An Effective Hyper-Heuristic Algorithm for Clustering Problem of Wireless Sensor Network

Chun-Wei Tsai[1], Wei-Lun Chang[2], Kai-Cheng Hu[2], and Ming-Chao Chiang[2(✉)]

[1] Department of Computer Science and Engineering, National Chung Hsing University, Taichung 40227, Taiwan, R.O.C.
cwtsai0807@gmail.com
[2] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, R.O.C.
m033040027@student.nsysu.edu.tw, spring69953@hotmail.com, mcchiang@cse.nsysu.edu.tw

Abstract. The basic idea of low-energy adaptive clustering hierarchy (LEACH) is not to select a particular set of sensors out of all the sensors as the cluster heads to avoid the problem of running out their energy quickly. Unfortunately, it may end up selecting an unsuitable set of sensors as the cluster heads. Inspired by these observations, an effective hyper-heuristic algorithm is presented in this paper to find out the transmission path that is able to give better results than the other algorithms compared in this research. In other words, the main objective of the proposed algorithm is to reduce the energy consumption of a wireless sensor network (WSN), by balancing the residual energy of all the wireless sensors to maximize the number of alive sensor nodes in a WSN. Experimental results show that the proposed algorithm can provide a better result in terms of the energy consumed by a WSN, meaning that the proposed algorithm provides an alternative way to extend the lifetime of a WSN.

Keywords: Wireless sensor network · Lifetime · LEACH

1 Introduction

Nowadays, wireless sensor network (WSN) is no longer something that is sitting on the corner of a laboratory. Several successful results [10,11,13] indicate that WSN has become part of our daily life in recent years. Its importance can also be found in [4] in which Harrop and Das reported that the market of WSN will grow to $1.8 billion in 2024. In [12], Reese estimated that about 24 million sensors of industrial WSNs will be installed in the next five years. Moreover, since WSN is the foundation of internet of things (IoT), industrial internet of things (IIoT), and even big data analytics systems, how to enhance the performance of a WSN has become an important area of research today. Also owing to inherent

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2017
J.-H. Lee and S. Pack (Eds.): QShine 2016, LNICST 199, pp. 97–109, 2017.
DOI: 10.1007/978-3-319-60717-7_10

limitation, the energies of the wireless sensors that are equipped with batteries are normally very small, the data transmission distance is still restricted to a small region. For these reasons, how to use the energy of sensors effectively will have a strong impact on the lifetime of a WSN. Several studies [15] have attempted to define the lifetime problem as an optimization problem for finding out the possible solutions, such as the cluster head election problem (CHEP) and routing problem.

One of the most well-known clustering method for solving the CHEP is the so-called low-energy adaptive clustering hierarchy (LEACH) [5]. One of the reasons is that it is very simple and easy to implement. Another reason that a large number of methods are built on LEACH or its variants because the original idea of LEACH is to avoid a sensor from being elected as the cluster head (CH) too often (i.e., too many times). This implies two things. The first is that the CH election procedure will not elect a particular sensor as the CH every time LEACH is performed. The second is that each sensor has a chance to be elected as a CH. It seems that LEACH provides a good way to avoid specific sensors from running out energy quickly; thus, it can reduce the energy consumption of a WSN. However, the observations of Hoang et al. [6], indicate that LEACH may not able to provide the best solution for the CHEP. To extend the lifetime of a WSN, several studies [6,8,15] attempted to combine metaheuristic algorithms with LEACH to further improve the performance of LEACH for the CHEP, but there is still plenty of room for the improvement. In this paper, an effective hyper-heuristic algorithm with LEACH is presented for solving the CHEP. The main idea of the proposed algorithm is to leverage the strength of different metaheuristic algorithms (e.g., genetic algorithm or particle swarm optimization) to provide a better solution for the CHEP.

2 Related Work

Just like the other emerging research problems, the lifetime problem of a WSN requires some effective and efficient algorithms to find out a *good solution* to enhance its performance in a reasonable time because it usually takes an unreasonable amount of time to find the optimal solution of these optimization problems. That is why we are looking for a good solution (i.e., approximate solution) instead of the best solution. However, *good way* to find out a *good solution* of these emerging research problems typically depends on the things we concern. It can be a fast search method or a method that is guaranteed to find out a good solution that is very close to the best solution.

The development of metaheuristic algorithms [1] is just like an epitome of the modern computer science and other relevant disciplines that need to use information systems because metaheuristic algorithms can be used in many different disciplines to find out a good solution in a reasonable time. As we mentioned previously, this provides us an alternative method to find out a good solution before we can find out the best solution (i.e., optimal solution) for these hard and complex optimization problems in a reasonable time, e.g., CHEP. However,

every metaheuristic algorithm has its advantages and disadvantages. That is, none of them can be fully replaced by another one. A good example is the genetic algorithm that is good at global search but bad at local search in most cases and explains the dilemmas of metaheuristic algorithms. An intuitive way to solve this issue is to combine GA with another search algorithm that is good at local search, such as combining GA with k-means to find a better result than does GA or k-means alone for the clustering problem [7]. Some recent studies called this kind of integration hybrid-heuristic algorithm [2]. However, if we simply combine two or more different algorithms into a search algorithm, the computation time of this new search algorithm will be increased significantly. The hyper-heuristic algorithm [3] provides an alternative way to solve this problem. Like hybrid-heuristic algorithm, hyper-heuristic algorithm also integrates two or more search algorithms into a single search algorithm. Unlike hybrid-heuristic algorithm that will use "all" the search algorithms at each iteration during the convergence process, hyper-heuristic algorithm will use only "one" of the search algorithms at each iteration during the convergence process. Because hyper-heuristic algorithm will use only one of the search algorithms at each iteration, the computation time can then be significantly reduced, compared to hybrid-heuristic algorithm. More precisely, this is how hyper-heuristic algorithm works. First, it will randomly choose one of the search algorithms to perform the search for a certain number of iterations. Then, it will randomly choose another one to replace the current one when the conditions for changing the search algorithms are satisfied. For hyper-heuristic algorithm, the conditions are used to determine the timing to change the search algorithm. The condition can be a fixed number of iterations, which means that each search algorithm will be performed for a fixed number of iterations before it is changed to another. Or it can be a predefined threshold, which means that the hyper-heuristic algorithm will switch to another search algorithm when the current one cannot find a better solution.

The hyper-heuristic algorithm still has some research problems to be addressed to further enhance its search performance. According to our observations, they can be summarized as follows:

- Time to change: This research issue is regarding the timing for changing the search algorithm (which is also referred to as the low-level heuristic (LLH) algorithm), which is a difficult problem for this kind of research. If we change the search algorithm too early, we might not be able to use its fully search ability. On the other hand, if we change the search algorithm too late, it might get stuck in a local optimum.
- Execution sequence: Another research issue is the execution sequence of these LLH algorithms. Until now, there is no specific execution sequence that can outperform the others in solving all the optimization problems.
- Passing the search experience: Since the LLH algorithms consist of both single-solution-based and population-based algorithms, the search process of which are very different, how to pass the searched solutions from one

LLH algorithm to another is also an important research issue for the hyper-heuristic algorithm.

From these observations, it can be easily seen that even though the hyper-heuristic algorithm can provide a better way to solve the optimization problem than the other heuristic algorithms, there is still plenty of room for the improvement.

3 The Proposed Algorithm

3.1 The Basic Idea

The basic idea of the proposed algorithm is to use a high performance hyper-heuristic algorithm (HHA) [14] called Effective Hyper-Heuristic Algorithm (EHHA) to solve the CHEP; that is, to find out suitable CHs for the CHEP. In order to further improve the performance of the hyper-heuristic algorithm we presented in [14], an additional operator, namely, recording pool, will be presented in this study. Also to eliminate redundant computations in the process of selecting the CHs, a check procedure will be used to determine whether to use EHHA or not at the very beginning of every round. For example, if there are 100 sensor nodes and p is set to 0.05, the number of CHs is $100 \times 0.05 = 5$. Note that p is the value to decide the number of CHs with a WSN which, based on the definition of [5], is set to 0.05 in this paper. If the minimum number of wireless sensors that can be selected as CHs is larger than 5, we will then perform the EHHA; otherwise, we will not perform EHHA. This means that all the remaining wireless sensors will be selected as the CHs.

3.2 The Effective Hyper-Heuristic Algorithm

The pseudocode of the proposed algorithm is as given in Fig. 1, which can be divided into two parts: initialization and the process of EHHA. Line 2 indicates that the proposed algorithm will first initialize all the parameters $\phi_{\max}$, ϕ_{ni}, r, p_p and p_b, where $\phi_{\max}$ denotes the maximum number of iterations for the selected LLH algorithm to run; ϕ_{ni} the maximum number of iterations if the selected LLH algorithm cannot improve the results; r the size of the recording pool to store the best solutions when changing the LLH algorithm; p_p the number of solutions to be changed by using the recording pool when changing the LLH algorithm; and p_b the percentage of solutions to be changed when generating a new solution. Line 3 inputs the data of a WSN that contains the location information of wireless sensors, the residual energy, and other information. Line 4 will then initialize the population of solutions $X = \{x_1, x_2, ..., x_N\}$, where N is the population size.

The main procedure of the proposed algorithm starts at line 6. As line 6 shows, it will first randomly select a LLH algorithm from the candidate pool. As far as this study is concerned, the LLH candidate pool consists of ant colony optimization (ACO), genetic algorithm (GA), particle swarm optimization (PSO),

```
1  /* Initialization */
2  Set up the parameters φmax, φni, r, pp, and pb
3  Input the information of sensors
4  Initialize the population of solutions X = {x1, x2, ..., xN}
5  /* The process of EHHA */
6  Randomly select a heuristic algorithm Hi from the candidate pool H
7  While the termination criterion is not met
8    Update the population of solutions X by using the selected algorithm Hi
9    Evaluate the fitness value of each solution after computing the energy level
10   F1 = Improvement_Detection(X)
11   F2 = Diversity_Detection(X)
12   If ψ(Hi, F1, F2)
13     Randomly select a new Hi
14     Save the best solution into the Recording Pool
15     Change part of population by using the Recording Pool
16   End If
17 End While
18 Transform the results to CHEP
```

Fig. 1. Outline of the effective hyper-heuristic algorithm.

and tabu search (TS). As lines 7–17 show, the selected LLH algorithm will be performed repeatedly until the termination criterion is met. The population will be changed by the selected LLH algorithm. If the selected LLH algorithms has performed $\phi_{\max}$ iterations, it will be stopped. Moreover, Eq. (1) indicates that EHHA can determine whether to switch to a new LLH algorithm, by using the function $\psi(H_i, F_1, F_2)$ and the parameters H_i, F_1 and F_2, where H_i denotes the LLH algorithm selected, F_1 the improvement detection operator, and F_2 the diversity detection operator. For single-solution-based heuristic algorithms (SSBHA), only F_1 is used whereas for population-based heuristic algorithms (PBHA), both F_1 and F_2 are used.

$$\psi(H_i, F_1, F_2) = \begin{cases} \text{false} & \text{if } H_i \in \mathbf{S} \text{ and } F_1 = \text{true}, \\ \text{false} & \text{if } H_i \in \mathbf{P} \text{ and } F_1 = \text{true and } F_2 = \text{true}, \\ \text{false} & \text{if } \phi_{\max} \text{ is not reached}, \\ \text{true} & \text{otherwise}, \end{cases} \tag{1}$$

where $\mathbf{S}$ denotes the set of SSBHAs; and $\mathbf{P}$ the set of PBHAs. If the function $\psi(H_i, F_1, F_2)$ returns true, the proposed algorithm will randomly select a new LLH algorithm to switch to while at the same time saving the so-far-best solution into the recording pool and changing part of the population by using the recording pool. In this way, some of the solutions will be changed while some of them will remain intact.

The Improvement_Detection(X) operator and the Diversity_Detection(X) operator [14] of the proposed algorithm are responsible for determining whether

the current LLH algorithm finds a better result or not and for measuring the search diversity of the proposed algorithm, respectively.

The Improvement_Detection(X) operator will return the value that F_1, as defined in Eq. (2), returns; that is, a false if the solution is not improved after ϕ_{ni} iterations in a row; otherwise, it will return a true.

$$F_1 = \begin{cases} \text{false} & \text{if the solution is not improved after } \phi_{\text{ni}} \text{ iterations,} \\ \text{true} & \text{otherwise.} \end{cases} \tag{2}$$

In other words, this operator will check to see if the solution found by the selected LLH algorithm H_i is an improvement or not at the ϕ_{ni} iteration. If H_i is an improvement, it will return a true to inform the high-level hyper center that it should keep using this selected LLH algorithm to find a better solution. However, if the selected LLH algorithm H_i cannot improve the current solution for ϕ_{ni} iterations, it will return a false to inform the high-level hyper center that it should switch to a new LLH algorithm to improve the quality of its solution.

The Diversity_Detection(X) operator is used to measure the search diversity of the proposed algorithm. It returns the value that F_2, as defined in Eq. (3), returns. The diversity of the initial solution $D(X_0)$ will be used as a threshold ω, i.e., $\omega = D(X_0)$. The proposed algorithm computes the diversity of the current solution $D(X)$ as the average of the distances between individual solutions. If the diversity of the current solution $D(X)$ is greater than the threshold ω, the operator will return a true, and EHHA will continue to explore the solutions using the original LLH. Otherwise, EHHA will randomly select a new LLH.

$$F_2 = \begin{cases} \text{true} & \text{if } D(X) > \omega, \\ \text{false} & \text{otherwise.} \end{cases} \tag{3}$$

3.3 Recording Pool

The traditional hyper-heuristic algorithm selects the LLH randomly. Although it is easy to implement, it might select an unsuitable LLH algorithm during the convergence process, and the results will degrade. In order to avoid this problem, a recording pool used to improve the search performance of the hyper-heuristic algorithm is presented in this section. Lines 14–15 depict how the recording pool is used to record the best solutions and to change part of the solutions when the function $\psi(H_i, F_1, F_2)$ returns a true. Initially, the recording pool is empty, so it will directly store the best solution into the recording pool. However, if the recording pool is full, it will retain only the best solution in the recording pool but remove all the other solutions. This mechanism ensures that a good solution can be produced from the recording pool and the exploration of good solutions can be continued based on these solutions. More precisely, the proposed algorithm will change only some of the solutions in the population with a predefined probability p_p. If all the solutions in the population are changed based on the solutions in the recording pool, it may end up having a population all the solutions of which are the same when the recording pool has one and only one solution. This

apparently will decrease the search diversity. EHHA will use the best solution in the recording pool as the basis for generating a new solution. Some bits in the new solution will be changed with a probability p_b. The changed bits are replaced by the corresponding bits of the solution in the recording pool. In brief, it will make the new solution similar to the solution in the recording pool, but not exactly the same.

3.4 The Other Operators of EHHA

For a WSN, each wireless sensor has a fixed location. But the way the solutions of the proposed algorithm are encoded is a virtual location, because some of the population-based heuristic algorithms adopted in this study use the centroids as the CHs of the CHEP. That is why the solutions of the EHHA are not the same as the input locations of the wireless sensors. For this reason, we need a mechanism for both the HHA and EHHA to transform the solutions to the input locations of sensors. In order to transform the solutions, the proposed algorithm will first select the nearest wireless sensor with a higher energy level as the first location of the solution obtained by the proposed algorithm; it will then select the second nearest wireless sensor with a higher energy level as the second location, and so on. In some cases, the number of wireless sensors with a higher energy level is not enough to fix all the locations. If this is the case, it will select the nearest wireless sensor with a lower energy level to fix its location. Then, it will select the second nearest wireless sensor with a lower energy level to fix its location, and so on until all the locations are fixed. The quality of the proposed algorithm is evaluated by the sum of squared errors, defined as follows:

$$\mathrm{SSE} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \parallel x_{ij} - c_i \parallel^2, \tag{4}$$

where k denotes the number of CHs, c_i the ith CH, n_i the number of wireless sensors belonging to c_i, and x_{ij} the wireless sensor belonging to c_i. In other words, the fitness of the solutions is measured by the total distance of transmission because the energy consumed by the transmission of data is significantly influenced by the distance; thus, the aim is to find the shortest transmission path.

4 Experimental Results

4.1 Experimental Environment and Parameter Settings

The empirical analysis was conducted on a PC with 2.67 GHz Intel Core i7 CPU and 4 GB of memory running Fedora 12 with Linux 2.6.32.26-175.fc12.x86_64, and the programs are written in C++ and compiled using g++. To evaluate the performance of EHHA for WSN with different sizes of area and different numbers of nodes using the first-order radio model [5], we compare it with LEACH,

Table 1. Parameter settings of the first-order radio model.

Parameters	Values
Initial energy (E_0)	0.5 J/node
Transmitter electronics (E_{elec})	50 nJ/bit
Receiver electronics (E_{elec})	50 nJ/bit
Data packet length (l)	4000 bits
Data aggregation energy (E_{DA})	5 nJ/bit/signal
Transmitter amplifier (ε_{fs}) if $d \leq d_o$	10 pJ/bit/m^2
Transmitter amplifier (ε_{mp}) if $d > d_o$	0.0013 pJ/bit/m^4

LEACH-GA [9], and hyper-heuristic algorithm (HHA) [3]. The parameter settings of the first-order radio model are as given in Table 1. The first-order radio model can be divided into the transmitter and receiver to transmit and receive detected data. The transmitter consists of the radio electronics and the power amplifier while the receiver consists of the radio electronics. The energy consumed by the transmitter transmitting an l-bit message over a distance d is defined by

$$E_{\text{Tx}}(l,d) = \begin{cases} l \times E_{\text{elec}} + l \times \varepsilon_{\text{fs}} \times d^2 & \text{if } d \leq d_o, \\ l \times E_{\text{elec}} + l \times \varepsilon_{\text{mp}} \times d^4 & \text{if } d > d_o, \end{cases} \tag{5}$$

where the threshold distance d_0 is defined as $\sqrt{\varepsilon_{\text{fs}}/\varepsilon_{\text{mp}}}$. E_{elec} is the energy consumed for transmitting and receiving a bit. $\varepsilon_{\text{fs}}d^2$ and $\varepsilon_{\text{mp}}d^4$ are the energy consumed by the amplifier, which depends on the distance between the transmitter and receiver. The energy consumed by a receiver in receiving an l-bit message is defined by

$$E_{\text{Rx}}(l) = l \times E_{\text{elec}}. \tag{6}$$

The percentage of CHs of LEACH is set equal to 0.05, and the maximum number of rounds is set equal to 10,000. The parameter settings of LEACH-GA are as follows: the crossover rate is set equal to 1.0; the mutation rate is set equal to 0.1; the population size is set equal to 10, and the maximum number of iterations is set equal to 10,000. For HHA and the proposed algorithm, the maximum number of iterations per run is set equal to 200. The population size is set equal to 20 for the PBHAs. The other parameter settings of the energy-effective algorithm are as shown in Table 2. Each simulation is carried out for 30 runs, and the results shown are the average of the 30 runs.

4.2 Results

Figure 2 gives the numbers of alive nodes for the 100 sensors and 100 m × 100 m area case with BS located in the middle, i.e., at (50, 50), of the WSN. It can be easily seen that LEACH-GA outperforms LEACH, because LEACH-GA finds a better probability for the CH selection. The results of HHA and EHHA are

Table 2. Parameter settings of the five energy-effective algorithms for WSN.

Algorithm	Parameters
ACO	Pheromone updating fact $\rho = 0.05$
	Choosing probability $q_0 = 0.5$
	Related influence weights $\alpha = \beta = 1$
GA	Crossover rate $c = 1$
	Mutation rate $m = 0.1$
PSO	Inertia weight $\omega = 0.5$
	Acceleration coefficient $c_1 = c_2 = 2.0$
TS	List size = 6
EHHA	Max iteration of LLH algorithm $\phi_{\max} = 50$
	Non-improved iteration threshold $\phi_{\text{ni}} = 5$
	Size of the recording pool $r = 10$
	Population change rate $p_p = 0.3$
	Bit change rate $p_b = 0.4$

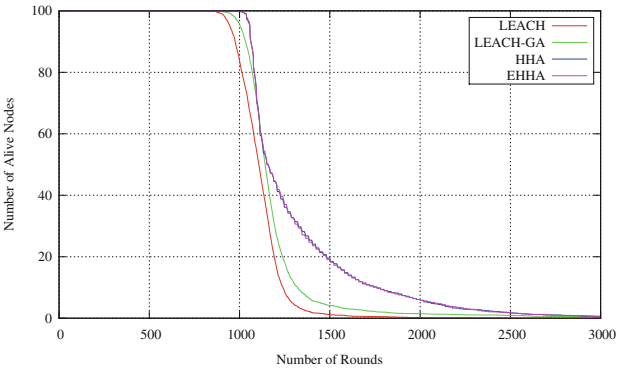

Fig. 2. Numbers of alive nodes for the 100 sensors and 100 m × 100 m area case with BS located at (50, 50).

better than LEACH and LEACH-GA, because HHA and EHHA finds a better distribution of CHs. LEACH may find a bad distribution of CHs that will affect the energy consumed. The death of the first node using the proposed algorithm is at round 928, LEACH is at round 853, and LEACH-GA is at round 874. So EHHA beats LEACH-GA by 54 rounds. At round 1500, the number of alive nodes using HHA and EHHA is 19, the number of alive nodes using LEACH is 1, and the number of alive nodes using LEACH-GA is 4. So EHHA has more alive nodes than LEACH and LEACH-GA for transmitting data to the BS. The result of EHHA is the same as that of HHA in this situation.

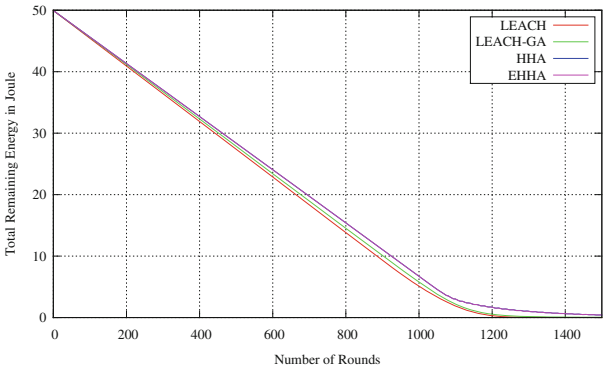

Fig. 3. Total remaining energy for the 100 sensors and 100 m × 100 m area case with BS located at (50, 50).

Figure 3 further shows the remaining energies for the 100 sensors and 100 m × 100 m area case with BS located in the middle, i.e., at (50, 50), of the WSN. The results of HHA and EHHA are similar. It can be easily seen that the remaining energy of HHA and EHHA is pretty much the same as that of LEACH and LEACH-GA, but the number of alive nodes of HHA and EHHA are more than the other approaches. According to our observation, this is because the proposed method takes into account the residual energy of nodes to decide which node will be used for long distance transmission and which node will be used for short distance transmission.

Tables 3 and 4 compare the proposed algorithm with the other clustering algorithms evaluated in terms of the number of alive nodes and the remaining energy. The results show that the proposed algorithm provides a better solution for the CHEP than the other clustering algorithms in most cases, especially for large and complex problems. When we use the number of alive nodes as a

Table 3. Results in terms of the number of alive nodes.

100 sensors, 100 m × 100 m, BS located at (50, 50).					500 sensors, 1000 m × 1000 m, BS located at (500, 500).				
Number of Rounds	Algorithm				Number of Rounds	Algorithm			
	LEACH	LEACH-GA	HHA	EHHA		LEACH	LEACH-GA	HHA	EHHA
500	100.0	100.0	100.0	100.0	500	9.1	11.0	60.9	63.0
1000	84.4	95.9	99.9	99.8	1000	3.4	6.2	27.9	29.1
1500	1.2	4.2	19.6	19.1	1500	1.3	4.1	14.7	15.9
2000	0.2	1.5	6.1	6.1	2000	0.7	2.8	7.5	8.4
2000 sensors, 1000 m × 1000 m, BS located at (500, 500).					4000 sensors, 1000 m × 1000 m, BS located at (500, 500).				
Number of Rounds	Algorithm				Number of Rounds	Algorithm			
	LEACH	LEACH-GA	HHA	EHHA		LEACH	LEACH-GA	HHA	EHHA
500	237.2	61.1	265.3	293.2	500	553.6	636.8	575.0	612.2
1000	4.7	4.9	123.1	164.7	1000	80.1	13.9	317.7	369.2
1500	0.8	1.6	42.8	105.9	1500	1.0	1.2	179.1	267.7
2000	0.4	0.9	17.5	63.4	2000	0.4	0.7	101.6	190.6

Table 4. Results in terms of the remaining energy (Joule) of sensors.

100 sensors, 100 m × 100 m, BS located at (50, 50).					500 sensors, 1000 m × 1000 m, BS located at (500, 500).				
Number of Rounds	Algorithm				Number of Rounds	Algorithm			
	LEACH	LEACH-GA	HHA	EHHA		LEACH	LEACH-GA	HHA	EHHA
500	27.4749	27.9474	28.4656	28.4673	500	0.594	1.159	5.8031	6.214
1000	5.1717	5.8642	6.7989	6.8032	1000	0.1084	0.5003	1.8051	2.0112
1500	0.0069	0.0488	0.4267	0.4234	1500	0.0261	0.2649	0.702	0.8173
2000	0.0012	0.0139	0.0791	0.0756	2000	0.0095	0.162	0.292	0.3282
2000 sensors, 1000 m × 1000 m, BS located at (500, 500).					4000 sensors, 1000 m × 1000 m, BS located at (500, 500).				
Number of Rounds	Algorithm				Number of Rounds	Algorithm			
	LEACH	LEACH-GA	HHA	EHHA		LEACH	LEACH-GA	HHA	EHHA
500	34.9547	3.6478	27.0588	36.9235	500	97.1901	103.0499	68.7917	85.8229
1000	0.0534	0.0671	6.9501	14.0022	1000	1.584	0.1787	24.5036	36.8481
1500	0.0066	0.0178	1.7937	6.6724	1500	0.0083	0.0111	10.4879	20.0203
2000	0.0021	0.0078	0.5273	2.7428	2000	0.0034	0.005	4.1607	10.0464

measure, as shown in Table 3, the proposed algorithm can keep more alive nodes than the other clustering algorithms. In other words, the difference (i.e., the number of alive nodes) between the proposed algorithm and the others will be larger when we increase the number of sensors while making the region to be covered larger. For example, the last case of Table 4, 4000 sensors in a region of size 1000 m × 1000 m, EHHA has about 190.6 sensors alive but HHA has only 101.6 sensors alive after 2000 rounds. The results show that the difference between the proposed algorithm and the others will become larger and larger, as the number of sensors and the size of the region increase. This implies that the proposed algorithm is a more scalable clustering algorithm than the others. The results of Table 4 show that the difference between these clustering algorithms will also be enlarged for the remaining energy of sensors. Even though the remaining energy of all the sensors of the proposed algorithm is less than the other clustering algorithms for a small CHEP, e.g., 100 sensors for a region of size 100 m × 100 m. But the proposed algorithm can provide a better result when the problem becomes larger and more complex, e.g., 500 sensors for a region of size 1000 m × 1000 m. Moreover, in the case of 4000 sensors for 1000 m × 1000 m, the difference between the proposed algorithm and HHA is about 5.8857 (= 10.0464 − 4.1607) which explains that for a larger and more complex problem, the difference will be more significant.

5 Conclusion

In this paper, we presented an improved hyper-heuristic algorithm for solving the CHEP. The results show that not only can it provide a better result than traditional clustering algorithms, such as LEACH, it can also provide a better result than LEACH with GA and simple hyper-heuristic algorithm. The results further show that the proposed algorithm can prolong the lifetime of most sensors in a WSN by reducing the energy they consume. According to our observations, the results also imply that the proposed algorithm can select a more suitable set

of sensors as the CHs than the other clustering algorithms, so the worse data transmission paths will be reduced. Since the results explain that this hyper-heuristic algorithm has potential for the CHEP, we will continue to seek possible ways to improve the performance of this method in the future, especially on adding more LLHs and developing an effective way to determine the execution sequence of LLHs.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the paper. This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST104-2221-E-197-005 and MOST104-2221-E-110-014.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
2. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: a survey. Appl. Soft Comput. **11**(6), 4135–4151 (2011)
3. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001). doi:10.1007/3-540-44629-X_11
4. Harrop, P., Das, R.: Wireless sensor networks (WSN) 2014–2024: forecasts, technologies, players. Technical report, IDTechEx (2015). http://www.idtechex.com/research/reports/wireless-sensor-networks-wsn-2014-2024-forecasts-technologies-players-000382.asp?viewopt=orderinfo
5. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of Annual Hawaii International Conference on System Sciences, pp. 1–10 (2000)
6. Hoang, D., Yadav, P., Kumar, R., Panda, S.: A robust harmony search algorithm based clustering protocol for wireless sensor networks. In: Proceedings of IEEE International Conference on Communications Workshops, pp. 1–5 (2010)
7. Krishna, K., Murty, M.: Genetic k-means algorithm. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **29**(3), 433–439 (1999)
8. Kulkarni, R.V., Venayagamoorthy, G.K.: Particle swarm optimization in wireless-sensor networks: a brief survey. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **41**(2), 262–267 (2011)
9. Liu, J.L., Ravishankar, C.V.: LEACH-GA: genetic algorithm-based energy-efficient adaptive clustering protocol for wireless sensor networks. Int. J. Mach. Learn. Comput. **1**(1), 79–85 (2011)
10. Losilla, F., Garcia-Sanchez, A.J., Garcia-Sanchez, F., Garcia-Haro, J., Haas, Z.J.: A comprehensive approach to WSN-based ITS applications: a survey. Sensors **11**(11), 10220–10265 (2011)
11. Potdar, V., Sharif, A., Chang, E.: Wireless sensor networks: a survey. In: Proceedings of the International Conference on Advanced Information Networking and Applications Workshops, pp. 636–641 (2009)
12. Reese, L.: Industrial wireless sensor networks. Technical report, Mouser Electronics (2015). http://www.mouser.com/applications/rf-sensor-networks/

13. Sang, Y., Shen, H., Inoguchi, Y., Tan, Y., Xiong, N.: Secure data aggregation in wireless sensor networks: a survey. In: Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 315–320 (2006)
14. Tsai, C.W., Huang, W.C., Chiang, M.H., Chiang, M.C., Yang, C.S.: A hyper-heuristic scheduling algorithm for cloud. IEEE Trans. Cloud Comput. **2**(2), 236–250 (2014)
15. Tsai, C.W., Hong, T.P., Shiu, G.N.: Metaheuristics for the lifetime of WSN: a review. IEEE Sens. J. **16**(9), 2812–2831 (2016)