methods to accomplish attack detection via linear iterations; while we develop policies of trust allocation in order to identify malicious nodes. By re-allocating trust on neighbors, each individual node will utilize up-to-date information from their neighbors to better observe any sign of misbehavior in the neighborhood. This difference allows us to characterize robustness of some specific networks in a more detailed way. In particular, we analyze agents' behavior in scale-free networks and provide conditions when the entire network would survive from an attack or encounter a breakdown.

The structure of this chapter is as follows. We characterize the reputation system in decentralized networks in Section 4.2. In particular, we provide the conditions under which the reputation system will be convergent. In Section 4.3, we analyze scale-free network's robustness and provide the conditions when scale-free networks will expectedly identify all existing malicious nodes and when they will fail to detect any potential attack. In the last section, we compare the number

of remaining good agents after an attack with respect to three types of network structure: grids, scale-free networks and small-world networks. The number of remaining good agents after an attack reflects the vulnerability of the underlying type of network structure. We show that scale-free networks are most robust to cyber attacks amongst the three types of network structures.

4.2 The Model

Consider a set $\mathcal{N} = \{1, \ldots, N\}$ of agents which are estimating a parameter $\theta \in \{0, 1\}$ based upon independent experimentation. Let $\mu_n^t = \Pr(\theta = 1 | \mathcal{H}_n^t)$ where $\mathcal{H}_n^t = \sigma(X_n^1, \ldots, X_n^t)$ and $\{X_n^1, \ldots, X_n^t\}$ is an i.i.d collection of random experiments conducted by agent $n \in \mathcal{N}$. Agents are of two types. A "good" agent will asymptotically produce the correct parameter estimation, that is,

$$\mu_n^t \to 1_\theta \qquad \text{with probability 1}$$

where $1_\theta = 1$ if $\theta = 1$ and $1_\theta = 0$, otherwise. On the contrary, a "bad" agent is bound to produce incorrect, inconsistent estimations so that there exists an $\eta > 0$

$$|\mu_n^t - 1_\theta| > \eta \qquad \text{eventually}$$

Agents are connected so that we may use a directed graph $G(V, E)$ to represent the network topology. Agents do not know their neighbors' type. They hope to benefit from information shared by neighbors, at the risk of potential exposure to "bad" agents who may provide wrong opinions. We now introduce a procedure to merge estimates of agents in a manner consistent with the "wisdom of the crowd".

4.2.1 Merge Estimates

Assuming estimates are broadcasted at every time period $t \geq 0$, first we consider the following linear combination rules for merging estimates from neighbors.

Assume the set of an agent i's neighbors is $\mathcal{N}_i = \{1, \ldots, N_i\} = \{j \mid e_{ji} = 1\}$. The merged estimate ω_i^t of agent i at time t is

$$\omega_i^t = \frac{\sum_{j \in \mathcal{N}_i} \beta_{ij}^{t-1} \omega_j^{t-1}}{\sum_{j \in \mathcal{N}_i} \beta_{ij}^{t-1}}$$

where β_{ij}^t is a weight assigned to neighbor j by agent i at time t. The ruls of updating weights will be described in Section 4.2.2.

In other words, let $\omega^t = (\omega_1^t, \ldots, \omega_N^t)^T$ be the merged estimates at time t, it is updated as

$$\omega^t = A_{t-1} \omega^{t-1}$$

where $N \times N$ matrix $A_t = (a_{ij}^t)$ is defined as

$$a_{ij}^t = \begin{cases} \dfrac{\beta_{ij}^t}{\sum_{k \in \mathcal{N}_i} \beta_{ik}^t} & j \in \mathcal{N}_i \\ 0 & j \notin \mathcal{N}_i \end{cases}$$

4.2.2 Update Weights

It can be seen that the higher value a_{ij}^t gets, the more confidence agent i puts on this neighbor j. With that being said, it is of importance to design a good mechanism of updating weights in the purpose of eliminating inaccurate information as well as enhancing valuable opinions.

We introduce a stochastic reputation scoring scheme to update weights of neighbors at each time t based on the concept of "wisdom of the crowd". It is involved in two steps. At time t

Step 1 Partition the set of neighbors $\mathcal{N}_i$ into two subsets $\mathcal{N}_i^+(t)$ and $\mathcal{N}_i^-(t)$, i.e.,

$$\mathcal{N}_i^+(t) \cap \mathcal{N}_i^-(t) = \emptyset \text{ and } \mathcal{N}_i^+(t) \cup \mathcal{N}_i^-(t) = \mathcal{N}_i \tag{4.1}$$

In particular, we assume the size of $\mathcal{N}_i^+(t)$ is no smaller than that of $\mathcal{N}_i^-(t)$.

Step 2 Given non-negative functions $h^t(x), g^t(x)$, update β_{ij}^t as follows

$$\beta_{ij}^t = \begin{cases} h^t(\beta_{ij}^{t-1}) & j \in \mathcal{N}_i^+(t) \\ g^t(\beta_{ij}^{t-1}) & j \in \mathcal{N}_i^-(t) \end{cases} \tag{4.2}$$

The specific way to classify the neighbors of agent i into $\mathcal{N}_i^+(t)$ or $\mathcal{N}_i^-(t)$ is flexible and can even be dependent on previous information upon time t. Nevertheless, we need two conditions to ensure convergence.

Condition 4.2.1. *For every neighbor j, it satisfies either*

$$P\left(\bigcap_{t=T}^{\infty}\{j \in \mathcal{N}_i^-(t)\}\right) \to 1 \qquad as \quad T \to \infty \tag{4.3}$$

or

$$P\left(\bigcap_{t=T}^{\infty}\{j \in \mathcal{N}_i^+(t)\}\right) \to 1 \qquad as \quad T \to \infty \tag{4.4}$$

The condition, in some sense, reflects the requirement of information consistency. Since the types ("good" or "bad") of neighbors are determined, it is expected that neighbors are likely to provide similar opinions in the long run given that they are of the same type . Hence, a neighbor will frequently appear in either $\mathcal{N}_i^+(t)$ or $\mathcal{N}_i^+(t)$, but not both asymptotically.

Furthermore, the way to increase or decrease weights is represented by functions h^t or g^t respectively. These functions are also flexible and can be time-varying. However, we do need an intuitive requirement on the weight-updating functions:

Condition 4.2.2. *The non-negative function h^t defined in Step 2 is increasing. Moreover, assuming a series of $\{a_t\}$ is defined by $a_{t+1} = h^t(a_t)$, $\{a_t\}$ is convergent for any initial value in the positive domain of h^0*

Condition 4.2.3. *The non-negative function g^t in Step 2 is decreasing. Assuming a series of $\{b_t\}$ is defined by $b_{t+1} = g^t(b_t)$, $\{b_t\}$ is convergent for any initial value in the positive domain of g^0*

The conditions on weight-updating functions play a role to eliminate the influence from neighbors who are in the minority in the long run, as well as to enhance the trustworthiness of the neighbors who consistently agree with the majority.

4.2.3 Convergence

We show that, under the conditions present in Section 4.2.2, the weights associated with neighbors will converge, regardless of the details of information classification and the functions used for weight updating.

Theorem 4.2.1. $\{\beta_{ij}^t\}$*, the sequence of neighbor* j*'s weights assigned by agent* i*, will converge under Condition (4.2.1), (4.2.2) and (4.2.3).*

Proof. Assume neighbor j satisfies Expression (4.4), we have

$$\begin{aligned} P\left(\lim_t sup\{j \in \mathcal{N}_i^-(t)\}\right) &= 1 - P\left(\lim_t inf\{j \in \mathcal{N}_i^+(t)\}\right) \\ &= 1 - \lim_N P\left(\bigcap_{n=N}^{\infty} \{j \in \mathcal{N}_i^+(t)\}\right) \\ &= 0 \end{aligned}$$

It shows that neighbor j will be classified into set $\mathcal{N}_i^-(t)$ for only finitely many times. It implies there exists a T_1 such that $j \in \mathcal{N}_i^+(t)$ for all $t > T_1$. Due to (4.2) and the assumption on h^t, we conclude that $\lim_{t\to\infty} \beta_{ij}^t = H$. Similarly, we conclude that $\lim_{t\to\infty} \beta_{ij}^t = 0$ as long as neighbor j satisfies Expression (4.3). So β_{ij}^t converges. □

Condition (4.2.1) may be allowed to relaxed into either

Condition 4.2.4. *For every neighbor* j*, either given* $\delta > 0$ *, it satisfies*

$$P\left(j \in \mathcal{N}_i^-(t)\right) \geq \delta \qquad \textit{eventually} \tag{4.5}$$

or it satisfies

$$P\left(\bigcap_{t=T}^{\infty}\{j \in \mathcal{N}_i^+(t)\}\right) \to 1 \qquad as \quad T \to \infty$$

or

Condition 4.2.5. *For every neighbor j, either given $\delta > 0$*

$$P\left(\bigcap_{t=T}^{\infty}\{j \in \mathcal{N}_i^-(t)\}\right) \to 1 \qquad as \quad T \to \infty$$

or

$$P\left(j \in \mathcal{N}_i^+(t)\right) \geq \delta \qquad eventually \tag{4.6}$$

We propose two corollaries that could relax Condition (4.2.1) with the requirement of more assumptions on weight-updating functions.

Corollary 4.2.1. *Assuming $h^t(x) = x$, the weight sequence $\{\beta_{ij}^t\}$ will be convergent if neighbor j satisfies Condition (4.2.4) and (4.2.3).*

Proof. Assume neighbor j satisfies Inequality (4.5). Then there exists a T_0 such that $P(j \in \mathcal{N}_i^-(t)) \geq \delta$ for all $t > T_0$. Construct an "imaginary" agent r in the following way: at time $t > T_0$ this agent will be classified into $\mathcal{N}_i^-(t)$ with probability δ. Then the series of events $\{r \in \mathcal{N}_i^-(t)\}_{t>T_0}$ are independent and $\sum_{t>T_0} P\left(\{r \in \mathcal{N}_i^-(t)\}\right) = \infty$. By applying the second Borel-Cantelli lemma ([26]), it follows

$$P\left(\lim_t sup\{r_t \in \mathcal{N}_i^-(t)\}\right) = 1$$

Note that $P\left(\{j \in \mathcal{N}_i^-(t)\}\right) \geq P\left(\{r \in \mathcal{N}_i^-(t)\}\right) = \delta$ for all $t > T_0$. Then we have

$$P\left(\lim_t sup\{j \in \mathcal{N}_i^-(t)\right) = 1$$

which implies that neighbor j will be classified into set $\mathcal{N}_i^-(t)$ for infinitely many t. Note that neighbor j's weight β_{ij}^t is unchanged if $j \in \mathcal{N}_i^+(t)$ but will be deducted based on g^t if $j \in \mathcal{N}_i^-(t)$. Then $P(\lim_t sup\{j \in \mathcal{N}_i^-(t)\}) = 1$ implies $\lim_t \beta_{ij}^t = 0$ under Assumption (4.2.3). The other case is directly from Theorem (4.2.1). □

Similarly, we have another relaxation as follows. Just as the previous corollary, it propose an exact form of the weight-deduction function g^t in order to relax Condition (4.2.1).

Corollary 4.2.2. *Assuming $g^t(x) = x$, the weight sequence $\{\beta_{ij}^t\}$ will be convergent if neighbor j satisfies Condition (4.2.5) and (4.2.2).*

4.2.4 Illustration

There are various ways to implement the set partition and to choose the score-updating functions in Section 4.2.2. One approach to illustrate the concept of "wisdom of the crowd" is as follows.

For example, at each iteration t, each agent n produces a sample estimate $\tilde{\theta}_n^t$ such that

$$\tilde{\theta}_n^t = \begin{cases} 1 & \text{with probability } \mu_n^t \\ 0 & \text{with probability } 1 - \mu_n^t \end{cases}$$

For each agent i's neighborhood, let

$$\mathcal{A}_0^t(i) = \{\, n \in \mathcal{N}_i |\, \tilde{\theta}_n^t = 0\} \qquad \mathcal{A}_1^t(i) = \{\, n \in \mathcal{N}_i |\, \tilde{\theta}_n^t = 1\}$$

Then define the majority set and the minority set such that

$$\mathcal{N}_i^+(t) = \begin{cases} \mathcal{A}_0^t & |\mathcal{A}_0^t| > |\mathcal{A}_1^t| \\ \\ \mathcal{A}_1^t & |\mathcal{A}_0^t| \leq |\mathcal{A}_1^t| \end{cases} \qquad \mathcal{N}_i^-(t) = \begin{cases} \mathcal{A}_1^t & |\mathcal{A}_0^t| > |\mathcal{A}_1^t| \\ \\ \mathcal{A}_0^t & |\mathcal{A}_0^t| \leq |\mathcal{A}_1^t| \end{cases} \tag{4.7}$$

It is obvious to see that this definition of partition on $\mathcal{N}_i$ satisfies the equations (4.1) in Step 1. Similar definition can be easily made when the true state of θ is $\theta = 0$.

As for the weight-updating function used for lowering the weights of minority in Step 2, we simply apply

$$h^t(\beta_{ij}^{t-1}) = \beta_{ij}^{t-1} \qquad \text{and} \qquad g^t(\beta_{ij}^{t-1}) = \alpha\beta_{ij}^{t-1} \quad \text{for some } \alpha \in (0,1)$$

They clearly satisfy Assumption (4.2.2) and (4.2.3).

In this scenario, we show that the merged estimates are convergent, and the limits are consistent with the opinion of the majority given a complete network.

Theorem 4.2.2. *ω^t, the vector of the merged estimates, converges to 1_θ w.p. 1. as long as the majority are "good" agents and the network is fully connected.*

Proof. For simplicity, we assume the true state of θ is $\theta = 1$. Since the network is fully connected, it is reasonable to concentrate on any single agent i. Define

$$E = \{n \in \mathcal{N}_i | \lim_{t\to\infty} \mu_n^t = 1\}$$
$$F = \{n \in \mathcal{N}_i | \ \left|\mu_n^t - 1_\theta\right| > \epsilon, \text{ eventually}\}$$

Since the number of neighbors is finite, there exists a uniform T_1 such that for $t > T_1$

$$\mu_j^t < 1 - \eta$$

for $j \in F$. And similarly, given $\epsilon > 0$, there exists a uniform T_2 such that for $t > T_2$

$$\mu_k^t > 1 - \epsilon$$

for $k \in E$. Note the majority of neighbors are "good", i.e., $|E| > |F|$. Hence, for a"bad" agent $j' \in F$

$$P\left(j' \in \mathcal{N}_i^-(t)\right) \geq \prod_{k\in E} \mu_k^t \prod_{j\in F} \left(1 - \mu_j^t\right) \geq (1-\epsilon)^{|E|} \eta^{|F|}$$

for $t > \max\{T_1, T_2\}$. Consider $\delta = (1-\epsilon)^{|E|}\,\eta^{|F|}$, then any "bad" agent in the neighborhood satisfies Inequality (4.3) in Condition 4.2.1. Now for a "good" agent $k' \in E$, also note that the partition of $\mathcal{N}_i^-(t)$ and $\mathcal{N}_i^+(t)$ is independent at each time. Hence,

$$P\left(\bigcap_{t=T}^{\infty}\{k' \in \mathcal{N}_i^+(t)\}\right) = \prod_{t=T}^{\infty} P\left(k' \in \mathcal{N}_i^+(t)\}\right) \geq \prod_{t=T}^{\infty}\prod_{k\in E}\mu_k^t \geq \prod_{t=T}^{\infty}(1-\epsilon)^{|E|}$$

when $T \geq \max\{T_1, T_2\}$. Since ϵ is arbitrary, it follows that

$$P\left(\bigcap_{t=T}^{\infty}\{j \in \mathcal{N}_i^+(t)\}\right) \to 1 \qquad \text{as} \quad T \to \infty$$

i.e., any good agent satisfies Inequality (4.4) in Assumption 4.2.1. Therefore, Corollary 4.2.1 implies that $\lim \beta_{ij}^t = 0$ if $j \in F$ and $\lim \beta_{ik}^t = \beta_k$ if $k \in E$. Then the merged estimates ω_t are convergent to 1_θ □

4.3 Robustness Analysis in Scale-Free Networks

Given a network $G = (V, E)$ of size N, assume each node is likely to be malicious with equal probability q, the question naturally arises how a well-behaved node in the network will be affected by those potential bad nodes using the implementation present in Section 4.2.4.

First of all, it is mindful to point out that whether or not a node will succeed to detect malicious behavior could be a random variable. However, based on Theorem 4.2.2, node i will find out all of his bad neighbors with probability 1 if and only if

$$d_i + 1 > 2p_i \tag{4.8}$$

where d_i is the degree of node i and p_i is the number of bad nodes connected to node i.

Now we discuss the condition that a node will expectedly detect all of its malicious neighbors given a scale-free network.

4.3.1 Distributions of Degrees and Malicious Nodes

Assume a scale-free network is generated by Barabasi-Albert procedure ([27]) using a complete network with N_0 initial nodes. The procedure adds a new node with $\Delta m (\Delta m \leq N_0)$ links (that will be connected to the nodes already present in the system) each time until the desired network size N is reached. How each link of the new node is constructed follows the preferential attachment, i.e., the probability with which a new node at time t will be connected to node i is $\frac{d_i^t}{\sum d_j^t}$.

Assume d_i^t is node i's degree at time t, then for $i > N_0$, we define

$$d_i^t = \begin{cases} 0 & t < i - N_0 \\ \Delta m & t = i - N_0 \end{cases} \tag{4.9}$$

As for $i \leq N_0$, it is obvious

$$d_i^0 = N_0 - 1 \tag{4.10}$$

because the initial network is complete.

Lemma 4.3.1. *Expectation of the number of node i's bad neighbors, p_i^t, is equal to expectation of node i's degree d_i^t at the same time, multiplied by the probability of a node to be malicious, q, i.e.,*

$$\mathbb{E}[p_i^t] = q\mathbb{E}[d_i^t] \tag{4.11}$$

Proof. According to the preferential attachment, the expectation of node i's degree d_i^{t+1} at time $t+1$ conditioning on the previous degree d_i^t is as follows

$$\begin{aligned} \mathbb{E}[d_i^{t+1} \mid d_i^t] &= d_i^t \left(1 - D_i^t\right)^{\Delta m} + \left(d_i^t + 1\right) \left(1 - \left(1 - D_i^t\right)^{\Delta m}\right) \\ &= d_i^t + 1 - \left(1 - D_i^t\right)^{\Delta m} \end{aligned}$$

where $D_i^t := \frac{d_i^t}{N_0(N_0-1)+2\Delta mt}$ is the probability that node i is connected to the newly-added node N^t for each link attachment (totally Δm links). Using Expression (4.9) and (4.10) as initial values, the degree expectation is

$$\mathbb{E}[d_i^{t+1}] = d_i^{\max\{0,i-N_0\}} + \mathbb{E}\left[\sum_{j=\max\{0,i-N_0\}}^{t}\left(1-\left(1-D_i^j\right)^{\Delta m}\right)\right] \tag{4.12}$$

where

$$d_i^{\max\{0,i-N_0\}} = \begin{cases} \Delta m & i > N_0 \\ N_0 - 1 & i \leq N_0 \end{cases}$$

Since each newly-added node could be bad with probability q as assumed, the expected number of bad neighbors of node i at time $t+1$ conditioning on d_i^t and p_i^t is

$$\mathbb{E}\left[p_i^{t+1} \mid d_i^t, p_i^t\right] = p_i^t + q\left(1-\left(1-D_i^t\right)^{\Delta m}\right)$$

Note the initial values of p_i^t are $p_i^{\max\{0,i-N_0\}} = qd_i^{\max\{0,i-N_0\}}$ because each node already presenting in the system is likely to be bad with probability q as well. Then, it concludes that

$$\mathbb{E}[p_i^{t+1}] = p_i^{\max\{0,i-N_0\}} + q\mathbb{E}\left[\sum_{j=\max\{0,i-N_0\}}^{t}\left(1-\left(1-D_i^j\right)^{\Delta m}\right)\right] = q\mathbb{E}[d_i^{t+1}]$$

□

4.3.2 Reliability of Scale-Free Networks

By Inequality (4.8) in the lemma, node i will expectedly detect all his malicious nodes if the desired network size N is reached (note $N = T + N_0$) if

$$\mathbb{E}[d_i^T] + 1 > 2\mathbb{E}[p_i^T]$$

Using Equation (4.11), we get

$$(2q-1)\mathbb{E}[d_i^{N-N_0}] < 1$$

It reaches the conclusion that node i will expectedly detect all the malicious neighbors when the network size is $N = T + N_0$ if

$$q \leq \frac{1}{2} \tag{4.13}$$

or

$$\mathbb{E}[d_i^{N-N_0}] < \frac{1}{2q-1} \quad q > \frac{1}{2} \tag{4.14}$$

where $\mathbb{E}[d_i^{N-N_0}]$ can be evaluated from Equation (4.12).

Thus, the characteristics, especially preferential attachment, of a scale-free network topology enables us to propose a theorem that provides the conditions under which the network will succeed or fail. Moreover, it also gives an approximation on the number of good nodes that will survive after a malicious attack.

Theorem 4.3.1. *Assume a scale-free network of size N is generated by BA procedure with the initial complete network of size N_0. Also assume the total link each new node adds to the system, Δm, is no more than N_0. Given the malicious rate q (the probability that a node is malicious), all the nodes in the network will expectedly succeed to detect all their malicious neighbors if*

$$q < \frac{1}{2N} + \frac{1}{2} \tag{4.15}$$

When $q \geq \frac{1}{2N} + \frac{1}{2}$, the expected number of nodes that will detect all their malicious neighbors is approximately

$$M = 2N\Delta m(\Delta m + 1) \sum_{k=\Delta m}^{\lceil \frac{1}{2q-1} \rceil - 1} [(k+2)(k+1)k]^{-1} \tag{4.16}$$

Especially, the network is expected to fail to detect any malicious node if

$$q \geq \frac{1}{2\Delta m} + \frac{1}{2} \tag{4.17}$$

Proof. Assume the degree distribution sequence is $h(k)$, i.e., the number of nodes with degree k is $h(k)N$. Obviously $h(k) = 0$ when $k < \Delta m$ or $k > N$. Condition (4.14) indicates, when $q > \frac{1}{2}$, the total number of nodes that will expectedly detect all their malicious neighbors is

$$M := \sum_{k=1}^{\lceil \frac{1}{2q-1} \rceil - 1} h(k)N = \sum_{k=\Delta m}^{\lceil \frac{1}{2q-1} \rceil - 1} h(k)N \tag{4.18}$$

It implies two extreme cases. First, $M = 0$ when $\Delta m > \lceil \frac{1}{2q-1} \rceil - 1$. Second, $M = N$, i.e. the the total number of nodes that can detect malicious nodes equals the network's size, when $N \leq \lceil \frac{1}{2q-1} \rceil - 1$. Due to Inequality (4.13) and (4.18), the two extreme cases result in Inequality (4.15) and Inequality (4.17). As for Equation (4.16), notice the fact (see [27]) that a scale-free network generated by BA procedure will approximately have a degree distribution sequence

$$h(k) = \begin{cases} 2\Delta m \dfrac{\Delta m + 1}{(k+2)(k+1)k} & k \geq \Delta m \\ 0 & k < \Delta m \end{cases}$$

Then, Equation (4.16) can be deduced from Condition (4.18). □

Theorem 4.3.1 suggests several implications of a scale-free network's vulnerability. First, a scale-free network is relatively robust in that it is expected to detect any malicious node if the probability each node becomes bad is less than $\frac{1}{2N} + \frac{1}{2}$, which is pretty costly for a hacker.

Second, a high-degree node (usually called a "hub" in a scale-free network) is more likely to fail to detect his malicious neighbors compared to small-degree nodes according to Inequality (4.14). That is to say, hubs are more vulnerable than

other nodes in a scale-free network. To reduce the chance of network breakdown, the condition suggests that it requests more security strengths on hubs.

Third, vulnerability of a scale-free network will be reduced if connectivity (Δm) increases. Thus, robustness can still be improved by raising network's connectivity even if the size of the network and the malicious rate are both fixed.

4.4 Network Vulnerability

Not only does Theorem (4.2.1) provide conditions on convergence but it also sheds insight into how network topology plays a role in network's resilience to malicious hacking.

To characterize the aftermath malicious nodes may cause, we first introduce the concept which defines the influence of malicious nodes to the opinion forming of a network. Due to the complexity of network topology and information exchange, it is possible that whether a "good" node will be affected by his malicious neighbor(s) remains stochastic. For example, a "good" node becomes"bad" with one-half chance and stays "good" with another one-half chance if exactly half of his neighbors(including himself) are malicious nodes in a fully-connected network using the implementation described in Section 4.2.4. Thus, the opinion evolution converges to a random variable other than a deterministic limit. For simplicity, we focus on the "good" nodes whose opinions will deterministically switch to bad ones.

Definition 4.4.1. *The smallest bad component $S(G, B_M)$ of a network $G = (\mathcal{N}, E)$, given the set of M initially bad agents $B_M = \{n \in \{n_1, \ldots, n_M\} \subseteq \mathcal{N} \mid |\mu_n^t - 1_\theta| \nrightarrow 0\}$, is the set of agents whose limits of opinions will deterministically be bad. That is, $S(G, B_M) = \{n \in \mathcal{N} \mid |\omega_n^t - 1_\theta| \nrightarrow 0\}$, where μ_n^t and ω_n^t are individual estimates and merged estimates of agent n at time t, respectively.*

It can be seen that the size of $S(G, B_M)$ provides a lower bound of how a malicious attack may impair the network. Furthermore, the size of the smallest

bad components varies in the location and the size of initially bad agents assigned to the network. That is, B_M will determine the value of $S(G, B_M)$ if the network is fixed. Although the size is as important as the location, the number of initially bad agents an attack is able to assign to the network is somewhat restricted due to cost budgets and other realistic concerns. It is natural for a hacker to choose the best locations of malicious nodes to optimize the aftermath of an attack if the size of nodes he can attack is fixed. That is, given a network G, a hacker is interested in finding out an optimal form of attack B_M^* such that $S(G, B_M^*) = \max_{B_M} S(G, B_M)$ when M is fixed.

Given a network G, we denote the largest size of the smallest bad components under M initially malicious nodes as

$$S^*(M) = S(G, B_M^*) = \max_{B_M} S(G, B_M) \tag{4.19}$$

We use Expression (4.19) to characterize the vulnerability of the given network G in terms of how many nodes will be attacked definitely and eventually given M initially "bad" agents.

Using the implementation described in Section 4.2.4, we run simulations on three types of networks to see how a network's structure may have impacts on its vulnerability to a potential attack. Each network contains 25 agents. The types of these networks are lattice, small-world and scale-free, respectively. The small-world network is generated by the Watts-Strogatz (WS) procedure ([27]) from a 5-regular network with rewiring probability $p = 0.2$. The scale-free network is generated by Barabasi-Albert (BA) procedure ([27]) from a 3-node complete network with $\Delta m = 2$.

Figure (4.1) depicts the simulation results by showing The relationship between M and $S^*(M)$ for the three types of network structure.

When the number of malicious nodes ($M \leq 4 << 25$) is much smaller than the network size, the scale-free network is slightly more vulnerable than both the small-world network and the lattice network. It arises from the notable characteristic

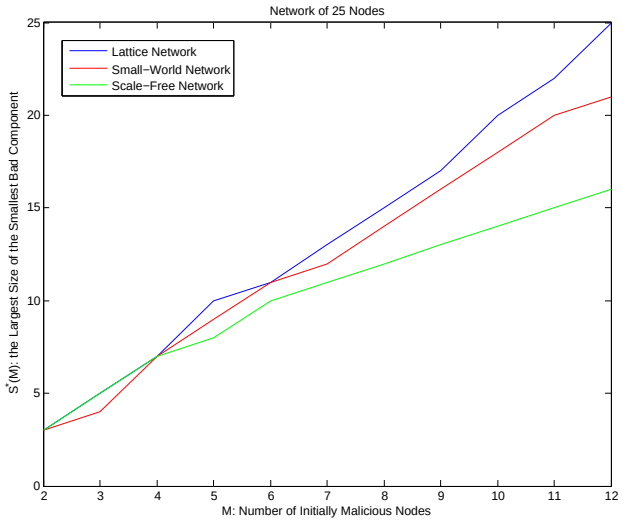

Figure 4.1: Network Vulnerability under Three Types of Network Structure

of scale-free networks, called "preferential attachment", which results in a small amount of nodes with a degree that greatly exceeds the average. Thus, attacking those "hubs" directly will affect more nodes in their neighborhood, compared to small-world networks and lattice networks. There does not exist those "hubs" in the latter two.

However, when the total number of malicious nodes increases, things turns around. As we can see, the lattice network is most vulnerable when facing massive attacks, followed by the small-world network. The scale-free network turns to be the most robust one. The results are consistent with other researchers' observations on Internet breakdowns ([28] and [29]).

There are several aspects that contributes to the simulation results. The first aspect is randomness of a network. In graph theory, randomness of a network is represented by the network's entropy:

$$I = \sum_{i=1}^{\text{max degree}} h_i \log(h_i)$$

where $h = (h_i)_{i=1,\ldots,\text{max degree}}$ is the degree sequence distribution of a network. Scale-free networks and small-world networks are generally considered as more "random" than a structured network, such as a lattice, in that they have a relatively

higher bits of entropy. For example, in our simulation the entropies of three types of networks are as follows:

	lattice	small-world	scale-free
entropy	1.0100	1.5651	1.6697

It can be seen that both the small-world network and the scale-free network contain more bits of entropy than the lattice, which makes them less structured. A less structured network is hard to be attacked in that it generally does not lose its connectedness if several nodes do not behave properly. Connectedness to other nodes in a network is the key for information exchange and aggregation. Failure that occurs in a more random network is less likely to have a dramatic impact on the entire network. Thus, both small-world networks and scale-free network are more robust than grids.

The second aspect that distinguishes the behavior of scale-free network from the other two is that a scale-free network's degree distribution asymptotically follows a power law. This characteristic of scale-free networks allows nodes to be arranged hierarchically. On one hand, most nodes have negligible impacts on the entire network because of their small degrees. On the other hand, even if several hubs are malicious, they will break the network into several isolated components amongst which information exchange becomes unavailable. Thus, rumors stop spreading out because of shortage of the network connection. This property of scale-free networks significantly contributes to their robustness to malicious attack.

At last, we need to point out that the individual degree of nodes is not the critical part to the network vulnerability. For instance, the lattice network of 25 nodes has an average degree of 4.2, while the small-world network is of 5, and the scale-free network is of 2.9. In fact, it is more likely to be how values of degrees are distributed among the nodes of a network that makes main contributions to network robustness. As we can see, entropy is based on the degree sequence distribution, indicating randomness of a network. Also, a scale-free network's extraordinary behavior against malicious attacks also comes from its degree hierarchy. With that

being said, the network robustness is determined by how nodes are connected with each other throughout the network.

Chapter 5

Case Study: Cyber-security Application to Wind Farms

Acknowledgments: This chapter would not have been possible without the work accomplished by Nathan Trantham, the graduate student in the department of Systems and Information Engineering at University of Virginia. Mr. Trantham explored the mechanics of general wind farms and run a vast number of simulations with the implementation of the distributed model-based algorithm discussed in Chapter 4. *I share the credit of my work, especially which is present in this chapter, with Nathan Trantham.*

5.1 Background

A wind turbine is a device designed to capture the kinetic energy available in wind and then convert this energy to usable electrical energy through a generator (see [30]). The operation of a wind turbine is based on the time averaged wind speed. Wind speed is broken up into three regimes: below cut-in speed , allowable wind speed, and above cut-out speed. When the averaged wind speed is below the cut-in speed or above cut-out speed, the turbine is commanded to remain in an idle state. When the wind speed is in between these thresholds, then the turbine is free to spin and generate electricity (see [31]). Despite the insignificant time

amount used in transitory states, it can be seen that a turbine is generally in one of the two primary stationary states: the idle state and the generative state.

In a common offshore wind farm (e.g. the Horns Rev wind farm in Denmark), a large number of turbines are grouped closely together (see [32]). The aggregate nature of wind turbines within a wind farm gives rise to an interesting and complex situation concerning the movement of wind through a wind farm. Temporary disagreements in state amongst turbines could happen due to the natural stochastic fluctuation in wind.

In a scenario where the initial wind speed is zero, as wind moves toward the wind farm it first will reach the most upwind array of turbines. Once the wind speed has surpassed the wind speed cut-in threshold for the generation state of the turbines supervisory controller, then that first array of turbines will begin to produce power. If the wind speed is near the threshold value, then the wind deficit created by the operation of the first array of turbines will be sufficient to not allow subsequent downwind arrays to operate. Only after the wind speed is high enough to compensate for all of the wind deficits from wind turbine wakes will the entire wind farm be in the generation state.

The finite-state machine structure of a wind turbines supervisory controller and the spatiotemporal coherence of wind speed within a wind farm make our model and the corresponding results present in Chapter 4 applicable to help detect cyber-security attack targeting at a wind farm. In particular, a dynamic reputation scheme is advantageous in this application in that the stochastic nature of the dynamic system governing the states of the turbines (i.e. wind).

5.2 Trust Dynamics Algorithm

As mentioned in Section 5.1, we assume there are two states in which each turbine could be: the idle state and the generative state. Each turbine will determine which state to stay based on the wind speed it detects on a regular basis and will switch between the two states when the wind speed changes.

In addition, it will be assumed that the turbines within the wind farm are spaced in a regular, grid-like manner as they are in virtually all large offshore wind farms (e.g. Horns Rev and Nysted). The eight turbines surrounding a central turbine will be defined as that turbines neighborhood and will be used for local collaboration and interaction. Turbines that reside on an edge of the farm will interact with the adjacent turbines that do exist. The set of the neighbors of turbine i is denoted by $\mathcal{N}_i$. Figure (5.1) visually depicts the neighborhoods of node 5 and 10.

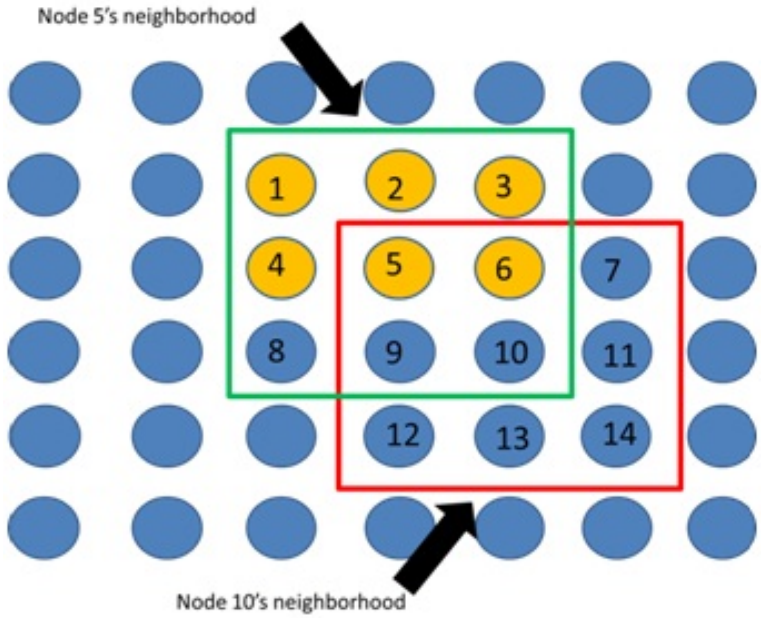

Figure 5.1: illustration of the neighborhood definition in a wind farm

Also, each turbine i has a reputation score γ_i^t bounded between 0 and 100 that is effectively an indicator of how often it agrees in state with the states of turbines nearest in physical proximity. Turbines initially are assigned a perfect reputation score of 100, but have their scores manipulated at regular intervals which are the same as those when they determine their states on a regular basis. A threshold reputation score is defined such that, when a particular turbine's reputation score drops below the said threshold, it will be deemed anomalous, indicating that a particular turbine has operated in such a suspicious manner in context to the operation of the turbines located physically nearby. In this way, the locally collaborative reputation scheme relies on the "wisdom of crowds" effect:

if a turbine is consistently in a different state from its neighboring turbines, then there is reason to suspect malfunctioning of that turbine.

Assuming turbine i's current state at time t is $S_i^t \in \{idle, generative\}$, define

$$s_i^{t+1} = \gamma_i^t + \sum_{j \in \mathcal{N}_i} c_j^t$$

where c_j is neighboring turbine j's contribution defined as

$$c_j^t = \begin{cases} \frac{b\gamma_j^t}{100} & S_j^t = S_i^t \\ \frac{p\gamma_j^t}{100} & S_j^t \neq S_i^t \end{cases}$$

we will update turbine i's reputation score by

$$\gamma^t = \begin{cases} 100 & \text{if } s_i^t > 100 \\ s_i^t & \text{if } 0 < s_i^t < 100 \\ 0 & \text{if } s_i^t < 0 \end{cases}$$

The reputation scoring updating scheme indicates that a turbine's reputation score at time $t+1$ is its reputation score at time t plus the contributions from its neighbors. A neighboring turbine's contribution is the product of the nominal penalty or bonus value times its scaled reputation score at previous iteration. By weighting the contribution of a neighbor turbine according to its reputation score, the efficacy of untrustworthy turbines (those with low reputation scores) is reduced. The values of nominal bonus b and penalty p can be unequal and are up to the user to set. For a more aggressive algorithm that can detect corruption quicker, but with the tradeoff of a higher false detection rate, the nominal penalty value can be set to a higher value than the nominal bonus value. The absolute and relative values of the nominal bonus and penalty values affect the behavior of the algorithm and this behavior is discussed more in the next section.

5.3 Simulation

To test the performance of the trust dynamics algorithm under various wind farm configuration, it is necessary to develop a realistic simulation for wind farms. In particular, the spatiotemporal profile of wind turbine states is required to complete the simulation configuration. The profile requires two basic information: a time series of wind speed and the spacial orientation of all of the turbine within a wind farm.

5.3.1 Simulation Development

In this simulation, we use the Horns Rev wind farm in Denmark as the test bed for the simulation of the trust dynamics algorithm. In the Horns Rev wind farm, the turbines are located in an 8 row $\times$ 10 column rectangular grid with 560 meters of spacing ([32]), as illustrated in Figure [5.2].

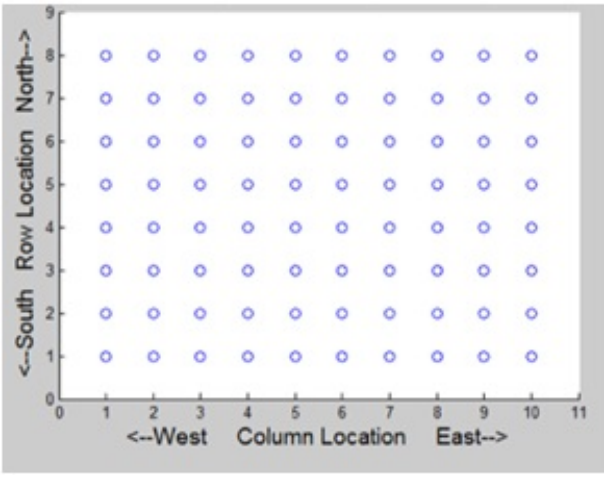

Figure 5.2: Illustration of Wind Turbine Spacing

As for the time series of wind speed for each turbine, we first generate a time series of the free stream wind speed prior to interaction with the wind farm. According to the data provided by [33]., wind speed in the Horns Rev wind farm is characterized by the Weibull distribution in Figure [5.3]. Because of the autocorrelation nature of wind, we use HOMER, a software tool developed by the National Renewable Energy Laboratory (NREL), to generate an appropriate and autocorrelated time series of wind speed as an input to the simulation. Then

we modified the time series of free stream wind speed for each turbine. The modification is based on two facts. First, the time series must be temporally shifted to compensate for the time required for wind to propagate through the wind farm. Therefore, the wind speed time series for each wind turbine must be shifted according to its location within the wind farm. Second, the wind speed needs to be adjusted to compensate for the wind deficits caused by turbine wakes. As discussed in Section 5.1, downstream turbines will experience a lower mean wind speed than their upstream counterparts.

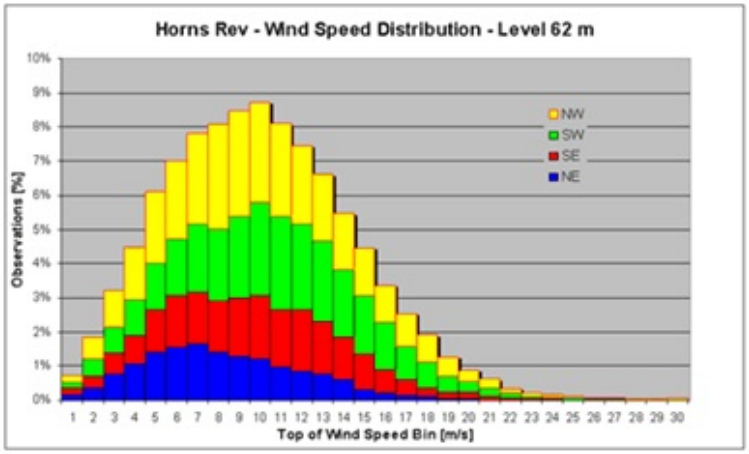

Figure 5.3: Weibull Distribution of Free Stream Wind Speeds

Taken into account these facts, a time series of wind speed for each turbine will be modified based on the time series of free stream wind speed generated by HOMER. And we use these as inputs for the simulation.

With the spatiotemporal profile of wind turbine states at hand, the trust dynamics algorithm can be applied. Initially at the first time step, it performs reputation score manipulations according to the algorithms details. Once the algorithm completes a particular time slice, it stores the reputation information and proceeds to the next time slice. This process is repeated iteratively until the entire profile is complete or it reaches a user-defined stop criterion (e.g. stop the simulation when X number of turbines have been identified).

5.3.2 Simulation Results

A variety of attack scenarios were tested to display the algorithm's behavior in various corruption contexts. Specifically, the magnitude of the parameter shift, location of corrupted turbines within the wind farm, and the quantity of corrupted turbines were the test variables manipulated to display the algorithm's performance in a variety of attack scenarios. The simulation was run for 160 iterations (800 minutes) where each iteration is 5 minutes. For the simulation results presented below, the algorithm's nominal bonus and penalty values were 4 and 8, respectively. The metrics presented to evaluate the algorithm's performance are false positives and false negatives.

Figure [5.4] demonstrates the algorithm's baseline performance, i.e., there is no corruption within the wind farm. The image on the right of Figure [5.4] depicts the locations of corrupted turbines. Blue circles indicate non-corrupted turbines. From the baseline simulation, it can be seen that when there are no corruptions to the wind farm the algorithm produces no false positives or false negatives.

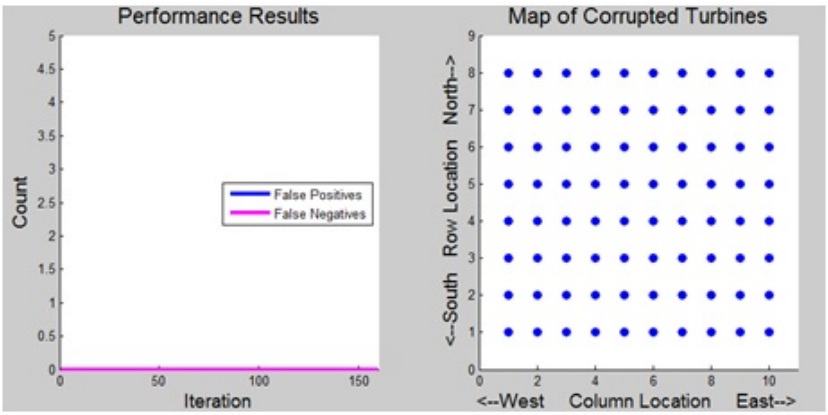

Figure 5.4: Baseline Simulation Results

Now we randomly generate 5 corrupted turbines within the wind farm, by changing the wind speed cut-in threshold from 4m/s (normal) to 12m/s (corruptive). Red squares in the right image of Figure [5.5] indicate the locations of corrupted turbines. From the performance results, it can be seen that the first few time steps produce 5 false negatives. This is normal behavior as it takes multiple iterations for individual turbines to have their reputation score decremented past the corruption

threshold score of 50. The additional false negative around the 60th iteration is a result of turbines having the ability to regain trustworthiness. In other words, turbines are not permanently deemed corrupt and are considered okay if their reputation score increases pass the threshold. This property may not be desirable in a real-world scenario, but was deliberately kept in for transparency and to demonstrate the difficultly of deciphering between corruption and the stochastic nature of wind.

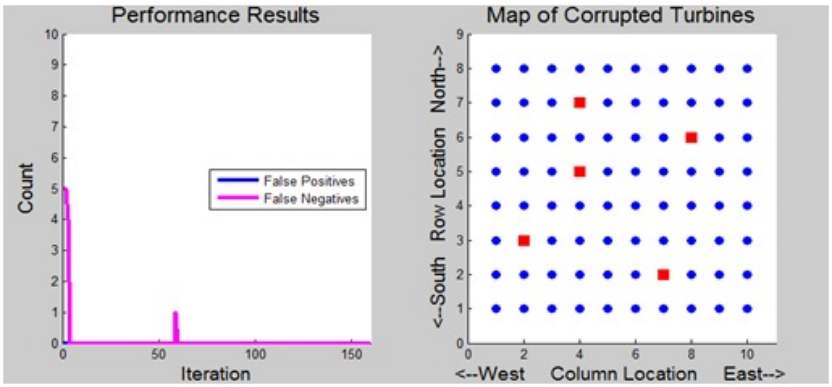

Figure 5.5: Simulation Results from a Small Set of Corrupted Turbines

The next test is to vary the level of turbine corruption, i.e. the magnitude of the parameter manipulation. A fixed topology of 4 columns of corrupted turbines can be seen below in the right image of Figure [5.6]. A total of 8 simulations were run where the corrupt turbines generation wind speed cut-in parameter was varied from 6m/s to 20m/s in 2m/s increments. The count of false positives and false negatives after 160 iterations is plotted against the corruption level in the left image of Figure [5.6]. The algorithm produced a substantial amount of false positives at the 6m/s corruption level and a few at the 8m/s level, but produced none at higher levels of corruption. This is the expected trend because smaller, more subtle parameter manipulations are more difficult to detect.

Now we change the number of corrupted turbines in our simulation. The locations of the corrupted turbines were randomly assigned and the quantity of corrupted turbines varied from 5 to 40 with increments of 5 turbines. The manipulation of the generation wind speed cut-in parameter was held constant at 12m/s. Figure [5.7] shows the history of false positives and false negatives for

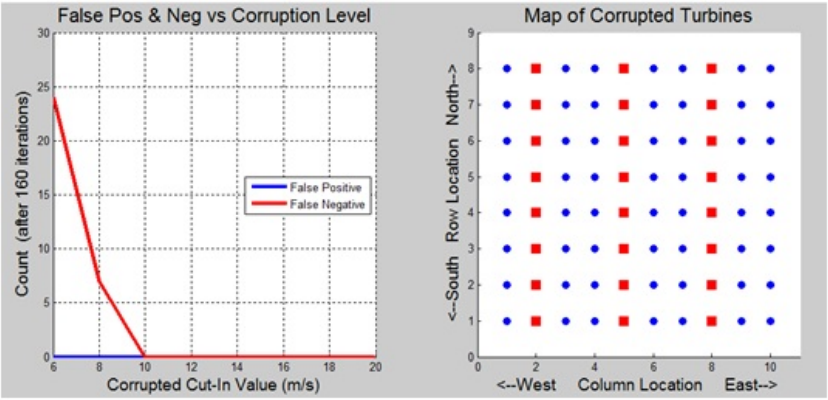

Figure 5.6: Simulation Results by Varing the Level of Corruption

each quantity of corrupted turbines. It can be seen that when 30 or more turbines are corrupted there are false positives and negatives reported at every iteration. The performance of the algorithm degrades as the number of corrupted turbines increases, as illustrated in Figure [5.8]. The algorithm is founded on the idea of "wisdom of the crowd", so it is expected to perform poorly as the number of corrupted turbines approaches 40 (half of the total number of turbines at Horns Rev wind farm).

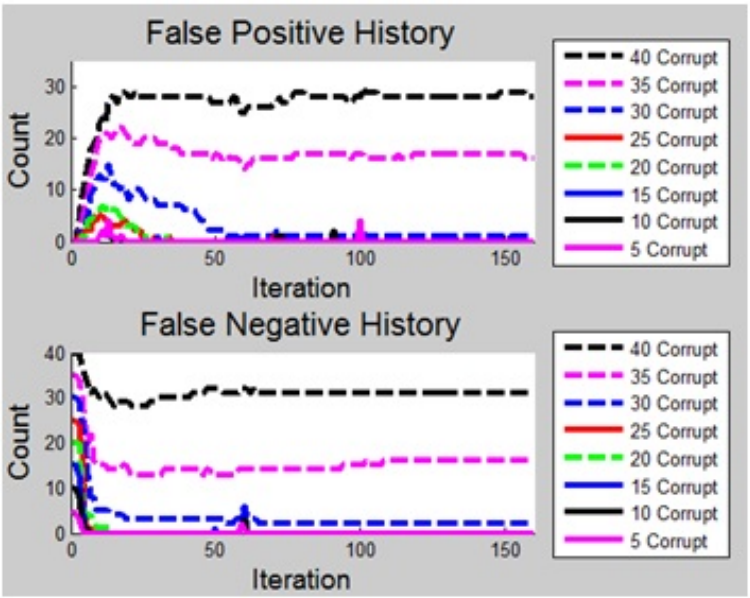

Figure 5.7: Simulation Results by Varying the Number of Corrupted Turbines

Lastly, we compare simulation results using various quantities and levels of turbine corruptions to provide a more complete demonstration of the algorithm's performance . Figure [5.9] depicts the false positive and false negative counts after 160 iterations for 64 different simulation settings. The locations of the corrupted

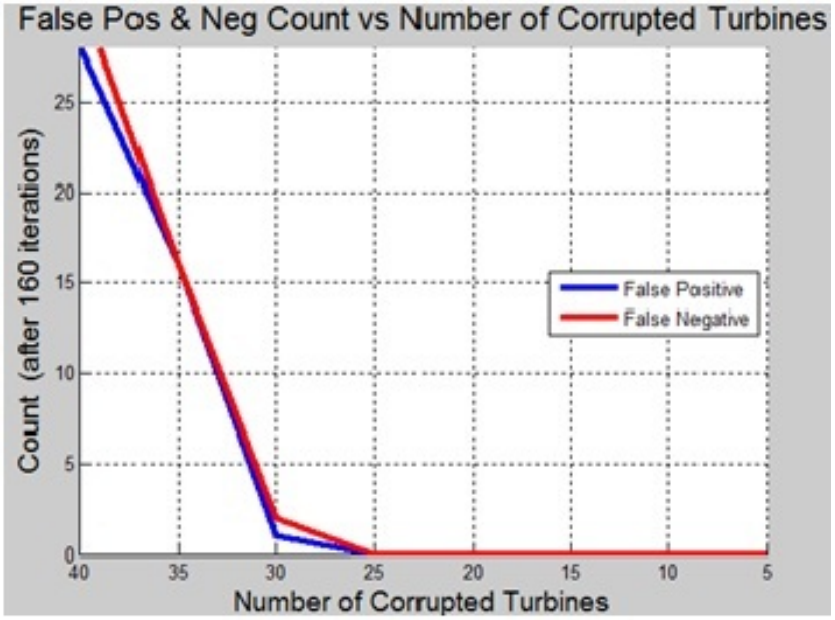

Figure 5.8: Simulation Results by Varying the Number of Corrupted Turbines

turbines were randomly generated and varied between 5 and 40 corrupted turbines. At each of these quantities, the level of corruption was varied from 6m/s to 20m/s. Congruous with results from before, the algorithm performs better when the quantity of corrupted turbines is low and the level of corruption is high.

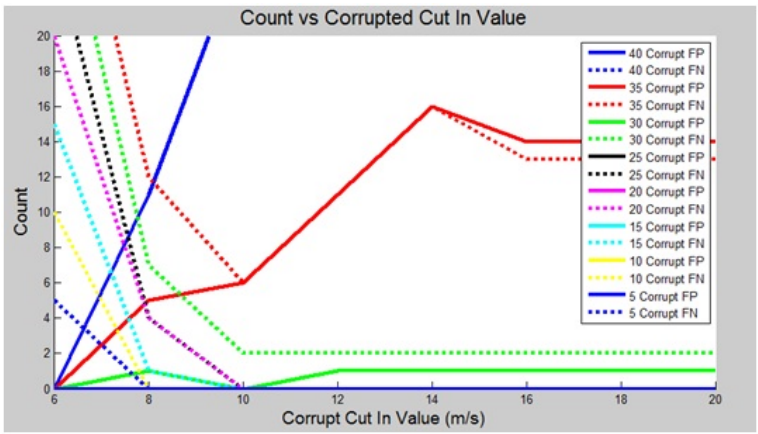

Figure 5.9: Simulation Results under 64 Different Simulation Settings

Remarks. The nominal bonus and penalty values used for the algorithm were 4 and 8, respectively. These values were chosen for the simulation tests because they yielded reasonably good results, but should not be considered optimal values. Manipulations to these values would affect detection time, detection probability, false positive rate, and false negative rate. For instance, a small bonus and large penalty would create a more aggressive algorithm. It would be aggressive in the

sense that it could more quickly identify a corrupted turbine, but with the tradeoff of a higher false positive rate. The exact values of the nominal bonus and penalty would be user-specified according to the user's desired algorithm characteristics.

5.3.3 Conclusion

The algorithm was test under various simulation environments where cyber-attacks were emulated through parameter manipulations to turbine supervisory controllers. It was found that it performed best when there were less than 30 corrupted turbines and the turbine's wind speed cut-in parameter was altered to a value of 10m/s or higher. When these conditions were present, the algorithm properly identified all corrupted turbines and yielded zero false positives or false negatives at the end of an 800 minute simulation. As the number of corrupted turbines approaches 40 (half of the total number of turbines at Horns Rev), the algorithm's performance degrades and begins to yield increasingly more false positives and false negatives. The algorithm is not applicable to situations when half or more of the turbines are corrupted due to its reliance on the "wisdom of the crowd" principle. When the magnitude of the wind speed cut-in parameter manipulation is small (value changed to 10m/s or less), the algorithm's performance degrades as the corrupted value approaches the nominal value of 4m/s. As the corrupted parameter value approaches the nominal parameter value, the corrupted turbine's state profile begins to more closely approximate the state profile that would be generated if that turbine were not corrupted. This sheds insight into why the algorithm has difficulty identifying turbines with subtle corruption. Based on these results, it is reasonable to conclude that the trust dynamics algorithm presented in this dissertation is a good option that could be used to help develop cyber-security solutions for wind power systems.

Chapter 6

Conclusion

People have been studying network structure for decades. And network structure is by no means an isolated research area. On the contrary, a number of fields of study require deep understanding of the underlying network topology. In this dissertation, we show that in certain cases network structure is of more importance than convention has it. Not only does it play a critical role in problems directly-related to network topology, such as cyber security in decentralized sensor networks, – which are intriguing problems in their own right, but it also provides an alternative insight into those problems that might be implicitly formulated into the framework of network structure, for example, global optimization.

By analyzing a general class of (single-thread) model-based search algorithms for global optimization, we reveals that the speed of convergence (worst-case) is associated with the "worst" possible combination of locally-optimal solutions that could ever be identified. In light of this finding, we use the concepts of function's landscape network to characterize the average convergence speed. This formulation smoothly allows a general global optimization problem to be fit into the framework of network structure. In particular, the average speed of convergence is characterized in terms of clusters (aggregation of states). A cluster posits a difficult computational challenge to the model-based search whenever new models generated by the algorithm (given a state in the cluster) do not vary significantly. Thus, the speed of convergence of the single-thread model-based search deteriorates in

problems with many difficult clusters whose basins of attraction are relatively large. It suggests that the worst and average performance of model-based algorithms for global optimization is highly linked to and can be characterized in terms of the objective function's landscape network.

We introduce a new interactive (multi-thread) version of the model-based search method to balance the efforts between exploration and exploitation. Interaction among multiple threads takes place through a relatively simple acceptance-rejection test. We show that the speed of convergence in the worst case increases exponentially in the number of threads.

We also provide a number of numerical examples (including hard high-dimensional optimization problems) solved by the algorithm we proposed in Chapter 3. The results illustrate that the interactive model-based algorithm works well not only in low-dimensional solution spaces but also in high-dimensional solution spaces. The average performance is greatly improved with the increase in the number of threads. In particular, the improvement is radical when the number of threads is changed from one to two. It suggests that it is highly recommended for problem solving to take into consideration the interaction between threads. Interaction helps balance the tradeoff between exploration and exploitation. Interaction amongst threads also makes information exchange possible and thereafter becomes useful for the identification of global minimum.

Unlike global optimization, we turn back to see a cyber-security problem directly related to network topology in Chapter 4 and 5. The interactive feature of the global optimization algorithm inspired us about the importance of information exchange and aggregation. However, the process of information exchange can be maliciously corrupted and even cause a severe security problem. We analyze the dynamics of trust in a social network of data-processing agents. Each agent is fitting into a statistical model for an independent and identically distributed stream of data. To score the reputation of other agents, we consider a simple rule in which every individual agent would downgrade an neighbor's trustworthiness if

his/her estimate differs significantly from most other estimates available to him. We show that the resulting reputation system will converge given the condition that each neighbor's estimate is asymptotically consistent, i.e., either to agree with the majority in a long run, or to constantly differ from most other estimates.

We implement the designated rule to a more specific form feasible for practical purposes. We show that, using this implementation of "wisdom of crowds", each individual agent will successfully identify all the malicious nodes in his/her neighborhood with probability one when the majority of his neighbors are well-behaved.

Using this implementation, we also consider how the entire network structure could play a role in network's vulnerability. To achieve this goal, we analyze a widely-observed category of network structure: scale-free networks. We show the conditions when a scale-free network will survive completely, together with when it will encounter an entire breakdown, on average. In addition, the number of agents that survived after a random attack is also approximately provided. We also take a glaze at other network topology, such as small-world networks and lattice networks. Simulation results are illustrated to compare the robustness among various types of network topology.

A more practical application, cyber security in wind farms, is studied using the algorithm implementation presented in Chapter 4. We test the algorithm performance under a simulation environment based on the real data resource from the Horns Rev wind farm in Denmark. It was found that the algorithm performed best when the number of corrupted turbines keep less than half of the total number, and when the turbines wind speed cut-in parameter was altered to a relatively high value. The simulation results are consistent with theoretical analysis and demonstrate the effect of "wisdom of the crowd . Based on these results, it is reasonable to conclude that the trust dynamics algorithm presented in this dissertation is a good option that could be used to help develop cyber-security solutions for wind power systems.

In a summary, this dissertation manages to analyze the implications of network structure that could work in various fields of study. We propose two algorithms to two distinct research areas. Both algorithms are designed and developed after a careful examination on the underlying network structure of the problems. By theoretical analysis and empirical examples, we demonstrate that the performance of the algorithms is pleasant in their own scopes.

Bibliography

[1] M. Srinivas and L. Patnaik, Generic Algorithms: A Survey, *IEEE Computer*, pp. 17-26 (1994)

[2] J. Kennedy and R. Eberhart, Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, IV. pp. 19421948 (1995).

[3] J. Hu, M. Fu and S. Marcus, A Model Reference Adaptive Search Method for Global Optimization, *Operations Research* Vol. 55 No. 3 pp. 549-568 (2007)

[4] Z.I. Botev, D.P. Kroese, R.Y. Rubinstein, P. L'Ecuyer, The Cross-Entropy Method for Optimization. In *Handbook of Statistics, Vol. 31: Machine Learning*, V. Govindaraju and C.R. Rao, Eds, North Holland (2011)

[5] J. Bekkera and C. Aldrich, The Cross-Entropy Method in Multi-Objective Optimization: An Assessment, *European Journal of Operational Research*, Vol. 211, No. 2, pp 112-121 (2011)

[6] J. Suzuki, A Markov Chain Analysis on Simple Genetic Algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. No. 4 pp. 655-659 (1995)

[7] R. Leary, Global Optimization on Funneling Landscapes. *Journal of Global Optimization* Vol.18, No. 4, pp. 367-383 (2000)

[8] J. F. Bonnans, J. C. Gilbert, C. Lemarechal and C. A. Sagastizabal, Numerical Optimization: Theoretical and Practical Aspects, 2nd edition, Springer (2006)

[9] D. H. Ackely, A Connectionist Machine for Genetic Hillclimbing, Academic Publishers, Boston (1987)

[10] T. Back, Evolutionary Algorithms in Theory an Practice, Oxford University Press (1996)

[11] A. O. Griewank, Generalized Decent for Global Optimization, *Journal of Optimization Theory and Applications*, Vol. 34 (1): pp 11-39 (1981)

[12] G. Brumfiel, High-energy Physics: Down the Petabyte Highwa, *Nature* Vol. 469: pp. 28283 (2011)

[13] M. B. Jones, M. P. Schildhauer, O.J. Reichman and S. Bowers, The New Bioinformatics: Integrating Ecological Data from the Gene to the Biosphere, *Annual Review of Ecology, Evolution and Systematics*, Vol. 37 (1): 519544 (2006).

[14] Y. Liu, B. Yang, D. Cao and Ma Nan, State-of-the-art in Distributed Privacy Preserving Data Mining, *Communication Software and Networks (ICCSN) 2011 IEEE 3rd International Conference*, pp.545-549 (2001)

[15] W. Dargie and C. Poellabauer, Fundamentals of Wireless Sensor Networks: Theory and Practice, John Wiley and Sons, New Jersey (2010)

[16] P. Michiardi and R. Molva, CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Network, *Communications and Multimedia Security Conference* (2002)

[17] M. Ripeanu, Peer-to-Peer Architecture Case Study: Gnutella Network, *P2P '01 Proceedings of the First International Conference on Peer-to-Peer Computing*, pp.99 . (2001)

[18] P. Resnick, R. Zeckhauser, E. Friedman and K. Kuwabara, Reputation Systems, *Communications of the ACM* (2000)

[19] Y. Sun, Z. Han, W. Yu, and K.J.R. Liu. A Trust Evaluation Framework in Distributed Networks: Vulnerability Analysis and Defense Against Attacks, *In Proc. of IEEE Infocom* (2006)

[20] T. Jiang and J.S. Baras, Trust Evaluation in Anarchy: A Case Study on Autonomous Networks, *In Proceedings of IEEE Infocom06* (2006)

[21] N.A. Lynch, Distributed Algorithm, Morgan Kaufmann, Massachusetts (1997)

[22] A. Jadbabaie, J. Lin, and A. S. Morse, Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules, *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988-1001, 2003.

[23] L. Xiao and S. Boyd, Fast Linear Iterations for Distributed Averaging, *Systems and Control Letters*, vol. 53, pp. 65-78, 2004.

[24] F. Pasqualetti, A. Bicchi, and F. Bullo, Consensus Computation in Unreliable Networks: A System Theoretic Approach, *IEEE Transactions on Automatic Control*, Vol. 57 No. 1 pp. 90-104 (2012)

[25] S. Sundaram and C. N. Hadjicostis, Distributed Function Calculation via Linear Iterations in the Presence of Malicious Agents, *American Control Conference*, pp. 1350-1361 (2008)

[26] R. Durrett, Probability: Theory and Examples, 3nd edition, Tomson Learning Inc, Belmont (2005)

[27] T. Lewis, Network Science: Theory and Practice, John Wiley and Sons Inc, New Jersey (2008)

[28] R. Cohen, K. Erez, D. ben-Avraham and S. Havlin, Resilience of the Internet to Random Breakdowns, *Physical Review Letters*, Vol. 85, No. 21, pp. 4626-4628 (2000).

[29] Duncan S. Callaway, M. E. J. Newman, S. H. Strogatz and D. J. Watts, Network Robustness and Fragility: Percolation on Random Graphs, *Physical Review Letters*, Vol. 85, No. 25, pp.5468-5471 (2000).

[30] L. Y. Pao and K. E. Johnson, A Tutorial on the Dynamics and Control of Wind Turbines and Wind Farms, *American Control Conference*, pp. 2076-2089 (2009)

[31] M. Garcia-Sanz and C. H. Houpis, Wind Energy Systems: Control Engineering Design, *CRC Press*, Boca Raton, FL (2012)

[32] Kristoffersen, The Horns Rev Wind Farm and The Operational Experience With the Wind Farm Main Controller, *Copenhagen Offshore Wind*, Denmark, p. 9 (2005)

[33] Tech-wise, Wind Resources at Horns Rev, *Tech-wise* (2002)