IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 16, 2020, accepted October 8, 2020, date of publication October 13, 2020, date of current version November 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3030629

Improve Energy Consumption and Signal Transmission Quality of Routings in Wireless Sensor Networks

WEI-CHANG YEH[1], (Senior Member, IEEE), YUNZHI JIANG[2], CHIA-LING HUANG[3], NEAL N. XIONG[4], CHENG-FENG HU[5], AND YUAN-HUI YEH[6]

[1]Integration and Collaboration Laboratory, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 300, Taiwan
[2]School of Mathematics and Systems Science, Guangdong Polytechnic Normal University, Guangzhou 510665, China
[3]Department of International Logistics and Transportation Management, KAINAN University, Taoyuan City 338, Taiwan
[4]Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK 74464, USA
[5]Department of Applied Mathematics, National Chiayi University, Chiayi 600, Taiwan
[6]Business School, Law School, University of New South Wales, Sydney, NSW 2052, Australia

Corresponding author: Yunzhi Jiang (jiangyunzhi@foxmail.com)

This work was supported in part by the Natural Science Foundation of China and Ministry of Science and Technology, R.O.C., under Grant 61702118, Grant MOST 102-2221-E-007-086-MY3, and Grant MOST 104-2221-E-007-061-MY3.

ABSTRACT Wireless (smart) sensor networks (WSNs) comprise a myriad of embedded wireless smart sensors. They play a cardinal role in the functioning of many applications, such as the Internet of Things, smart grids, smart production systems, and smart homes, which ultimately render them paramount instruments in the modern age. Recent advances in WSNs have resulted in the rapid development of sensors. However, WSNs will only able to achieve better execution efficiencies if their energy consumption - owing to limited battery life and difficulty of recharging - can be better controlled. Moreover, signal transmission quality determines WSN performance. Hence, two main concerns - energy consumption and signal transmission quality - should be addressed to improve the performance of WSNs. Thus, a new bi-objective simplified swarm optimization algorithm (bSSO) is proposed by employing the concepts of simple routing, SSO, and crowd distance. The performance and applicability of the proposed bSSO using eight different parameter settings are demonstrated through an experiment involving ten WSN benchmarks ranging from 100 to 1000 sensors. The proposed algorithm is then compared with NSGA-II, which is an algorithm widely used to solve multi-objective problems. The results show that the proposed bSSO can successfully achieve the aim of this work.

INDEX TERMS Wireless sensor network, energy consumption, network reliability, routing, algorithms.

I. INTRODUCTION

Wireless (smart) sensor networks (WSN) comprise devices embedded with wireless smart sensors and play a crucial role in the networking of many objects in real-world and daily life applications. The applications include healthcare and medical systems [1], industrial automation [2], Internet of Things (IoT) [3], energy systems [4], smart cities [5], the smart transportation industry [6], and the semiconductor industry [7]. Hence, wireless (smart) sensor networks (WSN) comprise devices embedded with wireless smart sensors and play a crucial role in the networking of many objects in real-world and daily life applications.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Li.

Owing to their greater flexibility over wired networks [3], [4], [17], [19], [22], sensors are deployed, operated, and embedded widely in devices, buildings, vehicles, and other items to model, gather, sense, investigate, and exchange data, to interconnect objects, and to improve production efficiency and offer more efficient resource consumption [1]–[3]. If the operation of a WSN system fails because of a shortage of the battery power of the sensors, then, signals, information, or data flow cannot successfully be transmitted through the network. Therefore, academia and the industry have placed plentiful and continuous attention on this topic and have made prime efforts in energy consumption researches for WSNs [4], [16]–[19].

Numerous methods have been proposed for reducing energy consumption in WSN systems. Mekonnen *et al.* [8] investigated a WSN prototype to minimize the energy consumption of video surveillance systems. Trapasiya and Soni [9] focused on achieving retransmission energy reduction in WSNs. Quang and Kim [10] proposed gradient routing to reduce industrial WSN energy consumption. Setiawan *et al.* [11] proposed a WSN energy-management policy to increase transfer efficiency. Liu *et al.* [12] designed WSNs to optimize their energy consumption. Chanak *et al.* [13] considered ways to deploy sensors to balance WSN energy consumption. Collotta *et al.* [14] and Kumar and Chaturvedi [15] aimed to optimize the energy efficiency of dynamic WSNs via fuzzy methods. Akram and Cho [16] optimized WSN energy consumption while considering the uncertainty of security attacks on the sensors using a fuzzy method [17].

The sequence of sensors that goes from the source sensor (via related links) to the sink sensor is called a routing path [17], [18], [34]. Routing paths forward signals via Wi-Fi, NFC, ZigBee, 4G/LTE, etc. From the user's perspective, service quality refers to whether a response, message, or signal from the system or themselves is able to arrive at its destination reliably, effectively, economically, and efficiently, anywhere and anytime [3], [19]–[23]. Hence, routing plays an important role in improving service quality and has been the focus of several works in recent literature for residual energy optimization [1], [17]–[23]. However, reported approaches lack the ability to find an optimal routing path based on more than one objective [3], [17]–[23].

It is thus necessary and important to balance various aspects when evaluating WSNs; for example, minimizing total energy consumption and maximizing system reliability are a pair of conflicting objectives in routing. Derived from real-life applications, this context sets up a bi-objective problem to find one routing from all available routings in a WSN to minimize energy consumption and increase service quality, which depends on the system's reliability.

In order to solve this NP-hard combinatorial optimization problem, which is essentially difficult to solve within polynomial time, a new systematic algorithm called bi-objective simplified swarm optimization (bSSO) is proposed in this paper to understand and manage this issue in practice. Then, bSSO is evaluated in terms of solution quality; these are the major contributions of the proposed bSSO. This problem-solving approach is based on simplified swarm optimization (SSO), which was originally developed by Yeh [24] and is a technique for finding and ordering nondominated solutions.

The contributions of this study are outlined below:

1. The novelty of the proposed bSSO in this study complements and strengthens the problem that reported approaches in the literature lack the ability to find an optimal routing path based on more than one objective [3], [17]–[23].
2. Derived from real-life applications, this context sets up a bi-objective problem to find one routing from all available routings in a WSN to minimize energy consumption and increase service quality, which depends on the system's reliability.
3. The proposed bSSO effectively solves the bi-objective problem in WSNs, which belongs to this NP-hard combinatorial optimization problem.

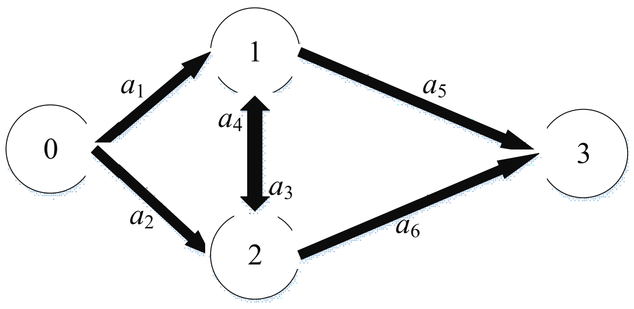


FIGURE 1. An example WSN with four nodes.

The remainder of this paper is structured as follows. Section II presents notations, assumptions, and the mathematical modeling of the proposed bi-objective problem to address energy consumption and signal transmission quality. Section III introduces the simplified swarm optimization (SSO) algorithm and the crowd distance, which are the basis of the proposed bSSO for updating solutions and ranking and selecting nondominated solutions, respectively. The proposed bSSO and its novelties are presented in Section IV. Section V shows a performance analysis of the proposed bSSO with eight different parameter settings by comparing it with the famous NSGA-II [25] in ten WSN benchmark problems with sensor numbers of 100, 200, 300, ..., and 1000. Our conclusions are given in Section VI.

II. OVERVIEW NOTATIONS, ASSUMPTIONS AND MATHEMATICAL PROBLEM DESCRIPTION

In this section, the mathematical model for energy consumption in WSNs is presented, together with the notations used in this study.

A. NOTATIONS

In graph theory, a path is a sequence of arcs connected by a sequence of vertices [34]. For example, there are at least four paths from node 0 to node 3, as shown in Figs. 1-2 [32], [35]. In this study, a routing refers to a special path from (source) sensor 1 to (sink) sensor n to denote the process for traffic in a WSN.

n:	number of sensors
m:	number of links
V:	a set of sensors $V = \{1, 2, \ldots, n\}$.
A:	a set of links $A = \{a_1, a_2, \ldots, a_m\}$
$e_{i,j}$, a_k:	the link that connects sensors i and j and the kth link in A.
β:	bits of data to be transferred from the source sensor 1 to the sink sensor n (unit: $\beta = 512$ bytes $= 4096$ bits)

D:	the distance function, which is defined as the Euclidean distance between paired sensor sensors, e.g., $D(1, 5) = 141.42$ if the positions of sensors 1 and 5 are (0, 0) and (100, 100), respectively. (unit: m)
$G(V, A, D)$:	a complete WSN with V, A, and D.
X:	$X = (x_1, x_2, \ldots, x_k)$ is a routing (solution) representing the way to send a signal in sequence from the source sensor 1 to sensor x_1, then to sensor x_2 and so on to sensor x_k before reaching the sink sensor n.
$\vert\bullet\vert$:	number of elements in $\bullet$.
E_e:	energy consumption in terms of electric power when transmitting or receiving data (unit: nJ/bit).
E_{fs} and E_{mp}:	amplifier parameters for transmission corresponding to the free-space and the two-ray models (unit of E_{fs}: pJ/bit/m^2 and unit of E_{mp}: pJ/bit/m^4), respectively.
$D_{\max}$	threshold distance to transfer data, with $D_{\max} = \sqrt{E_{fs}/E_{mp}}$
$E_r(\beta, \bullet)$:	energy consumption in link or routing $\bullet$ when receiving x bits of data. Note that $E_r(\beta, e_{i,j})$ is unaffected by $D(i, j)$. (unit: nJ)
$E_t(\beta, \bullet)$:	energy consumption in link or routing $\bullet$ when transmitting x bits of data. Note that $E_t(\beta, e_{i,j})$ is affected by $D(i, j)$. (unit: nJ)
$E_T(\beta, \bullet)$:	total energy consumption in link or routing $\bullet$ when receiving and transmitting x bits of data, i.e., $E_T(\beta, e_{i,j}) = E_r(\beta, e_{i,j}) + E_t(\beta, e_{i,j})$. (unit: nJ)
$R(\beta, \bullet)$:	transmission reliability of link or routing $\bullet$ when receiving and transmitting β bits of data.
$\mathrm{N_{sen}}$:	number of sensors used in the test problem.
$\mathrm{N_{run}}$:	number of runs for the algorithms.
$\mathrm{N_{gen}}$:	number of generations in each run.
$\mathrm{N_{sol}}$:	number of solutions in each generation.
$\mathrm{N_{non}}$:	number of selected temporary nondominated solutions.
$F_l(\bullet)$:	lth fitness function value of solution $\bullet$.
$Max(\bullet)$:	maximal value of $\bullet$.
$Min(\bullet)$:	minimal value of $\bullet$.
X_i:	ith solution for $i = 1, 2, \ldots, \mathrm{N_{sol}}$.
$x_{i,j}$:	jth variable in X_i for $i = 1, 2, \ldots, \mathrm{N_{sol}}$ and $j = 1, 2, \ldots, \mathrm{N_{var}}$.
P_i:	*pBest* in the ith solution for $i = 1, 2, \ldots, \mathrm{N_{sol}}$; P_i is the best solution among all solutions updated based on X_i.
$p_{i,j}$:	jth variable in P_i for $i = 1, 2, \ldots, \mathrm{N_{sol}}$ and $j = 1, 2, \ldots, \mathrm{N_{var}}$.
$gBest$:	index of the best solution among all solutions, i.e., $F(P_{gBest})$ is better than or equal to $F(P_i)$ for $i = 1, 2, \ldots, \mathrm{N_{sol}}$.
ρ_I:	random number generated uniformly within interval I.
c_g, c_p, c_w, c_r:	positive parameters used in SSO with $c_g + c_p + c_w + c_r = 1$.
C_g, C_p, C_w:	$C_g = c_g$, $C_p = C_g + c_p$, $C_w = C_p + c_w$ and $C_r = 1 - C_w$.
$\widehat{d}_{l,i}$:	$Min\{\frac{F_l(X_i)-F_l(X_j)}{Max(F_l)-Min(F_l)}\}$.
Π_t:	set of all solutions to generate or be generated in the t^{th} generation for $t = 1, 2, \ldots, \mathrm{N_{gen}}$.
π_t:	set of all temporary non-dominated solutions in Π_t.

B. ENERGY CONSUMPTION MODEL AND AN EXAMPLE

Suppose a WSN $G(V, A, D)$ with a set of sensors $V = \{1, 2, \ldots, n\}$ and a set of links $A = \{a_1, a_2, \ldots, a_m\}$. For example, assume the WSN shown in Figure 3 with the sensors set $\{1, 2, 3, 4\}$ and the links set $\{a_1, a_2, a_3, a_4, a_5\}$. Sensor 1 and sensor 4 are the source sensor and the sink sensor, respectively. All links are undirected.

A routing is a way in which an application's endpoints respond to client requests, e.g., there are only four possible routings in this example: (1, 2, 4), (1, 3, 4), (1, 2, 3, 4), and (1, 3, 2, 4), as shown in Fig. 3. From radio model [26], the energy consumption for transmitting and receiving x bits of data between paired sensors, such as sensors i and j, is modeled as follows:

$$E_t(\beta, e_{i,j}) = \begin{cases} \beta E_e + \beta E_{fs} D^2(i,j), & \text{if } D(i,j) \le D_{\max} \\ \beta E_e + \beta E_{mp} D^4(i,j), & o.w. \end{cases} \tag{1}$$

$$E_r(\beta, e_{i,j}) = \beta E_e \tag{2}$$

where

$$D_{\max} = \sqrt{E_{fs}/E_{mp}} \tag{3}$$

$D_{\max}$ is a threshold value. Note that Eq. (1) depends on both the received or transmitted number of data bits β and the distance, whereas Eq. (2) only depends on β.

The energy consumption of the sensors is included in the data transmitting and receiving parts in the WSNs. Thus, the total energy consumption for transmitting β bits of data from the source sensor 1 to the sink sensor n is formulated as follows in Eq. (4).

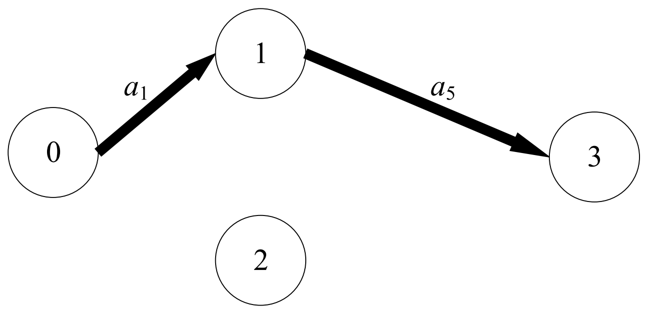
$$E_T = \sum_{e \in path} [E_t(\beta, e) + E_r(\beta, e)] \tag{4}$$

where e indicates the data transmitting routing that belongs to path, which demonstrates the path from the source sensor 1 to the sink sensor n in WSN.

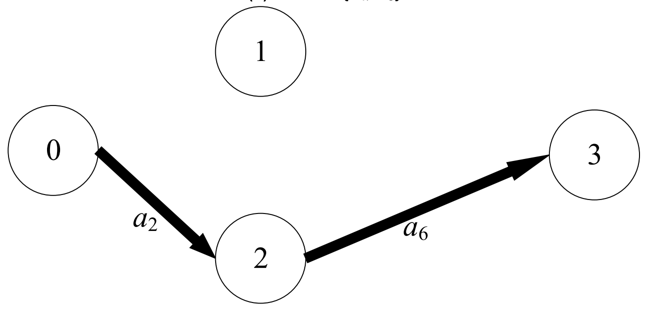
The unit of the transmitted data number has to be converted from bytes to bits, as in

$$\beta = 512 \text{ bytes} = 4096 \text{ bits}. \tag{5}$$

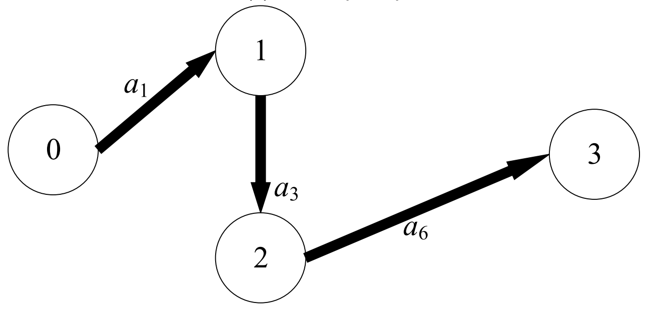
The energy consumption for receiving β bits of data according to Eq. (2) is calculated as shown in Eq. (6) for



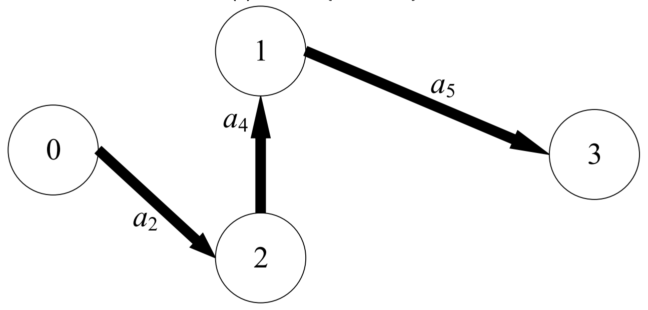
(*a*) *Path* 1*:* {*a*₁, *a*₅}



(*b*) *Path* 2*:* {*a*₂, *a*₆}



(c) *Path* 3*:* { *a*₁, *a*₃, *a*₆}



(d) *Path* 4*:* { *a*₂, *a*₄, *a*₅}

FIGURE 2. **Four paths in Fig. 1.**

receiving data at sensor 2 from source sensor 1 and for receiving data at sink sensor 4 from sensor 2.

$$E_r(50, e_{1,2}) = E_r(50, e_{2,4}) = \beta E_e = 204800(\text{nJ}) \quad (6)$$

Here, the calculation of the total energy consumption in a WSN can be done as in the following example. Suppose a WSN as the one shown in Fig. 3 and all the related parameter values listed in Table 1. Suppose that a data packet $\beta = 512$ bytes is transmitted on routing (0, 1, 3); the total energy consumption can be calculated as follows.

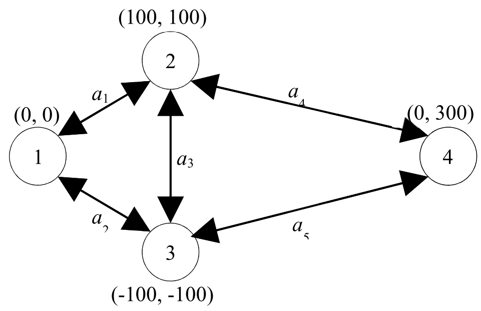


FIGURE 3. **An example WSN.**

TABLE 1. **Values of parameters.**

Parameter	Value
E_e	50 nJ/bit
E_{fs}	10 pJ/bit/m^2
E_{mp}	0.0013 pJ/bit/m^4
d_0	87 m
Data packet size	512 bytes
Number of packets transmitted	1
Source sensor coordinate	(0, 0)
Sensor 2 coordinate	(100, 100)
Sink sensor coordinate	(0, 300)

To calculate the energy consumption generated by the operation of transmitting x bits of data in the path, we first need to calculate the distance between each of the arcs in the path. The distances between source sensor 1 and sensor 2 and between sensor 2 and sink sensor 4, e.g., $D(1, 2)$ and $D(2, 4)$, can be calculated as follows via Eqs. (7) and (8), respectively.

$$\begin{aligned} D(1,2) &= \sqrt{(0-100)^2 + (0-100)^2} \\ &= 141.42 > D_{\max} = 87 \quad (7) \\ D(2,4) &= \sqrt{(100-0)^2 + (100-300)^2} \\ &= 223.607 > D_{\max} = 87 \quad (8) \end{aligned}$$

Because both $D(1, 2)$ and $D(2, 4)$ are larger than the threshold value $D_{\max} = 87$, the energy consumption generated by the operation of transmitting β bits of data from source sensor 1 to sensor 2 and from sensor 2 to the sink sensor 4 are calculated as follows.

$$E_r(50, e_{1,2}) = xE_e + xE_{mp}D^4(1,2) = 2334720(\text{nJ}) \quad (9)$$

$$E_r(50, e_{2,4}) = xE_e + xE_{mp}D^4(1,2) = 13312252962(\text{nJ}) \quad (10)$$

The total energy consumption for transmitting a data packet of $\beta = 512$ bytes through path (1, 2, 4) in the WSN shown in Fig. 3 is calculated via Eq. (11) as follows.

$$\begin{aligned} E_T(50,(1,2,4)) &= \sum_{i=0}^{l} [E_t(\beta, e_{v_i,v_{i+1}}) + E_r(\beta, e_{v_i,v_{i+1}})] \\ &= 13314997282(\text{nJ}) \quad (11) \end{aligned}$$

C. RELIABILITY

Network reliability is an important index for evaluating and validating the performance of various networks. Hence, it was considered as one of the objective functions in this study to improve WSN service quality. Routing reliability is defined as the probability that a routing function can meet our requirements [4]. Let *path* be the number of links connected to the sensors in routing *path*, and $R(\beta, e)$ be the transmission reliability of link e in *path* when receiving and transmitting β bits of data. In each routing, sensors are connected in series. A general formulation of the routing reliability p can be calculated simply using the following equation [4]:

$$R(\beta, path) = \coprod_{\forall e \in path} R(\beta, e) \tag{12}$$

D. MATHEMATICAL MODEL

In this study, we consider a bi-objective problem focusing on two significant factors, namely energy consumption [17]–[21] and reliability (i.e., transmission quality) [4], to improve WSN service quality [4]. Let $E_T(\beta, e)$ and $R(\beta, e)$ be the functions of the energy consumption and reliability of link e for x bits of data, respectively. The bi-objective problem is presented below:

$$\text{Min} \sum_{\forall e \in path} E_T(\beta, e) \tag{13}$$

$$\text{Max} \prod_{\forall e \in path} R(\beta, e) \tag{14}$$

where $E_T(\beta, e)$ and $R(\beta, e)$ are discussed in Sections II.B and II.C, respectively, for all routings p.

The objective functions in Eqs. (13) and (14) minimize energy consumption and maximize the reliability of routings separately. Both Eqs. (13) and (14) can be solved based on shortest-path problems (which are polynomial problems) if any one of both is considered individually [4]. However, the proposed bi-objective problem needs to consider both Eqs. (13) and (14) simultaneously, and the number of routings in the abovementioned mathematical model increases with the size of the problem [4], [16]. Hence, the proposed bi-objective problem is NP-hard and cannot be solved in polynomial time [16], [17]. Thus, in this paper, a new efficient algorithm to solve this important multi-objective problem is proposed.

III. SIMPLIFIED SWARM OPTIMIZATION (SSO) AND CROWDING DISTANCE

The proposed bSSO is based on SSO to find and update better solutions from generation to generation intelligently and uses crowding distance to rank and select nondominated solutions systematically. Hence, both SSO and crowding distance are introduced briefly in this section.

A. SSO AND AN EXAMPLE

SSO, which was first developed by Yeh in 2009 [24], is a simple but powerful Artificial Intelligence algorithm with the characteristics of being population-based (i.e., $N_{sol} > 1$), performing all-variable updates (i.e., all variables are updated), having a stepwise-function update mechanism, and being a hybrid of swarm intelligence (i.e., with a leader solution) and evolutionary computations (i.e., survival of the fittest). For each updated solution, the basic idea of SSO is that [3], [4], [24], [27], [28]:

1) a part of the updated variables is based on *gBest*, which is the index of the best solution among all existing solutions;

2) a part of the variables is based on its *pBest*, which is the best current solution in its evolutionary history;

3) a part of the variables is based on current variables;

4) the rest of the variables are randomly generated feasible variables.

c_g, c_p, c_w, and c_r are constant parameters that satisfy $c_g + c_p + c_w + c_r = 1$ and are used to decide the abovementioned proportions to balance the global search when exploring new search areas, the local search for exploiting the current solution space, and the random search to escape possible local traps. The update mechanism of SSO is very simple, efficient, and flexible [3], [4], [24], [27], [28], and its simplest form for updating variable $x_{i,j}$ is presented below:

$$x_{i,j} = \begin{cases} p_{gBest,j} & \text{if } \rho_{[0,1]} \in [0, C_g) \\ p_{i,j} & \text{if } \rho_{[0,1]} \in [C_g, C_p) \\ x_{i,j} & \text{if } \rho_{[0,1]} \in [C_p, C_w) \\ x & \text{if } \rho_{[0,1]} \in [C_w, 1]. \end{cases} \tag{15}$$

Owing to the simplicity of the stepwise-function update mechanism discussed above, it is easier to customize SSO by either replacing any item of its stepwise function with other algorithms, even hybrid algorithms in sequence or in parallel [3], to solve various problems than to customize other algorithms [3], [24], [27], [28]

Hence, after the invention of SSO, it became widely known as a swarm intelligence-based random optimization algorithm and has played as a very significant role in relevant studies of artificial intelligence. Furthermore, SSO has been applied in many studies to solve different types of problems in various fields [20], [29], [33]–[36].

Detailed steps of a simple SSO algorithm can be described as follows:

STEP S0. Generate $X_k = P_k$ randomly, find *gBest* such that $F(X_{gBest})$ is better than or equal to $F(X_k)$, and let $t = i = 1$, where $k = 1, 2, \ldots, N_{sol}$.

STEP S1. Update X_i and calculate $F(X_i)$ based on Eq.(15).

STEP S2. Let $P_i = X_i$ if $F(X_i)$ is better than $F(P_i)$. Otherwise, go to STEP S5.

STEP S3. Let $gBest = i$ if $F(P_i)$ is better than $F(P_{gBest})$.

STEP S4. Let $i = i+1$ and go to STEP S1 if $i < N_{sol}$.

STEP S5. Let $t = t+1$ if $t < N_{gen}$. Otherwise, halt.

SSO is much simpler than other famous AI algorithms, such as particle swarm optimization, which requires updating velocity and position vectors, genetic algorithms, which operate based on both crossovers and mutations, the estimated distribution algorithm, for which it is difficult to find an appropriate probability model, and the immune-system algorithm, which does not consider the interactions between variables [3], [4], [24], [27], [28].

B. CROWDING DISTANCE

A nondominated solution is a solution whose some of its objective functions are better than other solutions and some are not. The number of nondominated solutions is infinite in multi-objective problems, whereas the number of total solutions was always limited to a constant, namely N_{sol}, in this study. Hence, a few of the found nondominated solutions have to be abandoned if the total number of found nondominated solutions is larger than N_{sol}, which still has to maintain a good diversity of solutions efficiently. Therefore, the crowding distance [25] was adapted here to rank all found nondominated solutions first and select a certain number of nondominated solutions if the number of nondominated solutions found is larger than N_{sol}.

The overall crowding distance value is the sum of the individual shortest normalized distances corresponding to each objective and can be calculated as follows:

$$\sum_{i=1}^{|X_i|} \sqrt{\widehat{d}_{1,i}^{\,2} + \widehat{d}_{2,i}^{\,2}}, \tag{16}$$

where, for all routings X_i and $\widehat{d}_{l,i} = Min\{\frac{F_l(X_i)-F_l(X_j)}{Max(F_l)-Min(F_l)}\}$ for all nondominated solution X_j and $i \neq j$, $Min(F_l)$ is the minimal value of the lth objective function, and $Max(F_l)$ is the maximal value of the lth objective function.

Eqs. (1), (2) and (6) in this multi-objective problem are shortest path problems by letting $C(e)$, $E(e)$ or $T(e)$ be the distance of arc e, and they are able to solve the problem using a shortest path algorithm, e.g., the Dijstra algorithm, in polynomial time if such equation is considered individually [16], [17]. However, both Eqs. (3) and (5) are NP-hard [16], [17], even considered independently, since the longest path is an NP-Hard problem and cannot be solved in polynomial time [16]. Thus, traditional shortest path algorithms are unable to solve this problem.

If all paths from the source node to the sink node in the above mathematical model are known [16], [17], it may be possible to solve this multi-objective problem by substituting each path into Eqs. (1)-(6). Unfortunately, there are many paths in the above mathematical model, and their number will increase with the size of problem [16], [17]. There is thus also a need for a new, more efficient algorithm to solve this important multi-objective problem in IoT networks.

Furthermore, there may be many non-dominated solutions in multi-objective problems, and it is always inconvenient and difficult for decision-makers themselves to select one as an answer for the problem from all non-dominated solutions [18]–[22].

This study therefore proposes a new three-phase algorithm based on multi-criteria decision-making methods to overcome the above obstacles in the multi-objective problem to improve the service quality of IoT system technologies.

IV. PROPOSED BSSO

The proposed bSSO is a population, all-variable, and stepwise function-based soft-computing method, i.e., there are a fixed number of solutions in each generation and all variables must be updated based on the stepwise function with each solution. Details of the proposed bSSO are discussed in this section.

A. FLEXIBLE-LENGTH SOLUTION

A complete network is a graph in which any two nodes are connected by an edge. A WSN can be treated as a complete network because any pair of sensors is able to communicate with each other. In the proposed bSSO, each solution corresponds to a routing and is represented by a flexible-length vector that contains the sensor sequence in the routing path, considering the following property of WSNs.

Property 1: Each solution X includes at least one simple path from the source sensor to the sink sensor.

For example, both $p_1 = (1, 2, 4)$ and $p_2 = (1, 2, 3, 4)$ are routings from sensors 1 to 4 in Fig. 3, and there are two and three links in p_1 and p_2, respectively. Hence, any solution with no repeated sensors is a routing, even if such solution was generated randomly.

B. TEMPORARY NONDOMINATED SOLUTIONS AND ELITE SELECTION

Each nondominated solution is a solution that is not dominated by any other solution. However, a solution that is not dominated by other found solutions in the t^{th} generation may be dominated by a new solution in the $(i + 1)^{th}$ generation. Hence, a new concept called temporary non-dominated solution is introduced to represent solutions that are not dominated by any other solution over their evolutionary history.

Let Π_t be the set of all solutions to be generated in the t^{th} generation, π_t be the set of all temporary nondominated solutions in Π_t, and S_t be the selected solutions from Π_t to generate new solutions in the $(t + 1)^{th}$ generation. A new elite selection policy is, thus, proposed to select temporary nondominated solutions and maintain the diversity of temporary nondominated solutions using the crowd distance if necessary. In the proposed bSSO, we have

$$S_t = \begin{cases} \pi_t \cup \underset{\sim}{S_t} & \text{if } |\pi_t| \leq N_{sol} \\ \underset{\sim}{\pi_t} & \text{otherwise,} \end{cases} \tag{17}$$

where

$\underset{\sim\sim}{S_t}$ $=\{N_{sol} - |\pi_t|$ randomly selected solutions from $\Pi_t - \pi_t\}$

$\underset{\sim\sim}{\pi_t}$ $=\{$the top N_{sol} solutions in their crowd distances from $\pi_t\} \subseteq \pi_t$.

C. NOVEL UPDATE MECHANISM

Without loss of generality, both the source and sink sensors are removed from all solutions presented in the rest of this study. Similar to most versions of SSO, the N_{sol} of the proposed bSSO is fixed in each generation and each of the variables (sensors in this study) of all solutions must be updated based on a stepwise function. However, the stepwise function proposed here is totally different to that of any previous versions of SSO. The stepwise update function used in the

TABLE 2. Example of the update process in the proposed bSSO.

Variable	1	2	3	4	5
X_5	2	4	6	5	3
X^*	3	8	7		
ρ	.43	.65	.70	.99	
New X_5	3	4	6	$4^{\#}$	

"#" indicates that they are generated randomly

proposed bSSO is listed below for multi-objective problems for each solution X_i, with $i = 1, 2, \ldots, \mathrm{N_{sol}}$:

$$x_{i,j} = \begin{cases} x_i^* & \text{if } \rho_{[0,1]} \in [0, C_g) \text{ and } j < |X^*| \\ x_{i,j} & \text{if } \rho_{[0,1]} \in [C_g, C_w) \text{ and } j < |X_i| \\ x & \text{otherwise,} \end{cases} \tag{18}$$

where $X^* = (x_1^*, x_2^*, \ldots, x_k^*)$ is one of the temporary nondominated solutions selected from S_i randomly, $k = |X^*|$, $\rho_{[0,1]}$ is random number generated uniformly in [0, 1], $|X|$ is the number of variables in X, and the number of variables of the new solution generated from X_i is a random integer generated in the interval [Min{$|X_i|$, $|X^*|$}, Max{$|X_i|$, $|X^*|$}].

For example, let $X_5 = (2, 4, 6, 5, 3)$ and $X^* = (3, 8, 7)$ be selected randomly from the temporary nondominated solutions in S_i. Assume that the new length of the new X_5 is 3, which is a random number within [Min{$|X_5|$, $|X^*|$}=3, Max{$|X_5|$, $|X^*|$}=5] = {3, 4, 5}. Let $C_g = 0.50$, $C_w = 0.95$, and $\rho = (\rho_1, \rho_2, \rho_3, \rho_4) = (.43,.65,.7,.99)$. From Eq. (18), the process to update X_5 is displayed in Table 2, and the new $X_5 = (3, 4, 6, 4)$ is obtained.

Note that:

1) The number of variables in the new X_5 is not always greater than the largest among the old X_5 and X^* and not less than the smallest among the old X_5 and X^*.
2) Sensor IV appeared in the second and fourth positions, i.e., repeats twice, in the new X_5. Repeated sensors in a solution means that the signal formed a cycle, e.g., there is a cycle from sensors 4 to 6 and back to 4 in the new X_5. Hence, it is possible to have a new solution with cycles after implementing the proposed update mechanism.

D. REMOVE OF CYCLES

It is impossible to have repeated sensors in the routing path in which signals are sent in real-life applications. However, as shown in the example presented in Section 4.3, it is possible to have solutions with repeated sensors in the sequence after updating using the proposed update mechanism. Hence, we establish the following properties:

Property 2: If vector X contains a simple path, then the solution representing such a simple path is better than X.

Property 3: If sensor x appears at least twice in solution X, then solution X contains at least one cycle that starts and ends at sensor x.

Thus, a simple method is proposed based on the following property to remove cycles by shortening the solutions without affecting their feasibility.

Property 4: If sensor x appears in the ith position and the jth position in a solution, then the sensors from the $(i+1)$th positions to the jth position form at least one cycle and can be removed.

For example, the new $X_5 = (3, 4, 6, 4)$ presented in Table 2 can be simplified to a vector with only one dimension (3) after using Property 4 to remove the redundant cycle from sensors 4 to 6 to 4.

E. PSEUDOCODE OF THE PROPOSED BSSO

The details of the proposed bSSO are described in the following steps.

STEP 0. Generate X_i randomly, calculate $F_1(X_i)$ and $F_2(X_i)$, and let $t = 2$ and $S_1 = \Pi_1 = \{X_i|$ for all $i\}$, where $i = 1, 2, \ldots, \mathrm{N_{sol}}$.

STEP 1. Update X_i to $X_{\mathrm{Nsol}+i}$ based on Eq. (18) for $i = 1, 2, \ldots, \mathrm{N_{sol}}$, and let $S = \{X_{\mathrm{Nsol}+1}, X_{\mathrm{Nsol}+2}, \ldots, X_{2\times\mathrm{Nsol}}\}$ and $\Pi_t = (S_{t-1} \cup S)$.

STEP 2. Let π_t ={all temporary nondominated solutions in Π_t}.

STEP 3. If $|\pi_t| < \mathrm{N_{sol}}$, let S^*= {$\mathrm{N_{sol}} - |\pi_t|$ solutions selected randomly from $\Pi_t - \pi_t$}, $S_t = \{X_1, X_2, \ldots, X_{\mathrm{Nsol}}\}$, where X_i is the i^{th} solution in $\pi_t \cup S^*$, and go to STEP 5.

STEP 4. Let S^*= {$\mathrm{N_{sol}}$ solutions selected randomly from π_t} and $S_t = \{X_1, X_2, \ldots, X_{\mathrm{Nsol}}\}$, where X_i is the i^{th} solution in S^*.

STEP 5. If $t < \mathrm{N_{gen}}$, then let $t = t+1$ and go back to STEP 1. Otherwise, halt.

V. NUMERICAL EXPERIMENTS

Numerical experiments of the parameter-setting procedure and the performances of bSSO were carried out based on eight settings for ten benchmark problems, and the experimental results were compared with those obtained using NSGA-II [25] and are presented in this section. Note that NSGA-II is the best-known algorithm for multi-objective problems.

A. PARAMETER SETTINGS AND EXPERIMENTAL ENVIRONMENTS

Parameter setting always affects the performance of algorithms. To determine the parameters, i.e., C_g and C_w, of the proposed bSSO, eight different parameter settings were tested: (C_g, C_w) = (.25,.45), (.25,.70), (.50,.75), (.25,.95), (.50,.95), (.75,.95), (.50, 1.0), and (.75, 1.0). Hence, eight bSSOs with different parameter settings were tested in our numerical experiments. To make them easy to recognize, the bSSO with parameter settings (C_g, C_w) is denoted as bSSO(C_g, C_w), e.g., C_g =.25, C_w = 45 and $c_r = 1 - C_w$ =.55 in bSSO(.25,.45). Note that $C_g = c_g$, $C_w = C_g + c_w$ and $c_r = 1 - C_w$.

To demonstrate the performance of the proposed bSSO and select the best parameter settings, all bSSOs with different parameter settings were executed for ten benchmarks, namely $\mathrm{N_{sen}} = 100, 200, \ldots, 1000$, in a WSN. To further validate the superiority of the proposed bSSO, the results obtained with it were compared with those obtained with NSGA-II [25], which is the best-known multi-objective algorithm.

Both the proposed bSSOs with eight different parameter settings and NSGA-II were coded in DEV C++ on a 64-bit Windows 10 PC, implemented on an Intel Core i7-6650U CPU @ 2.20 GHz notebook with 16 GB of memory.

To make a fair comparison, for all algorithms, namely the eight bSSOs algorithms and NSGA-II, $\mathrm{N_{sol}} = \mathrm{N_{non}} = 50$, $\mathrm{N_{gen}} = 100$, and $\mathrm{N_{run}} = 500$, i.e., the same solution number, generation number, size of external repository, and run number for each benchmark problem was used, respectively. To guarantee that the final optimal solution can be obtained in each run, each algorithm executed 500 runs fairly.

The crossover rate $c_{\text{cross}} = 0.70$ and the mutation rate $c_{\text{mute}} = 0.30$ for NSGA-II so that the calculation number of the fitness function of all algorithms was limited to $\mathrm{N_{sol}} \times \mathrm{N_{gen}} = 5000$. This was also done to make a fair comparison.

B. PERFORMANCE METRICS

There are infinite real nondominated solutions and the Pareto front is the set of all real nondominated solutions. To represent the Pareto front using the limited number of solutions found using the algorithms, there are two major concerns when evaluating the solution quality of these algorithms: convergence metrics and diversity metrics. The convergence metrics measure the distances between all solutions in π_{Ngen} and the Pareto front, and the diversity metrics measure the diversity of solutions in π_{Ngen}, where π_{Ngen} is the set of all temporary nondominated solutions found in the last generation.

Among these metrics, the general distance (GD) [29] and spacing (SP) [30] are the most popular indexes used for the convergence metrics and diversity metrics, respectively. Let d_i be the shortest Euclidean distance between the Pareto front and the i^{th} temporary nondominated solution, say X_i^*, in π_{Ngen}, and let $\underline{d}$ be the average sum of all d_i for $i = 1, 2, \ldots, |\pi_{\mathrm{Ngen}}|$. The GD defined below is the average of sum of the squares of d_i [30]:

$$\mathrm{GD} = \frac{1}{|\pi_{N_{gen}}|}\sqrt{\sum_{i=1}^{|\pi_{N_{gen}}|} d_i^2}. \tag{19}$$

In general, the smaller the convergence metrics are, the closer to the Pareto front the solutions are and the better the solution quality is [30].

The definition of SP is very similar to that of standard deviation in probability theory and statistics and can be written as follows [30]:

$$\mathrm{SP} = \sqrt{\frac{1}{|\pi_{N_{gen}}| - 1}\sum_{i=1}^{|\pi_{N_{gen}}|} (\underline{d} - d_i)^2}. \tag{20}$$

In general, the smaller the SP is, the higher the diversity of solutions along the Pareto front and the better the solution quality becomes [30].

Both the GD and SP require knowing the Pareto front to be calculated according to their formulas [29], [30]; however, it is impossible to find the Pareto front because its number is infinite. To overcome this obstacle without conducting a nearly exhaustive method to simulate the Pareto Front, the eight bSSO-based algorithms and NSGA-II were used in $\mathrm{N_{run}} = 500$ simulations for each benchmark problem with $\mathrm{N_{sol}} = 50$, i.e., there are $500 \times 50 \times 2 = 50000$ solutions obtained in the end for each benchmark problem. These solutions, which are not dominated by any other solution, are grouped to form set π^* and create a simulated Pareto Front to calculate the GD and SP.

TABLE 3. **Number of temporary nondominated solutions obtained in 500 runs.**

$\mathrm{N_{sen}}$	NSGA-II	(.25,.45)	(.25,.70)	(.25,.95)	(.50,.75)	(.50,.95)	(.50,1.0)	(.75,.95)	(.75,1.0)
100	24538	24428	24580	24969	24600	24912	**24993**	24896	24982
200	24952	24957	24947	**25000**	24974	24998	**25000**	**25000**	24999
300	24979	24996	24990	**25000**	**25000**	24999	**25000**	**25000**	**25000**
400	24988	24999	24999	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**
500	24993	**25000**	24997	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**
600	24995	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**
700	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**
800	**25000**	24997	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**
900	24996	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**
1000	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**	**25000**

C. ANALYSIS OF RESULTS

All the results obtained in our experiments are listed in this section in tables based on the GD and SP to compare the converge and diversity of bSSO with eight different parameter settings and NSGA-II [25]. In these tables, the best of all algorithms and of all bSSO-based algorithms are indicated in bold and underlined, respectively.

A total of $\mathrm{N_{run}} \times \mathrm{N_{sol}} = 2500$ solutions were found for each algorithm and each benchmark problem. The number of found temporary nondominated solutions are listed in Table 3. From Table 3, it can be seen that almost all bSSO-based algorithms were better than NSGA-II in obtaining the maximum number of temporary nondominated solutions, i.e., 25000.

The number of final temporary nondominated solutions over that of all final solutions was up to 97.71% for all algorithms. The bSSO with parameters $(C_g, C_w) = (.50, 1.0)$, i.e., bSSO(.50, 1.0), was the best one among all algorithms. The NSGA-II only found all temporary nondominated solutions for $\mathrm{N_{sen}} = 700$, 800, and 1000.

An interesting observation can be made in that the number of obtained temporary nondominated solutions were always less than 2500 for smaller numbers of sensors, i.e., $\mathrm{N_{sen}} = 100$ and 200. The main reason for this is that the larger the sensor number is, the more temporary nondominated solutions can be found.

Tables 4 and 5 list the average values (after multiplying by 100) and the standard deviation, respectively, of the GD of all ten benchmark problems for all 500 runs for the bSSO algorithms with eight different settings and NSGA-II.

The NSGA-II is the best compared with other bSSO-based algorithms only for $\mathrm{N_{sen}} = 100$, and bSSO(.25,.45) is the best for $\mathrm{N_{sen}} = 200$, 300, 400, 500, and 600 in terms of average GD, as shown in Table 4. For sensor numbers from 700 to 1000, there is no fixed winner; however, bSSO(.25,.45) is still the second best for $\mathrm{N_{sen}} = 800$ and 900 and the third best for $\mathrm{N_{sen}} = 700$ and 1000 in terms of average GD.

TABLE 4. Average GD values (×100) obtained for all runs.

N_{sen}	NSGA-II	(.25,.45)	(.25,.70)	(.25,.95)	(.50,.75)	(.50,.95)	(.50,1.0)	(.75,.95)	(.75,1.0)
100	**0.0119**	0.0215	0.0284	0.0365	0.0181	0.0330	0.0753	0.1817	0.2072
200	0.1407	**0.0071**	0.0719	0.1263	0.0959	0.1562	0.2161	0.1649	0.1012
300	0.2639	**0.0804**	0.1154	0.1432	0.1195	0.1518	0.1845	0.3119	0.1618
400	0.4144	**0.1147**	0.1728	0.2046	0.1551	0.2724	0.1415	0.2933	0.2442
500	0.3157	**0.1012**	0.1562	0.2561	0.1193	0.2504	0.2353	0.2752	0.2175
600	0.6618	**0.1144**	0.1393	0.2939	0.2337	0.2897	0.3007	0.3312	0.1636
700	0.5880	0.1920	**0.1603**	0.3351	0.1626	0.3859	0.2631	0.2763	0.2667
800	0.6922	0.1800	0.3251	0.3355	0.3133	0.4028	0.3411	0.3573	**0.1653**
900	0.8072	0.2161	0.2242	0.2816	**0.1877**	0.4796	0.4650	0.2961	0.3430
1000	0.7977	0.2775	0.2341	0.4917	0.2866	0.3075	**0.2338**	0.3900	0.3563
average	0.4694	**0.1305**	0.1628	0.2508	0.1698	0.2729	0.2456	0.2878	0.2227

TABLE 5. Standard deviation of the GD values obtained for all runs.

N_{sen}	NSGA-II	(.25,.45)	(.25,.70)	(.25,.95)	(.50,.75)	(.50,.95)	(.50,1.0)	(.75,.95)	(.75,1.0)
100	**0.00594**	0.00651	0.00725	0.00698	0.00729	0.00750	0.01795	0.02417	0.02934
200	0.02238	**0.00355**	0.01499	0.02229	0.01804	0.02542	0.03505	0.02470	0.01869
300	0.03287	**0.01680**	0.01894	0.02238	0.02081	0.02317	0.03556	0.03460	0.02920
400	0.04411	**0.01983**	0.02409	0.02509	0.02324	0.03343	0.03010	0.03186	0.03658
500	0.03922	0.01981	0.02599	0.03176	**0.01807**	0.02917	0.03693	0.03545	0.03516
600	0.06172	**0.01859**	0.02033	0.03739	0.02946	0.03448	0.04311	0.03823	0.03152
700	0.06239	0.02732	0.02441	0.03890	**0.02539**	0.04179	0.05031	0.03366	0.04231
800	0.07034	**0.02578**	0.03867	0.03767	0.03666	0.04410	0.04870	0.03927	0.02631
900	0.07758	0.03036	0.03144	0.03666	**0.02882**	0.04851	0.05774	0.03669	0.04390
1000	0.07602	0.03723	**0.03077**	0.05613	0.03882	0.04026	0.03578	0.04071	0.06107
average	0.04926	**0.02058**	0.02369	0.03152	0.02466	0.03278	0.03912	0.03393	0.03541

Moreover, as shown in Table 4, the average of all average GD values of bSSO(.25,.45) was the best and there were only slight the differences between the average GD values of bSSO(.25,.45) and those of the best algorithms for N_{sen} = 700, 800, 900, and 1000. Similarly, bSSO(.25,.45) was better than the other algorithms in terms of the standard deviation of GD, as presented in Table 5.

Hence, bSSO(.25,.45) outperformed the other algorithms in terms of average GD and the standard deviation of GD in general, as can be seen in Tables 4 and 5. The reason for this is that all N_{sol} solutions were almost temporary nondominated solutions, as shown in Table 3, and these situations may have already occurred before $N_{gen} = 100$. Hence, items 1 and 2 in Eq. (18) are only exchanged information between temporary nondominated solutions in the generations before $N_{gen} = 100$, and only a larger c_r can result in a higher chance to explore new solution spaces in order to obtain an improved solution much closer to the Pareto front than before.

Additionally, as shown in Tables 4 and 5, the results tended to increase with N_{sen}, e.g., the average GD values were 0.0215 and 0.2775 for N_{sen} =100 and 1000 in Table 4, respectively, for bSSO(.25,.45). This situation occurs because the larger the size of an NP-hard problem is, the more difficult it is to solve it. Hence, the larger N_{sen} is, i.e., the size of the problem, the more difficult it is to find real and diverse nondominated solutions; in other words, most of the found temporary nondominated solutions are not real nondominated solutions, and these temporary nondominated solutions are farther from the Pareto front and with less diversity.

Tables 6 and 7 list the average (×100) and the standard deviation of the SP values obtained for all runs for each algorithm. The results for both the average and the standard deviation of the SP values were similar to those for the GD, i.e., bSSO(.25,.45) was still the best and NSGA-II was still the worst out of all algorithms, on average, in terms of average SP values and the standard deviation of the SP values.

Moreover, similar to Tables 4 and 5, the average and standard deviation of the SP values tended to increase with N_{sen}.

TABLE 6. Average SP values (×100) obtained for all runs.

N_{sen}	NSGA-II	(.25,.45)	(.25,.70)	(.25,.95)	(.50,.75)	(.50,.95)	(.50,1.0)	(.75,.95)	(.75,1.0)
100	**0.7597**	0.9252	1.1353	1.014	1.1633	0.9227	3.2641	9.1989	10.4266
200	9.6301	**0.2823**	4.7123	8.3362	7.057	10.4852	12.4793	9.884	5.1139
300	19.0268	**6.3252**	7.5637	8.1474	9.716	10.9566	13.6732	22.7547	10.1571
400	37.2386	**7.9015**	12.2122	12.8395	11.0625	18.3118	8.4399	18.3876	15.9766
500	25.3427	7.8569	14.7744	18.3937	**7.1903**	16.1105	17.4926	20.0585	15.8054
600	62.3549	**7.4826**	8.8729	26.8886	17.6137	25.7718	22.6079	26.6138	13.1509
700	64.7575	16.4243	**12.8357**	29.6709	13.4336	33.8712	22.8874	21.4778	19.8392
800	85.7096	14.2844	30.7586	25.2575	27.3773	33.987	26.5355	26.2583	**12.3682**
900	88.9079	19.0114	20.5262	26.879	**17.8059**	44.007	37.1461	27.6626	30.4527
1000	93.5602	28.6360	**20.2408**	50.9868	29.9936	32.4263	23.143	34.1234	33.6445
average	48.7288	**10.9130**	13.3632	20.8414	14.2413	22.6850	18.767	21.6420	16.6935

TABLE 7. Standard Deviation of the SP values (×100) obtained for all runs.

N_{sen}	NSGA-II	(.25,.45)	(.25,.70)	(.25,.95)	(.50,.75)	(.50,.95)	(.50,1.0)	(.75,.95)	(.75,1.0)
100	0.3798	0.3012	0.3070	**0.2318**	0.5384	0.2628	0.8521	1.4813	1.7277
200	1.7225	**0.1411**	1.1267	1.9455	1.5242	2.1083	2.2874	2.0157	1.1963
300	2.6713	1.4007	**1.3132**	1.3952	2.1060	1.9759	3.1790	3.0928	1.9436
400	4.5806	**1.6206**	1.9459	1.7023	1.8645	2.6663	1.8701	2.1857	2.5725
500	3.9425	1.8234	2.6607	2.6015	**1.3378**	2.0219	2.9240	2.8366	2.6969
600	7.8155	**1.2887**	1.5723	4.0023	2.5858	4.1122	3.4800	3.5083	3.1262
700	9.5949	2.5415	**2.1224**	3.9694	2.4779	4.5655	4.5630	2.9855	3.2382
800	12.3248	2.1816	4.0632	3.4127	3.8004	4.8969	4.0934	3.5494	**2.1021**
900	10.7541	**2.8609**	3.2601	4.0984	3.1801	5.2754	5.0356	4.4291	4.1244
1000	12.6321	4.7488	**3.5533**	6.6895	4.5920	5.0998	4.2720	3.9295	6.4057
average	6.6418	**1.8909**	2.1925	3.0049	2.4007	3.2985	3.2557	3.0014	2.9134

TABLE 8. Summary of the average results based on the values of c_r, c_g, and c_w.

	c_r	GD	SP		c_g	GD	SP		c_w	GD	SP
(.25,.45)	**.55**	**.13049**	**1.91298**	(.75,.95)	.75	.28779	21.64196	(.25,.95)	.70	.25045	2.84136
(.25,.70)	.30	.16277	13.36321	(.75,1.0)	.75	.22268	16.69351	(.50,1.0)	.50	.24564	18.76690
(.50,.75)	.25	.16918	14.24132	(.50,.75)	.50	.16918	14.24132	(.50,.95)	.45	.27293	22.68501
(.25,.95)	.05	.25045	2.84136	(.50,.95)	.50	.27293	22.68501	(.25,.70)	.45	.16277	13.36321
(.50,.95)	.05	.27293	22.68501	(.50,1.0)	.50	.24564	18.76690	(.75,1.0)	.25	.22268	16.69351
(.75,.95)	.05	.28779	21.64196	(.25,.45)	**.25**	**.13049**	**1.91298**	(.50,.75)	.25	.16918	14.24132
(.50,1.0)	0	.24564	18.76690	(.25,.70)	.25	.16277	13.36321	(.75,.95)	.20	.28779	21.64196
(.75,1.0)	0	.22268	16.69351	(.25,.95)	.25	.25045	2.84136	(.25,.45)	**.20**	**.13049**	**1.91298**

TABLE 9. Summary of standard deviations based on the values of c_r, c_g, and c_w.

	c_r	GD	SP		c_g	GD	SP		c_w	GD	SP
(.25,.45)	**.55**	**.02058**	**1.8909**	(.75,.95)	.75	.03393	3.0014	(.25,.95)	.70	.03152	3.0049
(.25,.70)	.30	.02369	2.1925	(.75,1.0)	.75	.03541	2.9134	(.50,1.0)	.50	.03912	3.2557
(.50,.75)	.25	.02466	2.4007	(.50,.75)	.50	.02466	2.4007	(.50,.95)	.45	.03278	3.2985
(.25,.95)	.05	.03152	3.0049	(.50,.95)	.50	.03278	3.2985	(.25,.70)	.45	.02369	2.1925
(.50,.95)	.05	.03278	3.2985	(.50,1.0)	.50	.03912	3.2557	(.75,1.0)	.25	.03541	2.9134
(.75,.95)	.05	.03393	3.0014	(.25,.45)	**.25**	**.02058**	**1.8909**	(.50,.75)	.25	.02466	2.4007
(.50,1.0)	0	.03912	3.2557	(.25,.70)	.25	.02369	2.1925	(.75,.95)	.20	.03393	3.0014
(.75,1.0)	0	.03541	2.9134	(.25,.95)	.25	.03152	3.0049	(.25,.45)	**.20**	**.02058**	**1.8909**

Hence, again, the larger the N_{sen}, the more difficult it is to find real nondominated solutions, solutions close to the real nondominated solutions, or solutions with a high distribution.

Tables 8 and 9 summarize the average results and the standard deviation for both GD and SP values based on the values of c_r, c_g, and c_w, respectively. An interesting result is that the trend of the best results in both Tables 8 and 9 are very similar, e.g., the algorithm with the best average or standard deviation of GD also has the best average or standard deviation values for SP.

The results for the c_r values listed in Tables 8 and 9 further confirm the observation discussed above in that a higher c_r (> 0.5) results in better GD and SP values. Besides, lower values for both c_g and c_w result in better GD and SP values, which again confirm that higher c_r values result in better GD and SP because $c_g + c_w + c_r = 1$.

Hence, in general, the proposed bSSO(.25,.45) algorithm was closer to the simulated Pareto front with a better diversity than the other proposed bSSO algorithms with different parameter settings and NSGA-II, as shown in Tables 3–9 for each test.

VI. CONCLUSION

A nascent bi-objective WSN problem focusing on energy consumption and service quality in terms of routing reliability has been propounded in this study to find non-dominated routings for the purpose of ameliorating the service quality issues of WSNs. Both the objectives considered are some of the major concerns for users of the service provided by WSNs. Thus, we proposed a new bi-objective simplified swarm optimization (bSSO) algorithm in response to this bi-objective problem.

The bSSO algorithm is based on a novel update mechanism. It integrates crowding distance, cycle removal, and a new stepwise update function. The performance and applicability of the proposed bSSO algorithm was tested on ten benchmarks for sensor networks ranging from small to large, namely for $N_{\text{sen}} = 100, 200, \ldots, 1000$ in a WSN with eight different parameter settings. A comparison with the results obtained with the NSGA-II was also drawn. From our experimental results, we concluded that the bSSO algorithm with $c_g = 0.25$ and $c_w = 0.45$ outperformed the others in terms of the average and standard deviation values of both GD (convergence metrics) and SP (diversity metrics) in this problem.

The limitation of the proposed model is that it only considers from the signal from the source node to the sink node and all signals in the routing paths obey the conservation law [33]. In the future study, we are going to generalize the proposed model to consider these signals from many resource nodes to many sink nodes without 100% satisfying the conservation law to meet the practical requirement [36]–[38].

Also, we plan to extend our current research to focus on the convergence rate for providing performance comparison for the goal of attaining a better result than that of a fixed number of run trials and to also provide the time-complexity that was conveyed in [31]. In addition, we will focus more on obtaining general results and findings for a certain application. As well as, the proposed method in this study aims to be verified and validated through practical applications such as Internet of Things (IoT), advanced driver assistance system (ADAS), and drone.

ACKNOWLEDGMENT

The authors would like to thank the anonymous editor and the referees for their constructive comments and recommendations, which significantly improved this article.

REFERENCES

[1] N. Dey, A. S. Ashour, F. Shi, S. J. Fong, and R. S. Sherratt, "Developing residential wireless sensor networks for ECG healthcare monitoring," *IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 442–449, Nov. 2017.

[2] A. Cenedese, M. Luvisotto, and G. Michieletto, "Distributed clustering strategies in industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 228–237, Feb. 2017.

[3] W.-C. Yeh and J.-S. Lin, "New parallel swarm algorithm for smart sensor systems redundancy allocation problems in the Internet of Things," *J. Supercomput.*, vol. 74, no. 9, pp. 4358–4384, Sep. 2018.

[4] M. Wang, W.-C. Yeh, T.-C. Chu, X. Zhang, C.-L. Huang, and J. Yang, "Solving multi-objective fuzzy optimization in wireless smart sensor networks under uncertainty using a hybrid of IFR and SSO algorithm," *Energies*, vol. 11, no. 9, p. 2385, Sep. 2018, doi: 10.3390/en11092385.

[5] B. C. Csaji, Z. Kemeny, G. Pedone, A. Kuti, and J. Vancza, "Wireless multi-sensor networks for smart cities: A prototype system with statistical data analysis," *IEEE Sensors J.*, vol. 17, no. 23, pp. 7667–7676, Dec. 2017.

[6] M. Gao, P. Wang, Y. Cao, R. Chen, and D. Cai, "Design and verification of a rail-borne energy harvester for powering wireless sensor networks in the railway industry," *IEEE Trans. Intell. Transport. Syst.*, vol. 18, no. 6, pp. 1596–1609, Jun. 2017.

[7] K. Colins, L. Li, and Y. Liu, "Analysis of a statistical relationship between dose and error tallies in semiconductor digital integrated circuits for application to radiation monitoring over a wireless sensor network," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 5, pp. 1151–1158, May 2017.

[8] T. Mekonnen, P. Porambage, E. Harjula, and M. Ylianttila, "Energy consumption analysis of high quality multi-tier wireless multimedia sensor network," *IEEE Access*, vol. 5, pp. 15848–15858, 2017.

[9] S. D. Trapasiya and H. B. Soni, "Energy efficient policy selection in wireless sensor network using cross layer approach," *IET Wireless Sensor Syst.*, vol. 7, no. 6, pp. 191–197, Dec. 2017.

[10] P. T. A. Quang and D.-S. Kim, "Enhancing real-time delivery of gradient routing for industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 61–68, Feb. 2012.

[11] D. Setiawan, A. A. Aziz, D. I. Kim, and K. W. Choi, "Experiment, modeling, and analysis of wireless-powered sensor network for energy neutral power management," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3381–3392, Dec. 2018.

[12] J. Liu, K. Xiong, P. Fan, and Z. Zhong, "Resource allocation in wireless powered sensor networks with circuit energy consumption constraints," *IEEE Access*, vol. 5, pp. 22775–22782, 2017.

[13] P. Chanak, I. Banerjee, and H. Rahaman, "Load management scheme for energy holes reduction in wireless sensor networks," *Comput. Electr. Eng.*, vol. 48, pp. 343–357, Nov. 2015.

[14] M. Collotta, G. Pau, and D. G. Costa, "A fuzzy-based approach for energy-efficient Wi-Fi communications in dense wireless multimedia sensor networks," *Comput. Netw.*, vol. 134, pp. 127–139, Apr. 2018.

[15] P. Kumar and A. Chaturvedi, "Fuzzy-interval based probabilistic query generation models and fusion strategy for energy efficient wireless sensor networks," *Comput. Commun.*, vol. 117, pp. 46–57, Feb. 2018.

[16] M. Akram and T. H. Cho, "Energy efficient fuzzy adaptive selection of verification nodes in wireless sensor networks," *Ad Hoc Netw.*, vol. 47, pp. 16–25, Sep. 2016.

[17] A. Khatri, S. Kumar, O. Kaiwartya, N. Aslam, N. Meena, and A. H. Abdullah, "Towards green computing in wireless sensor networks: Controlled mobility-aided balanced tree approach," *Int. J. Commun. Syst.*, vol. 31, no. 7, p. e3463, May 2018, doi: 10.1002/dac.3463.

[18] L. Farhan, R. Kharel, O. Kaiwartya, M. Hammoudeh, and B. Adebisi, "Towards green computing for Internet of Things: Energy oriented path and message scheduling approach," *Sustain. Cities Soc.*, vol. 38, pp. 195–204, Apr. 2018.

[19] M. Sajwan, D. Gosain, and A. K. Sharma, "Hybrid energy-efficient multi-path routing for wireless sensor networks," *Comput. Electr. Eng.*, vol. 67, pp. 96–113, Apr. 2018.

[20] H. Huang, J. Zhang, X. Zhang, B. Yi, Q. Fan, and F. Li, "EMGR: Energy-efficient multicast geographic routing in wireless sensor networks," *Comput. Netw.*, vol. 129, pp. 51–63, Dec. 2017.

[21] A. E. Fawzy, M. Shokair, and W. Saad, "Balanced and energy-efficient multi-hop techniques for routing in wireless sensor networks," *IET Netw.*, vol. 7, no. 1, pp. 33–43, Jan. 2018.

[22] X. Wang, J. Ma, S. Wang, and D. Bi, "Distributed energy optimization for target tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 1, pp. 73–86, Jan. 2010.

[23] B. Li, W. Wang, Q. Yin, R. Yang, Y. Li, and C. Wang, "A new cooperative transmission metric in wireless sensor networks to minimize energy consumption per unit transmit distance," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 626–629, May 2012.

[24] W. C. Yeh, "A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 200–9192, 2009.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[26] J. Cecílio and P. Furtado, *Wireless Sensors in Heterogeneous Networked Systems*. Cham, Switzerland: Springer, 2014, ch. 2, pp. 5–25.
[27] W.-C. Yeh, "Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 1, pp. 250–261, Jan. 2012.
[28] W.-C. Yeh, "New parameter-free simplified swarm optimization for artificial neural network training and its application in the prediction of time series," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 661–665, Apr. 2013.
[29] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," *Evol. Comput.*, vol. 8, no. 2, pp. 21–29, 1999.
[30] J. R. Schott. (1995). *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. [Online]. Available: http://hdl.handle.net/1721.1/11582
[31] H. Yetgin, K. T. K. Cheung, and L. Hanzo, "Multi-objective routing optimization using evolutionary algorithms," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Paris, France, Apr. 2012, pp. 3030–3034.
[32] W.-C. Yeh, C. Bae, and C.-L. Huang, "A new cut-based algorithm for the multi-state flow network reliability problem," *Rel. Eng. Syst. Saf.*, vol. 136, pp. 1–7, Apr. 2015.
[33] W.-C. Yeh, Y.-C. Lin, and Y. Y. Chung, "Performance analysis of cellular automata Monte Carlo simulation for estimating network reliability," *Expert Syst. Appl.*, vol. 37, no. 5, pp. 3537–3544, May 2010.
[34] W.-C. Yeh, "A simple universal generating function method to search for all minimal paths in networks," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 6, pp. 1247–1254, Nov. 2009.
[35] W.-C. Yeh, "A simple universal generating function method for estimating the reliability of general multi-state node networks," *IIE Trans.*, vol. 41, no. 1, pp. 3–11, Nov. 2008.
[36] W.-C. Yeh, "Evaluating the reliability of a novel deterioration-effect multi-state flow network," *Inf. Sci.*, vol. 243, pp. 75–85, Sep. 2013.
[37] W.-C. Yeh, "Evaluation of the one-to-all-target-subsets reliability of a novel deterioration-effect acyclic multi-state information network," *Rel. Eng. Syst. Saf.*, vol. 166, pp. 132–137, Oct. 2017.
[38] W.-C. Yeh, "Methodology for the reliability evaluation of the novel learning-effect multi-state flow network," *IISE Trans.*, vol. 49, no. 11, pp. 1078–1085, Nov. 2017.

WEI-CHANG YEH (Senior Member, IEEE) received the M.S. and Ph.D. degrees from the Department of Industrial Engineering, University of Texas at Arlington. He is currently a Distinguished Professor with the Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Taiwan. Most of his research is focused around algorithms, including exact solution methods and soft computing. He has published more than 250 research articles in highly ranked journals and conference papers, and has been awarded the Outstanding Research Award twice, the Distinguished Scholars Research Project once, and an Overseas Research Fellowship twice by the Ministry of Science and Technology in Taiwan. He has been invited to serve as an Associate Editor of the two top reliability related journals, namely, the IEEE Transactions on Reliability and *Reliability Engineering and System Safety*. He proposed a novel soft computing algorithm called the simplified swarm optimization (SSO) and demonstrated the simplicity, effectiveness, and efficiency of his SSO for solving NP-hard problems. He has been granted 50 patents and earns an International Fellow, the Guoguang Invention Medal as well as the titles of Outstanding Inventor of Taiwan and Doctor of Erudition, by the Chinese Innovation and Invention Society.

YUNZHI JIANG was born in 1982. He received the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 2012. Since 2018, he has been an Associate Professor with the School of Mathematics and Systems Science, Guangdong Polytechnic Normal University, Guangzhou. He is the author of more than 20 articles. His research interests include theoretical foundation and application of evolutionary algorithms, image segmentation, text information detection, and segmentation.

CHIA-LING HUANG received the Ph.D. degree in industrial engineering and management from National Chiao Tung University, Hsinchu, Taiwan. She is currently an Associate Professor with the Department of International Logistics and Transportation Management, Kainan University. Her research interests include reliability, network analysis, and statistical application.

NEAL N. XIONG received the Ph.D. degree in software engineering from Wuhan University, and the Ph.D. degree in dependable networks from the Japan Advanced Institute of Science and Technology. He worked with the Wentworth Technology Institution, Georgia State University, for many years. He is currently an Associate Professor with the Department of Mathematics and Computer Science, Northeastern State University. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

CHENG-FENG HU received the B.S. degree from National Tsing Hua University, Hsinchu City, Taiwan, and the Ph.D. degree from North Carolina State University, Raleigh, NC, USA, in 1993 and 1997, respectively. Her research interests include fuzzy optimization and decision making and financial engineering.

YUAN-HUI YEH is currently pursuing the degree with the Business School, Law School, University of New South Wales, Sydney, NSW, Australia. Her research interests include the network applications, performance evaluations, and analysis.