Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2018
Aug 26-29, 2018, Quebec City Convention Center, Quebec City, Canada

DETC2018-86006

MODELING TRUST IN SELF-ORGANIZING SYSTEMS WITH HETEROGENEITY

Hao Ji
IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, California, 90089
haoji@usc.edu

Yan Jin
IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, California, 90089
yjin@usc.edu
(*corresponding author)

ABSTRACT

Self-organizing systems (SOS) possess the potential of performing complex tasks in uncertain situations with adaptability. Despite the benefits of self-organizing, it is also subject to the influence of unpredictable behavior of individual agents and environment noises. In hostile situations for example, individual performance of self-organizing agents may deteriorate and this can lead to system malfunction, posing great challenge for the design of SOS. In this paper, we propose a trust based model as a design approach to SOS in consideration of the capability heterogeneity of the agents. A box-pushing task was presented and studied. Trust is measured using beta probability distribution, which takes into account both the positive and negative interactions between the agents. The simulation results have shown that our trust model ensures favorable interactions among agents and leads to increased system effectiveness and conditional system efficiency improvement in comparison to SOS without using a trust model.

Keywords: trust-based model; complex system, self-organizing system; heterogeneity.

1 INTRODUCTION

Self-organizing systems consist of elementary agents that work collaboratively to achieve complex system level behavior without global knowledge. Design of SOS takes a bottom-up approach and the system complexity can be reached through local interaction of agents [1,2]. Complex system design by using a self-organizing approach has many advantages, such as increased adaptability, scalability and reliability [3,4]. In addition, self-organizing systems tend to be more robust to external changes and more resilient to system damage or partial malfunction [5-6]. A typical example of self-organizing systems would be a swarm of robots. In such systems, robots are usually compact in size, adopt simple interaction rules and are with limited functionality. Such systems often consist of a large number of homogenous robots [7]. Collaborative behavior of the swarm robots emerges and is used to apply to situations such as unmanned aerial vehicle patrolling, search and rescue, distributed sensing, traffic control, and box-pushing [5].

In real life, individual agents are not always ideal. Robots may have different speed and power due to production variance, wear and tear [8-9]. Products that come from the same production line may have variations in dimensions. The presence of heterogeneity within agents adds extra noise and uncertainty to SOS and causes unexpected or unwanted system behavior, which is usually outside of control. Thus, SOS design is under great challenge when the assumption of homogeneity of agents can not be met in real applications. Therefore, *how can we design a SOS to achieve complex tasks when agents have differences in competency*? *How can agents have a better sense of its partner's capability and make informed decisions during the self-organizing process*?

In social sciences, social norm is generated through the interaction of socially active agents and is considered an agreement made by the members of the society [10]. By following 'norm of cooperation', people are able to avoid unnecessary conflicts and the average utility among group members can be increased [10]. Agents who do not obey the social norm are considered untrustworthy while a good social norm follower receives higher trust among other agents. SOS can be considered as a social society where agents follow the same set of rules and interact with their bounded rationality [11].

Agents with different levels of competency react differently to the social norm and their trustworthiness can be updated by other agents through observation. When a given agent's trust level falls below a trust threshold, it is no longer trusted [12]. Untrusted agents will be weighted less or ignored by other agents during cooperative tasks. In these cases, trust ensures good interactions and discourages norm breakers. Thus, it plays a key role in keeping the social-rule enforced.

Like in the social society, devising a trust based model into SOS can take into account the uncertainty of the individual competencies and increase system's resistance to internal perturbations or malfunctions. As updating trust is a dynamic process, it can be considered as a dynamic optimization tool for SOS.

In the rest of this paper, we first review the relevant work of SOS and trust based modeling. Next, we present a beta-probability based trust model in design of complex systems and show how dynamic trust and distrust can be formed. Then, we introduce a box-pushing problem and develop a trust based SOS framework as a solution. The box-pushing case study is presented and the results analyzed. Finally, conclusions are drawn from the case study: Trust based SOS model increases system effectiveness in general; depending on the number of heterogeneous agents in SOS, system efficiency either increases or decreases.

2 RELATED WORK

2.1 Artificial Self-organizing Systems

An artificial self-organizing system is a system that is designed by human and has the emergent behavior and adaptability similar to nature [1]. Much research has been done regarding design of an artificial self-organizing system. Werfel developed a system of homogenous robots to build a pre-determined shape using square bricks [13]. Beckers et al. introduced a robotic gathering task where robots have to patrol around a given area to collect pucks [14]. As robots prefer to drop pucks in high density areas, the collective positive feedback loop contributes to a dense group of available pucks [2,14]. Khani et al developed a social rule based regulation approach in enforcing the agents to self-organize and push a box toward the target area [5-6]. Swarms of UAVs can self-organize based on a set of cooperation rules and accomplish tasks such as target detection, collaborative patrolling and formation [15-18]. Chen and Jin used a field based regulation (FBR) approach to guide self-organizing agents to accomplish complex tasks such as approaching long-distance targets while avoiding obstacles [19]. Price investigated into the use of genetic algorithm (GA) in optimizing Self-organizing multi-UAV swarm behavior. He tested the effectiveness of GA algorithm for both homogenous and heterogeneous UAV in accomplishment of 'destroying retaliating target' task [20].

2.2 Trust based Modeling

In the literature, trust is defined as a 'mental state of a trustor who is willing to run the risk of vulnerability with the expectations on positive intentions or behavior from the trustee' [21]. Trust implicitly means potential cooperation between the agents and risk of uncertainty. Typically, a trust model consists of three aspects: trust beliefs, trust decisions and trust actions [22]. Trust beliefs measure the probabilistic based belief that an agent has in other agents' ability of accomplishing the cooperative task; trust decisions are individual assessment of either trust or distrust on a third party given its current belief. Trust actions are the actions that agents take based on its trust decisions [22].

Castelfranchi [22] classified trust models into three different categories: the logical approach, the socio-cognitive approach and computational approach. Logical approach begins with mathematical logics for simulating trust relationships [23]. The socio-cognitive model of trust considers the cognitive ingredients of trust and describes trust by taken into consideration the mental ingredients of trust, its value and actions [22]. Computational approach takes advantage of the computational power and quantify trust in terms of trust values.

Yu and Singh [24] suggested using information from referrals as an alternative source for information on building trust about other interacting agents. In ReGreT model developed by Sabater and Sierra [25-26], trust incorporates three different sources of information: direct experiences, information from other agents and social structures. Travos is a Bayesian based trust model in the context of inaccurate information [27]. The author in the article assumed interaction between agents is either positive or negative and used Bayesian posterior calculation to update trust value based on existing prior information [27]. Trust theory in game theoretical application includes prisoner dilemma situation: both prisoners need to trust each other and cooperate to achieve maximum system-level gain [28]. If one person defects while another cooperates, defectors receive satisfactory gain while cooperators losses [28]. Defection is good for individual utility, but is detrimental at the system level.

Research on trust model falls in categories of wireless sensor network [29], P2P (peer to peer) [30], multi-robot patrolling [9], team formation [30-31], and human-computer interaction [32]. Pippin [8-9] adopted a centralized auction approach to dynamic update the trust from central controller towards collaborating agents. As one of the robots' patrol speed slows down, nearby robot is able to jump in and help with the low performance agents through command from the center [8-9]. Tae Kyung Kim proposed a trust model based on fuzzy logic for efficient communication between wireless source node and destination node in the network [33]. If a given sensor node has high trust value, other nodes will send data to it. In field of human robot interaction, Mahasalem investigated how human's perception of unusual robot behavior have an effect on their interaction choices and the willingness for cooperation with the robot [34]. In eBay and Amazon [35], the aggregate rating from the customer can be used as an indication of the product's trustworthiness.

In SOS, the system level complexity needs to be reached in order to perform complex tasks. However, individual heterogeneity brings uncertainty and inherent noise or perturbations to the SOS, which makes desired emergent behavior unfeasible or outside of control. GA has been applied to optimize SOS in the face of heterogeneity, however, the model itself has to go through many trial runs against various fitness functions in order to find the optimal solution, which often only works for particular initial situations and is not applicable to changing or unknown heterogeneity situations.

Moreover, existing trust models applied in engineering systems are limited in their own application and the tradeoffs of applying a trust model are hardly analyzed. It is crucial for designers to provide guidelines as to how to design a trust based framework in heterogeneous SOS and how to analyze the design tradeoffs. Such area is often omitted in the literature and is the focus of this paper.

3 BETA PROBABILISTIC TRUST MODEL

3.1 Illustration of Trust Model

As past performance is a good indicator of an agent's future performance, trust based model, which tasks into account previous interaction outcomes, can serve as an indication of future behavior of the agent. A simple way to look into agents' heterogeneity is from a probabilistic perspective. We assume individual competency can be represented as the percentage of times an individual agent is able to successfully perform a required task. Also, such outcome can be observed by other agents and classified as either positive or negative experiences. Thus, each observation result can be simplified as a single Bernoulli experiment. Beta probability method is a useful tool in evaluating the outcome of a binomial Bernoulli experiment. Therefore we adopt a beta probability approach inspired by the TRAVOS model to measure trust [27].

We denote a set of agents as $A = \{a_1, a_2, \ldots, a_n\}$. During each round of simulation, several pairs of agents $\{a_x, a_y\} \subseteq A$ will interact with each other based on relative positions and sensor information. A trustor $a_{tr} \in A$ would deem an interaction with a trustee $a_{te} \in A$ successful if trustee a_{te} completes its obligation. The outcome of the interaction is represented as a binary value $O_{atr,ate}$.

$O_{atr,ate} = 1$ means a positive or successful observation, whereas $O_{atr,ate} = 0$ indicates an unsuccessful observation, as expressed in equation (1).

$$O_{atr,ate} = \begin{cases} 1, if\ positive\ observation \\ 0, if\ negative\ observation \end{cases} \quad (1)$$

At a given time $t \in Z$ and $t > 0$, at most one interaction outcome can be observed between two agents, which can be either 1 or 0 . Agents at any given time t keeps track of a history of the positive and negative interaction outcomes, which can be denoted as a tuple $H_{atr,ate\,(t)} = (m, n)$. Here, m and n represents the number of *positive* and *negative* outcomes observed from a_{tr} towards a_{te}, respectively.

The tendency of a_{te} to fulfill its obligation is represented as $B_{atr,ate} \in [0,1]$. It is a representation of the likelihood that a_{te} will fulfill its task during interaction with a_{tr} from the perspective of a_{tr}, shown in equation (2). For instance, $B_{atr,ate} = 0.5$ means the likelihood that a_{te} will fulfill 50% of its obligation.

$$B_{atr,ate} = p(O_{atr,ate} = 1) \quad (2)$$

If complete information about a_{te} is obtained from a_{tr}, then the probability that a_{te} generates positive outcome can be expressed by $B_{atr,ate}$ [27]. However, as complete knowledge can not be assumed since observation results are limited in quantity, from a Bayesian point of view, it is best we use expected value of $B_{atr,ate}$ as an estimation of trust [27].

Confidence is another measure that an agent a_{tr} has in evaluation of the trust value and is denoted as $\gamma_{atr,ate}$. Here, confidence measures how accurate the trust value is given the number of past observations that an agent had. Basically, more observation results give an agent higher confidence in its evaluation of trust value towards another agent. The detailed definition and evaluation measures of trust and confidence are illustrated below.

Trust of an agent a_{tr} towards another agent a_{te} is represented as $\tau_{atr,ate}$, which is a_{tr}*'s* estimation of the probability that a_{te} will generate successful outcomes during interaction and is represented as the expected value of $B_{atr,ate}$ given a_{tr}'s history knowledge $H_{atr,ate(t)}$ in period t, as shown in equation (3).

$$\tau_{atr,ate} = E[B_{atr,ate} \mid H_{atr,ate(t)}] \quad (3)$$

To be able to determine the expected value of trust, a probability density function (pdf) should be modeled to measure the relative probability that $B_{atr,ate}$ will adopt a certain value. In Bayesian analysis, the beta pdf has been widely used as a tool to model distribution of a random variable like $B_{atr,ate}$, representing probability of a binary event (either 0 representing unsuccessful outcome or 1 indicating successful outcome) [27]. The standard formula for beta distributions is given in equation (4), where α and β dictate the shape of the density function and is related to positive m and negative n observations according to equation (5) and (6).

$$f(B_{\text{atr, ate}} \mid \alpha, \beta) = \frac{(B_{atr,ate})^{\alpha-1}(1-B_{atr,ate})^{\beta-1}}{\int_0^1 U^{\alpha-1}(1-U)^{\beta-1}dU} \quad (4)$$

$$\alpha = m + 1 \quad (5)$$

$$\beta = n + 1 \quad (6)$$

Example plots for the beta distribution are shown in Figure 1. Here, the horizontal axis represents the possible values of $B_{atr,ate}$ and the vertical axis represents the probability density distribution. And the shape of the distribution represents the degree of uncertainty over the true value of $B_{atr,ate}$.

At the initial stage of the trust updating process, no prior observation results are gained from a_{tr} towards a_{te} . So, $m=n=0$, it would be reasonable to assume all possible values of $B_{atr,ate}$ are equally likely, here, with $\alpha=\beta=1$, the beta distribution becomes uniform. As time proceeds, a number of positive observation and negative observation would be collected by a_{tr} towards a_{te}, and α and β would be updated based on equation (5) and (6) accordingly . Then the trust value $\tau_{atr,ate}$ can be calculated using the derived mathematical equation representing expected value of beta distribution, shown in equation (7). The expected trust value for agent i towards another agent j is based on the number of positive and negative experiences it observed from agent j and can be represented as,

$$E_{trust\ i,j} = \frac{\alpha}{\alpha+\beta} \tag{7}$$

Given a set of history observations $H_{atr,ate}$ and the elapsed time t, the trust value τ can be expressed as

$$\tau = [E_{trust\ i,j} \mid H_{atr,ate}(t)] \tag{8}$$

However, the current formula for trust value does not differentiate between cases where a_{tr} has enough observation experience about trustee and cases where trustor has little experience. So, we added confidence metric $\gamma_{atr,ate}$ to measure how accurate the trust value is and it is defined as the posterior probability that the exact value of $B_{atr,ate}$ falls with a margin of error ε around τ. In our simulation, ε is defined 0.1. Confidence value is expressed as

$$\gamma_{atr,ate} = \frac{\int_{\tau-\varepsilon}^{\tau+\varepsilon} X^{\alpha-1}(1-X)^{\beta-1}dX}{\int_0^1 U^{\alpha-1}(1-U)^{\beta-1}dU} \tag{9}$$

Figure 1 shows an example of the beta distribution of trust value with different number of positive and negative observations. Left column graphs (a), (b), (c) show an example simulation of probability distribution of trust value of a third agent who has 10% competency (an agent is able to successfully fulfill its obligation 10% of the times). And right column shows how probability distribution of trust value changes with a full competent agent.

Before agent gains observation of another agent, the number of positive and negative observation is 0. So, according to equation (5) and (6), $\alpha=1$, $\beta=1$, the probability density distribution becomes uniform as shown in Figure 1(a) and this means that all trust value is equally likely from the point of view of trustor towards trustee. As time goes on, agent gains more observation, when agent has 1 positive observation and two negative observation, $\alpha=1+1=2$, $\beta=2+1=3$, the distribution is moving towards left hand and is becoming more steep, confidence in the expected trust value increases to 0.339. As more observation is gained, as shown in plot (c), the distribution is becoming more centered around 0.1, and agent's confidence in its judgment of expected trust value increases to 0.983 and trustor can become very sure that the other agent under consideration has an expected trust value of around 0.107. Similarly, Figure 1 (d) (e) (f) show how the trust value changes for a full competency agent. Initially, there is no observation, so the distribution becomes uniform. After 3 successful observation, distribution shifts to the right and expected trust value becomes 0.8 and confidence value γ = 0.416. As more observation is collected, the beta distribution forms the shape of (f), in which α = 10 and β =1 and expected trust value equals 0.909 and confidence level γ = 0.88.

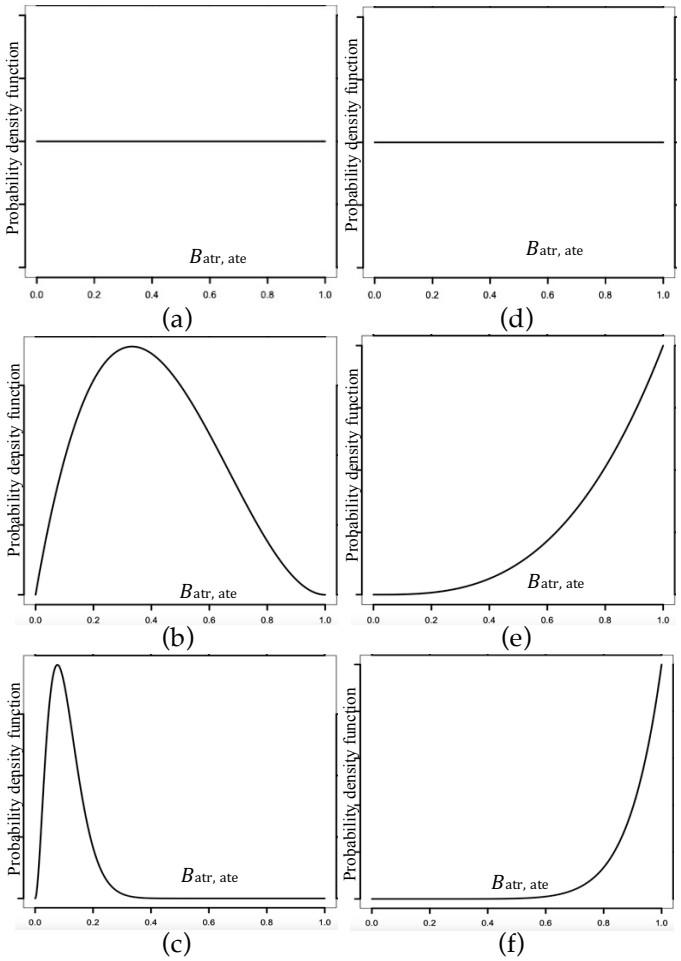


Figure 1. An example of the probability density distribution of trust value with increasing number of observations (a) α= 1, β=1, ε=0.1, τ= 0.5, γ= 0.2 (b) α= 2, β=3, ε=0.1, τ= 0.4, γ=0.339 (c) α= 3, β=25, ε=0.1, τ= 0.107 , γ=0.983 (d) α= 1, β=1, ε=0.1, τ= 0.5, γ= 0.2 (e) α= 4, β=1, ε=0.1, τ= 0.8, γ=0.416 (f) α= 10, β=1, ε=0.1, τ= 0.909, γ=0.88

In the trust modeling literature, Marsh [12] proposed a trust cooperation threshold concept. He argued that once expected trust value reaches a given threshold, an agent is considered trusted and vice versa. In our simulation, when an agent measures trust towards another agent, it compares the trust value with the cooperation threshold and measures confidence value with respect to confidence threshold, shown in equation (10). If both of them are larger than the associated threshold, the third

agent under consideration is trusted. Here, we set cooperation threshold and confidence threshold to be 0.5, which is the neutral point between 0 and 1 to prevent bias towards trust assessment.

$$Trust\ Decision = \begin{cases} trusted & if\ \tau > 0.5\ \&\ \gamma > 0.5 \\ untrusted & Otherwise \end{cases} \quad (10)$$

3.2 Accuracy of Trust Assessment

As discussed above, individual competency can be considered as a percentage rate of obligation fulfillment and each observation can be either successful or unsuccessful. Beta trust model is able to measure such binomial Bernoulli experiment and suggest trust decision. So, how quickly can our proposed trust model converge to reasonable trust decisions given the observation experience? Can trustworthy agents always be identified as trustworthy?

We analyzed three different trust cases where individual agent has 100%, 90%, 80% competency. In industrial applications, three-sigma and six-sigma concept [36] has been introduced to help engineers decide at which error rate should they reject a product. One standard deviation is a value where around two thirds of data fall within the range of one sigma below average and one sigma above average [36]. In SOS, because system is more resilient than traditional engineering system and thus is more tolerable of individual error rate. Therefore, we consider 80% competency agents as acceptable case for trustworthiness, which is around 13% more than one standard deviation case.

We use p as the percentage of times an agent is able to generate successful outcomes. So, each observation of an individual action can be considered as a Bernoulli experiments and given the total number of observation n, the probability of k positive observation $P(k|n)$ is

$$P(k|n) = \binom{n}{k} p^k (1-p)^{n-k} \quad (11)$$

For total number of n observation, there is possibility of 0, 1, 2, … n positive outcomes. The accuracy of successful measurement of trust is sum of all individual probability $P(k|n)$ that meets the trust threshold and confidence threshold of beta distribution, which can be calculated using the simple algorithm shown in Figure 2.

1. **Set** k= 0
2. **Set** Prob = 0
3. **While** (k ≤ n)
4. {
5. **If** (τ = > 0.5 & γ > 0.5)
6. {$Prob = Prob + \binom{n}{k} p^k (1-p)^{n-k}$}
7. k = k + 1
8. }

Figure 2 Algorithm for computing accuracy of trust assessment

Here, *Prob* represents the accuracy of trust measurement, n is the total number of observation, k is the number of positive observation. α = k +1, β = n-k+1, $\tau = \frac{\alpha}{\alpha+\beta} = \frac{k+1}{n+2}$,

$$\gamma_{atr,ate} = \frac{\int_{\tau-\varepsilon}^{\tau+\varepsilon} X^{\alpha-1}(1-X)^{\beta-1} dX}{\int_0^1 U^{\alpha-1}(1-U)^{\beta-1} dU}.$$

Figure 3 shows the results of the plot based on algorithm described in Figure 2. The horizontal axis represents the number of total observations; vertical axis represents how likely trust model is able to successfully predict the trustworthiness of an agent given the number of observations. In terms of 100% competency agent, it only takes 5 runs to successfully predict trustworthy agents and it takes 8 and 10 observations for the trust accuracy of 90% and 80% competency agents to converge to 1. There is a slight drop of trust accuracy assessment with 6 observation compared to 5 observations in 90% and 80% competency settings. However, the trust accuracy quickly converges to 1 afterwards.

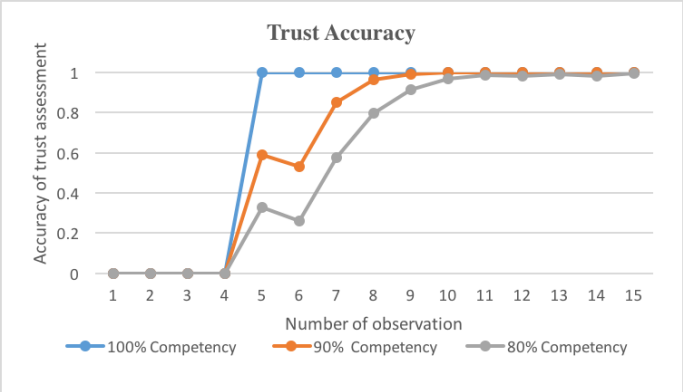


Figure 3 Probability of correct trust assessment vs. number of observation with respect to different competency levels of agents

The number of 4 observations is a tipping point for all three settings of trust evaluation. Below 4 observations, it is unlikely that a trustworthy agent would be discovered. This is consistent with the real-life trust building where it takes a number of positive interactions for people to build trust towards others. Moreover, as the number of observation becomes large, trust value is able to converge to unity and agent is able to make high accuracy trust assessment.

4 A TRUST BASED MODEL OF SELF-ORGANIZING SYSTEMS

Field is considered a mathematical representation of the influence present in the physical environments [37]. In physics, gravitational field, electronic field and magnetic field is used very common. In biology, chemical field can influence the development of living creatures [38]. Inspired by the natural field, we present a two field approach to govern agent behavior in SOS [37]. Based on information from task field and social

field, and trust value, social rule based behavior regulation (SRBR) can suggest optimal actions for agents to take.

4.1 Task Field

Task field of SOS is considered the field generated by the task environments such as repulsion of obstacles and attraction of the target [39]. In the self-organizing box-pushing example from previous research [19], agents do not have explicit communication between each other but make movements based only on perceived task field strength. In a task environment where there are m targets (represented by θ) and n obstacles (represented by β), the task field can be shown as,

$$tField = FLD_T(\theta 1, \theta 1, \ldots \theta m, \beta 1, \beta 2, \ldots \beta n), \quad (12)$$

where FLD_T indicates an operator for task field generation.

4.2 Social Field

Besides task field, agents' behavior in SOS can also be affected by their peers. Social field is generated by an agent's neighbors and adds an another layer of information to the self-organizing agents. It can be considered an aggregation of the relationships between the individual agent and the other agents it is interacting with.

For an agent i with n neighboring agents within its communication range, social field can be represented as,

$$sField = FLD_S(\varphi 1, \varphi 2, \ldots \varphi n), \quad (13)$$

where FLD_s represents a generation operator of social field. ψ_1, ψ_2, $\ldots \psi_n$ indicates the relationship between agent i and his neighboring n agents.

4.3 Social Rule based Behavior Regulation

The application of task field and social field allows us the opportunity to devise social structuring into the SOS to explicitly distinguish between the two fields and perform complex tasks [6,39]. We define a term called 'social rule based behavior regulation' (SRBR) as a mapping between the two field, trust value and the agent actions. When agents have task field, social field and trust information in place, social rules define recommended actions for agents to take. Relationship between suggested action and task field, social field, trust value can be shown in equation (14).

$$\text{Action} = \text{SRBR (tField, sField, trust value)} \quad (14)$$

where SRBR means the social rule based behavior regulation. tField represents task field and sField represents social field. While each agent makes decisions based on social rules, the combined effect will generate emergent complex behavior at the system level.

4.4 Trust based Framework for SOS

Trust model can be used in combination with task field, social field, and social rules to tackle the SOS uncertainty and its framework is shown in Figure 3.

Trust is measured based on observation of selected partners within observation range. By observation, we mean observations of positive and negative behaviors from other agents with respect to following social rule. Whenever an agent violates the social rule, its behavior is considered a negative experience from other agent's point of view; strictly following the social rule is considered a positive action. After trust is measured, agents compare expected trust value with cooperation threshold and its confidence of expected trust value with respect to confidence threshold. If trust value and confidence towards an agent is greater than the threshold, the agent is consider trusted and vice versa. At the same time, social rules combine information from task field, social field and trust decisions (trust or distrust) and suggest actions. The collective effort made by each single agent will affect the emergent system performance, and this will in turn lead to another round of observation by the agents. The iterative loop will continue until systems complete required task. Upon completion, system performance can be measured in terms of system effectiveness and system efficiency.

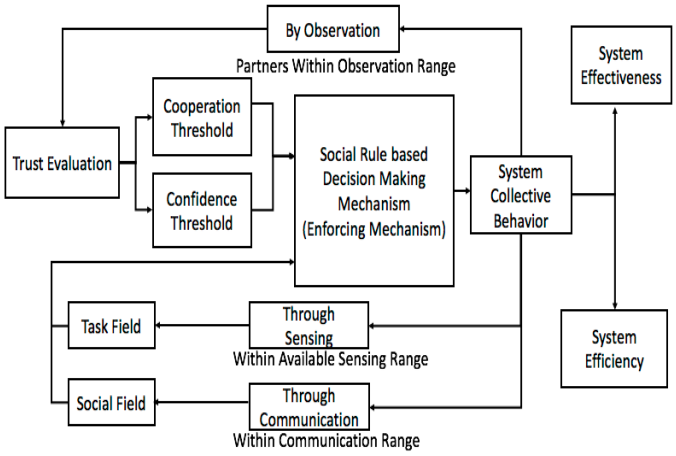


Figure 3. The trust based model for Self-organizing System with heterogeneity

5 CASE STUDY

5.1 The Box-pushing Problem

The box-pushing problem is often classified as a piano mover's problem or trajectory planning [40]. Numerous mathematical and topological solutions have been proposed in the past [40]. In our paper, we are using a self-organizing approach to solve box-pushing problem in which agents work collectively, follow simple rules and push the box towards a target without global knowledge.

The box-pushing case study was modeled in Netlogo platform [41], a multi–agent simulation software. A R extension package was installed and enables communication between

Netlogo and R software, which allows agents to analyze beta distributions and evaluate trust values.

In box-pushing simulation, we took into account the possibility of presence of various number of heterogeneous agents and integrated a trust model with SOS. We showed the design tradeoffs of application of trust model in SOS. In detail, we investigated into how trust among agents can have an impact on system effectiveness and efficiency with respect to different number of incompetent agents. Such an understanding will provide helpful guidance on future SOS design.

A graphical illustration of box-pushing case study is shown in Figure 4. As we can see from the figure, multiple agents with limited sensing and communication range, and different capability (competency of pushing the box) need to self-organize to push and rotate the box towards the goal/target. During the transportation process, as there is an obstacle and walls along agents' path and as the road becomes narrower, agents couldn't just simply push the box but need to rotate the box when necessary [5,6], which adds complexity to the task. When the center of the box reaches the same horizontal x-coordinate of the target, the simulation is deemed a success.

There are seven major task summarized as below:

T1 = <Move><Box> to <Goal>
T2 = <Rotate><Box> to <Goal>
T3 = <Move><Box> away from <Walls>
T4 = <Move><Box> away from <Obstacle>
T5 = <Sense><Task Field>
T6 = <Sense><Social Field>
T7 = <Evaluate><Trust> on <Agents' competency>

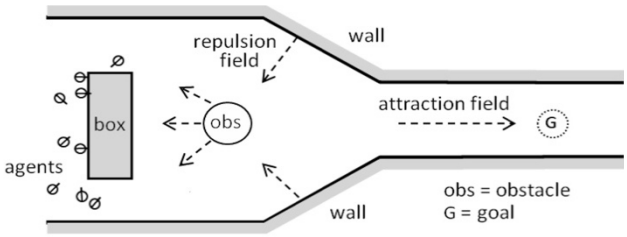


Figure 4 Box-pushing task illustration [5]

In Netlogo, distance is measured by patch-widths (pw). Patch is a single square area in the simulation environment [41]. As an example, the box in our simulation is 5 pw wide and 11 pw long.

In box-pushing, agents have **limited sensing and communication capabilities**. They can broadcast their information to nearby agents but not to the whole team. They can also measure direction and distances, reason with simple rules, and have limited storage of observation information. These assumption is in line with the definition of "minimalist" robot [42] and is reasonable with the current applications of physical robot hardware [43].

Box neighborhood can be simplified as six regions [6,44], as shown in Figure. 5. In determining agent's neighborhood, an agent looks for agents in its own zone, and if it is on a long edge of the box, agents in adjacent zone on the long edge and agents in the opposite zone along vertical axis. For instance, the red agent in region 3 can sense the field strength of blue and green agents and itself. It can also count the number of agents in region 2, 3 and 5.

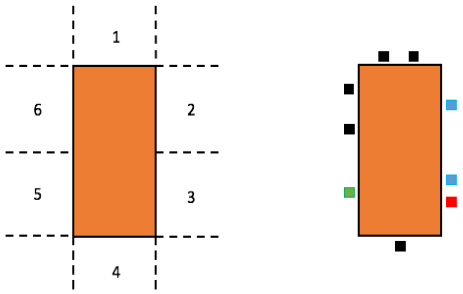


Figure 5. The six regions of box neighborhood

Box dynamics is based on a simplified physical model. Its movement depends on simulated force and torque. Forces equals the sum of vector forces of every pushing agent. Each push carries same amount of force, which acts from the agents towards the box. Sum of 9 pushes will move the box 1 pw in a given direction. Torque is assumed to be exerted on the centroid of the box and equals to sum of moment arm of all vector forces of the pushing agents. 3 pushes with a moment arm of 5pw each will rotate the box 1 degree. We assume box carries a large moment of inertia and when it hits the obstacle, which is considered rather small, it will continue its movement until its expected end position is reached.

In our simulation, there are **total 9 pushing agents** and individual agent's competency vary depending on the simulation setup. We assume heterogeneous agents are always able to move to suggested positions of box neighborhood by following social rule but are limited by their competency of pushing the box. **Competent agents** are able to push the box 100% of the times. **Incompetent agents** can only successfully push the box 10% of times—i.e., only 10% of the total of number of pushes during the simulation are effective with a uniform distribution of time ticks. This simulates the real-life systematic error/noise or malfunction of force execution mechanism of physical robots. Need to mention that if entire robot is broken down and robots can not even move, it will make zero contribution to the box movement and such situation is not considered in our case study.

Defining **rules for the behavior** of agent on the individual-level is quite a challenge due to limited knowledge of agents. Moreover, although social rules can be designed for homogenous self-organizing box-pushing agents [5], when heterogeneity is presented, uncertainty might generate unexpected system behavior, such as hitting the obstacle and etc. How can we mitigate such uncertainty in SOS and accomplish task with reasonable time and good performance even with individual heterogeneity? And if so, what are the design tradeoffs?

5.2 Task field, social field and trust model

In order to facilitate heterogeneous self-organizing agents in accomplishing box-pushing tasks, we devise task field, social field and trust model into the system.

Task field: Task field in box-pushing case consists of an attraction field from the target and gradient like repulsion field from the walls & obstacles as warning to agents in case of dangerous situations. Here, attraction field from target is expressed in equation (14)

$$\text{tField} = \text{Ve}^{-\lambda \text{d}} \quad (15)$$

where tField represents task field, V and λ are design constants, we choose V to be 80 and lambda is 0.1, d represents the distance between selected position and the target zone.

Task field from the walls and obstacles is gradient based and is below 0. For position within 1,2 and 3 patch radius of obstacles and walls, it is -10, -8 and -6 respectively. In simulation environment, positions with higher field values are more preferable for agents. During the box movement, agents always seek low field position in the box neighborhood and try to push the box towards a position with higher field strength value [5,6]. At the same time, agents make predictions on preferred torque and force of box movement based on task field information and its position of box neighborhood. As shown in Figure. 6, red agent in box region 3 can obtain task field information from the green, blue agents and itself, represented by $\text{tField}_{\text{green}}$, $\text{tField}_{\text{blue}}$ and $\text{tField}_{\text{red}}$. Based on the task field information of nearby regions, agents estimate on the direction of preferred torque and force of box movement. They will then compare information from social field, trust value and make suggested movements. For instance, as $\text{tField}_{\text{green}}$ is smaller than $\text{tField}_{\text{red}}$ and also smaller than $\text{tField}_{\text{blue}}$, according to our simulation algorithm, the preferred torque would be positive (counter-clockwise) and preferred force is towards target. Red agent has a tendency to move to a green region where it will contribute to preferred torque or force.

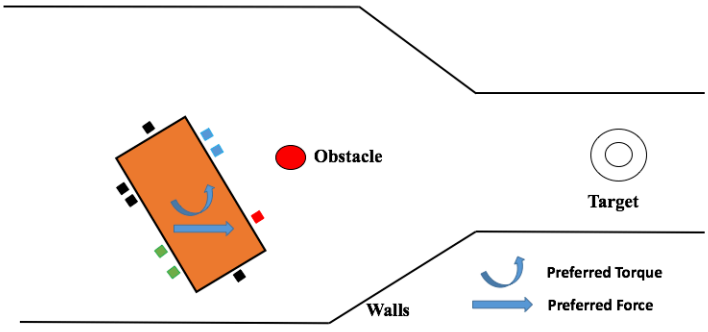


Figure 6. Illustration of preferred torque and preferred force for red agent in box-pushing task

Figure 7 is the pseudo code of how the red agent in box region 3 in Figure 6 calculates its preferred torque and force. Agents in other regions estimate their preferred torque and force following similar logic and we will not give details of algorithm for agents in other regions here.

Social field: Social field is an abstraction of relationships between agents. Here, social field represents agent's sense of how many agents are in its neighboring regions and their relative positions, which has already been illustrated in Figure 5. It will be used combined with trust model to calculate expected torque and forces illustrated below.

```
1. If tField_green <= tField_red
2.       If tField_green <= tField_blue
3.             Set preferred torque positive
4.             Set preferred force towards-goal
5.       Else
6.             If |tField_blue – tField_red| < 2
7.                   Set preferred force away-from-goal
8.             Else
9.                   Set preferred force away-from-goal
10.                  Set preferred torque positive
11. Else
12.      If tField_blue < tField_red
13.            If |tField_blue–tField_red | < 2
14                   If tField_blue < 0
15.                   Set preferred force away-from-goal
16.            Else
17.                   Set preferred torque positive
18.                   Set preferred force away-from-goal
```

Figure 7. Pseudo code for estimating preferred torque and force based on task field information

Trust model: In box-pushing task, agents have two kinds of actions. One is to per suggested action from SRBR, move to a recommended region in a box neighborhood, the other one is performing pushing action after moving to the recommended box region. We assume individual agent in box-pushing task are able to follow SRBR and successfully move to suggested box region, but their pushing behavior is limited by their competency. Whenever agents push the box, it is assessed by nearby agents as a positive experience and if agents fail to push the box, it is deemed a negative experience. Agents assess the trustworthiness of other agents by adopting the trust model described in section 3 and the dynamic of trust updating process of box-pushing case study can be seen in Figure 1.

If agent *j* is considered by agent *i* to be trustworthy, it is considered by agent *i* to be competent enough to make the future push effort. And if it is not considered trustworthy, agent *i* will not count agent *j* in making the future push effort. Once trust information is gained, agents make predictions on expected force and expected torque (different from preferred torque and force): sum of vector forces of all trustworthy agents and sum of torque of all trustworthy agents. When the number of agents in a specific region is unknown to an agent due to its limited sensing ability, agent estimate the number to be rounded sum of all agents within sensing range divided by number of observed regions.

5.3 Social Rule based Behavior Regulation

Social rule based behavior regulation makes suggested actions for agents based on obtained task field, social field and

trust information. Agents first sense task and social filed information of nearby agents and itself. Based on obtained information, it makes estimations on direction of preferred torque and force. Then agent makes trust decisions (either trustworthy or untrustworthy) based on observations of other agents. It counts number of perceived competent agents in different neighborhood and calculate expected torque and force. By comparing expected torque and force with preferred torque and force, agents make associated actions: When preferred torque and force matches with expected torque and force, agents stay at the current position; otherwise, it moves to a region where torque or force conflict could be alleviated. Figure 8 below shows a pseudo code for SRBR:

1.**Sense** *task field* of neighboring agents and myself
2. **Sense** *social field* of neighboring agents and myself
3. **Make** estimations on *preferred torque* and *force*
4. **Apply** *trust model* and measure *trustworthiness* of agents
5. **Calculate** *perceived number of competent agents* in different regions
6. **Calculate** *expected torque* and *expected force*
7. **If** *expected torque calculation* matches with *preferred torque*
8. **If** *expected force calculation* matches with *preferred force*
9. **Stay** at *current position*
10. **Else**
11. **Move to** region where *force conflict* can be alleviated
12. **Else**
13. **If** *expected force calculation* matches with *preferred force*
14. **Move to** *region* where *torque conflict* can be alleviated
15. **Else**
16. **Move to** *region* where *torque or force conflict* can be alleviated

---

Figure 8. Pseudo code for Social rule based behavior regulation algorithm

5.4 Simulation Environment Setup

Figure 9 summarizes the simulation environment setup. Independent variables are the input/control variables in the simulation. Here, we vary the number of incompetent agents and compare how the use of trust model affects system performance with respect to without the use of trust model. To explain, when agents do not use trust model, they make predictions on expected torque and force value based only on observed presence of number of agents in nearby region rather than evaluate real competency of agents in those regions using trust model.

There are two dependent variables we use for performance measurement:

Success rate: the percentage of number of runs where agents can successfully push the box towards goal without hitting obstacles.

Time duration: the total simulation time steps it takes for agents to push the box to target. Time duration is measured in ticks in Netlogo, similar to real-time seconds. Only time duration of successful runs is incorporated into the time duration calculation.

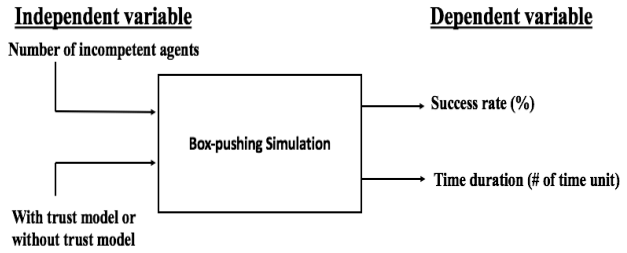


Figure 9. Box-pushing environment setup, independent variables vs dependent variables

6 RESULTS AND DISCUSSIONS

6.1 Typical Failure Modes

Figure 10 shows a visual representation of some typical failure and undesirable modes of box movement with motion trace examples when heterogeneous agents are presented. Figures 10 (a) through (d) are the results of running the simulation case of 7 incompetent agents and without trust model applied. It can be seen that incompetent agents result in sudden, excessive momentum, which leads to pushing the box towards the red obstacle, as shown in (a). Sometimes incompetent agents cause insufficient or excessive rotation torque, resulting in edge of the box hitting the red obstacle, as shown in (b). Besides failure mode of box-pushing, undesirable box movements are not critical but should be reduced, such as tumbled pushing trajectory in picture (c) or pushing the box slightly off the target in the vertical axis in picture (d).

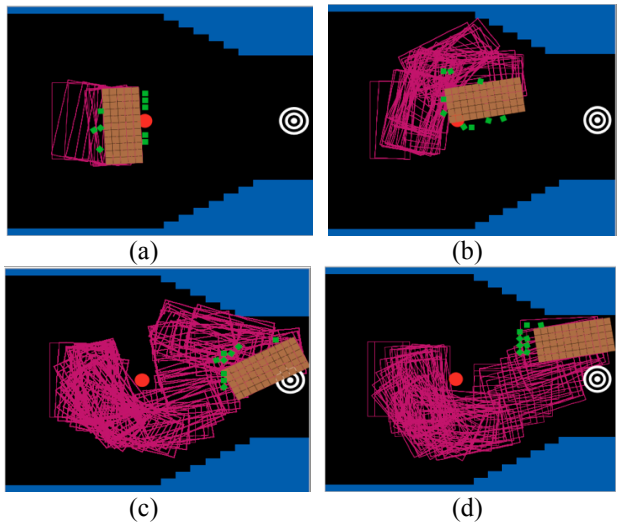


Figure 10. (a) failure with excessive pushing momentum (b) failure with insufficient/excessive rotation torque (c) undesirable tumbled trajectory (d) undesirable ending position of box

6.2 Simulation Results

Applying a trust model should help decrease failure rate and optimize the moving trajectory of the box movement given task filed, social field information and the social rules applied. Figure 11 shows the box-pushing trajectory with motion traces of 7 incompetent agents and with the application of trust model. We can see that box movement trajectory is more smooth and the box ending position is closer to the target than without trust model applied.

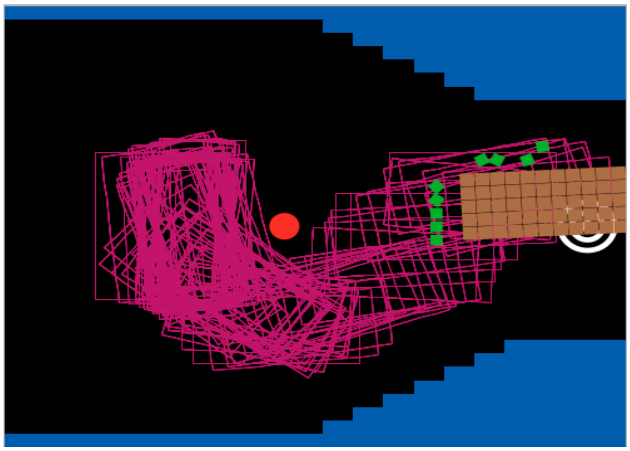
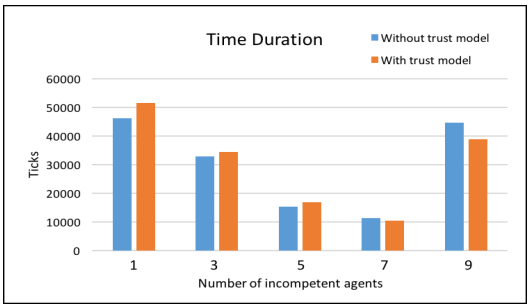
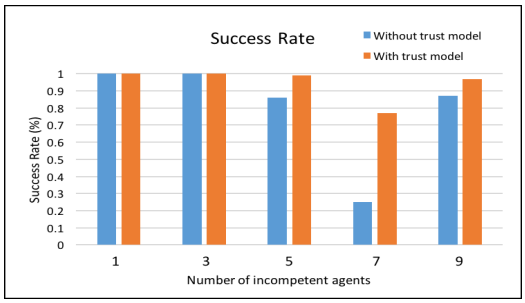


Figure 11. Visual representation of box movement with motion traces for 7 incompetent agents with trust model applied.



(a)



(b)

Figure 12. Time duration & success rate of box pushing task for without & with trust model under different number of incompetent agents

Figure 12 shows the statistical results of the simulation. Each simulation is composed of 100 runs and we take the average of all simulation results to maintain statistical significance.

As shown in Figure 12(a), as number of incompetent agents gets larger (up to 5), incorporating trust between agents first increase the time for accomplishing the task and then decrease the time for finishing the task compared to without use of trust model and this difference is statistically significant ($p < 0.05$) according to the results of two sample t tests. The might be due to the fact that building trust between agent requires a certain number of observations. As there are relatively more competent agents in these settings compared to 7 incompetents and 9 incompetent cases, the initial trust building process can be chaotic and would have errors in its assessment, which results in some loss of system efficiency. This means integration of a trust model with SOS does not necessarily increase system efficiency. It only increases system efficiency when number of incompetent agents reach a threshold value.

Also, as the number of incompetent agent increase (for both without and with the trust model), time for task accomplishment first decreases and then increases, which implies incorporating some incompetent agents during the box-pushing task might be beneficial to the system efficiency as compared to when system is with full competent agents. This can be explained by characteristics of the box-pushing task: box-pushing task requires both cooperation and conflict resolution. Too few agents are not able to achieve cooperation in pushing box, while too many agents result in too much conflict and is not beneficiary either. So, optimal number of competent box-pushing agents should be neither too large nor too small. In our case, either 2 competent agents, 7 incompetent agents or 4 competent agents 5 incompetent agents is a good number for achieving optimal system efficiency. Such findings are also consistent with the finding from past literature [5] where moderate social structuring is crucial for collaborative box-pushing. Here, social structuring means the percentage of time agents are able to follow the social rule, which is similar to our simulation setup, where competency means percentage of time agents are able to make the push effort enforced by social norm.

In Figure12 (b), we can see that SOS can be very tolerant of incompetent agents in terms of success rate. Even there are 5 incompetent agents, the system can still maintain high success rate for both trust model and without trust model situation. This proves that SOS has high resiliency [3,4]. We also noticed that when the number of incompetent agent increases to 5, the success rate drops a little bit. Applying trust model can move the success rate back to almost 100%, which is larger than without trust model ($p < 0.05$ according to the results of two sample t tests). However, when 7 agents are incapable in the system, the success rate drops dramatically to 25%. With trust building among agents, system is able to maintain around 75% success rate, a large increase than 25% (difference is statistically significant, $p < 0.05$ according to the results of two sample t tests). What is also interesting is when incompetent agents are 9, the system success rate goes back to high percentage even without use of trust model, this can be explained by the fact that

pushing forces of all incompetent agents lead to a more balanced state of box movement compared to when there are 7 incompetent agents (individual unbalanced pushing effort of the 2 competent agents may not be corrected by the rest of 7 incompetent agents), so success rate remains high in this case. Still, Adoption of a trust model can assist agents push the box with almost 100% success rate when system is full of incompetent agents.

7 CONCLUSIONS

When tasks become more complex, self-organizing systems can be a useful approach to achieve task goals. From a system design perspective, there are two levels of design parameters that require careful design attentions, one is individual level (i.e., agent level) and the other social level (i.e., system level or interaction level). Our previous research has demonstrated that for homogeneous self-organizing systems, where all participating agents are equally capable and competent, devising social structures plays an important role for increased system level effectiveness and efficiency [5]. In real world applications, however, agent competencies cannot be guaranteed since some agents may experience damage for various reasons. The system heterogeneity due to the partial loss of competency among agents adds another layer of complexity to the design of self-organizing systems. In this paper, we presented a trust modeling based approach to deal with this issue, which essentially is to add a layer of mutual behavior prediction capability to agents.

While social structuring allows agents to explicitly assume their social roles (i.e., system level roles) through shared social rules [5], it does not provide any information of the functional capability and competence of agents. When the effective coordination among agents is demanded by tasks, an agent must predict such information based on its own observation. In this paper, a beta probabilistic trust model is introduced and applied to the box-pushing case study. By setting up the cooperation threshold and confidence threshold, an agent can effectively make trust decisions (i.e., trust or do not trust another agent being able to perform the push action) and coordinate more effectively.

The simulation results of our proposed trust model based approach have led us to some interesting findings:

- Incorporating a trust model into SOS can improve the system effectiveness (i.e., success rate) in general. The results show that positive effect of adding trust model does not depend on the number of incompetent agents. The positive impact of trust modeling is especially strong when the system performed poorly with 7 incompetent agents.
- Depending on the number/percentage of incompetent agents, the system efficiency (i.e., time duration) first decreases then increases. In another word, efficiency of the system might suffer when system does not have many incompetent agents with trust model applied.

It is worth mentioning that the above findings are now limited to the box-pushing type of tasks. For more complex tasks, each agent in a system needs to perform multiple and different combinations of functions, just as different components in a system perform different functions. In such highly heterogeneous situations, in order for an agent to effectively self-organize with others, it will need to establish a *complete trust profile* of other agents through observation. Such a trust profile will include competence trust values of a set of functions. Furthermore, by moving from observing of single agent's behavior to observing the patterns of multi-agent behaviors, more advanced trust calculation can be developed. We plan to explore further along these two directions.

This paper was based on the work supported in part by the Monohakobi Tech-nology Institute (MTI) and Nippon Yusen Kaisha (NYK). The authors are grateful to the MTI team for their discussions and insights on this research.

REFERENCES

[1] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH computer graphics, 21(4), 25-34.

[2] Ashby, W. R. (1991). Requisite variety and its implications for the control of complex systems. In Facets of systems science (pp. 405-417). Springer US.

[3] Chiang, Winston, and Yan Jin. "Design of Cellular Self-Organizing Systems." IDETC/CIE. 2012.

[4] Humann, J., Khani, N., & Jin, Y. (2014). Evolutionary computational synthesis of self-organizing systems. AI EDAM, 28(3), 259-275.

[5] Khani, N., Humann, J., & Jin, Y. (2016). Effect of Social Structuring in Self-Organizing Systems. Journal of Mechanical Design, 138(4), 041101.

[6] Khani, N., & Jin, Y. (2015). Dynamic structuring in cellular self-organizing systems. In Design Computing and Cognition'14(pp. 3-20). Springer, Cham.

[7] Kennedy, J. (2006). Swarm intelligence. In Handbook of nature-inspired and innovative computing (pp. 187-219). Springer US.

[8] Pippin, C. E. (2013). Trust and reputation for formation and evolution of multi-robot teams (Doctoral dissertation, Georgia Institute of Technology).

[9] Pippin, C., & Christensen, H. (2014, May). Trust modeling in multi-robot patrolling. In Robotics and Automation (ICRA), 2014 IEEE International Conference on (pp. 59-66). IEEE.

[10] Elster, J. (1989). Social norms and economic theory. Journal of economic perspectives, 3(4), 99-117.

[11] Simon, H. A. (1982). Models of bounded rationality: Empirically grounded economic reason (Vol. 3). MIT press.

[12] Marsh, S. P. (1994). Formalising trust as a computational concept.

[13] Werfel, J. (2012). Collective construction with robot swarms. In Morphogenetic Engineering (pp. 115-140). Springer Berlin Heidelberg.

[14] Beckers, R., Holland, O. E., & Deneubourg, J. L. (1994, July). From local actions to global tasks: Stigmergy and collective robotics. In Artificial life IV (Vol. 181, p. 189).
[15] Dasgupta, P. (2008). A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(3), 549-563.
[16] Ruini, F., & Cangelosi, A. (2009). Extending the Evolutionary Robotics approach to flying machines: An application to MAV teams. Neural Networks, 22(5), 812-821.
[17] Lamont, G. B., Slear, J. N., & Melendez, K. (2007, April). UAV swarm mission planning and routing using multi-objective evolutionary algorithms. In Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on (pp. 10-20). IEEE.
[18] Wei, Y., Madey, G. R., & Blake, M. B. (2013, April). Agent-based simulation for UAV swarm mission planning and execution. In Proceedings of the Agent-Directed Simulation Symposium (p. 2). Society for Computer Simulation International.
[19] Chen, C., & Jin, Y. (2011). A behavior based approach to cellular self-organizing systems design. ASME Paper No. DETC2011-48833.
[20] Price, I. C., & Lamont, G. B. (2006, December). GA directed self-organized search and attack UAV swarms. In Proceedings of the 38th conference on Winter simulation (pp. 1307-1315). Winter Simulation Conference.
[21] Rousseau, D. M., Sitkin, S. B., Burt, R. S., & Camerer, C. (1998). Not so different after all: A cross-discipline view of trust. Academy of management review, 23(3), 393-404.
[22] Castelfranchi, C., & Falcone, R. (2010). Trust theory: A socio-cognitive and computational model (Vol. 18). John Wiley & Sons.
[23] Castelfranchi, C., Falcone, R., & Lorini, E. (2009). A Non-reductionist Approach to Trust.
[24] Yu, B., Singh, M. P., & Sycara, K. (2004, August). Developing trust in large-scale peer-to-peer systems. In Multi-Agent Security and Survivability, 2004 IEEE First Symposium on (pp. 1-10). IEEE.
[25] Sabater, J., & Sierra, C. (2001, May). Regret: A reputation model for gregarious societies. In Fourth workshop on deception fraud and trust in agent societies (Vol. 70, pp. 61-69).
[26] Sabater, J., & Sierra, C. (2002, July). Reputation and social network analysis in multi-agent systems. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: Part 1 (pp. 475-482). ACM.
[27] Teacy, W. L., Patel, J., Jennings, N. R., & Luck, M. (2006). Travos: Trust and reputation in the context of inaccurate information sources. Autonomous Agents and Multi-Agent Systems, 12(2), 183-198.
[28] Poundstone, W. (1993). Prisoner's Dilemma/John von Neumann, Game Theory and the Puzzle of the Bomb. Anchor.
[29] Mármol, F. G., & Pérez, G. M. (2011). Providing trust in wireless sensor networks using a bio-inspired technique. Telecommunication systems, 46(2), 163-180.
[30] Jarvenpaa, S. L., & Leidner, D. E. (1998). Communication and trust in global virtual teams. Journal of Computer-Mediated Communication, 3(4), 0-0.
[31] McDonough, E. F., Kahnb, K. B., & Barczaka, G. (2001). An investigation of the use of global, virtual, and colocated new product development teams. Journal of product innovation management, 18(2), 110-120.
[32] Baecker, R. M. (Ed.). (2014). Readings in Human-Computer Interaction: toward the year 2000. Morgan Kaufmann.
[33] Kim, Tae Kyung, and Hee Suk Seo. "A trust model using fuzzy logic in wireless sensor network." World academy of science, engineering and technology 42.6 (2008): 63-66.
[34] Salem, Maha, et al. "Would you trust a (faulty) robot?: Effects of error, task type and personality on human-robot cooperation and trust." Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction. ACM, 2015.
[35] Resnick, P., & Zeckhauser, R. (2002). Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. In The Economics of the Internet and E-commerce (pp. 127-157). Emerald Group Publishing Limited.
[36] Binder, Robert V. "Can a manufacturing quality model work for software?." IEEE Software 14.5 (1997): 101-102.
[37] Humann, J., Khani, N., & Jin, Y. (2014). Evolutionary computational synthesis of self-organizing systems. AI EDAM, 28(3), 259-275.
[38] Haken, H. (1978). Nonequilibrium phase transitions and bifurcation of limit cycles and multi-periodic flows. Zeitschrift für Physik B Condensed Matter, 29(1), 61-66.
[39] Jin, Y., & Chen, C. (2014). Field based behavior regulation for self-organization in cellular systems. In Design Computing and Cognition'12 (pp.605-623). Springer, Dordrecht.
[40] LaValle, Steven M. Planning algorithms. Cambridge university press, 2006.
[41] Wilensky, U. (2001). Modeling nature's emergent patterns with multi-agent languages. In Proceedings of EuroLogo (pp. 1-6).
[42] Jones, Chris, and Maja J. Mataric. "Adaptive division of labor in large-scale minimalist multi-robot systems." Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on. Vol. 2. IEEE, 2003.
[43] Groß, Roderich, et al. "Autonomous self-assembly in swarm-bots." IEEE transactions on robotics 22.6 (2006): 1115-1130.
[44] Humann, J., Khani, N., & Jin, Y. (2016). Adaptability Tradeoffs in the Design of Self-Organizing Systems. In ASME 2016 IDETC (pp.V007T06A016). American Society of Mechanical Engineers

