Computer Networks 000 (2016) 1–14

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An effective WSN deployment algorithm via search economics

Chun-Wei Tsai*

Department of Computer Science and Information Engineering, National Ilan University, Yilan 26047, Taiwan

ARTICLE INFO

Article history:
Received 31 July 2015
Revised 9 December 2015
Accepted 3 January 2016
Available online xxx

Keywords:
Metaheuristic algorithm
Wireless sensor network
Deployment problem

ABSTRACT

The development of a powerful search mechanism to find a good solution is the current research direction of studies on metaheuristic algorithms; however, most of the developed mechanisms will search and check the possible solutions without knowledge of the overall landscape of the solution space during the convergence process. To make each search during the convergence process as effective as possible, this paper presents a new metaheuristic algorithm called search economics (SE) to solve the deployment problem of wireless sensor networks. The main distinguishing features of the SE are twofold: the first is its capability to depict the solution space based on the solutions that have been checked by the search algorithm, and second is its capability to use the knowledge thus obtained, i.e., the "landscape of the solution space," during the search process. On the basis of these concepts, the investment in a search process will be more meaningful and thus less easy to fall into a local optimum during early iterations. The experimental results show that the proposed algorithm can provide a result for the deployment problem that is significantly better than those provided by the state-of-the-art metaheuristic algorithms evaluated in this study in terms of the quality.

1. Introduction

It has become a clear and definite goal of various organizations and companies to apply the internet of things (IoT) [1–5] to all kinds of systems, approaches, and environments that are encountered in our daily life. In addition, the IoT can be applied because its unlimited possibilities have been illustrated by several successful application domains such as smart homes, agriculture farms, weather reconnaissance, wearable computing, and industry management [6–10] Among them, the industrial internet of things (IIoT) [11] is a promising area of research, as it plays the roles of accelerating production procedures, creating new hybrid business models to enhance competitive power, and using intelligent technologies to improve the performance of a company. From the wireless sensor network (WSN) to the IoT, and then, to the IIoT, enhancing the performance of the whole system has been formulated as complex optimization problems that include coverage, lifetime, and deployment problems (DPs). The DP [12] is very important because its planning strategy will affect not only the coverage and lifetime of the whole system but also the transmission rate of the sensors. It is typically assumed that the solution to the DP is to place a given set of sensors in a certain area that would either maximize the sensing coverage and network connectivity or minimize the energy consumption [13,14].

A high-performance method for solving the DP of a WSN [15–19] is an important research issue because it will also be the foundation of the IoT and the IIoT. However, although random, grid-based, and deterministic methods have been widely applied to DPs [20], they cannot find an approximate or optimal solution within a reasonable time through using limited computation resources, as most DPs are NP-hard [21]. An alternative way to deal with such problems is to use a metaheuristic algorithm

* Tel.: +886 952685667.
E-mail address: cwtsai0807@gmail.com

http://dx.doi.org/10.1016/j.comnet.2016.01.005
1389-1286/

because it is able to find an approximate solution within a reasonable time compared to exhaustive search and deterministic algorithms. Several recent studies [12] have shown that simulated annealing (SA), tabu search (TS), and genetic algorithms (GAs) can be used to find a better result than rule-based algorithms for the DP of a WSN and the IoT. One of the important reasons is that the metaheuristic algorithm uses some stochastic procedures during the convergence process; thus, it can avoid falling into a local optimum during early iterations, i.e., the early stages of the search. However, in most cases, a metaheuristic algorithm may get stuck in the same region or on particular solutions for a long time when the search process approaches the convergence state, i.e., later stages of the search. These redundant searches will degrade the performance of the metaheuristic algorithm by wasting a large amount of invested computing resources. A good example is searching the same region several times.

According to our observation, most studies on metaheuristics are focused on how to enhance the performance of a search algorithm (e.g., reducing the computation time of the search or increasing the accuracy of the result); few are focused on how to depict the solution space. As a result, most metaheuristics have only short-term memory for the searched solution space, thus making them easily fall into a local optimum when the search diversity degrades to a certain degree. To make each search as meaningful as possible and not easily fall into a local optimum, this study presents a novel metaheuristic algorithm by using the information of the solution space, computing resources, and current solution to enhance the *value of each search*. The main contributions of this paper can be summarized as follows:

(1) The main search process is designed from scratch for distributed computing; thus, no global information needs to be kept. Such a design makes it easy to apply a metaheuristic algorithm to cloud computing. By using the information related to the computing resources and solution space, the search algorithm can then determine the direction or region that is worth being searched later, just like we have to understand the market circumstances before we invest. To achieve this goal, a new search algorithm, called vision search (VS), is presented in Section 2.4.
(2) The proposed algorithm will survey the solution space during the search process to provide information about the solution space that it has already checked for the search algorithm. To achieve this goal, a new mechanism for metaheuristics called marketing research (MR) is discussed in Section 2.5.
(3) A detailed description of how the proposed algorithm works for a DP is given in Section 3 to show the possibility of SE for real optimization problems.

The remainder of the paper is organized as follows. Section 2 presents the basic idea and details of the proposed algorithm. In addition, a simple example is presented in this section to explain how to use the proposed algorithm. Section 3 begins with a description of the simulation environment. Then, two different optimization problems are used to evaluate the performance of SE and other algorithms. Section 4 gives a brief review of metaheuristics to show the differences between the proposed algorithm and other metaheuristics. Finally, Section 5 draws the conclusions and gives some future research directions for this research.

2. Search economics

2.1. Notation

To simplify the discussion that follows, the following notation is used throughout the rest of the paper.

s, s_i	a set of investments (solutions) for the optimization problem in question, i.e., $s = \{s_1, s_2, \ldots, s_n\}$, where s_i is the investment of the i-th searcher, and n is the number of searchers and investments.
r, r_j	a set of regions in the market (solution space), i.e., $r = \{r_1, r_2, \ldots, r_h\}$, where r_j is the j-th region of the market, and h is the number of regions.
r_j^b	the best-so-far good (solution) of the region r_j.
m, m_{jk}	a set of goods (sampling nodes) in all regions in the market, i.e., $m = \{m_{11}, \ldots, m_{1w}, \ldots, m_{h1}, \ldots, m_{hw}\}$, where m_{jk} is the k-th good in the j-th region, and w is the number of goods in each region.
v^i, v^i_{jk}	a set of possible investments of the i-th searcher, which is defined as the crossover of s_i and all goods in m, i.e., $v^i = \{v^i_{jk}\}$, where $v^i_{jk} = s_i \otimes m_{jk}$, and $\otimes$ is the crossover operator.
e, e^i_j	a set of expected values; that is, e^i_j is the expected value for the i-th investment in the j-th region.
t^a_j	the number of times that the j-th region has been invested in (searched). Initially, $t^a_j = 1$; its value will be increased by 1 every time the j-th region is searched.
t^b_j	the number of times the j-th region has not been invested in (searched). Initially, $t^b_j = 1$; its value will be increased by 1 for every iteration that the j-th region is not searched but will be reset to 1 if it is searched.

Note that the solution space in this study is called the market, and the result of a search (i.e., the solution of a search) is referred to as an investment. Moreover, the solution space is divided into a set of subspaces called regions. On the basis of the concept of an investment, a searcher is an agent who determines which region is worth being invested in, i.e., worth being searched.

2.2. Concept

Most of the metaheuristics search for a solution without the information about the solution space, i.e., in a way similar to a traveler who does not carry a map. However, even without a map to guide the traveler in the right direction, he or she may still have a chance to reach the destination. However, the traveler may have a "very small chance" to reach the destination, especially when the geographical environment is complicated and large. A "very small chance" means that the event of reaching the destination is incredibly unlikely to happen. In a very rare case, the traveler may still reach the destination after many trials and errors. Nevertheless, we just do not know "how much of an unnecessary long way" the traveler has traveled. The traveler may *never* find the right way to reach the destination until he or she gives up. Similarly, we still have a chance to make money from a haphazard investment, even if we do not understand the market. The bottom line is that we do not waste too much money on an

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

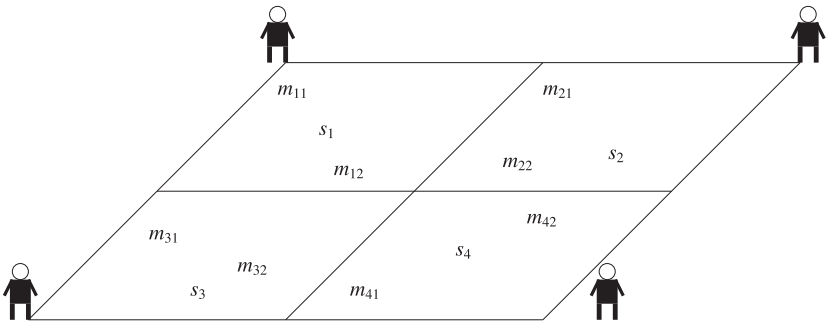

Fig. 1. The basic idea of SE.

```
1  SearchEconomics()
2  {
3     s = Initialization()
4     m = ResourceArrangement(s, r)
5     While the termination criterion is not met
6        s = VisionSearch(s, m)
7        m = MarketingResearch(s, m)
8     End
9     Output s
10 }
```

Fig. 2. Outline of the proposed algorithm.

investment, which can be regarded as the computing resource that has to be invested in the right region in the solution space to find the optimal solution of the optimization problem in question.

The basic idea of the proposed algorithm is that *you get whatever solution for which you actually pay*. In other words, the idea is to reduce the number of redundant searches in the search process, thus making each search of metaheuristics as meaningful as possible, instead of just a wild guess in the traditional way of searching. This is why a search of metaheuristics in the solution space is regarded as an investment in this study. This concept implies that how the search of metaheuristics is performed and how the candidate solutions are measured will be very different. Here are some of the concerns. The first is the important information for the search—the number of times a region is searched. This means that if the objective values of the solutions of two different regions are the same, their potentials may still be different because one region may have been searched many more times than other regions. This is just like mining. Spending 100 days finding gold that is worth 100 U.S. dollars in a region is very different from spending 1 h finding gold that is also worth 100 U.S. dollars in the same region. The second is that the regions that have not been searched for a long time will have a higher chance to find better solutions than the other regions that have been searched frequently. The third is how to make metaheuristics work in a parallel computing environment. Therefore, the parameters and information of the proposed algorithm cannot be centralized. In response to a parallel computing environment, where the computing resources (nodes) may be dynamically increased or decreased, the design of the proposed algorithm also takes into account this implementation issue. Each computer node will be assigned a searcher and region so that each computer node can be used to search for candidate solutions and record the searched information. This means that the proposed algorithm can increase the number of search directions and record more searched information as the number of computer nodes increases. By using this design, even though each computer node of SE does not have global information, the proposed algorithm can still be used to search for a solution, just like it has the global information because the searched information will be exchanged between searchers and regions.

As shown in Fig. 1, we assume that there are four computer nodes (resources), and the proposed algorithm will create four searchers, each of which are associated with an investment s_i (i.e., a solution). The solution space (called the market in this study) will then be divided into four regions, each of which have two goods m_{jk} (i.e., candidate solutions) to depict the geography of the region to which they belong. On the basis of this concept, when the proposed algorithm gets one more computer node, it can split one of the regions that has a high potential to find a better result into two regions to effectively use the additional computing resources. On the other hand, SE can merge two regions into one region if there is a shortage of computing resources. These two strategies mean that the proposed algorithm is very suitable for the environments of parallel computing and cloud computing. Moreover, the goods of SE are used to draw the geography of the subsolution space, just like the sampling points. With the information of goods (e.g., the sampling nodes of each region), the searchers of SE own more information than other metaheuristics for solving the optimization problem.

Fig. 2 shows that the proposed algorithm consists of three main operators. The ResourceArrangement (RA) operator plays the role of assigning the searchers to the regions of the market, the VisionSearch (VS) operator plays the role of searching, and the MarketingResearch (MR) operator plays the role of keeping track of the information of each region for the search operators.

2.3. Resource arrangement

As shown in Fig. 3, the RA operator) divides the market (the solution space) into h regions. After that, RA will

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

```
ResourceArrangment(s, r)
{
    r = Divide the solution into h regions
    For j = 1 to h
        For k = 1 to w
            m_jk = Random(r_j)
        End
        r_j^b = Best(m_j)
    End
    For i = 1 to n
        s_i = Assign(m)
    End
}
```

Fig. 3. Outline of the resource arrangement operator.

```
VisionSearch(s, m)
{
    v = Transition(s, m)
    e = ExpectedValue(v, m, t^a, t^b)
    s = Determination(v, e)
}
```

Fig. 4. Outline of the vision search operator.

first create w possible goods (candidate solutions) for each region r_j—all randomly—and then find the best good r_j^b of each region r_j. The i-th searcher will randomly choose a good in the i-th region to invest in, which means that the i-th searcher will be assigned to the i-th region if $n = h$ at the very beginning of the search; otherwise, some of the regions may be randomly assigned either more than one searcher or none at all.

2.4. Vision search

As shown in Fig. 4, the VS operator consists of three suboperators to transit, evaluate, and determine the solution. Basically, this operator is similar to a traditional metaheuristic algorithm, implying that it can be replaced by a traditional metaheuristic algorithm as long as the evaluation and determination operators of the traditional metaheuristic algorithm can be adapted to fit with the spirit of the proposed algorithm. The *transition operator* is similar to that of the genetic algorithm for exchanging information between solutions; that is, in this study, the transition operator consists of the crossover and mutation operators of the GA. Unlike the genetic algorithm, in which the information exchange is between chromosomes in the same population, the information exchange in the proposed algorithm is between the "investment" of the searchers and the "goods" of the regions. Since there are n searchers in total and w goods (candidate solutions) in each region, the transition operator will create a set of temporary candidate solutions v, where v_{jk}^i is obtained by exchanging the information between the investment of the i-th searcher s_i and the goods in all regions m, i.e., all of the goods in the market.

For the evaluation operator of SE, the fitness of the good of the i-th searcher at the j-th region is evaluated in terms of the expected value e_j^i by using the ExpectedValue operator, unlike traditional metaheuristics, in which the fitness of a solution is evaluated in terms of the fitness or objective value. The main difference between the evaluation operator of SE and the evaluation operator of other metaheuristics is that it uses information about the number of iterations to search for the same region, how long the region has not been searched, the objective values of the solutions, and the objective values of the possible solutions to evaluate the relative quality of the solutions. The expected value of the ExpectedValue operator is defined by

$$e_j^i = T_j V_j^i M_j. \tag{1}$$

In brief, Eq. (1) says that the expected value e_j^i consists of three pieces of information, each of which is described below.

(1) The status of the investment in the j-th region is defined as

$$T_j = \frac{t_j^b}{t_j^a}, \tag{2}$$

which is a measure of the return of an investment in a particular region of the market. In other words, it aims to reduce the number of redundant searches in a region or to avoid falling into a local optimum for a long time.

(2) The objective value of the i-th searcher is defined as

$$V_j^i = \frac{\sum_{k=1}^{w} f(v_{jk}^i)}{w}, \tag{3}$$

which is a measure of the potential of the investment of the i-th searcher in the j-th region based on the temporary candidate solutions in v.

(3) The proportion or weight of the objective value of the best-so-far solution in each region of the market is defined as

$$M_j = \frac{f(r_j^b)}{\sum_{j=1}^{h} f(m_j)}, \tag{4}$$

where r_j^b is the best-so-far solution of the j-th region, as defined in Section 2.1.

The *determination operator* of SE is the tournament operator of the GA but with a minor change. That is, the i-th searcher has to decide which region to invest in, although it is initially associated with the i-th region. Since the proposed algorithm is designed from scratch to not centralize all of the information, in addition to v_{ii}, the determination operator will also randomly choose some of the values of v_{ij} for $i \neq j$ to enter the tournament process and then choose the best one to be the investment of the i-th searcher.

2.5. Marketing research

As shown in Fig. 5, the Marketing Research (MR) operator contains two kinds of operations: (1) update the market, i.e., keep the information of the solutions that have been checked, and (2) accumulate the values of t^a and t^b. For the Update(s, m) operator, the main idea is to record everything about the solutions that have been checked to improve the search performance. However, there is simply no way to save all of the searched solutions, i.e., the search

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

```
1 MarketingResearch(s, m)
2 {
3     m = Update(s, m)
4     t^a = Accumulation1(s, m)
5     t^b = Accumulation2(s, m)
6 }
```

Fig. 5. Outline of the marketing research operator.

history, because no matter how large it is, memory space is limited. The trade-off of the update operator is to save as much of the information of the searched solutions (investments) as it can to achieve this goal. In this study, we try to keep k goods (candidate solutions) in each region. When finding a better good in region r_j (i.e., there exists a v^i_{jk} such that $v^i_{jk} > r^b_j$), the update operator will use this good to replace one of the current goods in the same region. One possible solution to this problem is to compress the searched solutions into a single solution, e.g., use the mean of the searched solutions to represent the searched solutions. For example, pattern reduction [22] can be used to detect and use the common structure of the subsolutions in a region to represent a set of searched solutions. In addition to relying on compression or geoinformatics techniques to keep track of the information of searched solutions, random sampling is a makeshift method to use before we find a novel method to keep track of all of the information of the search process of SE.

The other important operation of MR is the accumulation operator. As shown in Fig. 5, t^a_j denotes how many times the j-th region has been invested in (searched), whereas t^b_j denotes how many times the j-th region has not been invested in. If a searcher invests in the j-th region, then t^b_j will be set to 1, and t^a_j is set to $t^a_j + 1$ (i.e., $t^a_j = t^a_j + 1$). Then, SE can use Eq. (2) to measure the potential of each region. For example, if the objective value of good (solution) m_{11} is equal to the objective value of good m_{21} but $T_1 > T_2$, then it implies that good m_{11} has a higher potential than good m_{21} because either SE invests *more* computing resources searching region 2 than searching region 1 (i.e., $t^a_1 < t^a_2$) or SE invests *fewer* computing resources searching region 1 than searching region 2 (i.e., $t^b_1 > t^b_2$). On the basis of this concept, SE can avoid searching a particular region in the search space; rather, it will search more regions that have a higher potential to find a better solution.

2.6. Summary

It can be easily understood that the basic idea of SE is to use the "limited computing resources" to search for a solution in a "huge solution space." Since the computing resources we have are generally less than the computation costs we have to spend if we want to check every possible solution in the solution space, the priority is then to make each search as meaningful as possible. This means that although the basic idea of a metaheuristic algorithm is a strategic guess, we have to figure out how to avoid duplicate and irrelevant guesses (i.e., searches). This is something similar to good marketing research that can

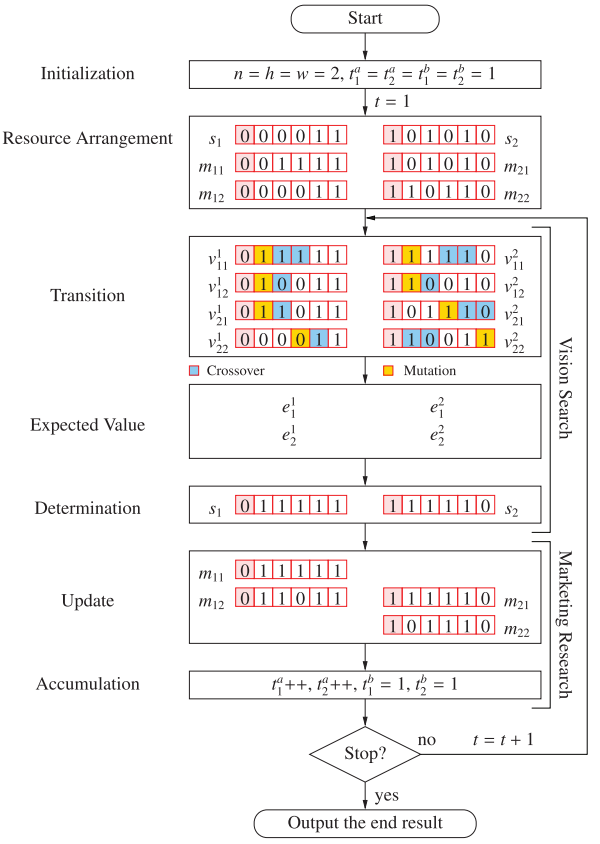

Fig. 6. A simple example illustrating how SE works.

help us increase the successful rate of investment. Further, this means that SE has to know the reasons for searching a particular region. In summary, SE will first map the solution space (i.e., the marketing research of SE) and then search regions in the solution space that have a higher potential than the other regions to make a search as meaningful as possible. This is the main idea of the proposed algorithm, which aims to find a solution with limited resources. If SE gets more computation resources during the convergence process, it will invest the resources to search regions that have a higher potential than the other regions by using Eq. (1).

A simple example is shown in Fig. 6 to illustrate how SE, which consists of the initialization, RA, VS, and MR operators, works. In this example, the parameters n, h, and w are set to 2; the parameters t^a_1, t^a_2, t^b_1, and t^b_2 are set to 1 initially. Because h and w are set to 2 (i.e., two searchers, each of which have two goods), eight temporary solutions are created, but only four v^i_{jk} will be retained to represent the possible investments of each searcher s_i of the transition operator. More precisely, each searcher s_i will perform the crossover operator four times; the first two times are with m_{11} and m_{12}, whereas the other two times are with m_{21} and m_{22}. This will create eight temporary solutions. In order to reduce the number of solutions that has to be kept in memory, the proposed algorithm only retains the better ones from each crossover procedure. This means

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

that SE will keep only the best solution for each crossover or information exchange procedure. This is why there are only four occurrences of v_{jk}^1 in the rectangle of the transition operator in Fig. 6. In addition to using the crossover operator to create the temporary solutions, the mutation operator will then be used to perturb these temporary solutions. The expected value e_j^i of each searcher s_i for the j-th region will be computed and then used by the determination operator of SE to estimate the potentials of the possible search directions. For example, the proposed algorithm will then try to find the best of v_{21}^2 and v_{22}^2 to replace s_2 if $e_2^2 \geq e_1^2$. In brief, s_1 and s_2 will be updated by the determination operator. This example shows that VS consists of the transition, expected value, and determination operators, whereas MR consists of the update and accumulation operators. VS and MR will be performed at each iteration of SE. After performing VS, SE will then update m_{11}, m_{12}, m_{21}, and m_{22} if a better result than m_{ik} is found by s_i as well as t_1^a, t_2^a, t_1^b, and t_2^b. In summary, VS plays the role of "searching" for the solution while MR plays the role of "surveying" the solution space during the convergence process.

3. Results

An empirical analysis is conducted on an IBM X3400 machine with a 2.0 GHz Xeon CPU and 8GB of memory running CentOS 5.0 with Linux 2.6.18, and the programs are written in C++ and compiled using g++. The OneMax and deployment problems are used to show the performance of the proposed algorithm. The solution representation and the objective function of these two problems will also be given to show how to apply the proposed algorithm to optimization problems. The source code of SE will be available at https://sites.google.com/site/cwtsai0807/search-economics.

3.1. OneMax problem

The problem used to evaluate the performance of the proposed algorithm is the OneMax problem [23]. It is an optimization problem with the goal of maximizing the number of ones in a bit string. That is, it is defined as follows:

Definition 1. Given a string $x = \langle x_1, x_2, \ldots, x_N \rangle$, where N is the length of the string x, and $x_i \in \{0, 1\}$, the goal is to maximize

$$F(x) = \sum_{i=1}^{N} x_i. \tag{5}$$

Apparently, the optimal solution of this problem is a bit string with all ones. For instance, if $N = 5$, then the optimal solution is $x = \langle 1, 1, 1, 1, 1 \rangle$. The OneMax problem is used as the touchstone of the proposed algorithm SE, as it can be easily transformed into other optimization problems. For example, for the bin-packing and OneMax problems, the main difference is $F(x)$. Thus, all we have to do is to change $F(x)$ to compute the profit of a bin. This is why we choose the OneMax problem—to make the description of SE as easy to understand as we can.

Table 1
Dataset for the OneMax problem.

Dataset	# of bits N
DSO-1	10
DSO-2	50
DSO-3	100
DSO-4	500
DSO-5	1000

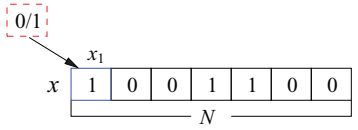

Fig. 7. How the OneMax problem is represented in SE.

3.1.1. Data sets and parameter settings

As shown in Table 1, five datasets with $N = 10$ up to $N = 1000$ are used to evaluate the performance of the proposed algorithm for the OneMax problem.

In this study, we compare the performance of SA [24], a GA [25], and SE. The parameter settings of SA are as follows. The number of neighbors is set to 7. The parameter settings of the GA are as follows. The population size is set to 8, the crossover rate is set to 0.8, and the mutation rate is set to 0.001. Tournament selection and the one-point crossover are used as the determination and transition operators. The parameter settings of SE are as follows. The number of searchers n is set to 4, the population size h is set to 4, the number of possible goods (the number of samples of each region) w is set to 2, $t_j^a = 1$, and $t_j^b = 1$, initially, i.e., at the first iteration. Note that for the OneMax problem, we assume that $n = h$ in this study. Note that because the OneMax problem is not NP-hard, random sampling is used as the update operator of the MR of the proposed algorithm. Further, some additional checks are added to the determination of the VS; that is, the searcher chooses the new investment if it is better than the current investment. Each experiment is carried out for 30 runs, and the number of iterations for each run is set to 1000. All of the experimental results shown are the average of 30 runs.

3.1.2. Representation of SE for the OneMax problem

Since OneMax is a simple optimization problem, most of the search procedure of SE can be referred to in Section 2. The most important things we need to take into account are nothing more than how to encode each solution of search economics, how to divide the solution space into regions, and what attention the searchers of SE have to pay for.

As shown in Fig. 7, for the OneMax problem, the encoding allows each solution x of SE to be divided into N subsolutions, each of which assume a value of 0 or 1. In this study, h is set to 4, meaning that the solution space will be divided into four different subspaces. One way to do this is to fix the value of the first two bits (i.e., $2 = \log_2(4)$) for each solution. For $N = 7$, this means that the first region r_1 will be $\{00XXXXX\}$, r_2 will be $\{01XXXXX\}$, r_3 will be

Table 2
Comparison of the quality.

Dataset	SA	GA1	GA2	SE
DSO-1	**100.0%**	**100.0%**	**100.0%**	**100.0%** (0.00)
DSO-2	99.2%	97.9%	99.3%	**100.0%** (0.00)
DSO-3	96.5%	93.7%	96.3%	**100.0%** (0.00)
DSO-4	91.6%	72.6%	80.9%	**98.6%** (0.70)
DSO-5	89.8%	63.3%	70.2%	**93.8%** (1.63)

{10*XXXXX*}, and r_4 will be {11*XXXXX*}, where X is assumed to take a value of 0 or 1. Each search of r_j can only change the last five bits to find the possible solutions. When the searchers in different regions want to exchange their solutions, only the last five bits will be touched. That is, the first two bits will remain intact. The only exception is that the first two bits will be changed during the convergence process when a searcher moves to another region to search for a possible solution. In this case, the first two bits of its searched solutions (investments) will be updated accordingly, i.e., to the particular value of the first two bits of the new region.

3.1.3. Experimental results

As shown in Table 2, SE provides better results than the other metaheuristics compared in this study. The results of the simple GA are denoted GA1 and GA2 in Table 2. For GA1, the number of chromosomes is set to 8, the crossover rate is set to 0.8, and the mutation rate is set to 0.1. For GA2, the number of chromosomes is set to 20, the crossover rate is set to 0.8, and the mutation rate is set to 0.2. Although SA gives better results than GA1, it does not mean that the performance of GA is worse than that of SA. According to our observation, this is because the performance of the GA might be affected by several factors. In addition to the population size, crossover rate, mutation rate, crossover operator, mutation operator, and selection operators, even the way the fitness value is computed will have an impact on the final results of the GA. These results demonstrate that the fine-tuning ability (i.e., the local search ability) of SA for a simple optimization problem is better than that of a simple GA.

The results in Table 2 show that the proposed algorithm can provide better results than the other two metaheuristics compared in this study. Note that the values in parentheses in this table represent the standard deviation of the results of 30 runs when applying SE to the dataset. According to our observations, the proposed algorithm can prevent the search diversity from decreasing or moving towards particular search directions because the search directions depend not only on the objective or fitness values but also on the information of the solution space that is also taken into account by SE. The results in Table 2 also show that the proposed algorithm provides results that are close to the optimal solution using only a simple transition operator to exchange information. This means that the search performance of the proposed algorithm can be enhanced by using the transition and determination operators of different metaheuristics.

Fig. 8 shows the convergence of SE for three OneMax problems (problems with $N = 10$, 100, and 1000). Because the initial solutions are randomly generated, the distance between the solution found by SE to the optimal solution starts at about 0.5, whereas 1.0 indicates that the solution found by SE is exactly the same as the optimal solution (i.e., 100%). From these results, it can be easily seen that the proposed algorithm can find an approximate solution or even the optimal solution if we invest enough computing resources. For example, for the OneMax problems with $N = 10$ and $N = 100$, the proposed algorithm can find the optimal solutions at iterations 20 and 570, respectively. Although SE cannot find the optimal solution of the problem with $N = 1000$ within 1000 iterations, the trends show that the results might be improved with further iterations.

This is why we conducted another simulation to better understand the performance of SE. Fig. 9 shows that the proposed algorithm is almost capable of finding the optimal solution after 4895 iterations on average. As a matter of fact, there is no guarantee that SE will find the optimal solution in every run, but the results show that it

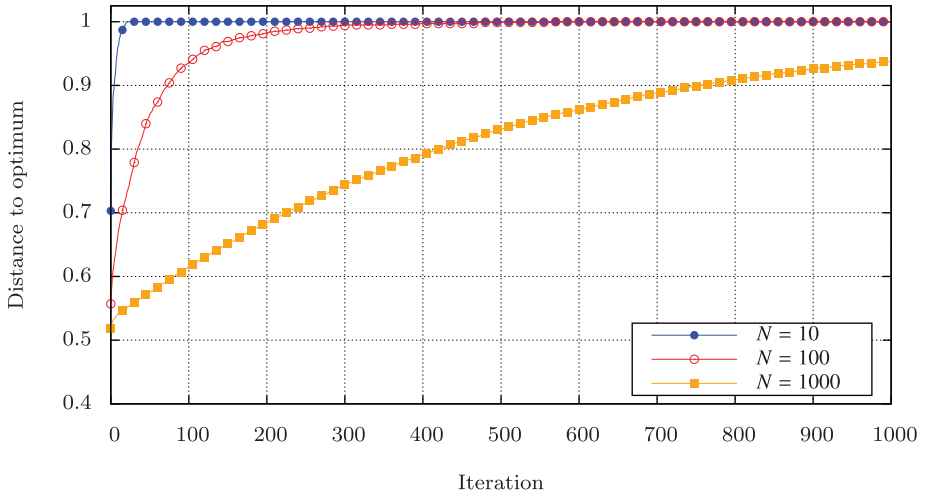

Fig. 8. The convergence of SE for the OneMax problem.

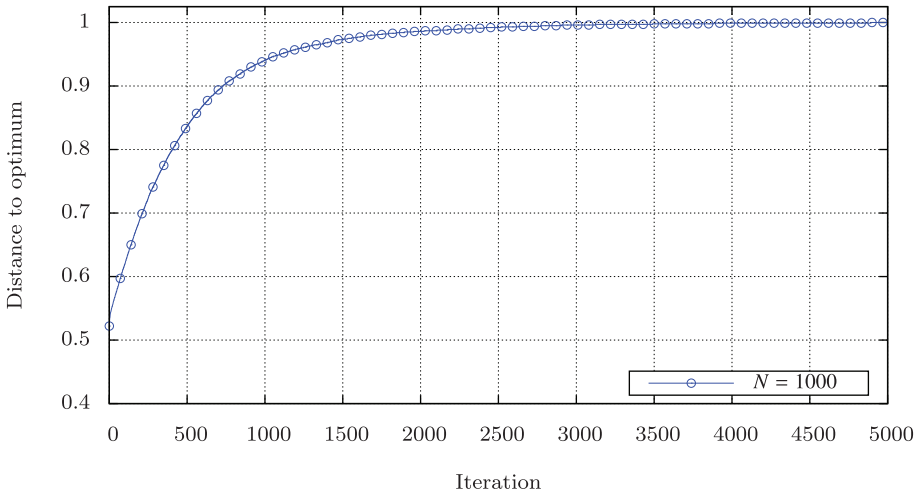

Fig. 9. The convergence of SE for the OneMax problem.

outperforms the other metaheuristics compared in this study. These results also show a distinguishing feature of SE; that is, SE does not quickly converge to a local optimum. The reason is that the search diversity will not quickly degrade, even at the later iterations of the convergence process. That is, each region still has a good chance to be searched, whereas the other metaheuristics may have already converged to particular regions. However, this does not mean that SE can use only this kind of transition operator; rather, SE can actually use other transition operators to enhance its search performance for the optimization problem.

3.2. Deployment problem

To evaluate the performance of the SE for a DP, a simple definition of a DP is to maximize the coverage of all sensors, which can be defined as follows:

Definition 2. Given a set of wireless sensors $x = \{x_1, x_2, \ldots, x_\varphi\}$, a set of targets $g = \{t_1, t_2, \ldots, t_\varepsilon\}$, and a set of candidate positions $p = \{p_1, p_2, \ldots, p_\omega\}$, where $\varphi \leq \omega$, each sensor x_i will be deployed to a candidate position p_j. A solution to the DP is a set of positions $y = \{y_1, y_2, \ldots, y_\varphi\}$ to which the set of given sensors are deployed. The overall coverage rate of a sensor for the targets is defined as

$$c = \frac{\sum_{i=1}^{\varepsilon} c_i}{\varepsilon} \times 100\%, \tag{6}$$

and

$$c_i = \begin{cases} 1, & \text{if there exists } y_j \text{ such that } d(y_j, t_i) < r, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where $d(a, b)$ is the distance between a and b, and r is the radius of coverage of a sensor.

In short, the search algorithm is used to find an approximate solution to deploy the sensors to the appropriate positions to cover as many targets as it can. The solution s found by the search algorithm can be regarded as the solution y of the DP. Therefore, we can easily evaluate the results of the search algorithm.

Table 3
Dataset for the DP.

Dataset	φ	ε
DSD-1	30	100
DSD-2	30	250
DSD-3	30	500
DSD-4	10	100
DSD-5	20	100
DSD-6	40	100
DSD-7	50	100

3.2.1. Data sets and parameter settings

As shown in Table 3, seven datasets are used to evaluate the performance of the proposed algorithm for the DP. Each dataset has φ sensors to be deployed for ε targets spread in an area with a size of 10 by 10. This area is divided into a grid of squares, each of which has a size of 1 by 1, and the possible sensor deployment positions are the corners of each square. This implies that there are total of $11 \times 11 = 121$ possible positions (i.e., $\omega = 11 \times 11 = 121$). Moreover, the radius of coverage of each sensor is set to 1. Fig. 10 shows the three kinds of distributions of targets to be used in this study. More precisely, Fig. 10(a) shows the distribution to be used for DSD-1, DSD-4, DSD-5, DSD-6, and DSD-7; Fig. 10(b) shows the distribution to be used for DSD-2; and Fig. 10(c) shows the distribution to be used for DSD-3.

In this section, we compare the performance of a simple genetic algorithm (GA) [25] and SE for the DP. The parameter settings of the GA are as follows. The population size is set to 8, the crossover rate is set to 0.8, and the mutation rate is set to 0.2. Tournament selection and the one-point crossover are used as the determination and transition operators. The parameter settings of SE are as follows. The number of searchers n is set to 8, the number

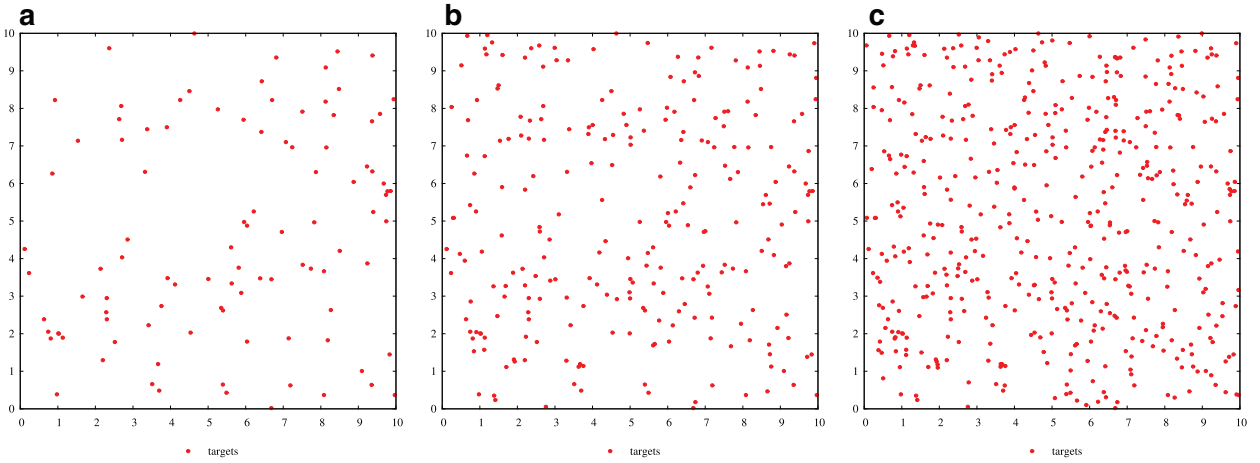

Fig. 10. The distribution of targets. (a) 100 targets, (b) 250 targets, and (c) 500 targets.

of possible goods w is set to 2, $t_j^a = 1$, and $t_j^b = 1$ initially. Note that for the DP in this study, we further assume that $n = h$. The main search procedure of SE for the DP is essentially the same as that for the OneMax problem, except for the representation and objective function to be discussed in Section 3.2.2. Similar to the simulation of OneMax problem, each experiment is carried out for 30 runs, but the number of iterations for each run is set to 500. All of the experimental results shown are the average of 30 runs.

3.2.2. Representation of SE for the DP

For the DP, the encoding of each solution x of SE can be divided into ω subsolutions, each of which assumes a value of 0 or 1, which is similar to SE for the OneMax problem. As shown in Fig. 12, h is set to 3, meaning that the solution space will be divided into three disjoint subspaces, and ω is set to 9. The value of each subsolution p_i indicates whether or not a sensor will be deployed at this position, with 1 meaning that a sensor will be deployed at this position and 0 meaning that no sensor will be deployed at this position. In this example, a sensor will be deployed at positions p_1, p_4, p_5, and p_9, but no sensors will be deployed at positions p_2, p_3, p_6, p_7, and p_8.

For the RA operator, SE will use a different way to divide the solution space into subspaces, which is different from that of SE for the OneMax problem. As shown in Fig. 12, SE will first divide the candidate positions into a certain number of subsets, e.g., $\lfloor \log_2(\omega) \rfloor = 3$ subsets, each of which correspond to a region. In this example, the subsets of the candidate solutions (the regions) are $r_1 = \{p_1, p_2, p_3, X, X, X, X, X, X\}$, $r_2 = \{X, X, X, p_4, p_5, p_6, X, X, X\}$, and $r3 = \{X, X, X, X, X, X, p_7, p_8, p_9\}$, where X takes a value of 0 or 1. At the initialization step, at least one sensor will be randomly deployed to a position in the j-th subset of the candidate positions (i.e., to a position in the r_j region) first, and then, SE will randomly deploy the remaining sensors to the other candidate positions of this region at which no sensor has been deployed.

By using this concept, we can make each searcher search the possibilities of its region. To ensure that each region is searched by at least a searcher, the double-check procedure has to make sure that at least one sensor is deployed in each region (i.e., each subset of candidate positions). For illustration, let us assume that the goal of the DP is to search for an optimal solution for the deployment of four sensors in a predefined space. For region r_1, SE will first deploy a sensor at p_1, p_2, or p_3 and then deploy the remaining three sensors to the other positions at which no sensor has been deployed. For region r_1, SE will use the double-check procedure to avoid the case of no sensors being deployed at the positions in the subset $\{p_1, p_2, p_3\}$. Except for the representation of the solution, solution-space division method, double-check procedure, and objective function,[1] the search procedure of SE for the DP is similar to that for the OneMax problem, meaning that the only things that need to be changed are those mentioned above.

Table 4
Results of the coverage rate of the GA and SE for the DP.

Dataset	# of targets	GA (8)	GA (16)	SE
DSD-1	100	87.9%	88.5%	**94.1**% (1.31)
DSD-2	250	81.5%	82.2%	**87.7**% (1.64)
DSD-3	500	79.1%	79.1%	**84.2**% (1.26)

3.2.3. Experimental results

As summarized in Table 4, the deployment results of the GA and SE after 500 iterations can cover most of the targets. More precisely, GA (8) and GA (16) in Table 4 indicate that the population sizes of the GA are equal to 8 and 16, respectively. The main difference between DSD-1, DSD-2, and DSD-3 is the number of targets (i.e, 100, 250, and 500, respectively). The results in Table 4 show that SE can find better results than the GA. Even after increasing the population size of the GA from 8 to 16, the proposed algorithm still provides a better result than the GA, and the difference is more than 5.1%. These results show that the

[1] Note that the OneMax problem uses Eq. (5) as the objective function, whereas the deployment problems uses Eq. (6) as the objective function.

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

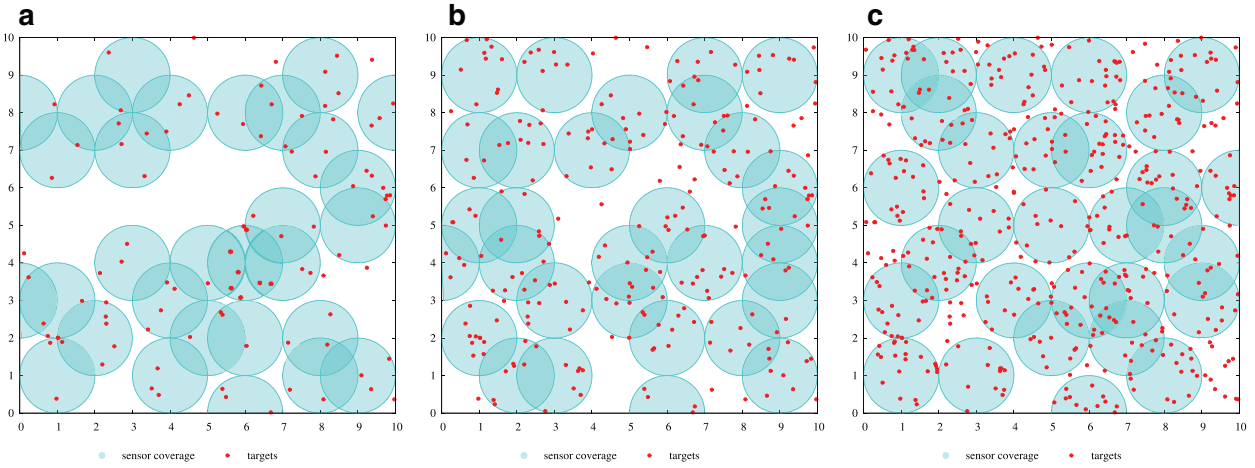

Fig. 11. The deployment results of SE after 1000 iterations. (a) 100 targets, (b) 250 targets, and (c) 500 targets.

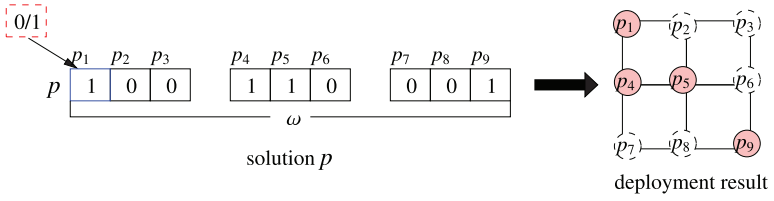

Fig. 12. Representation of SE for the DP.

Table 5
Coverage rate of SE for the DP for different numbers of sensors.

Dataset	# of sensors	SE
DSD-4	10	53.9% (1.19)
DSD-5	20	80.5% (1.80)
DSD-1	30	94.1% (1.31)
DSD-6	40	100.0% (0.00)
DSD-7	50	100.0% (0.00)

Table 6
Results of the GA and SE for the DP for different numbers of iterations.

Dataset	# of iterations	GA (8)	GA (16)	SE
DSD-1	500	87.9%	88.5%	**94.1%** (1.31)
DSD-1	10,000	92.4%	92.6%	**96.7%** (1.00)
DSD-3	500	79.1%	79.1%	**84.2%** (1.26)
DSD-3	10,000	82.7%	82.9%	**87.8%** (0.70)

proposed algorithm possibly provides better results than the GA or other metaheuristics. The results in Fig. 11 show that some of the targets cannot be covered by increasing the number of targets; that is, not all of the holes and corners are covered by all the sensors. Two possible solutions for increasing the coverage rate of the targets are by (a) increasing the number of sensors so that much more space can be covered and (b) improving the deployment results of the search algorithm by using the same number of sensors.

The results in Table 5 for the use of 10–50 sensors to cover 100 targets further show that the number of sensors has a strong impact on the coverage rate. By using DSD-1 as the baseline, the coverage rate normally can be significantly improved if additional sensors are deployed for the same number of targets, as the results of DSD-6 and DSD-7 show. On the contrary, the coverage rate will decrease as the number of sensors decreases. These results demonstrate that increasing the number of sensors is an effective way to increase the coverage rate of a WSN, the IoT, and even the IIoT.

Although the results in Table 5 show that the coverage rate can be improved by increasing the number of sensors, it will also increase the overall investment costs. A solution to this problem is to make the search algorithm find a better deployment result. Typically, the final results of most metaheuristics cannot be improved once the search process approaches a stable state (i.e., converges to a local optimum), even though more computing resources are invested.

The results in Table 6 show that the proposed algorithm is able to improve the final deployment results by investing more computing resources. For example, the coverage rate of SE is 94.1% after 500 iterations, but it can be improved to 96.7% by increasing the number of iterations, i.e., 10,000 iterations in the case. These results show that the proposed algorithm can also provide a better result than the GA, as the number of iterations is increased to 10,000. These results imply that the proposed algorithm will not easily fall into a local optimum, even if the search result is quite close to the global optimum, and more computing resources will be useful for improving the final results. Unlike other metaheuristics that may easily

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

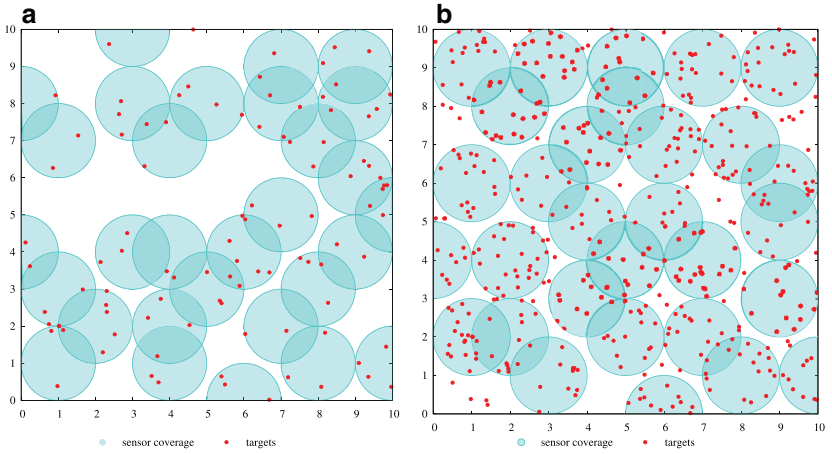

Fig. 13. The deployment results of SE after 10,000 iterations. (a) 100 targets and (b) 500 targets.

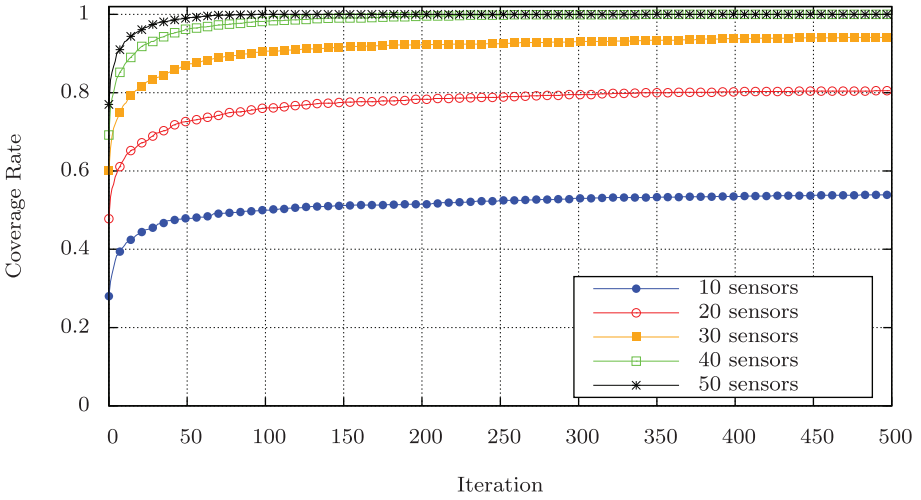

Fig. 14. The convergence of SE by using different numbers of sensors (i.e., 10–50 sensors) for the 100 targets of the DP.

get stuck at some solution (e.g., an approximate solution), SE will continue to improve the search result if more computing resources are invested. With this search tool at hand, we can then easily determine how many computing resources to invest to find the "appropriate result."

The deployment results in Fig. 13 for 100 and 500 targets after 500 and 10,000 iterations show that the deployed sensors will attempt to fit the distribution of the targets; thus, the total overlap in coverage between different sensors is less compared to the results in Fig. 11(a) and (c). It can be easily seen from these results that SE is able to find a "good solution" to the DP for different target distributions. Fig. 14 shows the convergence of SE using different numbers of sensors for 30 targets, which illustrates that the search performance of the proposed algorithm is quite stable for different situations or problems.

Fig. 15 shows the convergence results of the proposed algorithm when deploying 30 sensors to cover 100 and 500 targets. It can be easily seen from these results that if we can invest more computing resources, e.g., performing SE for more iterations, the search results can typically be improved. For example, the blue line in Fig. 15 shows that the search result of SE for 100 targets can be continuously improved if additional iterations are invested. These results are similar to those of SE for the OneMax problem in the sense that before the optimal solution is found, better results are obtained after additional iterations. These results also demonstrate that the assumption of this study, "You get whatever solution for which you actually pay," can be realized by using SE to solve the DP.

4. Related work

The development of metaheuristics can be dated back to the 1950s or even earlier; the year 2000 can be imagined as a watershed of these search algorithms. Until now, finding an approximate or optimal solution within an reasonable time by using limited computing resources is still vital for metaheuristic algorithms. Some discussions of the variety of developmental trajectories for state-of-the-art metaheuristics presented before the year 2000 can be found in [26,27]. These algorithms can be divided into two

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

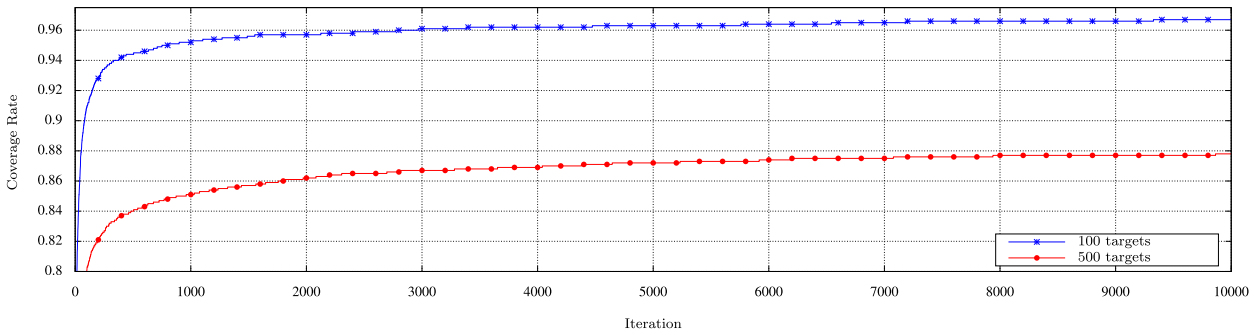

Fig. 15. The convergence of SE by using 30 sensors for 100 and 500 targets of the DP with 10,000 iterations.

groups by a well-known method, namely, single-solution-based algorithms (SSBAs) versus population-based algorithms (PBAs). The main difference is in the number of candidate solutions (directions) that is used at a time during the search process. SA [24] and TS [28] fall into the SSBA group, whereas the GA [25], ant colony optimization (ACO) [29], particle swarm optimization (PSO) [30], and differential evolution (DE) [31] are in the PBA group.

After the year 2000, many new metaheuristics were presented to solve complex optimization problems. Swarm intelligence has become an important research field because several new metaheuristics are inspired by the behavior of swarms [32–34]. In addition to the behavior of swarms, natural phenomena inspire other ways of thinking in the development of new metaheuristics, such as harmony search [35], the gravitational search algorithm [36], and coral-reef optimization [37]. In addition to the input and output operators, most metaheuristics contain four important operators [38]:

- *Initialization*: This operator is typically used to randomly create the initial solution for the convergence process. To enhance the search performance of a metaheuristic algorithm, some studies [39,40] have used a refinement procedure to get a better initial solution to start with.
- *Transition*: This operator usually plays the role of perturbing the current solution or exchanging subsolutions (information) between searched solutions.
- *Evaluation*: After the transition operator is performed, metaheuristics need to use an evaluation operator to evaluate the new solution.
- *Determination*: On the basis of the results of the evaluation operator, the determination operator can then be used to choose the search directions that would allow a good solution to be found at later iterations.

Except for the initialization operator that will be performed only once, transition, evaluation, and determination are the main operators that will be performed at each iteration. By using these operators to search for the solution, most metaheuristics will possess great intelligence, just like the observations of the convergence of a GA by the authors in [25,41], who say that

> *short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.*

These observations can also be used to say that most metaheuristics will have a chance to find a better solution gradually during the convergence process. Because the basic idea of metaheuristics is not to search for every possible solution in the solution space to find the global optimum, they may fall into a local optimum at early iterations for some optimization problems, which is usually referred to as the premature problem. To prevent the search process from getting stuck in a particular region of the solution space owing to a local optimum, a large number of studies attempted to develop an efficient method to deal with this problem. For example, allowing the search process to probe worse directions intermittently to avoid falling into a local optimum is used in SA, whereas the use of the mutation operator to perturb subsolutions (i.e., genes) is adopted by GAs to avoid searching for the same solutions.

In addition, some studies [28–30,35,42] attempted to use a different way to enhance the search performance of metaheuristics by keeping the search results for a certain number of iterations instead of just the solution of the current iteration. Two advantages can thus be easily imagined: the first one is that metaheuristics can avoid the premature problem, and the second one is that metaheuristics can use a better searched result to accelerate the convergence process. Now, let us look back on the study of Glover and Laguna [28]; the basic idea of TS is to keep some searched solutions to prevent the search process from searching the same solutions again and again in the near future. The way to keep the searched solutions to develop a high-performance metaheuristic algorithm, of course, can also be found in swarm intelligence [29,30]. For instance, ACO [29] uses the so-called pheromone table to accumulate and share the information of searched solutions along the way (i.e., from the initialization step to the current iteration), and PSO [30] uses the global best and personal best to keep the best-so-far solution of all of the particles and the best-so-far solution of each particle, respectively. In addition to the TS, ACO, and PSO, recent studies attempted to keep the structure of better searched solutions (e.g., promising candidate solutions)

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

such as the estimation of distribution algorithm (EDA) [42] and harmony search (HS) [35]. Both the EDA and HS attempted to build a probabilistic model to keep the structure of the better solutions they searched before and then use this probabilistic model to create a new solution.

Most metaheuristics can only keep some of the searched solutions because of the limited computing resources and memory space. This is why the length of the tabu list is usually set to a small number to keep only some searched solutions, and the EDA and HS use a probabilistic model to keep promising searched solutions. However, as we mentioned in Section 2.2, without knowledge of the solution space, the search process of a metaheuristic may converge to a local optimum, e.g., search the same regions again and again. This is why we need a new metaheuristic algorithm that is able to keep track of all of the searched solutions to avoid searching for redundant solutions (thus degrading the search diversity) and to search for solutions that have not been searched before to increase the search diversity.

5. Conclusion

This paper presents an effective method called SE to enhance the performance of metaheuristic algorithms by keeping track of the information collected from the search process, which includes not only the objective values of the candidate solutions but also the parts of the solution space that have been explored. The solution presented in this study is to depict the solution on the basis of the information we have in hand so that the search algorithm can dynamically invest computing resources in a region in the solution space that has a higher potential to find a better solution, meaning that SE will do its best to invest the computing resources it has in the regions where a better solution can be found. With the addition of computing resources (computer nodes) to the search process, the proposed algorithm will divide a region into two new regions by design and then assign the searcher belonging to the original region and a new searcher to these two new regions to achieve the goal of parallel computing on the fly.

The simulation results of the OneMax problem and DP show that the proposed algorithm is able to find better results than the other metaheuristics compared in this paper. Some of the simulations further show that the proposed algorithm is able to find the global optimum if we invest enough computing resources, which implies that SE will not easily fall into a local optimum. Although it is unable to store all of the searched solutions for the time being, SE now uses lossy compression to keep track of all of the searched solutions (i.e., not just the best-so-far solution but also the others). However, the ultimate goal of SE is to use lossless compression to keep track of all of the searched solutions. This is why this paper can be regarded as the starting point of this study. In addition to developing a better way to depict the solution space, the goal is to apply SE to other optimization problems in the future to demonstrate the performance of the proposed algorithm.

Acknowledgment

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 Contract MOST104-2221-E-197-005.

References

[1] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[2] D. Bandyopadhyay, J. Sen, Internet of things: applications and challenges in technology and standardization, Wirel. Pers. Commun. 58 (1) (2011) 49–69.
[3] D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac, Internet of things: vision, applications and research challenges, Ad Hoc Netw. 10 (7) (2012) 1497–1516.
[4] J. Wan, C. Zou, S. Ullah, C.F. Lai, M. Zhou, X. Wang, Cloud-enabled wireless body area networks for pervasive healthcare, IEEE Netw. 27 (5) (2013) 56–61.
[5] M. Chen, Y. Hao, Y. Li, C.F. Lai, D. Wu, On the computation offloading at ad hoc cloudlet: architecture and service modes, IEEE Commun. Mag. 53 (6) (2015) 18–24.
[6] D.J. Cook, M. Youngblood, E.O. Heierman III, K. Gopalratnam, S. Rao, A. Litvin, F. Khawaja, Mavhome: an agent-based smart home, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications, 2003, pp. 521–524.
[7] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[8] T. Bhattasali, R. Chaki, N. Chaki, Study of security issues in pervasive environment of next generation Internet of Things, in: Computer Information Systems and Industrial Management, vol. 8104, Springer, Berlin Heidelberg, 2013, pp. 206–217.
[9] M. Chen, Y. Zhang, Y. Li, S. Mao, V. Leung, EMC: Emotion-aware mobile cloud computing in 5G, IEEE Netw. 29 (2) (2015a) 32–38.
[10] M. Chen, Y. Zhang, Y. Li, M. Hassan, A. Alamri, AIWAC: affective interaction through wearable computing and cloud technology, IEEE Wirel. Commun. 22 (1) (2015b) 20–27.
[11] Accenture Technology, Driving Unconventional Growth Through the Industrial Internet of Things, 2015, Available online at https://www.accenture.com/us-en/_acnmedia/Accenture/next-gen/reassembling-industry/pdf/Accenture-Driving-Unconventional-Growth-through-IIoT.pdf, (accessed 03.12.15).
[12] C.W. Tsai, P.W. Tsai, J.S. Pan, H.C. Chao, Metaheuristics for the deployment problem of WSN: a review, Microprocess. Microsyst. 39 (8) (2015) 1305–1317.
[13] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (4) (2002) 393–422.
[14] C.W. Kang, J.H. Chen, An evolutionary approach for multi-objective 3D differentiated sensor network deployment, in: Proceedings of the International Conference on Computational Science and Engineering, vol. 1, 2009, pp. 187–193.
[15] Y. Zou, K. Chakrabarty, Sensor deployment and target localization in distributed sensor networks, ACM Trans. Embed. Comput. Syst. 3 (1) (2004) 61–91.
[16] A. Krause, C. Guestrin, A. Gupta, J. Kleinberg, Near-optimal sensor placements: maximizing information while minimizing communication cost, in: Proceedings of International Conference on Information Processing in Sensor Networks, 2006, pp. 2–10.
[17] J. Wu, S. Yang, Optimal movement-assisted sensor deployment and its extensions in wireless sensor networks, Simul. Model. Pract. Theory 15 (4) (2007) 383–399.
[18] M. Younis, K. Akkaya, Strategies and techniques for node placement in wireless sensor networks: a survey, Ad Hoc Netw. 6 (4) (2008) 621–655.
[19] B. Wang, H.B. Lim, D. Ma, A survey of movement strategies for improving network coverage in wireless sensor networks, Comput. Commun. 32 (13-14) (2009) 1427–1436.
[20] K. Xu, G. Takahara, H. Hassanein, On the robustness of grid-based deployment in wireless sensor networks, in: Proceedings of the International Conference on Wireless Communications and Mobile Computing, 2006, pp. 1183–1188.
[21] N. Megiddo, K.J. Supowit, On the complexity of some common geometric location problems, SIAM J. Comput. 13 (1) (1984) 182–196.

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005

[22] C.W. Tsai, S.P. Tseng, M.C. Chiang, C.S. Yang, A framework for accelerating metaheuristics via pattern reduction, in: Proceedings of the Conference on Genetic and Evolutionary Computation, 2010, pp. 293–294.
[23] J. Schaffer, L. Eshelman, On crossover as an evolutionary viable strategy, in: Proceedings of the International Conference on Genetic Algorithms, Morgan Kaufmann, 1991, pp. 61–68.
[24] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[25] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[26] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[27] F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Springer US, 2003.
[28] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.
[29] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 26 (1) (1996) 29–41.
[30] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[31] R. Storn, P.K. P., Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Technical report, TR-95-012,ICSI, 1995.
[32] H. Abbass, MBO: marriage in honey bees optimization a haplometrosis polygynous swarming approach, in: Proceedings of Computation Congress on Evolutionary Computation, vol. 1, 2001, pp. 207–214.
[33] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: Proceedings of the Nature Inspired Cooperative Strategies for Optimization, vol. 284, 2010, pp. 65–74.
[34] X.S. Yang, Firefly algorithms for multimodal optimization, in: Proceedings of the International Conference on Stochastic Algorithms: Foundations and Applications, 2009, pp. 169–178.
[35] Z.W. Geem, J.H. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[36] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.
[37] S. Salcedo-Sanz, J.D. Ser, S. Gil-López, I. Landa-Torres, J.A. Portilla-Figueras, The coral reefs optimization algorithm: an efficient metaheuristic for solving hard optimization problems, in: Proceedings of the Applied Stochastic Models and Data Analysis International Conference, 2013, pp. 751–758.
[38] C.W. Tsai, J. Rodrigues, Metaheuristic scheduling for cloud: a survey, IEEE Syst. J. 8 (1) (2014) 279–297.
[39] M. Todorovski, D. Rajicic, An initialization procedure in solving optimal power flow by genetic algorithm, IEEE Trans. Power Syst. 21 (2) (2006) 480–487.
[40] W.F. Gao, S.Y. Liu, L.L. Huang, Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique, Commun. Nonlinear Sci. Numer. Simul. 17 (11) (2012) 4316–4327.
[41] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, second extended ed., Springer-Verlag New York, Inc., New York, NY, USA, 1994.
[42] P. Larraanaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Norwell, MA, USA, 2001.

Chun-Wei Tsai received the Ph.D. degree in Computer Science from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2009. He was a postdoctoral fellow with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, before joining the faculty of the Department of Applied Geoinformatics and the faculty of the Department of Applied Informatics and Multimedia, Chia Nan University of Pharmacy & Science, Tainan, Taiwan, in 2010 and 2012, respectively. He joined the faculty of the Department of Computer Science and Information Engineering, National Ilan University, Yilan, Taiwan, as an assistant professor in 2014. His research interests include metaheuristics, data mining, internet technology, and combinatorial optimization.

Please cite this article as: C.-W. Tsai, An effective WSN deployment algorithm via search economics, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.01.005