RAIRO-Oper. Res. 50 (2016) 825–843
DOI: 10.1051/ro/2016061

RAIRO Operations Research
www.rairo-ro.org

LOGIC GATE-BASED EVOLUTIONARY ALGORITHM FOR THE MULTIDIMENSIONAL KNAPSACK PROBLEM-WIRELESS SENSOR NETWORK APPLICATION

Ayet Allah Ferjani[1], Noureddine Liouane[1] and Pierre Borne[2]

Abstract. Evolutionary algorithms (EAs) are predominantly employed to find solutions for continuous optimization problems. As EAs are initially presented for continuous spaces, research on extending EAs to find solutions for binary spaces is in growing concern. In this paper, a logic gate-based evolutionary algorithm (LGEA) for solving some combinatorial optimization problems (COPs) is introduced. The proposed LGEA has the following features. First, it employs the logic operation to generate the trial population. Thereby, LGEA replaces common space transformation rules and classic recombination and mutation methods. Second, it is based on exploiting a variety of logic gates to search for the best solution. The variety among these logic tools will naturally lead to promote diversity in the population and improve global search abilities. The LGEA presents thus a new technique to combine the logic gates into the procedure of generating offspring in an evolutionary context. To judge the performance of the algorithm, we have solved the NP-hard multidimensional knapsack problem as well as a well-known engineering optimization problem, task allocation for wireless sensor network. Experimental results show that the proposed LGEA is promising.

Mathematics Subject Classification. 90C27.

Received May 11, 2015. Accepted August 30, 2016.

1. Introduction

Evolutionary algorithms (EAs) such as genetic algorithm (GA) [9], particle swarm optimization (PSO) [7], differential evolution algorithm (DE) [21], artificial bee colony (ABC) [11] and ant colony optimization (ACO) [6] have been successfully applied to various complex optimization problems. The algorithm is based on mechanisms found in nature. To search for the optimal solution, a population of solutions evolves by constantly simulating selection and recombination/mutation operators, creating a new set of individuals, everyone competing with others according to their fitness [20].

The original EAs are simple and efficient, but they are predominantly presented for continuous space. As many optimization problems are defined in the binary space, research on extending EAs to solve binary combinatorial optimization problems (COPs) has become in growing concern in recent years.

Keywords. Evolutionary algorithm, logic gate, multidimensional knapsack problem, task allocation, wireless sensor network.

[1] Electrical Engineering Department, National Engineering School of Monastir, Monastir, Tunisia.
ayetferjani@gmail.com; Noureddine.Liouane@enim.rnu.tn

[2] École Centrale de Lille, BP 48, 59651 Villeneuve d'Ascq cedex, France. pierre.borne@ec-lille.fr

Article published by EDP Sciences

Initially, Kennedy and Eberhart [12] proposed the binary PSO (BPSO) to extend the PSO for binary problems. The algorithm is characterized by a space transformation technique. The position is denoted as a real vector, and thus a sigmoid function is employed to transform the position into its corresponding solution. Several improved versions of BPSO have been developed. For instance, Chuang *et al.* [5] proposed a BPSO model that uses chaotic maps for parameter adaptation. Bansal and Deep [2] introduced a modified version of BPSO in order to increase the exploration ability. Recently, chih *et al.* [4] proposed a BPSO with time-varying acceleration coefficients for the multidimensional knapsack problem. These BPSO algorithms have presented promising performance on some benchmark problems. However, the binary coding scheme can be applied to limited types of COPs. The GA has been widely considered to solve COPs [14,18]. However, the drawback of genetic methods is that it may face up to the slow convergence rate and easily get stuck in local optimum due to its limited exploration when solving NP-hard problems.

There have been only few attempts for presenting "pure" binary mechanisms based on the evolutionary concepts. Particularly, the logic operation has been shown to be effective in implementation of novel EAs for binary problems. For instance, Kiran *et al.* [13] proposed a XOR-based artificial bee colony algorithm for binary optimization (binABC). Marandi *et al.* [15] introduced the Boolean PSO to the Design of a Dual-Band Dual-Polarized Planar Antenna. All these algorithms adopt the XOR tool to optimize binary problems. However, to achieve better performance, more logic tools should be explored.

As a result, this paper proposes a logic gate-based evolutionary algorithm (LGEA) to extend the application of EAs for solving some optimization problems in binary space. LGEA features the following characteristics. First, it employs the logic operation to generate the trial population. Thereby, LGEA replaces common space transformation rules and classic recombination and mutation methods. Second, it is based on exploiting a variety of logic gates to search for the best solution. The logic gate implements either of the OR, AND, NOR, NAND, XOR or XNOR gate via trial and error. The variety among these logic tools will naturally lead to promote diversity in the population and improve global search abilities. The LGEA presents thus a new technique to combine the logic gates into the procedure of generating offspring in an evolutionary context. The algorithm has been tested by solving the multidimensional knapsack problem (MKP) [8] as well as an engineering optimization problem, task allocation in wireless sensor network [24]. In the experiments, the algorithm is compared with the existing binary evolutionary approaches. Experimental results show that the LGEA is promising.

The rest of this paper is organized as follows. Section 2 presents the proposed LGEA. The performance of the LGEA algorithm on the MKP is evaluated in Section 3 and compared against the results existing in literature. Section 4 is devoted to task allocation for wireless sensor network. The conclusions are finally summarized in Section 5.

2. Logic gate-based evolutionary algorithm

In this section, the logic gate-based evolutionary algorithm (LGEA) is described. To improve the ability of other EAs to solve binary problems, LGEA applies the logic gate operation to perform the binary perturbation, where a variety of logic operators are exploited. In comparison with BPSO and GA, the LGEA is characterized by a small number of control parameters. In the proposed method, the logic gate mechanism, called also "logic mutation" is defined by a Boolean function that performs a logical mathematical relationship between two binary candidate solutions to produce a new binary solution. At generation zero, the initial population is uniformly and randomly generated within the search space. Thereafter, LGEA evolves the population towards the promising solutions through repeated cycles of logic gate mechanism, crossover and selection. The main procedure of LGEA is explained as follows.

2.1. Logic gate mechanism

After initialization, the algorithm performs the logic gate mechanism to generate a mutant vector V_i^G related to a target vector X_i^G in the current population. The proposed algorithm can be designed by a variety of logic gate strategies inspired from the Differential Evolution (DE) algorithm [21]. The DE algorithm employs the

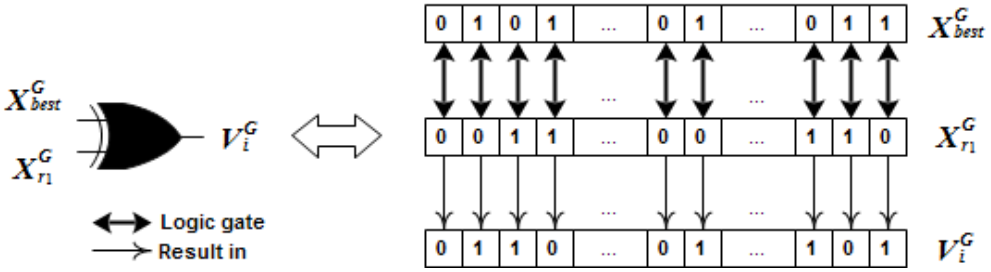

FIGURE 1. Logic gate mechanism with XOR-best2rand.

differential mutation while the LGEA exploits the logic mutation. Moreover, DE works with real parameters while LGEA performs with binary variables. Let NP be the number of individuals in the population, then for each target vector X_i^G at generation G, its corresponding mutant vector $V_i = (v_{i,1}, v_{i,2}, v_{i,3}, \ldots, v_{i,D})$, $i = 1, \ldots NP$ can be created via certain logic gate strategy as follows.

(1) "best2rand"

$$V_i^G = X_{\text{best}}^G \oplus X_{r_1}^G \tag{2.1}$$

(2) "rand2rand"

$$V_i^G = X_{r_1}^G \oplus X_{r_2}^G \tag{2.2}$$

(2) "old2rand"

$$V_i^G = X_i^G \oplus X_{r_1}^G \tag{2.3}$$

The indices r_1, r_2 are randomly chosen from the interval $[1, NP]$ and are also different from the index i. X_{best}^G is the best individual vector with the best fitness value at generation G. $\oplus$ is a logic gate, which implements either of the following operators.

"XOR" (2.4)

"AND" (2.5)

"OR" (2.6)

"XNOR" (2.7)

"NAND" (2.8)

"NOR" (2.9)

To undergo the logic mutation, the logic operator is selected by trial and error in order to find the optimized solution. For instance, in (2.10), the mutant vector V_i^G is generated by XOR-best2rand strategy as defined in (2.1) and (2.4). Figure 1 shows logic gate mechanism with XOR-best2rand.

$$V_i^G = XOR\left(X_{\text{best}}^G, X_{r_1}^G\right) \tag{2.10}$$

2.2. Crossover mechanism

To increase the diversity of population, each of the target vector X_i^G and its associated mutant vector V_i^G undergo crossover mechanism to produce a trial vector $U_i^G = (u_{i,1}^G, u_{i,2}^G, u_{i,3}^G, \ldots, u_{i,D}^G)$. In basic versions, EAs employ the one-point crossover [17] defined as follows. Given a crossover rate CR, the bits of trial vector U_i^G are inherited from the associated mutant vector V_i^G, beginning from a randomly determined index till the first time $rand() > CR$; $rand()$ is a random number between $[0, 1]$. The remaining bits of the trial vector U_i^G are copied from the associated target vector X_i^G. Figure 2 shows the crossover mechanism.

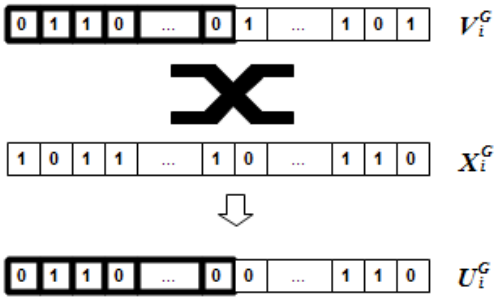

FIGURE 2. Crossover mechanism.

2.3. Selection mechanism

The fitness values of the newly generated trial vectors are evaluated. Then, a selection mechanism is processed. The fitness value of each trial vector $f(U_i^G)$ is compared to its related target vector $f(X_i^G)$ in the present population. If the trial vector has better or equal fitness value than the associated target vector, this gets replaced by the trial vector in the next generation. Otherwise, the old vector is retained. The selection mechanism can be expressed as follows.

$$X_i^{G+1} = \begin{cases} U_i^G, \text{if } f(U_i^G) \geq f(X_i^G) \\ X_i^G, \text{otherwise} \end{cases} \tag{2.11}$$

The computational procedure of the LGEA is summarized below.

Step 1. Initialize the parameters and the population with random binary solutions.
Step 2. Evaluate the fitness value for each individual.
Step 3. Generate the mutant vector V_i^G for each target vector X_i^G *via* one of the strategies (2.1)−(2.3) using one of the logic gates (2.4)−(2.9).
Step 4. Generate the trial vector U_i^G for each target vector X_i^G by crossover operator as described in Figure 2.
Step 5. Evaluate fitness value of the trial population.
Step 6. Select individual between the target and trial vector as defined in (2.11).
Step 7. If the stopping criterion is not reached go to Step 3, else return the individual with the best fitness as the solution.

3. PERFORMANCE ON THE MULTIDIMENSIONAL KNAPSACK PROBLEM

In this section, the multidimensional knapsack problem (MKP) is used to verify the effectiveness of the LGEA and especially to show the potential of the logic gate mechanism for binary optimization. Our main motivation for choosing the MKP as a test problem is due to the fact that it can be viewed as a general framework for any type of binary problems with positive coefficients. Yet, the problem itself is difficult to solve (NP-hard). Moreover, due to its practical importance, the MKP has been subject to many investigations in different domains [16]. In particular, there are some publications in the field of evolutionary computation related the MKP [4].

3.1. Problem description

Generally, the knapsack problem can be explained as the following idea. Suppose a hitch-hiker wants to fill a knapsack. There are items available to select, but the capacity of the knapsack is limited. The hitch-hiker tries to maximize the total profit of the items in the knapsack while not overloading it.

The multidimensional knapsack problem (MKP) is a generalization of the standard knapsack problem. The MKP can formally be described as follows. Given a set of n items with profits c_j and a set of m knapsacks

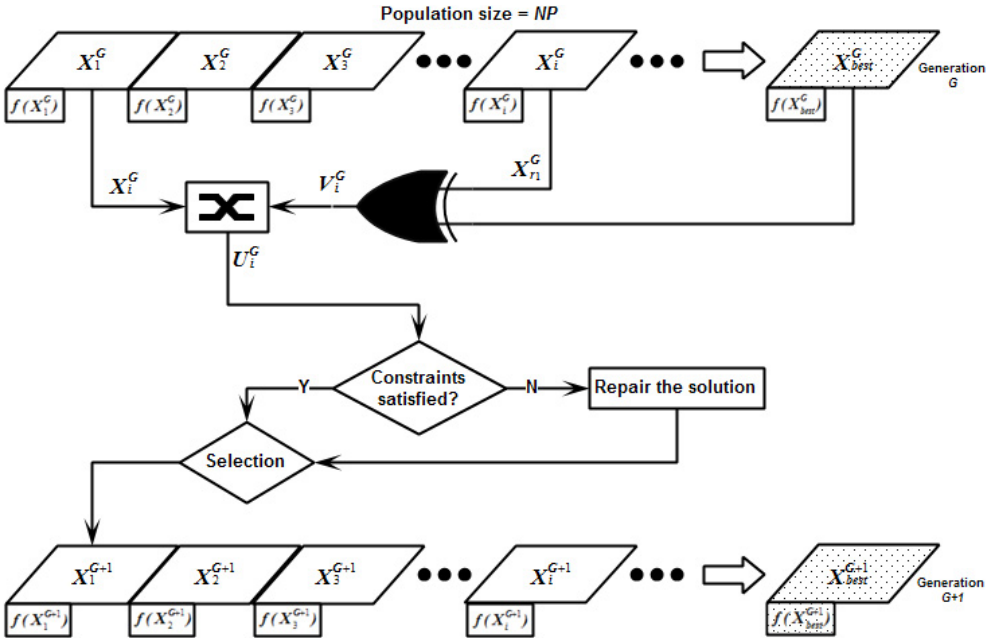

FIGURE 3. Flowchart of the LGEA operation with XOR-best2rand for solving the MKP.

with capacities $b_i (k = 1, \ldots, m)$. Each item i requires a_{ij} units of weight in each knapsack i. The variable x_j denotes the decision whether or not the item j is selected for given knapsacks ($x_j = 1$ if item j is selected in the knapsack; $x_j = 0$ otherwise). The goal is to find the subset of items that yield the maximum profit (3.1) without exceeding the knapsack capacities – knapsack constraint (3.2).

$$\text{Maximize } f(X) = C.X \tag{3.1}$$

$$\text{Subject to } A.X \leq B \tag{3.2}$$

$$X \in \{0,1\}^n$$

$$\text{with } C \in N^{*n},\ A \in N^{m*n},\ B \in N^m$$

3.2. LGEA for the MKP

The LGEA is used in this section to solve the MKP. Figure 3 shows a simplified flowchart of the LGEA operation with XOR-best2rand for solving the MKP. The step by step implementation is described as follows.

Step 1. Initialize the parameters and the binary individuals. The binary solution $x^0_{i,j}$ is randomly initialized by (3.3).

$$x^0_{i,j} = \begin{cases} 1, & \text{if} \quad rand() < p_0 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where $rand()$ is a random number uniformly distributed in the range $[0, 1]$; p_0 is the desired percentage of selected items. However, the above randomly created solution X_i^G may not respect the knapsack constraint. If the knapsack capacity violates the knapsack constraint, the bits of infeasible solution are repaired by converting from one to zero until the infeasible solution is transformed to the feasible one.

Step 2. Evaluate the fitness value of each initialized solution. Compute the individual with the best fitness value.

Step 3. Generate the mutant vector V_i^G *via* best2rand strategy using one of the logic gates (2.4)−(2.9). For example, we employ XOR-best2rand as defined in (2.10):

$$V_i^G = XOR(X_{\text{best}}^G, X_{r_1}^G)$$

where X_{best}^G is the individual with the best fitness value at generation G and X_{r1}^G is a randomly chosen solution from the current population.

Step 4. Generate the trial vector U_i^G by crossover operator as described in Figure 2.

Step 5. For the newly generated trial solutions, evaluate the knapsack constraints according to (3.2). If the knapsack constraints are violated, the infeasible solution would be repaired as the mentioned repair process.

Step 6. Evaluate fitness value of the newly generated trial population according to (3.1).

Step 7. Select individual between target population and trial population as defined in (2.11).

Step 8. If the stopping criterion is not reached go to Step 3, else return the individual with the best fitness value as the ultimate solution for the MKP.

3.3. Experimental results and comparison studies

In this section, we present the numerical experimental results of the MKP based on the proposed LGEA. The performances of LGEA are evaluated on several experiments using benchmarks of the operations research library (OR-library) [3]. The experiments are performed on Windows 7/ Pentium ® Dual- Core CPU T4500 2.3 GHz machine and implemented in Matlab. In this study, we compare the LGEA algorithm with the other existing evolutionary-based approaches.

In the experiments in this section, according to the analyses in Section 2, the configurations of switching the logic gate ($\oplus$) between different operators (XOR, OR, AND, XNOR, NOR, NAND) by trial and error is used. To improve the convergence, we select best2rand strategy. The crossover rate (CR) is taken in the range of 0.005-0.1 to preserve diversity in the population. The maximum number of generations ($G_{\max}$) was 20 000. The simulation experiments were based on a total of 30 runs.

In the first experiment, we compare the LGEA algorithm with BPSO-based algorithms, namely: CBPSO1 [5], MBPSO [2], BPSOTVAC [4] and CBPSOTVAC [4] on SENTO, WEING, WEISH, HP and PB knapsack benchmarks. We employ for that the five performance criteria proposed in [4]: the success ratio (SR), the mean absolute deviation (MAD), the mean absolute percentage of error (MAPE), the minimal error (LE) and the standard deviation of the solution (SD). The comparative results are presented in Tables 1, 2 and 3. The results clearly show that LGEA outperforms the binary PSO algorithms on the different performance criteria with a significant difference. Moreover, while it is difficult for the BPSO algorithms to obtain the optimal solutions, LGEA manages to find the optimal solutions in all instances.

Another experiment is performed to compare LGEA with three other binary population-based algorithms, including two genetic algorithms (OGA [14] and GADS [18]) and a fish swarm algorithm (IbAFSA) [1] on Peterson knapsack benchmarks. The performance criteria are listed in Table 4, where SR denotes the success rate, and AE, LE and SDE represent the average, least and standard deviation of errors, respectively. According to LE, compared with the GAs, LGEA is able to find the optimal solutions in all instances. According to AE, LGEA performs better than the other methods. These results prove that LGEA has better consistency to achieve relatively good solutions in different runs.

In the last experiment, we solve problems with higher dimension to demonstrate the potential of the LGEA. We have considered Glover and Kochenberger knapsack benchmarks whose size is ranging from 100 to 2500 items and the number of constraints is between 15 and 100. The LGEA is compared with a binary fruit optimization

TABLE 1. Comparison between LGEA and MBPSO, CBPSO1, BPSOTVAC and CBPSOTVAC algorithms on SENTO and WEING testing problems.

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Sento1	30	60	MBPSO	0.16	44.81	0.0058	229	43.23
			CBPSO1	0	198.29	0.026	302	49.92
			BPSOTVAC	0.57	8.74	0.0011	34	11.52
			CBPSOTVAC	0.39	136.28	0.021	3146	357.78
			LGEA	**1**	**0**	**0**	**0**	**0**
Sento2	30	60	MBPSO	0.03	24.85	0.0029	81	18.8
			CBPSO1	0	103.32	0.012	150	25.78
			BPSOTVAC	0.27	9.42	0.001	38	7.04
			CBPSOTVAC	0.2	53.53	0.0063	633	101.03
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing1	2	28	MBPSO	0.82	110.79	0.0008	801	250.43
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.92	51.25	0.0004	1961	281.98
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing2	2	28	MBPSO	0.65	117.45	0.0009	1700	314.08
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.88	123.19	0.0009	3341	545.5
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing3	2	28	MBPSO	0.11	1053.2	0.0112	3500	876.78
			CBPSO1	1	0	0	0	0
			BPSOTVAC	0.92	6.42	0.00007	160	25.53
			CBPSOTVAC	0.75	173.07	0.0019	3789	672.42
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing4	2	28	MBPSO	0.76	570.6	0.0049	4001	1270.8
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.97	42.83	0.0004	3774	378.58
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing5	2	28	MBPSO	0.52	1629.21	0.017	4778	1923.5
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.94	85.62	0.0009	4728	572.82
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing6	2	28	MBPSO	0.36	310.2	0.0023	1340	322.4
			CBPSO1	1	0	0	0	0
			BPSOTVAC	0.97	11.7	0.00009	390	66.86
			CBPSOTVAC	0.87	91.71	0.0007	2460	343.45
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing7	2	105	MBPSO	0.02	660.86	0.0006	6111	1130.6
			CBPSO1	0	32 690.6	0.0308	42 425	5002
			BPSOTVAC	0	281.23	0.00026	2069	383.74
			CBPSOTVAC	0	11 272.9	0.011	154 486	30 020
			LGEA	**1**	**0**	**0**	**0**	**0**
Weing8	2	105	MBPSO	0.03	5824.7	0.0095	44731	4704.3
			CBPSO1	0	118 166	0.234	160 402	15 988
			BPSOTVAC	0.35	1872.44	0.0030	6463	2000.9
			CBPSOTVAC	0.20	27 128.4	13.7	623 862	75 169
			LGEA	**1**	**0**	**0**	**0**	**0**

TABLE 2. Comparison between LGEA and MBPSO, CBPSO1, BPSOTVAC and CBPSOTVAC algorithms on WEISH testing problems.

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Weish1	5	30	MBPSO	0.82	10.9	0.0024	114	26.34
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.94	5.45	0.0012	248	32.81
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish2	5	30	MBPSO	0.55	8.39	0.0018	123	18.01
			CBPSO1	0.79	1.05	0.00023	5	2.04
			BPSOTVAC	0.64	1.8	0.00040	5	2.41
			CBPSOTVAC	0.66	4.12	0.0009	231	23.12
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish3	5	30	MBPSO	0.63	20.54	0.0051	141	34.98
			CBPSO1	1	0	0	0	0
			BPSOTVAC	0.99	0.63	0.00015	63	6.3
			CBPSOTVAC	0.95	9.21	0.0024	394	52.69
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish4	5	30	MBPSO	0.96	1.76	0.0004	56	8.99
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.99	8.59	0.0023	859	85.9
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish5	5	30	MBPSO	0.99	0.54	0.00012	54	5.4
			CBPSO1	1	0	0	0	0
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.98	8.11	0.0021	742	74.45
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish6	5	40	MBPSO	0.32	15.36	0.0028	56	14.39
			CBPSO1	0.65	5.47	0.00098	34	7.92
			BPSOTVAC	0.59	6.68	0.00121	18	8.19
			CBPSOTVAC	0.53	23.21	0.0044	518	79.28
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish7	5	40	MBPSO	0.64	10.2	0.0018	122	18.92
			CBPSO1	0.83	3.45	0.0006	25	7.79
			BPSOTVAC	0.96	0.7	0.00013	18	3.45
			CBPSOTVAC	0.78	19.17	0.0036	511	71.95
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish8	5	40	MBPSO	0.44	7.24	0.0013	72	13.07
			CBPSO1	0.64	0.72	0.00013	2	0.96
			BPSOTVAC	0.79	0.42	0.00008	2	0.82
			CBPSOTVAC	0.68	8.84	0.0016	418	42.81
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish9	5	40	MBPSO	0.78	10.61	0.0021	200	25.65
			CBPSO1	0.96	1.36	0.0003	34	6.69
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.85	13.01	0.0027	641	65.7
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish10	5	50	MBPSO	0.56	10.84	0.0017	83	22.17
			CBPSO1	0.07	42.57	0.0068	141	33.65
			BPSOTVAC	0.91	1.43	0.0002	68	9.56
			CBPSOTVAC	0.67	57.16	0.0102	1394	188.63
			LGEA	**1**	**0**	**0**	**0**	**0**

TABLE 2. Continued.

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Weish11	5	50	MBPSO	0.4	29.48	0.0053	167	43.95
			CBPSO1	0.05	79.9	0.0144	191	47.06
			BPSOTVAC	0.88	7.42	0.0013	113	25.72
			CBPSOTVAC	0.62	110.85	0.028	2245	403.03
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish12	5	50	MBPSO	0.65	16.35	0.0026	226	35.68
			CBPSO1	0.09	57.3	0.0092	191	42.35
			BPSOTVAC	0.89	0.29	0.00005	19	1.91
			CBPSOTVAC	0.71	107.5	0.020	1497	304.43
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish13	5	50	MBPSO	0.87	8.47	0.0014	155	25.19
			CBPSO1	0.15	59.33	0.0098	159	40.97
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.85	38.62	0.0075	1725	180.04
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish14	5	60	MBPSO	0.66	16.09	0.0023	100	25.95
			CBPSO	0	210.47	0.031	347	66.79
			BPSOTVAC	0.98	0.62	0.00089	31	4.36
			CBPSOTVAC	0.79	116.23	0.021	2127	364.66
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish15	5	60	MBPSO	0.72	10.55	0.0014	70	18.64
			CBPSO1	0	193.51	0.0266	359	59.99
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.8	161.45	0.030	2978	554.35
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish16	5	60	MBPSO	0.44	7.66	0.0011	87	17.49
			CBPSO1	0	139.24	0.0195	259	47.07
			BPSOTVAC	0.54	1.16	0.00016	8	1.71
			CBPSOTVAC	0.43	143.29	0.023	1931	367.29
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish17	5	60	MBPSO	0.56	5.76	0.0007	41	7.38
			CBPSO1	0	91.8	0.0107	170	32.79
			BPSOTVAC	1	0	0	0	0
			CBPSOTVAC	0.72	85.29	0.011	1035	227.16
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish18	5	70	MBPSO	0.38	14.65	0.0015	94	18.4
			CBPSO1	0	259.34	0.0278	367	54.01
			BPSOTVAC	0.75	2.79	0.00029	15	5.25
			CBPSOTVAC	0.53	99.14	0.011	1595	275.53
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish19	5	70	MBPSO	0.55	20.83	0.0027	149	33.67
			CBPSO1	0	429.39	0.059	615	83.65
			BPSOTVAC	0.65	4.9	0.0006	35	7.13
			CBPSOTVAC	0.62	169.45	0.028	3060	489.37
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish20	5	70	MBPSO	0.53	10.81	0.0011	69	15.99
			CBPSO1	0	336.41	0.037	528	89.41
			BPSOTVAC	0.78	3.78	0.0004	20	7.53
			CBPSOTVAC	0.69	117.89	0.015	2482	410.74
			LGEA	**1**	**0**	**0**	**0**	**0**

TABLE 2. Continued.

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Weish21	5	70	MBPSO	0.61	17.85	0.0019	88	24.97
			CBPSO1	0	347.65	0.039	499	84.42
			BPSOTVAC	0.74	6.06	0.0007	24	10.41
			CBPSOTVAC	0.67	125.78	0.016	2574	378.38
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish22	5	80	MBPSO	0.33	29.73	0.0033	112	31.55
			CBPSO	0	674.21	0.082	935	94.69
			BPSOTVAC	0.16	15.12	0.00169	18 6.63	
			CBPSOTVAC	0.17	172.8	0.024	3063	486.71
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish23	5	80	MBPSO	0.24	29.65	0.0036	126	35.43
			CBPSO1	0	670.43	0.087	902	119.64
			BPSOTVAC	0.85	1.11	0.00013	36 5.11	
			CBPSOTVAC	0.58	179	0.026	3114	437.23
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish24	5	80	MBPSO	0.27	17.48	0.0017	70	18.09
			CBPSO1	0	367.36	0.0373	499	56.96
			BPSOTVAC	0.7	3.04	0.00029	31	6.44
			CBPSOTVAC	0.55	113.72	0.012	1841	295.79
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish25	5	80	MBPSO	0.29	15.13	0.0015	61	13.39
			CBPSO1	0	449.77	0.0475	648	86.21
			BPSOTVAC	0.49	4.54	0.00045	24	7.09
			CBPSOTVAC	0.32	112.43	0.013	2321	361.88
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish26	5	90	MBPSO	0.31	27.32	0.0028	114	24.27
			CBPSO1	0	895.39	0.103	1122	126.39
			BPSOTVAC	0.36	11.44	0.0012	47	12.81
			CBPSOTVAC	0.28	270.13	0.04	4084	710.77
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish27	5	90	MBPSO	0.65	23.7	0.0024	203	48.62
			CBPSO	0	967.43	0.109	1205	120.34
			BPSOTVAC	0.99	0.39	0.00004	39	3.9
			CBPSOTVAC	0.83	211.46	0.028	3915	640.43
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish28	5	90	MBPSO	0.64	15.21	0.0016	144	26.72
			CBPSO1	0	980.45	0.115	1266	122.32
			BPSOTVAC	0.87	2.99	0.00031	23	7.77
			CBPSOTVAC	0.62	368.74	0.06	4387	887.33
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish29	5	90	MBPSO	0.46	26.73	0.0029	154	34.74
			CBPSO1	0	981.44	0.117	1180	108.56
			BPSOTVAC	0.86	3.19	0.0003	79	10.09
			CBPSOTVAC	0.48	384.5	0.057	4891	854.5
			LGEA	**1**	**0**	**0**	**0**	**0**
Weish30	5	90	MBPSO	0.38	11.6	0.001	57	14.48
			CBPSO1	0	548.1	0.0516	760	87.58
			BPSOTVAC	0.87	0.52	0.00005	4	1.35
			CBPSOTVAC	0.63	203.79	0.021	2836	491.81
			LGEA	**1**	**0**	**0**	**0**	**0**

TABLE 3. Comparison between LGEA and MBPSO, CBPSO1, BPSOTVAC and CBPSOTVAC algorithms on HP and PB testing problems.

Problem	#Benchs	#Items	Algorithm	SR	MAD	MAPE	LE	SD
Hp1	4	28	MBPSO	0.1	37.52	0.0112	109	25.52
			CBPSO1	0.64	5.5	0.0016	30	7.84
			BPSOTVAC	0.38	11.44	0.0034	33	10.69
			CBPSOTVAC	0.29	14.1	0.0042	72	13.69
			LGEA	**1**	**0**	**0**	**0**	**0**
Hp2	4	35	MBPSO	0.11	46.22	0.015	136	39.15
			CBPSO1	0.73	4.58	0.0014	19	7.65
			BPSOTVAC	0.67	6.51	0.0021	116	13.95
			CBPSOTVAC	0.59	12.39	0.0039	114	21.35
			LGEA	**1**	**0**	**0**	**0**	**0**
Pb1	4	27	MBPSO	0.11	32.62	0.0107	104	24.32
			CBPSO1	0.61	5.93	0.0019	30	7.92
			BPSOTVAC	0.46	9	0.0029	30	9.44
			CBPSOTVAC	0.4	10.26	0.0033	61	10.52
			LGEA	**1**	**0**	**0**	**0**	**0**
Pb2	4	34	MBPSO	0.16	44.69	0.014	154	39.31
			CBPSO1	0.8	4.28	0.00135	95	11.49
			BPSOTVAC	0.73	4.5	0.00142	31	7.68
			CBPSOTVAC	0.51	14.45	0.0046	87	18.73
			LGEA	**1**	**0**	**0**	**0**	**0**
Pb4	2	29	MBPSO	0.27	2639.8	0.029	4751	1803
			CBPSO1	0.99	2.03	0.00002	203	20.3
			BPSOTVAC	0.91	228.1	0.0025	3233	797.1
			CBPSOTVAC	0.84	304.33	0.0033	3498	875.1
			LGEA	**1**	**0**	**0**	**0**	**0**
Pb5	10	20	MBPSO	0.08	49.42	0.024	117	24.36
			CBPSO	0.99	0.17	0.00008	17	1.7
			BPSOTVAC	0.84	2.72	0.0013	17	6.26
			CBPSOTVAC	0.8	3.4	0.0016	17	6.83
			LGEA	**1**	**0**	**0**	**0**	**0**
Pb6	30	40	MBPSO	0.28	27.36	0.038	147	29.12
			CBPSO1	0.55	8.47	0.011	34	10.99
			BPSOTVAC	0.5	8.7	0.012	31	9.99
			CBPSOTVAC	0.54	17.74	0.028	351	40.17
			LGEA	**1**	**0**	**0**	**0**	**0**
Pb7	30	37	MBPSO	0.05	19.89	0.019	82	16.29
			CBPSO1	0.41	5.64	0.0055	22	5.88
			BPSOTVAC	0.47	5.43	0.0053	20	5.71
			CBPSOTVAC	0.4	13.05	0.013	126	24.25
			LGEA	**1**	**0**	**0**	**0**	**0**

algorithm (bfoa2 [23]) and a hybrid EDA-based algorithm (HEDA [22]). The results of the comparison are given in Table 5, where Min.Dev and Ave.Dev are the minimum and average percentage deviations from the best-known values, respectively. The results show that LGEA is also competitive on bigger MKP instances. Compared with bfoa2 and HEDA algorithms, LGEA is able to obtain relatively better average deviations, and the minimal deviations of S-CLPSO in all instances are less than 1%. For the GK08 knapsack benchmark, according to Table 6, LGEA manages to yield better solution than the best known one. This proves that the logic gate mechanism in LGEA is contributing to binary optimization. Overall, the performance results reveal that LGEA is promising.

TABLE 4. Comparison between LGEA and OGA, GADS and IbAFSA algorithms on Peterson testing problems.

Prob.	Method	SR	AE	LE	SDE
PT2	OGA	–	–	0.00	–
	GADS	100.00	0.00	0.00	0.00
	b-AFSA	100.00	0.00	0.00	0.00
	LGEA	**100.00**	**0.00**	**0.00**	**0.00**
PT3	OGA	–	–	0.00	–
	GADS	100.00	0.00	0.00	0.00
	b-AFSA	100.00	0.00	0.00	0.00
	LGEA	**100.00**	**0.00**	**0.00**	**0.00**
PT4	OGA	–	–	0.00	–
	GADS	100.00	0.00	0.00	0.00
	b-AFSA	100.00	0.00	0.00	0.00
	LGEA	**100.00**	**0.00**	**0.00**	**0.00**
PT5	OGA	–	–	10.00	–
	GADS	73.33	2.67	0.00	4.50
	b-AFSA	33.33	17.67	0.00	21.28
	LGEA	**100.00**	**0.00**	**0.00**	**0.00**
PT6	OGA	–	–	0.00	–
	GADS	6.67	52.70	0.00	**27.70**
	b-AFSA	16.67	58.83	0.00	50.00
	LGEA	**20.00**	**42.60**	0.00	38.75
PT7	OGA	–	–	13.00	–
	GADS	0.00	**155.10**	93.00	**31.80**
	b-AFSA	6.67	94.53	0.00	56.43
	LGEA	**20.00**	173.40	**0.00**	145.57

TABLE 5. Comparison between LGEA and bFOA2 and HEDA2 algorithms on Glover and Kochenberger testing problems.

Problem	n × m	Best known	LGEA		bFOA2		HEDA2	
			Min.Dev	Ave.Dev	Min.Dev	Ave.Dev	Min.Dev	Ave.Dev
GK01	100×25	3766	**0.3983**	**0.6638**	0.5576	0.7554	0.6107	0.9360
GK02	100×50	3958	**0.4295**	**0.5375**	0.6569	0.8518	0.7580	0.9790
GK03	150×25	5650	**0.6726**	**0.8673**	0.7965	0.9150	0.9381	1.1531
GK04	150×50	5764	0.8848	**0.9739**	**0.8675**	1.0279	1.0930	1.2673
GK05	200×25	7557	**0.9528**	1.2783	1.0057	**1.1930**	1.2439	1.4960
GK06	200×50	7672	**0.6257**	**0.9515**	0.8472	0.9802	1.1210	1.4436
GK07	500×25	19, 215	**0.9732**	**1.0503**	1.4312	1.5194	1.7018	1.8460
GK09	1500×25	58, 085	**1.0485**	**1.0737**	2.1744	2.2816	2.4585	2.5461
GK10	1500×50	57, 292	**0.9879**	**1.0075**	1.7437	1.7905	2.0177	2.1170
GK11	2500×100	95, 231	**1.1173**	**1.1209**	1.5037	1.5738	1.7043	1.7348

TABLE 6. Simulation results of GK08 testing problem.

Problem	n × m	LGEA	Best known
GK08	500 × 50	**18 808**	18 801

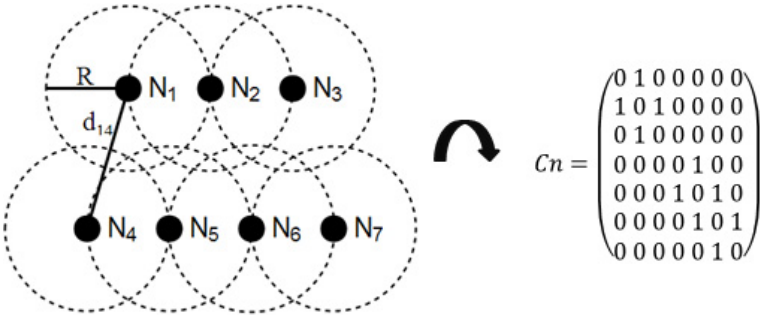

FIGURE 4. The network model and its adjacency matrix.

4. Application to task allocation for wireless sensor network

The second application is of the LGEA is an engineering optimization problem in the domain of in-network processing that is concerned with task allocation for wireless sensor network (WSN). Yang *et al.* [24] have been applied a Binary Particle Swarm Optimization (BPSO) variant recently. Authors in [10,19] considered the standard Genetic Algorithm (GA) tool to provide the well-performing task allocation scheme. In this work, the proposed LGEA is used to search for the best task allocation scheme. The problem is firstly formulated as a multiobjective constrained optimization problem. The individuals of LGEA are encoded with binary matrices to represent candidate task allocation schemes. The task-workload and the connectivity are considered as constraints to guarantee the completion of each task and important data transaction among the selected nodes. A fitness function including the number of active nodes and their load distribution is designed to evaluate the quality of each solution.

4.1. Problem definition

The network model, task model and cost functions are presented to develop our framework for the task allocation problem.

4.1.1. The network model

The WSN is composed of a number of heterogeneous sensor nodes deployed randomly in the monitoring region. The network topology is modelled by a weighted undirected graph $G = (N, R)$ as described in Figure 4; $N = \{N_j : j = 1, 2, \ldots, n\}$ is the set of vertices corresponding to the network nodes. R is the set of edges corresponding to the direct communication link among nodes. There is no direction for the edges since all the nodes have the same maximum transmission range the communications are bidirectional. Each node N_j is characterized by (a_j, b_j), a_j is the load for the processor of node N_j and b_j is includes the load for transmitting and receiving a data packet. The weight on each edge, $d_{i,j}$, is equivalent to the direct communication link between N_i and N_j. When $d_{i,j}$ is smaller than the maximum transmission range, the two nodes are directly connected to each other. The network model can be denoted by its adjacency matrix $Cx = (Cx_{i,j})_{n\times n} \in \{0,1\}^{n\times n}$, which informs of the direct communication among nodes.

$Cx_{i,j} = 1$ *if* node i is directly connected to node j.

$Cx_{i,j} = 0$ *if* node i is not connected to node j.

4.1.2. The task model

A WSN application can be executed by a sequence of processing tasks. The tasks and their performance order can be represented by the Directed Acyclic Graph (DAG).The set of vertices are expressed as $M = \{M_i : i = 1, 2, \ldots, m\}$. Each task is characterized by its computation workload $P = \{P_i : i = 1, 2, \ldots, m\}$ and its communication workload $L = \{L_i : i = 1, 2, \ldots, m\}$. The workload of each task respects the energy supply

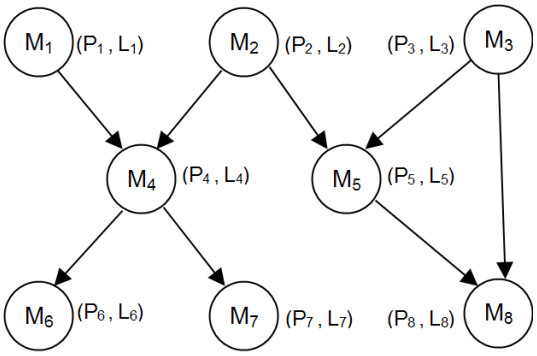

FIGURE 5. DAG for the task model.

of a single node. The direction of edges describes the priority among different tasks. For a given edge, if M_i is the immediate predecessor of M_j, M_j must be executed after M_i has been achieved. A DAG for tasks [24] is shown in Figure 5.

4.1.3. Cost functions

To achieve cooperative in-network processing in WSN with less energy consumption and longer network lifetime, the total number of active nodes, both of the computation and communication load uniformity are considered as cost functions.

(1) *The number of active nodes*: A node that is selected for a specialized task is named as an active node. The number of active nodes for all the tasks in WSN

can be expressed by an $m \times n$ matrix. An element of C is $c_{i,j}$, which denotes whether the node N_j is active for task M_i. When N_j is selected for M_i, $c_{i,j} = 1$. Otherwise $c_{i,j} = 0$. Then, the total number of active nodes in WSN for all the tasks is repersented by (4.1).

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} \tag{4.1}$$

(2) *The computation load uniformity*: The equilibration of loads of sensor nodes plays a key role in balancing the energy consumption. The computation energy of sensor nodes for all the tasks in WSN can be expressed by a $m \times n$ matrix A. $a_{i,j}$ is an element of A. When the node N_j is not selected for task M_i, $a_{i,j} = 0$. Otherwise, $a_{i,j}$ represents a certain amount of computation load. In this study, the computational load uniformity of WSN can be measured by the deviation of computation load of the sensor nodes among all the tasks (DP) as (4.2).

$$DP = \sum_{i=1}^{m} |A_i - P_i| \tag{4.2}$$

$A_i = \sum_{j=1}^{n} a_{i,j}$ is the total computation load of all the nodes assigned with the task M_i, P_i is the computation workload of task M_i, which is also the expectation of energy consumption for computation for all the sensor nodes.

(3) *The communication load uniformity*: The communication energy of sensor nodes for all the tasks in WSN can be expressed by a $m \times n$ matrix B. $b_{i,j}$ is an element of B. When the node N_j is not selected for task M_i,

$b_{i,j} = 0$. Otherwise, $b_{i,j}$ represents a certain amount of communication load. The communication load uniformity can be calculated by (4.3).

$$DL = \sum_{i=1}^{m} |B_i - L_i| \tag{4.3}$$

$B_i = \sum_{j=1}^{n} b_{i,j}$ is the total communication load of all the nodes assigned with the task M_i, P_i is the communication workload of task M_i, which is also the expectation of energy consumption for communication for all the sensor nodes.

4.1.4. Problem formulation

An application is decomposed of m processing tasks to be simulated in the WSN. The set of tasks is represented by $M = \{M_i : i = 1, 2, \ldots, m\}$. For a specific task M_i, a group of k sensor nodes $N_s = \{N_{sk} : k = 1, 2, \ldots, n\}$ is selected. The selected nodes must have adequate resource capabilities for completing the different task workloads. Moreover, they must be connected with each other to ensure the prerequisite data exchange. The aim of task allocation is to obtain a scheme that can achieve the best global network performance under the required constraints on task workload and connectivity. In this work, we propose optimization metrics for a task allocation scheme. The metrics include the total number of active nodes, total deviation of computation loads and total deviation of communication loads of the sensor nodes among all the tasks. Therefore, the task allocation problem can be formulated as a multiobjective constrained optimization problem. The general optimization formulation of the objective function with weights for each metric is expressed by (4.4).

$$\text{minimize } f(X) = \alpha \times C + \beta \times DP + \gamma \times DL \tag{4.4}$$

where C is total number of the active nodes in WSN; DP and DL are the total computation and communication load deviations of the sensor nodes among all the tasks.

(1) *Task workload constraint*: According to the DAG of tasks, each task requires a workload on computation as well as on communication. Moreover, the assigned group of nodes must be capable to accomplish the task. A task M_i needs a total computation load P_i and a total communication load L_i. To finish P_i, the total computation load of the selected nodes A_i should be larger than P_i. To finish L_i, the total computation load of the selected nodes B_i should be larger than L_i. The constraints on task workloads for the current individual are expressed as (4.5).

$$\begin{cases} A_i \geq P_i \\ B_i \geq L_i \end{cases} \tag{4.5}$$

The constraint on workload is then expressed by the limitation of the minimal load of the selected nodes for a specialized task. The constraint on task workload is satisfied only when (4.5) is satisfied.

(2) *Connectivity constraint*: The connectivity would tell whether the selected nodes of a specialized task ensure direct communication. When two nodes are connected, the distance between them must be within a range distance R. The connectivity of the sensor nodes is defined in matrix Cx. As a result of this, the nodes that violate the constraint on connectivity will be eliminated.

4.2. Task allocation using LGEA

The binary solution stands for a potential task allocation scheme for the WSN. It is encoded into a $m \times n$ binary matrix $X_i^G \in [0, 1]^{m \times n}$, where m is the total number of tasks in and n is the total number of sensor nodes in the WSN.

Each row of the solution expresses an allocation representation of a particular task. The priority of the tasks is described by the order of rows in the solution, *i.e.* the tasks with the highest priority are in the first rows and the tasks with the lowest priority are in the last rows.

$$X_i^G = \begin{bmatrix} x_{i,11} & \dots & x_{i,1n} \\ \vdots & \ddots & \vdots \\ x_{i,m1} & \dots & x_{i,mn} \end{bmatrix}$$

The binary value of each element in X_i^G denotes whether a node is selected for a particular task. For example, $x_{i,pq}$ is an element of X_i^G. $x_{i,pq} = 1$ means that the q^th node is selected for the p^th task. $x_{i,pq} = 0$ means that the q^th node is not selected for the p^th task.

4.2.1. LGEA-based task allocation

The objective of the proposed algorithm is to find out the optimal task allocation scheme that achieves the best overall performance of the WSN. Each task is assigned to a proper group of nodes. The following implementation steps describe the whole process:

Step 1. Create the DAG of the tasks and the graph for WSN. Initialize the parameters for tasks and WSN.
Step 2. Set the control parameters for LGEA.
Step 3. Initialize the binary solution with the satisfaction of task workload and connectivity constraints. The population is randomly initialized in the binary domain as (4.6).

$$x_{i,pq} = \begin{cases} 1, & \text{if} \quad rand() < p_0 \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

where *rand* is a random number; p_0 is the pseudo probability of being "1" for the components of individuals in the initial population. p_0 stands for the desired percentage of the selected node q for the task p.
Step 4. Generate the offspring individuals using the logic mechanism. For example, we exploit the XOR-rand2rand strategy as defined in (2.2)−(2.4).
Step 5. Fulfill the constraint of connectivity for the trial solutions. Evaluate the constraint on task workload using (4.5). If the constraint on task workload is not satisfied, then return to Step 4. Otherwise, go on to Step 6.
Step 6. Evaluate the fitness value of the newly generated individuals according to the fitness function f as (4.4).
Step 7. Update the current population as follows: if the fitness of the trial individual is better than the old value, the trial individual will be selected for the next generation.
Step 8. When the maximum number of generations is not reached go back to Step 4. Otherwise, return the best individual.

4.3. Simulations

We design a simulation environment for the task allocation problem in order to test the effectiveness of LGEA. Assume that the WSN is composed of a heterogeneous sensor nodes' number deployed in a monitoring area of 50 m × 50 m. The nodes have different characteristics on computation and communication. The computation and communication loads for a single sensor node are set to be uniformly distributed in the range of $[1, n]$, n is the number of sensor nodes. The number of tasks and their execution order are randomly initialized in the simulation. Suppose that the computation load and communication load for each task is set to be uniformly distributed in the range of $[1, m]$, m is the number of tasks.

To confirm the improvement by the proposed logic gate mechanism, the proposed algorithm was compared with two existing binary evolutionary algorithms: the GA and the BPSO algorithm. The maximum generation number for all methods is 1000; for the BPSO the inertia weights are: $w_{\max} = 0.9$, $w_{\min} = 0.4$, the inertia

TABLE 7. Experimental results of LGEA, BPSO and GA.

Algorithm		LGEA	BPSO	GA
MF		**5.1251**	15.9246	22.1475
Best solution	f	**3.2542**	9.9464	13.2603
	C	**13**	18	28
	DP	**2.5955**	7.4976	8.5887
	DL	**1.2491**	6.0850	9.5811

MF = mean best fitness.

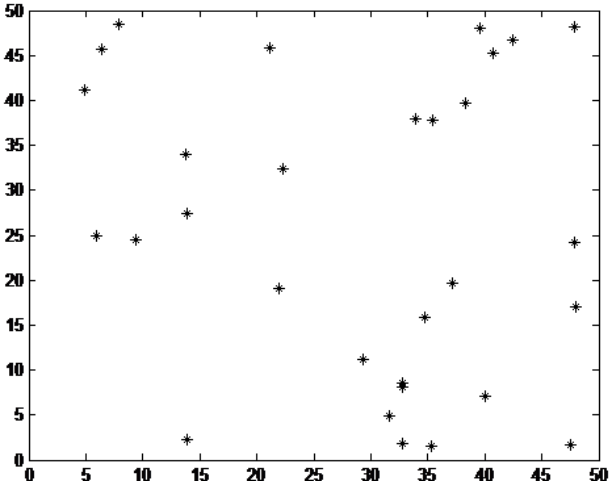

FIGURE 6. The problem set up (* represents the sensor node).

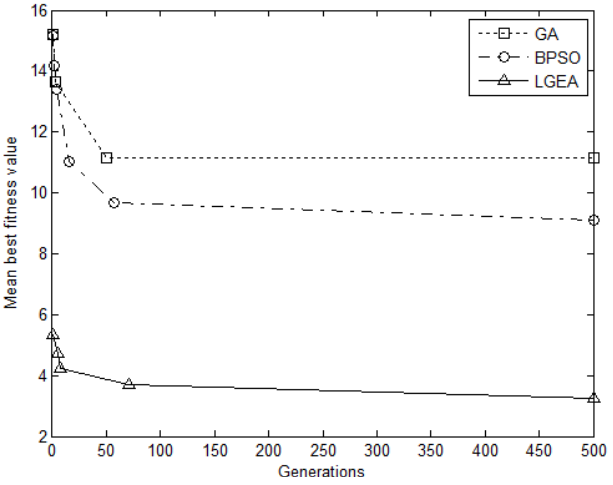

FIGURE 7. Convergence curves of the mean best fitness.

constant is $c = 2.0$ and the velocity constant $v_{\max}$ is 6.0; for the GA the mutation rate is 0.2. For all the simulations, the default number of tasks and sensor nodes are set to be 8 and 30 respectively. Figure 6 shows the monitoring area for the WSN. Experimental results for performance comparison are listed in Table 7 in which

C, DP and DL incorporated in the best solution are also given. It shows that the proposed LGEA can achieve the best performance on the different metrics. The convergence curve of the mean best fitness value is depicted in Figure 7. It can be seen that the BPSO and GA have slower convergence speeds and get easily stuck in local minima. However, the LGEA needs less number of generations than BPSO and GA to converge to the global best value.

5. Conclusion

A logic gate-based evolutionary algorithm (LGEA) has been proposed. In order to solve binary space problems, LGEA performs the binary perturbation using the logic gate mechanism. Common space transformation mechanisms and classic recombination and mutation methods are replaced by the logic gate mechanism. Thus, LGEA represents a new technique to combine the concept of logic gates into the procedure of generating offspring in an evolutionary algorithm. We have experimentally investigated the performances of our algorithm on several benchmarks of the multidimensional knapsack problem (MKP). The algorithm was compared with existing binary EAs and showed a superior performance to them. To demonstrate the usefulness of the algorithm, we have solved an engineering optimization problem, task allocation for wireless sensor network. Experimental results have proved the potential of the logic gate mechanism.

References

[1] M.A.K. Azad, A.M.A.C. Rocha and E.M.G.P. Fernandes, Solving multidimensional 0–1 knapsack problem with an artificial fish swarm algorithm. In *Proc. 12th Int. Conf. Comput. Sci. and Its Applications (ICCSA 2012)*, Salvador de Bahia, Brazil. Springer Berlin Heidelberg (2012) 72–86.

[2] J.C. Bansala and K. Deep, A modified binary particle swarm optimization for knapsack problems. *Appl. Math. Comput.* **218** (2012) 11042–11061.

[3] J.E. Beasley, Or-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** (1990) 1069–1072.

[4] M. Chih, C.-J. Lin, M.-S. Chern and T.-Y. Ou, Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem. *Appl. Math. Model.* **38** (2014) 1338–1350.

[5] L.-Y. Chuang, C.-H. Yang and J.-C. Li, Chaotic maps based on binary particle swarm optimization for feature selection. *Appl. Soft Comput.* **11** (2011) 239–248.

[6] M. Dorigo and G.D. Caro, Ant colony optimization: a new meta-heuristic. In *Proc. 1999, Congr. Evolutionary Computation (CEC 99)*, Washington, USA (1999).

[7] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory. In *Proc. of the Sixth Int. Simp. on Micro Machine and Human Science (MHS '95)* (1995) 39–43.

[8] A. Fréville, The multidimensional 0–1 knapsack problem: An overview. *European J. Oper. Res.* **155** (2004) 1–21.

[9] J. Holland, Genetic algorithms. *Sci. Amer.* **267** (1992) 66–72.

[10] Y. Jin, J. Jin, A. Gluhak, K. Moessner and M. Palaniswami, An intelligent task allocation scheme for multihop wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **23** (2012) 444–451.

[11] D. Karaboga and B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization. In *Proc. of the 12th Int. Fuzzy Syst. Assoc. World Congr. on Foundations of Fuzzy Logic and Soft Computing (IFSA 2007)*, Cancun, Mexico. Springer Berlin Heidelberg (2007) 789–798.

[12] J. Kennedy and R.C. Eberhart, A discrete binary version of the particle swarm algorithm. In *IEEE Int. Conf. Syst., Man, and Cybernetics*. Florida, USA (1997) 4104–4108.

[13] M.S. Kiran and M. Gündüz, Xor-based artificial bee colony algorithm for binary optimization. *Turkish J. Electr. Eng. Comput. Sci.* **21** (2013) 2307–2328.

[14] H. Li, Y.-C. Jiao, L. Zhang and Z.-W. Gu, Genetic algorithm based on the orthogonal design for multidimensional knapsack problems. In *Proc. of Second Int. Conf. Natural Computation (ICNC 2006)*, Xi'an, China. Springer Berlin Heidelberg (2006) 696–705.

[15] A. Marandi, F. Afshinmanesh, M. Shahabadi and F. Bahrami, Boolean particle swarm optimization and its application to the design of a dual-band dual-polarized planar antenna. In *2006 IEEE Conf. on Evolutionary Computation (CEC 2006)*, Vancouver, Canada. *IEEE Comput. Intell. Soc.* (2006) 3212–3218

[16] H. Pirkul, A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Res. Logist.* **34** (1987) 161–172.

[17] A.K. Qin, V.L. Huang and P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.* **13** (2009) 398–417.

[18] M. Sakawa and K. Kato, Genetic algorithms with double strings for 0–1 programming problems. *Eur. J. Oper. Res.* **144** (2003) 581–597.

[19] R. Shams and F.H. Khan, Solving wireless network scheduling problem by genetic algorithm. *Glob. Enginners Technologists Rev.* **2** (2012) 10–13.

[20] D. Stefanoiu, P. Borne, D. Popescu, F.G. Filip and A. ElKamel, *Optimization in Engineering Sciences: Approximate and Metaheuristic Methods*. John Wiley (2014).

[21] R. Storn and K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11** (1997) 341–359.

[22] L. Wang, S.-Y. Wang and Y. Xu, An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem. *Expert Syst. Appl.* **39** (2012) 5593–5599.

[23] L. Wang, X.-L. Zheng and S.-Y. Wang, A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. *Knowledge-Based Syst.* **48** (2013) 17–23.

[24] J. Yang, H. Zhang, Y. Ling, C. Pan and W. Sun, Task allocation for wireless sensor network using modified binary particle swarm optimization. *IEEE Sensors J.* **14** (2014) 882–892.