Evolutionary Constraint-based Multiobjective Adaptation for Self-Organizing Wireless Sensor Networks

Pruet Boonma and Junichi Suzuki
Department of Computer Science
University of Massachusetts, Boston
Boston, MA 02125, USA
{pruet, jxs}@cs.umb.edu

ABSTRACT

Wireless sensor applications (WSNs) are often required to simultaneously satisfy conflicting operational objectives (e.g., latency and power consumption). Based on an observation that various biological systems have developed the mechanisms to overcome this issue, this paper proposes a biologically-inspired adaptation mechanism, called MONSOON. With MONSOON, each application is designed as a decentralized group of software agents. This is analogous to a bee colony (application) consisting of bees (agents). Agents collect sensor data on individual nodes, and carry the data to base stations. They perform this data collection functionality by autonomously sensing their local and surrounding environment conditions and adaptively invoking biological behaviors such as pheromone emission, replication, reproduction and migration. Each agent has its own behavior policy, as a gene, which defines how to invoke its behaviors. MONSOON allows agents to evolve their behavior policies (i.e., genes) and simultaneously adapt to conflicting objectives. In addition to consider multiple objectives equally, MONSOON also allows agents to evolve in a constraint-based (or intentionally-biased) manner. A constraint is defined as an upper or lower bound for each objective. Simulation results show that MONSOON allows agents (WSN applications) to adapt to dynamics of the network (e.g., node/link failures) through evolution and simultaneously satisfy conflicting objectives in a self-organizing manner.

Keywords

Biologically-inspired networking, evolutionary and adaptive sensor networks, self-organizing sensor networks

1. INTRODUCTION

Autonomous adaptability is a key challenge in wireless sensor networks (WSNs) [1, 3, 2, 21]. With minimal intervention to/from human operators, WSN applications are required to adapt their operations to dynamic conditions in the network (e.g., network traffic and node/link failures). A critical issue in this challenge is that each WSN application tends to have conflicting operational objectives. For example, in data collection applications, the success rate of data transmissions from individual nodes to base stations is an important operational objective because higher success rate ensures that base stations have more data for operators to better understand operational environments and make better informed decisions. At the same time, the latency of data transmissions from individual nodes to base stations is another important operational objective. Lower latency ensures that base stations can collect sensor data for operators to understand operational environments quickly and make timely decisions. Success rate and latency conflict with each other. For improving success rate, hop-by-hop recovery is often applied; however, this can degrade latency. For improving latency, nodes may transmit data to base stations with the shortest paths; however, success rate can degrade because of traffic congestion on the paths.

In order to address this adaptability issue, the authors of the paper envision self-organizing[1] WSN applications that understand operational objectives and simultaneously satisfy them against the dynamics of network environments. The authors observe that various biological systems have developed the mechanisms necessary to realize this vision. For example, a bee colony self-organizes to satisfy conflicting objectives simultaneously [22]. Those objectives include maximizing the amount of collected honey, maintaining the temperature in a nest and minimizing the number of dead drones. If bees focus only on foraging, they fail to ventilate their nest and remove dead drones. Given this observation, the proposed application architecture, called BiSNET/e (Biologically-inspired architecture for Sensor NETworks, evolutionary edition), applies key biological mechanisms to design self-organizing adaptive WSN applications.

Figure 1 shows the BiSNET/e runtime architecture. The BiSNET/e runtime operates atop TinyOS on each node. It consists of two software components: *agents* and *middleware platforms*, which are modeled after bees and flowers, respectively. Each WSN application is designed as a decentralized group of agents. This is analogous to a bee colony (application) consisting of bees (agents). Agents read/collect sensor data on platforms (flowers) atop individual nodes, and carry the data to base stations on a hop-by-hop basis, in turn, to a backend server (the MONSOON server in Figure 1), which is

[1]Self-organization is a process in which a system's internal components autonomously react to environmental changes, interact with each other and create an ordered/stable state (equilibrium point) without being guided by any outside sources [5, 9].

modeled after a nest of bees. Agents perform this data collection functionality by autonomously sensing their local and surrounding network conditions and adaptively invoking biological behaviors such as pheromone emission, replication, reproduction, migration and death. A middleware platform runs on each node, and hosts one or more agents (Figure 1). It provides a series of runtime services that agents use to perform their functionalities and behaviors.

This paper focuses on a key mechanism in BiSNET/e, called MONSOON[2], which is an evolutionary adaptation mechanism for agents. Each agent possesses its own behavior policy, as a gene, which defines how to invoke its behaviors. MONSOON allows agents to evolve their behavior policies via genetic operations (e.g., mutation and crossover) and simultaneously adapt them to conflicting objectives (success rate, latency and power consumption) in dynamic netowrk environments. MONSOON also frees application designers from anticipating all possible network conditions and tuning their agents to the conditions at design time. Instead, agents can autonomously evolve and tune their behavior policies. This significantly simplifies the implementation and maintenance of agents (i.e., WSN applications).

In addition to consider multiple objectives equally, MONSOON allows agents to adapt their behavior policies in a constraint-based (or intentionally-biased) manner. A constraint is defined as an upper or lower bound for each objective. For example, a tolerable (upper) bound may be defined for the latency objective. This feature allows agent designers to flexibly specify their specific requirements (or priorities) on objectives. Also, constraints can often improve evolution speed by dedicating agents to satisfy the constraints.

This paper is organized as follows. Section 2 overviews the BiSNET/e runtime, particularly agent behaviors, and Section 3 describes the design of MONSOON. Section 4 evaluates MONSOON with a series of simulation results. Sections 5 and 6 conclude with some discussion on related work.

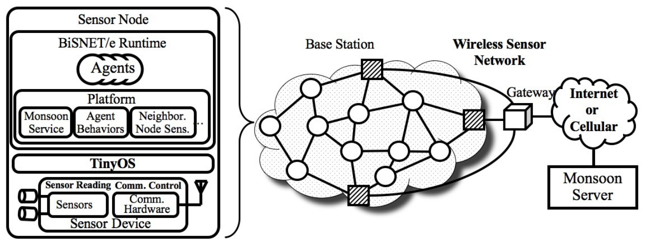


Figure 1: BiSNET/e Runtime Architecture

2. THE BISNET/E RUNTIME

BiSNET/e is currently designed to implement data collection applications. Every node reports its sensor reading to a base station periodically (i.e., at each data collection cycle). At the beginning of a WSN operation, an agent is deployed on each node. It has a randomly-generated behavior policy.

2.1 BiSNET/e Agent

Each agent consists of *attributes*, *body* and *behaviors*. *Attributes* carry descriptive information on an agent. They include agent's behavior policy (gene), sensor data to be reported to a base station, time stamp of the sensor data, and ID of a node where the sensor data is captured.

Body implements the functionalities of an agent: collecting and reporting sensor data to a base station.

Behaviors are the actions inherent to all agents. Inspired by biological entities (e.g., bees), agents sense their local and surrounding environment conditions, and behave according to the conditions without any intervention from/to other agents. This paper considers the following six behaviors.

(1) Food gathering and consumption: Biological entities strive to seek food for living. For example, bees gather nectar to produce honey. In BiSNET/e, each agent collects sensor data (as nectar) on the local platform and gains *energy* (as honey)[3] in each data collection cycle. Agents periodically consumes a constant amount of energy for living.

(2) Pheromone emission: Agents may emit different types of pheromones: *migration* and *alert pheromones.* They emit migration pheromones on their local nodes when they migrate to neighboring nodes. Each migration pheromone references the destination node an agent has migrated to. Agents also emit alert pheromones when they fail migrations within a timeout period. Each alert pheromone references a possibly failed node that an agent could not migrate to. Each pheromone has its own concentration. The concentration decays by half at every data collection cycle. A pheromone disapears when its concentration becomes zero.

(3) Replication: Agents make a copy of themselves when they gain energy in each data collection cycle. A child agent is placed on the platform that its parent resides on, and it inherits the parent's gene (behavior policy). Replicated agents are intended to move toward base stations to report collected sensor data.

(4) Migration: Agents may move from one node to another. Migration is used to transmit agents (sensor data) to base stations. Each agent chooses a migration destination node by sensing three types of pheromones available on the local node: base station, migration and alert pheromones.

Each base station periodically propagates *base station pheromones* to individual nodes in the network. Their concentration decays on a hop-by-hop basis. Using base station pheromones, agents can sense where base stations exist approximately, and move toward the base stations by climbing pheromone's concentration gradient[4].

An agent may move to a base station by following a migration pheromone trace on which many other agents have traveled. The trace can be the shortest path to the base station. Conversely, an agent may goes off a migration pheromone trace and follows another path to a base station when the concentration of migration pheromones is too high on the trace (i.e., when too many agents have followed the trace). This avoids separating the network into islands. The network can be separated with the migration paths that too many agents follow, because the nodes on the paths consume more power and go down earlier than the others.

An agent may also avoid moving to a node referenced by an alert pheromone. This allows agents to reach base stations by bypassing link/node failures.

(5) Reproduction: Once agents arrive at the MON-

[2]Multiobjective Optimization for Network of Sensors using an evOlutionary algOrithm with coNstraints

[3]In BiSNET/e, the concept of energy does not represent the amount of physical power, but logically affects agent behaviors.

[4]Base station pheromones are designed after the Nasonov gland pheromone, which guides bees to move toward their nest [8].

SOON server (a nest for agents; see Figure 1), they may produce their offspring with other agents (mating partners). A reproduced agent inherits a gene (behavior policy) from its parents via crossover, and mutation may occur on the inherited gene. Some of reproduced agents that adapt to the current network conditions are dispatched to the network, and they perform a generation change by taking over existing agents running on individual nodes.

Reproduction is intended to evolve agents so that the biological entities that fit better to the environment become more abundant. It retains the agents whose fitness to the current network conditions is high (i.e., the agents that have effective behavior policies, such as moving toward a base station in a short latency), and eliminates the agents whose fitness is low (i.e., the agents that have ineffective behavior policies, such as consuming too much power to reach a base station). Through successive generations, effective behavior policies become abundant in agent population while ineffective ones become dormant or extinct. This allows agents to adapt to dynamic network conditions.

(6) Death: Agents periodically consume energy for living, and expend energy to invoke their behaviors. (The energy costs to invoke behaviors are constant for all agents.) Agents die due to lack of energy when they cannot balance energy gain and expenditure. The death behavior is intended to eliminate the agents that have ineffective behavior policies. For example, an agent would die before arriving at a base station if it follows a too long migration path. When an agent dies, the local platform removes the agent and releases all resources allocated to the agent.

2.2 A Sequence of Agent Behaviors

Figure 2 shows a sequence of behaviors that each agent performs on a node in each data collection cycle. An agent reads sensor data (as nectar) with the underlying sensor device and gains a constant amount of energy (as honey). Given the energy intake (E_F), each agent updates its energy level as follows.

$$E(t) = E(t-1) + E_F \tag{1}$$

$E(t)$ is the current energy level of the agent, and $E(t-1)$ is the agent's energy level in the previous data collection cycle. t is incremented by one at each data collection cycle. If an agent's energy level ($E(t)$) becomes very low (below the death threshold: T_D), the agent dies due to starvation[5].

An agent replicates itself in each data collection cycle. A replicating (parent) agent splits its energy units to halves ($\frac{E(t)-E_R}{2}$), gives a half to its child agent, and keeps the other half. E_R is the energy cost for an agent to perform the replication behavior. A child agent contains the sensor data that its parent collected, and carries it to a base station.

Each replicated agent migrates toward a base station on a hop by hop basis. On each intermediate node, it examines Equation 2 to determine which next node it migrates to.

$$WS_j = \sum_{t=1}^{3} w_t \frac{P_{t,j} - P_{t_{min}}}{P_{t_{max}} - P_{t_{min}}} \tag{2}$$

An agent calculates this weighted sum (WS_j) for each neighboring node j, and moves to a node that generates the highest weighted sum. t denotes pheromone type; P_{1j}, P_{2j} and P_{3j} represent the concentrations of base station, migration and alert pheromones on the node j. $P_{t_{max}}$ and $P_{t_{min}}$ denote the maximum and minimum concentration of P_t among neighboring nodes.

When an agent is migrating to a neighboring node, it emits a migration pheromone on the local node. If the agent's migration fails, it emits an alert pheromone. Each alert pheromone spreads to one-hop away neighboring nodes.

```
for each data collection cycle
do
    Read sensor data and gain energy ($E_F$).
    Update energy level ($E(t)$).
    if $E(t) <$ the death threshold ($T_D$)
      then Invoke the death behavior.
    Invoke the replication behavior to make a child agent.
    Give the half of the current energy level to
     a replicated (child) agent.
    for each migrating agent
    do
        Determine the destination node of migration.
        Emit a migration pheromone on the local node.
        Migrate to a neighboring node.
        if Migration fails
          then
            Emit an alert pheromone on the local node
            Propagate it to neighboring nodes.
```

Figure 2: A Sequence of Agent Behaviors in Each Data Collection Cycle

2.3 Agent Behavior Policy

Each agent's behavior policy (gene) contains a set of weight values in Equation 2 ($w_t, 1 \leq t \leq 3$). w_1 and w_3 are non negative, and w_2 can be negative. These weight values govern how agents perform the migration behavior. For example, if an agent has zero for w_2 and w_3, the agent ignores migration and alert pheromones, and moves toward the base stations by climbing the concentration gradient of base station pheromones. If an agent has a positive value for w_2, it follows a migration pheromone trace on which many other agents have traveled. A negative w_2 value allows an agent to go off a migration pheromone trace and follow another path toward a base station. If an agent has a positive w_3, it moves to a base station by bypassing link/node failures.

Agents have randomly-generated behavior policies at the beginning of WSN operation. MONSOON allows them to dynamically evolve their behavior policies according to the current network conditions.

3. MONSOON

MONSOON is a constraint-based evolutionary multiobjective adaptation mechanism designed for agents in BiSNET/e. It allows agents to heuristically adapt to multiple objectives simultaneously. This adaptation process is performed through *elite selection* and *genetic operations*. The elite selection process evaluates the agents that arrive at base stations, based on given objectives, and chooses the best (or elite) ones. Then, elite agents are propagated to the network in order to to perform genetic operations and reproduce an offspring (next generation) agent on each node. Elite selection is performed in the MONSOON server (see Figure 1), and genetic operations are performed in each node.

3.1 Adaptation Objectives

[5]If all agents are dying on a node at the same time, a randomly selected agent will survive. At least one agent runs on each node.

With MONSOON, agents consider three conflicting objectives: *latency*, *cost* and *success rate* of their migration (i.e., data transmission) from individual nodes to base stations.

(1) Latency represents the time required for an agent to travel to a base station from a node where the agent is born (replicated). In this paper, as depicted in Equation 3, latency is measured as a ratio of this agent travel time to the physical distance (PD) between a base station and a node where the agent is born. The MONSOON server knows the location of each node with a certain localization mechanism.

$$Latency = \frac{agent\ travel\ time}{PD} \tag{3}$$

(2) Cost represents the amount of energy required for an agent to travel to a base station from a node where the agent is born. In this paper, cost is measured with the topological distance between between a base station and a node where the agent is born (TD), the transmission range (radius) of each node, and TD.

$$Cost = TD \cdot \frac{transmission\ range}{PD} \tag{4}$$

(3) Success Rate is measured as the ratio of the number of agents that arrive at base stations to the number of nodes.

MONSOON allows agents to evolve their behavior policies so that they maximize success rate and minimize latency and cost.

3.2 Elite Selection

Figure 3 shows how elite selection occurs at the MONSOON server in each data collection cycle. The first step is to obtain three objective values (i.e., latency, cost and success rate) from each of the agents that reach the MONSOON server via base stations. If an agent's objective value does not meet a given constraint, the agent is eliminated. Then, each of the remaining agents is evaluated whether it is dominated by another agent. An agent is considered to be dominated if another agent outperforms it in all of the three objectives.

In the next step, a subset of non-dominated agents are selected as elite agents. This is performed with a hypercube space, which a three dimensional space whose axes represent three objectives (i.e., latency, cost and success rate). Each axis of the hypercube space is divided so that the space is divided into small cubes. Each non-nominated agent is plotted in this hypercube space based on their objective values. A single agent is randomly selected from each cube as an elite agent. This elite agent selection is designed to maintain the diversity of elite agents' genes. The diversification of agent genes contribute to improve agents' adaptation even to unanticipated network conditions.

Figure 4 shows an example hypercube space. Each axis is divided into two ranges; therefore, eight cubes exist. Thus, the maximum number of elite agents is eight. In this example, six (A to F) non-dominated agents are plotted in the hypercube space. Three agents (B, C, and D) are plotted in the lower left cube, while the other three agents (A, E, and F) are plotted in three different cubes. From the lower left cube, only one agent is randomly selected as an elite agent. A, E, and F are selected as elite agents because they are in different cubes.

3.3 Genetic Operations

Once elite agents are selected, the MONSOON server propagates them to each node in the network. They are propagated with base station pheromones.

An agent performs the reproduction behavior on each node through genetic operations (crossover and mutation) when elite agents arrive at the node. As a mating partner, the agent selects one of the elite agents that has the most similar gene. Gene similarity is measured with the Euclidean distance between the values of two genes.

Reproduction occurs with a certain reproduction probability. During reproduction, a child agent inherits the half of its gene from its parent agent and the other half from its parent's mating partner. Mutation occurs on the child agent's gene with a certain mutation probability by randomly changing gene values within a predefined value range.

4. SIMULATION RESULTS

This section shows a set of simulation results to evaluate MONSOON and BiSNET/e. MONSOON is implemented in Java and connects to BiSNET/e which is implemented in TinyOS and simulated in TOSSIM.

In each simulation, a WSN consists of 100 sensor nodes uniformly deployed in a 300x300 meters observation area. Sensor nodes' communication range is 30 meters. A base station is deployed on the northwestern corner of the observation area. The base station connects to MONSOON via emulated serial port connection. This simulation assumes that each sensor node has to report a sensor reading every 5 minutes; thus, a generation of MONSOON is also in 5 minutes period. For genetic operation, the reproduction probability is set to 0.75 and mutation probability is 0.025.

To measure the degree of self-organization of sensor network, a system entropy is measured for each generation. Generally speaking, entropy represents the disorderedness of the system. Therefore, when the entropy becomes lower, system becomes more self-organized. In this paper, entropy is measured from the probability that agents' performance becomes similar, i.e. self-organized, or dissimilar, i.e. disordered. To evaluate the similarity of agents' performance, the performance of agent is presented as a set of states, or cubes, in a hyper-cube, which each axis of the hypercube represents each objective. Each state, or cube, has objective values' range assigned from hypercube. The probability

```
Empty the archive
while true
do  Empty the population pool.
    Collect agents from the network.
    Add collected agents to the population pool.
    Move agents from the archive to the population pool.
    Empty the archive
    for each agent of the ones in the population pool
    do  Obtain each objective value.
        if An objective value breaks a constraint
          then Remove the agent from the population pool.
    for each agent of the ones in the population pool
    do  if not dominated by all other agents in
        the population pool
          then Add the agent to the archive.
    Select elite agents from the archive.
    Propagate elite agents to the network.
```

Figure 3: Elite Selection in the MONSOON server

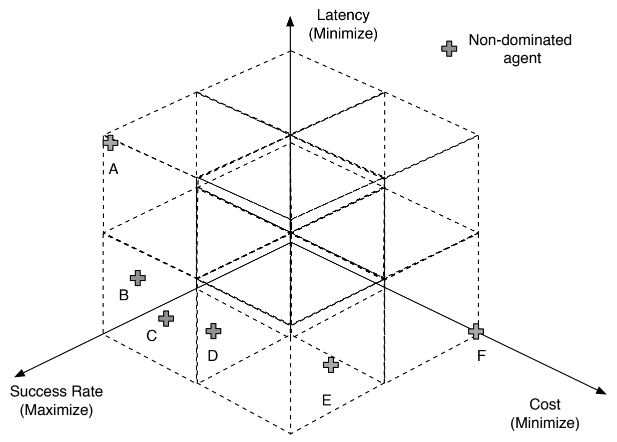


Figure 4: An Example Elite Selection

of each state is measured from the number of agents whose objectives values are according to state's objectives range over the total number of collected agents. Thus, entropy can be measured from following equation;

$$H = \sum_{i \in S} p_i \log(p_i) \quad (5)$$

Where S is the set of system state, p_i of the probability of the state i.

In each figure in this section, the X-axis depicts the simulation time, in generations. The left Y-axis shows average cost, latency, and entropy while the right Y-axis shows the success rate. The average cost is presented in a unit of topological over physical distance, i.e., hop count over 30 meters. The unit of latency is second over physical distance, i.e., 30 meters and the unit of success rate is percentage. The value of entropy is normalized to be between zero and one.

4.1 Simulation Results without Constraints

Figure 5 (a) shows the average value of each objectives from collected agents and the entropy of the system in each generation when all sensor nodes are performed without malfunction. From the figure, After six generations, MONSOON starts improve the performance of the WSN; it can be observed from increasing of success rate, and decreasing of latency and cost. With in 16 generations, the performance is converged such that the latency is close to zero second over 30 meters, cost is close to one hop over 30 meters and the success rate is 100%. Moreover, MONSOON allows the sensor network to become more self-organized which can be observed from the decreasing of entropy value. The simulation result shows that MONSOON can autonomously improve the performance, i.e. objective values, of WSN, and allows WSN to becomes more self-organized.

Figure 5 (b) portraits a scenario when 25 nodes are added randomly into the network at the 20th generation. At first, after deploy the new sensor nodes, the success rate are dropped dramatically because the gene of the agents on the new sensor nodes are random; therefore, agent from the new sensor nodes cannot migrate efficiently toward the base station. Also, the agents from new sensor nodes disturb agents from the other nodes such that some agents from the other node cannot successfully move to the base station. This can be observed from the increasing of cost and latency as well. However, sooner or later, the gene of the new sensor nodes is improved and the success rate becomes higher while network cost and latency become lower.

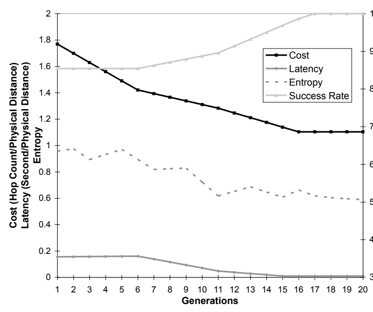
From the figure 5 (b), the entropy value also increases at 21st generation from 0.6 to almost 1.0, which indicates the increasing of disorderedness of the sensor network. However, the entropy level decreases continuously between 22nd and 38th generation and stay the same lowest level as before 20th generation. MONSOON is able to autonomously adjusts the operational parameters of redeployed sensor nodes to retain the performance and order of WSN.

From the figure 5 (c), at 20th generation, 25 sensor nodes are randomly selected and deactivated to simulate sensor node failure. Therefore, only 75 nodes are working properly. At 20th generation, the success rate of the WSN drops dramatically because of the sudden change in the WSN, similar to latency and cost.

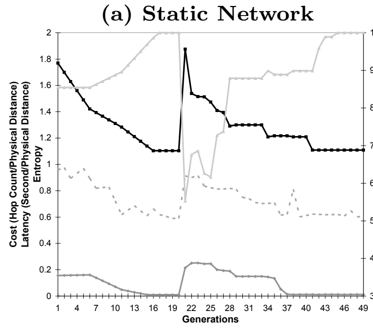
Agents on sensor nodes close to deactivated nodes still try to migrate to the deactivated nodes; hence, the migration may be unsuccessful or taking longer time and/or distance. However, after 21st generation, the three objective values are improved continuously and converge at the 30th generation. the For the entropy value, at 21st generation, entropy increases because of changes in sensor network, then it continuously decreases again until reach the lowest level at 40th generation. The simulation results shows that MONSOON allows WSN to survives a partial sensor network failure by adjusting the operational parameters of WSN to be suitable to the changes in network condition.

Figure 5 (d) shows the result of a simulation when 20 sensor nodes are deactivated at the 20th generation. In contrast with the simulation in Figure 5 (c) which the sensor nodes to be deactivated are selected randomly, the sensor nodes are selected in spatially-correlated fashion such that the sensor nodes who are located in the middle of observation area are selectively deactivated. Hence, the sensor network contains a hole in the middle of the network. Compared with the result in figure 5 (c), MONSOON takes longer time to improve the success rate of the WSN. The success rate converges at about 52nd generation to approximately 98%. The cost and latency also show the similar trend. Particularly, after 52nd generation, the average value of cost and latency are higher than the values just before 20th generation because agents have to detour in a longer path to avoid the hole in the middle of the network. Nevertheless, the entropy level shows the same trend as the previous simulations. The simulation results shows that MONSOON allows WSN to survives a spatially-correlated sensor nodes failure by adjusting the operational parameters of WSN to be suitable to the changes in network condition.

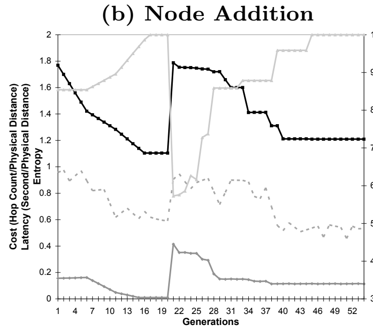
Figure 5 (e) shows the result of a simulation which has two base stations deployed at the northwestern and southeastern corner of the observation area initially. Then, at the 20th generation, the base station at the southeastern corner is deactivated. From the figure, at the 21st generation, the success rate drops sharply to about 45% from about 100% in the 20th generation because more than a half of the agents still try to move to the base station at the southeastern corner. However, the success rate is improved successively and reach the same level as before the base station is deactivated at the 37th generation.



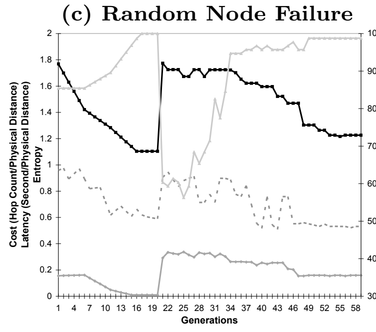
(a) Static Network



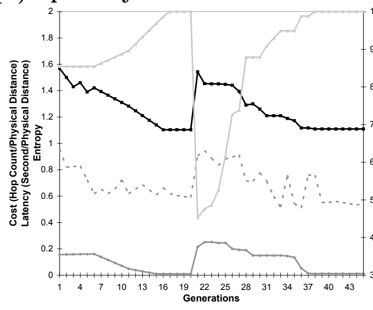
(b) Node Addition



(c) Random Node Failure

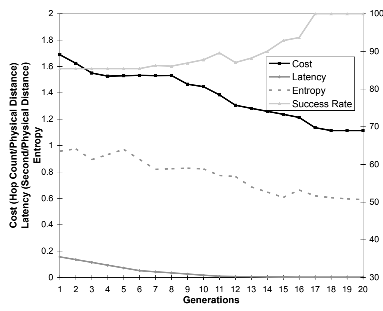


(d) Spatially-Correlated Node Failure

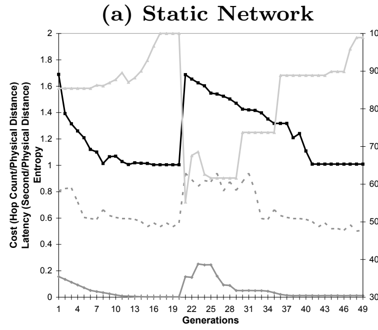


(e) Base Station Failure

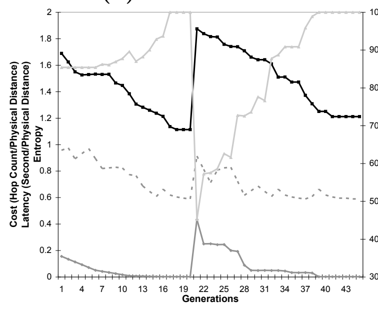
Figure 5: Objective Values without Constraints



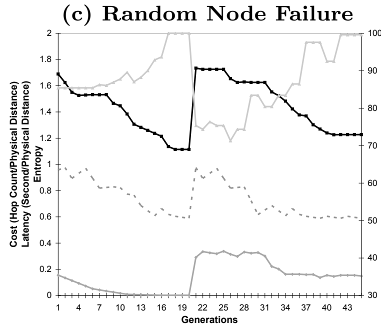
(a) Static Network



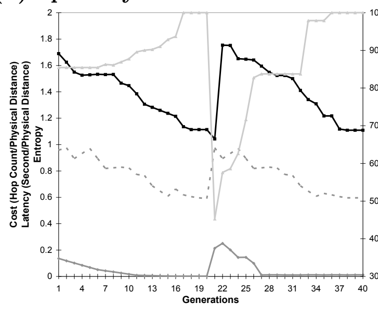
(b) Node Addition



(c) Random Node Failure

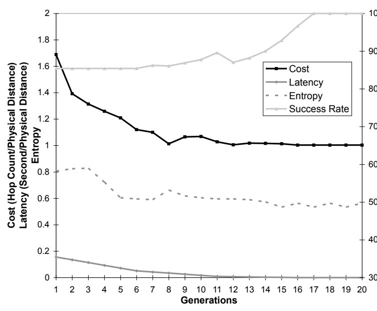


(d) Spatially-Correlated Node Failure

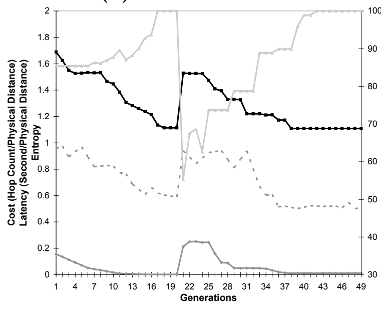


(e) Base Station Failure

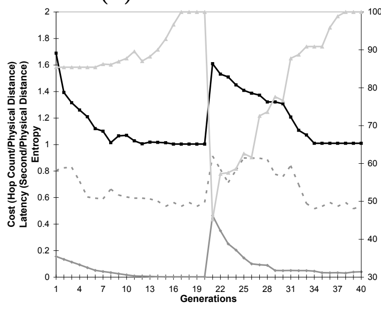
Figure 6: Objective Values with a Latency Constraint



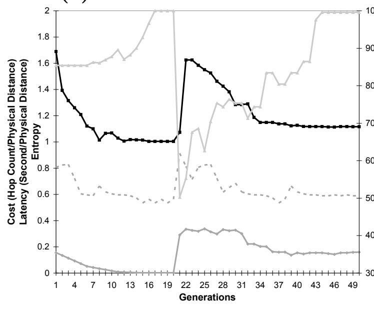
(a) Static Network



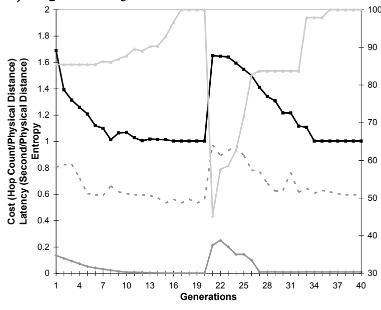
(b) Node Addition



(c) Random Node Failure



(d) Spatially-Correlated Node Failure



(e) Base Station Failure

Figure 7: Objective Values with Latency and Cost Constraints

Network cost, latency, and entropy level show the same trend. MOSOON allows WSN to survives a base station failure by autonomously directing all agents to remaining base station.

4.2 Simulation Results with a Latency Constraint

In this section, a user-defined constraint is assigned to the latency such that the average latency of agents is expected to be lower than 0.05 second.

Figure 6 (a) shows the average value of each objectives from agents collected in each generation. From the figure, MONSOON focuses on decreasing the latency to meet the constraint. Hence, the average latency is decreased faster than the other objectives. In particular, the latency is optimized at 11th generation, compared with 15th generation in figure 5 (a) while cost is optimized at 18th generation, compared with 16th generation in figure 5 (a). The success rate shows the same trend as cost. Nevertheless, after 16 generation, all objectives are converged close to optimized value, i.e. minimum or maximum bound. Moreover, MONSOON allows the sensor network to become more self-organized which can be observed from the decreasing of entropy. This simulation result shows that MONSOON is able to optimize the operational parameters regarding to user-defined constraint.

Figure 6 (b) has the same simulation parameter as the simulation in figure 5 (b). Similar to figure 6 (a), latency is improved faster than cost and success rate both before and after node redeployment. The entropy level also decreases after the 21st generations. This simulation result shows that MONSOON allows WSN to adapt to changes in network condition and is able to optimize the operational parameters regarding to user-defined constraint.

Figure 6 (c), 6 (d), and 6 (e) have the same simulation parameter as in the simulation in figure 5 (c), 5 (d) and 5 (e), respectively. The simulation results in those three figures show similar trend as in figure 6 (b); thus, the simulation results show that MONSOON allows WSN to adapt to changes in network condition and is able to optimize the operational parameters regarding to user-defined constraint.

4.3 Simulation Results with Latency and Cost Constraints

In this section,constraints are assigned to both latency and cost. The latency is expected to be lower than 0.05 second over 30 meters while the cost is expected to be lower than 1.05 hop over 30 meters.

From the figure 7 (a), MONSOON focuses on decreasing the latency and cost concurrently to meet the constraints. Hence, the average latency and cost are decreased faster than the success rate. Particularly, the success rate remains unimproved till 10th generation, in contrast with the success rate in figure 5 (a), which is improved since 6th generation. However, all objective values are optimized at 17th generation. Thus, MONSOON is able to optimize the operational parameters regarding to multiple user-defined constraint.

Figure 7 (b) has the same simulation parameter as the simulation in figure 5 (b). From the figure, after 21st generation, latency and cost are improved rapidly while success rate improved slowly because of MONSOON gives more priority on latency and cost. Particularly, latency and cost become optimized at about 40th generation, compared with 43rd generation in figure 7 (b). Figure 7 (c), 7 (d) and 7 (e) show similar trends as in figure 7 (b). The simulation results show that MONSOON allows WSN to adapt to changes in network condition and is able to optimize the operational parameters regarding to multiple user-defined constraint.

4.4 Memory Footprint

Table 1 shows the memory footprint of the BiSNET/e runtime in a MICA2 mote, and compares it with the footprint of Blink (an example program in TinyOS), which periodically turns on and off an LED and Agilla, which is a mobile agent platform for WSNs [7]. The BiSNET/e runtime is lightweight in its footprint thanks to the simplicity of the biologically-inspired mechanisms in BiSNET/e. BiSNET/e can even run on a smaller-scale nodes, for example, TelosB, which has 48KB ROM.

Table 1: Memory Footprint in a MICA2 Mote

	RAM (KB)	ROM (KB)
BiSNET	2.0	26.0
Blink	0.04	1.6
Agilla	3.59	41.6

5. RELATED WORK

This work is an extension to the authors' prior work, BiSNET [4]. [4] shows that BiSNET allows agents to autonomously adapt to network conditions. However, it did not investigate evolutionary adaptation (i.e., MONSOON); agent behavior policies were manually configured and fixed. Unlike BiSNET, BiSNET/e allows agents to dynamically adapt their behavior policies even to unanticipated network conditions such as node failures, base station failures and node additions.

Agilla proposes a programming language to implement mobile agents for sensor networks, and provides a runtime system (interpreter) to operate agents on TinyOS [7]. On the other hand, BiSNET/e does not focus on investigating a new programming language for sensor networks. BiSNET/e agents and Agilla agents have a similar set of behaviors such as migration and replication. Both of them are also intended to be used for similar applications (e.g., wildfire detection). However, Agilla does not address the research issues that BiSNET/e focuses on; adaptation, self-organization, and optimization. In addition, BiSNET/e focuses on its design simplicity and runtime lightweightness. As shown in table 1, BiSNET/e is much more lightweight than Agilla.

[23] proposes a generic communication primitive for sensor networks. The primitive hides lower level implementation, i.e. network communication, while maintain the ability of programmer to control over the communication behavior of sensor node. The authors use a biological communication mechanism, called pheromone, as the communication primitive. A set of pheromone properties, i.e. type, strength, source, and payload, and instructions, i.e. deposit and smell is provided to application developer to be used as communication mechanism between sensor node. Different from BiSNET/e which is designed base on biological system from the bottom up, the pheromone concept in this work is not properly integrated into the other part of sensor software development. Hence, application developers have to deal with two different levels of concept, a high level concept of

pheromone, and low level concept of sensor node programming.

There are several research efforts to apply evolutionary algorithm to sensor networks. Evolutionary algorithm is used for network cluster-based routing [15, 12, 13, 6], data processing [11], localization [24] and sensor node placement [10, 25]. In [15, 12, 13, 6], genetic algorithm is used to find a set of cluster head which will be used as relaying point for sending data form sensor nodes to base stations. Particularly, [6] claims that a multi-objective optimization using genetic algorithm is used in their work; however, multiple objectives employed in their work are combined into a single fitness function. [11] uses genetic algorithm to analyze sensor data, i.e. find approximate polynomial from partial sensor data. [24, 10, 25] focus on using genetic algorithm for spatial aspect of WSN, i.e., localization and optimized sensor node placement. Most of the mentioned works assume network to be static; thus, they can not perform well in dynamic environment.

In [14, 20, 17], evolutionary multi-objective optimization is used for finding optimal location of each sensor nodes in a sensor network. Moreover, [18, 19] propose to use evolutionary multi-objective optimization for data routing in sensor networks. In contrast with MONSOON, their optimization process performs completely in a central server, which can leads to scalability problem. MONSOON is carefully designed to perform partially in central server and each sensor nodes in order to minimize extra energy consumption and be scalable.

A constraint-based multi-objective optimization is used in [16] for routing in wireless sensor networks. The objectives considered in this paper are such as distance to the destination. The constraints considers in the paper are such as energy levels of sensor nodes along a route and hazardous level of the route. Similar to MONSOON, the optimization process in this work finds an optimal route for each sensor in a decentralize manner. Each node can make their own decision which intermediate node it should forward the data to, in order to send the data to a destination. The node makes decision by applied a rule set provided by central server to current property, e.g. energy level, of each neighbors. In contrast with MONSOON, the central server cannot autonomously improve the rule set based on current condition of the sensor networks. Thus, the proposed framework in this paper cannot adapt well to the changes in network condition. Moreover, human administrators have to carefully design the rule set in order to archive the required objectives. MONSOON, on the other hand, can autonomously archive the objectives without any human intervention.

6. CONCLUSION

This paper describes a biologically-inspired adaptation mechanism for WSNs. It allows WSN applications to autonomously adapt to dynamics of the network (e.g., node/link failures, base station failures and node addition) through evolution and simultaneously satisfy conflicting objectives (e.g. success rate, latency and energy consumption) in a self-organizing manner. Thanks to simple biologically-inspired mechanisms, the proposed mechanism is implemented lightweight.

7. REFERENCES

[1] K. Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks. *Elsevier Ad Hoc Networks*, 3(3):325–349, 2005.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Elsevier Journal of Computer Networks*, 38:393–422, 2002.

[3] J. Blumenthal, M. Handy, F. Golatowski, M. Haase, and D. Timmermann. Wireless sensor networks - new challenges in software engineering. In *Proc. of Emerging Technologies and Factory Automation*, September 2003.

[4] P. Boonma and J. Suzuki. BiSNET: A biologically-inspired middleware architecture for self-managing wireless sensor networks. *Elsevier J. of Computer Networks*, 51, 2007.

[5] S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula, , and E. Bonabeau. *Self Organization in Biological Systems*. Princeton University Press, 2003.

[6] K. P. Ferentinos and T. A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Elsevier J. of Computer Nets.*, 51(4), 2007.

[7] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *Proc. of Int'l Conf. on Distributed Computing Systems*, June 2005.

[8] J. B. Free and I. H. Williams. The role of the nasonov gland pheromone in crop communication by honey bees. *Int'l J. of Behavioural Biology, Brill Publishing*, 41, 1972.

[9] C. Gershenson and F. Heylighen. When can we call a system self-organizing? In *Proc. of the 7th European Conference on Artificial Life*, 2003.

[10] H. Y. Guo, L. Zhang, L. L. Zhang, and J. X. Zhou. Optimal placement of sensors for structural health monitoring using improved genetic algorithms. *Smart Materials and Structures*, 13(3):528–534, 2004.

[11] J. Hauser and C. Purdy. Sensor data processing using genetic algorithms. In *Proc. of the 43rd IEEE Midwest Symposium on Circuits and Systems*, 2000.

[12] S. Hussain and A. W. Matin. Hierarchical cluster-based routing in wireless sensor networks. In *Proc. of the 5th Int'l Conf. on Info. Processing in Sensor Nets*, 2006.

[13] S. Jin, M. Zhou, and A. S. Wu. Sensor network optimization using a genetic algorithm. In *Proc. of Multiconf. on Systemics, Cybernetics and Informatics*, 2003.

[14] D. B. Jourdan and O. L. de Weck. Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility. In *Proc. of SPIE Defense and Security Symposium*, 2004.

[15] R. Khanna, H. Liu, and H. Chen. Self-organisation of sensor networks using genetic algorithms. *Inderscience Int'l J. of Sensor Networks*, 1(3):241–252, 2006.

[16] D. Mahjoub and H. El-Rewini. Adaptive constraint-based multi-objective routing for wireless sensor networks. In *Proc. of the IEEE Int'l Conf. on Pervasive Services*, 2007.

[17] A. M. Raich and T. R. Liszkai. Multi-objective genetic

algorithm methodology for optimizing sensor layouts to enhance structural damage identification. In *Proc. of the 4th Int'l Workshop on Structural Health Monitoring*, 2003.

[18] R. Rajagopalan, C. Mohan, P. Varshney, and K. Mehrotra. Multi-objective mobile agent routing in wireless sensor networks. In *Proc. of IEEE Congress on Evolutionary Comp.*, 2005.

[19] R. Rajagopalan, P. K. Varshney, K. G. Mehrotra, and C. K. Mohan. Fault tolerant mobile agent routing in sensor networks: A multi-objective optimization approach. In *Proc. of the 2nd IEEE Upstate New York Workshop on Communication and Networking*, 2005.

[20] R. Rajagopalan, P. K. Varshney, C. K. Mohan, and K. G. Mehrotra. Sensor placement for energy efficient target detection in wireless sensor networks: A multi-objective optimization approach. In *Proc. of Annual Conf. on Information Sciences and Systems*, 2005.

[21] P. Rentala, R. Musunuri, S. Gandham, and U. Sexena. Survey on sensor networks. In *Proc of Int'l Conf. on Mobile Computing and Networking*, 2001.

[22] T. Seeley. *The Wisdom of the Hive*. Harvard University Press, 2005.

[23] L. Szumel and J. D. Owens. The virtual pheromone communication primitive. In P. B. Gibbons, T. Abdelzaher, J. Aspnes, and R. Rao, editors, *Proc. of the Second IEEE International Conference on Distributed Computing in Sensor Systems*, volume 4026 of *Lecture Notes in Computer Science*, pages 135–149. Springer, June 2006.

[24] V. Tam, K. Y. Cheng, and K. S. Lui. Using micro-genetic algorithms to improve localization in wireless sensor networks. *J. of Comm., Academy Publisher*, 1(4), 2006.

[25] J. Zhao, Y. Wen, R. Shang, and G. Wang. Optimizing sensor node distribution with genetic algorithm in wireless sensor network. In *Proc. of Int'l Symp. on Neural Nets.*, 2004.