EPiC Series in Computer Science

Volume 36, 2015, Pages 296–311

GCAI 2015. Global Conference on Artificial Intelligence

Simulated Annealing Application to Maximum Lifetime Coverage Problem in Wireless Sensor Networks

Antonina Tretyakova[1] and Franciszek Seredynski[1]

Cardinal Stefan Wyszynski University
Warsaw, Poland
[a.tretyakova, f.seredynski]@uksw.edu.pl

Abstract

Energy optimization problem in Wireless Sensor Networks (WSN) is a backbone of efficient performance of sensor network consisting of small devices with limited and non-recovering battery. WSN lifetime maximization problem under assumption of that the coverage is main task of the network is known as Maximal lifetime coverage problem (MLCP). This problem belongs to a class of NP-hard problems. In this paper we propose a novel simulated annealing (SA) algorithm to solve MLCP. The proposed algorithm is studied for high dense WSN instances under different parameter setup.

1 Introduction

WSN is a set of huge number of small devices enabling to monitor surroundings, gather information about environment and perform many other tasks. They are getting involved in many spheres of human vital activities, such as agriculture, healthcare, biomedicine, environment observation, etc. For many missions, sensors are randomly distributed over the target field, where human access is limited or impossible, therefore batteries of sensors cannot be usually rechargeable or renewable. Exhaustion of battery charge implies the change in topology of WSN, quality of its work and reduction of its lifetime.

Maximal lifetime coverage problem in WSN can be stated as a combinatorial problem with given target field and WSN instance, where a target field is represented by a set $POIs = \{1, 2, ..., N_{POIs}\}$ of N_{POIs} integer numbers and a set of sensors $S=\{s_1, s_2, ..., s_N\}$ is represented by a family of N subsets from $POIs$, i.e. $s_i \subseteq POIs$ for all i=1, 2, ..., N, so that i-th subset is related to those POIs which are within coverage area of the i-th sensor. Consider homogeneous network with circle coverage area of radius R_s called sensing range of a sensor. We assume that all sensors initially possess the same battery charge given by an integer number b and during each time of sensor's activity battery limit decreases by one energy unit. The goal of a network is to support a given coverage ratio q until energy supply of all sensors are exhausted. This condition called coverage requirement means that at least q-th part of POIs should be sensed by an active set of sensors. The goal is to find a maximal number of subsets of active sensors,

each sensor can be included in no more than b subsets and each of such subsets should meet the coverage constraint. The number of the found subsets corresponds to lifetime of the network.

Simulated annealing (SA) is one of the nature inspired algorithms leading from the physical process of crystal growth. Firstly SA was proposed and developed by Kirkpatrick, Gelatt and Vecchi [8], and later by Johnson et al. [7], Koulamas [9], Abramson [1] and Emden-Weinert [3]. SA is based on the physical annealing process applied in glass and metal metallurgy. The goal of this process is crystallization of metal via its heating and gradual cooling. In slow annealing the global decrease of system energy is arisen during the crystallization. However, there are exist system states with energy increased for a while. Due to this fact it is possible to exit from the local optimum, which the system is found in.

During the last years the efficiency of simulated annealing technique to problems of discrete optimization was proven. These approaches were applied to different combinatorial hard-computational problems, among which are graph colouring problem [6], [7], flow shop and job shop scheduling problem [9], school time tabling problem [1], airline crew-pairing problem [3], etc. There are also more recent applications of SA, among which are router nodes placement problem [15], parametric sensitivity in fuzzy control systems [12], dynamic robot path planning [10], replenishment policy for ATO supply chain [5]. In this paper, we propose a novel SA algorithm to solve MLC problem in WSNs.

The remainder of the paper is organized as follows. In next section, we introduce simulated annealing algorithm application to MLCP in WSN. Section 3 and 4 contains parameter setup and experimental results with discussion. Conclusion are presented in the last sections. More results are added in appendix of the paper.

2 Simulated Annealing algorithm solves MLCP

SA algorithm has four initial parameters: temperature T, a cooling ratio r, temperature length L, frozen threshold θ. Pseudo code of the general scheme of SA is presented in Algorithm 1 and consists of the following steps. Two cycles are performed on the randomly taken solution Sol, called accepted solution. The external cycle is executed until the temperature T drops below θ. Each external cycle step the temperature is decreased according to a cooling scheme. During the internal cycle, which is repeated L times, a random neighbouring solution Sol_{neigh} is generated and evaluated by a function f as well as accepted solution. In case of the worse neighbour, it is accepted instead of previous one with probability $e^{-\frac{\Delta}{T}}$, where

$$\Delta = f(Sol) - f(Sol_{neigh}) \tag{1}$$

otherwise, the best of these two is accepted for the next iteration.

SA provides comparatively good solutions for many problems of combinatorial optimization. However, there exists a fine parameter tuning problem which means that appropriate set of parameters highly impacts on the algorithm's performance and depends on a given problem. For SA algorithm running, the following entities should be defined:

- search space, which should contain all feasible solutions, i.e. the solutions meeting all constraint posed by the problem,
- neighbourhood, which is defined by a heuristic of low computational cost and enables to achieve good solutions,
- initial parameters: temperature T_0 and a cooling scheme with a set of cooling parameters.

Algorithm 1 SA algorithm to MLCP

Input :
Initial temperature T=100
A cooling ratio $r \in$ [0,1)
Temperature length L
Frozen threshold θ
Random solution Sol
while $notfrozen$ **do**
  for $i \leftarrow 1$ to L **do**
    evaluate accepted solution $f(Sol)$
    pick a random neighbour $Sol_{neigh}(Sol)$
    evaluate neighbouring solution $f(Sol_{neigh})$
    let $\Delta = f(Sol) - f(Sol_{neigh})$
    if $\Delta \leq 0$ **then**
      $Sol <-- Sol_{neigh}$
    else
      $Sol <-- Sol_{neigh}$ with probability $e^{\frac{\Delta}{T}}$
    end if
    i=i+1,
  end for
  reduce temperature $T = r \times T$,
end while
Output :
the last accepted solution

Initial temperature and a cooling scheme have a key meaning for the quality of results provided by the algorithm. Beginning from too low temperature or too rapid cooling enables to converge the local optimum. Too high temperature or very slow cooling follows the increase in computation time need to obtain the result. We consider linear cooling scheme, where temperature is changing as follows:

$$T_{i+1} = T_i - \alpha, \tag{2}$$

where a cooling factor $\alpha \in (0, 1)$, d is usually set to one [2], c is greater than or equal to the largest energy barrier in the problem [11] [4] and i is a current step in the external cycle, and T_i is temperature of the system in the $i - th$ step. The acceptance probability is assumed to be equal to zero for negative temperatures.

2.1 Solution encoding

A solution is encoded as an $T_{max} \times N$ table (further we call it as a *schedule* or a *schedule solution*), where T_{max} is predefined in time intervals meeting the following condition:

$$b < Lifetime(q) < T_{max} << N \times b, \tag{3}$$

where T_{max} should to be set greater than $Lifetime(q)$ (2) and q should be less than complete coverage provided by the network with all activated sensors. The upper bound $N \times b$ arises as a maximal number of different subsets, each of which consists of one element (such elements N) and, according to battery capacity requirement, can appear in the schedule b times.

In the schedule the parameter T_{max} is related to a number of columns. A number N is a number of sensors in the sensor network and concerns the number of rows in the schedule solution's table. A column of the schedule contains a network of active sensors during the certain time interval. Each row of the table is related to one of the sensors and represents its schedule of activity over all period of time T_{max}.

Binary coding is used, so that "1"s value corresponds to active state of a sensor, "0" corresponds to a sleeping state. Cells of the table are filled by "1"s and "0"s values in such a way that battery capacity restriction is met, i.e. each row of the table contains b ones and $T_{max} - b$ zeros.

A schedule solution is associated with a sequence of T_{max} numbers called a *coverage string*, i.e

$$coverage\ string = \{cov(1), ..., cov(T_{max})\}, \tag{4}$$

where for every $i = 1, ..., T_{max}$ $cov(i)$ is counted according to (1) and $|POIs_{obs}| = |\cup_{j=1} POIs_{obs}(s_j)|$.

Algorithm 2 Pseudocode - Encoding a "one" solution

```
Input : Sol
for i ← 1 to b do
  for j ← 1 to N do
    set i − th bit in the j-th row to be equal to 1,
    j = j + 1,
  end for
  i = i+1,
end for
Output : Sol
```

2.2 Evaluation function

As evaluation function, which should be maximized, in MLCP is assumed Lifetime(q) metric defined as a number of time intervals, during which coverage requirement is met, i.e.

$$Lifetime(q) = \sum_{i=1}^{T_{max}} 1|_{cov(i) \geq q} \tag{5}$$

The whole timeline of a schedule is divided into two sequences: Redundant and Unsatisfactory. All time units when active sensors cover more than q-th part of POIs constitute Redundant Sequence (RS) of WSN schedule, the rest time units are included in Unsatisfactory Sequence (US) [13].

2.3 Generating neighbouring solutions

The method of solutions generating is related to the search space's definition and a structure of MLCP's solution. The first solution is created as "one" solution such that the first b in each row in the solution's array are fulfilled by "1" and the rest are filled by "0" (see, Algorithm 2). Generating the sequential solutions is based on a swap of a pair of cells of opposite values. A number of changing pairs, let us call it a neighbourhood size, relatively to a schedule's size

provides a probability of a cell's change. For example, in case of changing two random cells, we obtain a solution in 1-neighbourhood from a given solution.

In SA algorithm, we consider problem knowledge-wise scheme to generate neighbouring solutions which consists in follows.

The pseudo code of the neighbours generating scheme is presented in Algorithm 3. The additional information about solution is computed such as coverage of each time slot according which the time line is divided on RS, ES and US.

Algorithm 3 Pseudocode - problem knowledge - wise neighbourhood

Input :
Sol
k
compute RS, US
for $i \leftarrow 1$ to k **do**
  for $j \leftarrow 1$ to $RS.size()$ **do**
    choose random "1" gene from $i-th$ RS column, the row of the gene let us denote as l
    find the first "0" gene from US and $l-th$ row
    change the values of the pair of chosen genes
    $j = j + 1$
  end for
  $i = i + 1$
end for
Output :
Chromosome

3 Experimental setup

In this section, we present some results of experimental study of the proposed SA algorithm to solve MLCP.

SA need to be set up values of a set of parameters, which are influential in perfection of algorithm's performance to solve the given problem. In the same time the problem defines an input parameters set, for which the algorithm provides the best results under the certain values algorithm's parameters. The algorithm was studied under the wide range of problem instances.

The experimental study was conducted in two steps. Firstly, several experiments were made in order to estimate the best values for parameters of the algorithms. The next step of experiments was to compare these algorithms with the best parameters sets. For this purpose a set of problem instances was created, which are defined by target field, WSN instance and required coverage level, i.e.

- Target field 100 × 100 (m^2)
- Distance between POIs (g) {1, 5, 10} m
- A number of POIs {100, 400, 10000}
- A number of sensors (N) {100, 200, 300}
- Battery capacity b {20, 40, 60, 80, 100}
- Sensing range (R_s) {1, 2, 5, 10, 20} m

Table 1: Parameter set for SA.

Parameter	Values
Solution size	
A size of a schedule (T_{max})	10 $\times b$
Algorithm parameters	
Neighbourhood type	{random, knowledge-wise}
Neighbourhood size k_{neigh}	{1, 5, 10, 20, 30, 40, 50}
Cooling schemes and	linear ($\alpha = 0.5$)
Initial temperature	$T_0 = 100$
Input data	
Target field	100 $\times$ 100 (m^2)
Distance between POIs (g)	{1, 5, 10} m
A number of POIs	{100, 400, 10000}
A number of sensors (N)	{100, 200, 300}
Battery capacity b	{20, 40, 60, 80, 100}
Sensing range (R_s)	{1, 2, 5, 10, 20} m
Coverage ratio (q)	{0.75, 0.8, 0.85, 0.9, 0.95, 1.0}

- Communication range (R_c) 42.5 m
- Coverage ratio (q) {0.75, 0.8, 0.85, 0.9, 0.95, 1.0}
- A size of a schedule (T_{max}) 10 $\times b$

The whole parameter setup for the SA is summarized in Table 1.

For evaluating the solutions, we rely on our network simulator, written in Java. The experiments were run on standard PC computer with two cores 1.66GHz CPU and 1GB RAM.

Let us denote SA algorithm for a chosen parameters set of values as SA-$neigh_{k_{neigh}}$-$coolingScheme$, where $neigh$ is K in case of knowledge-wise (Algorithm 3), k_{neigh} is a neighbourhood size from the set {1, 5, 10, 20, 30, 40, 50}, and a $coolingScheme$ is lin in case of linear with parameters such as indicated in Table 1. For example, SA-K_{10}-lin is a denotion for SA algorithm with knowledge-wise neighbours' choice with linear cooling scheme and neighbourhood size equal to 10. Let us assume default values as follows:

- Target field 100 $\times$ 100 (m^2),
- Distance between POIs g = 10 (m),
- A number of POIs 100,
- A number of sensors $N = 100$,
- Battery capacity $b = 20$,
- Sensing range $R_s = 20$ (m),
- Coverage ratio $q = 0.9$.

3.1 Target field, POIs

Sensors are randomly deployed over the target field F of dimensions $(L \times L)m^2$, where in all experiments we assume $L = 100$. POIs are uniformly distributed over the target field F in every g m, where $g = 20$.

3.2 WSN instances

We consider WSN consisting of a number N of sensors equal to 100, 200 and 300, respectively. For each value of N, we created 3 instances, which differ by random allocation of sensors, so 9 instances were used in experimental study. Each instance is described as Instance $num-sensors\ instance-order-number$, so e.g. Instance23 means the third instance with N=200 sensors. Each sensor possess a sensing range R_s equal to $20m$, and each node has a battery capacity b from the set {20, 40, 60, 80, 100}. The size of an schedule solution depends on a number of sensors N in the WSN instance and battery capacity b. These two parameters are strongly impact on a lifetime of the WSN.

WSN density In order to prolong lifetime of WSN in the same target field a larger amount of sensors is deployed. We created instances of with a number N of sensors in the range {100, 200, 300} distributed in the target field of the same size.

Sensing range The sensing range has an impact on a number of sensors which is necessary to be activated in order to achieve required coverage level q. We assume a sensing range (R_s) values in the range {1, 2, 5, 10, 20} m.

Battery capacity Battery capacity b parameter is defined in time units dependently on initial battery capacity of a sensor and a number of time units during which the sensor is activated. Therefore, we study the performance of the algorithm under the $b \in$ {20, 40, 60, 80, 100}. In case of the same initial battery capacity the higher value of b corresponds to shorted time unit value in the schedule timeline division.

Randomness of deployment For each WSN density we created three random deployments. For each value of N, we created 3 instances, which differ by random allocation of sensors. Each instance is described as Instance $num-sensors\ instance-order-number$, so e.g. Instance23 means the third instance with N=200 sensors.

3.3 Coverage ratio q

Preserving complete area coverage is a desirable objective, but, sometimes, to achieve high coverage ratio may be more practical interest. Analysis and simulations in [18] [16] [17] and [14] have shown that network coverage lifetime can be greatly prolonged if only preserving partial coverage. As we have discussed above, full coverage is not achievable by arbitrary instance of WSN. Therefore, in experimental study we use required coverage ratio $q \in$ {0.75, 0.8, 0.85, 0.9, 0.95}for WSN with $R_s \in$ {10, 20} and full coverage for $R_s = 20$.

4 Experimental Study

4.1 Influence of neighbourhood size on the lifetime of WSN

Table 2 contains the results of Lifetime(0.9) averaged over the 25 runs of each of the algorithms. For the algorithm seven variants of neighbourhood size for SA algorithms were studied with linear cooling scheme. Table 2 presents maximal, average with standard deviation of values of Lifetime(0.9), a number of times of maximum achieved and average performance time per run of SA-K_k-lin, where $k \in$ {1, 2, 3, 4, 5, 10, 15} for Instances 1, 21 and 31 and battery capacity

$b = 20$. Among the different neighbourhood size the best one is provided by SA-$N_1 - lin$ independently on WSN instance. Therefore, in next experiments we assume one-neighbourhood.

4.2 Impact of a density of WSN deployment on the lifetime of WSN

Table 4 contains the result of Lifetime(0.9) averaged over the 25 runs of each of the SA with one-neighbourhood and linear cooling scheme. To compare impact of density of WSN deployment on the lifetime of WSN the results obtained for instances with higher density N equal to 200 and 300 are compared with the results provided for Instance1 with twice and three times taken, respectively. In such a way, the best Lifetime(0.9) obtained for Instance 21 and Instance 31 by SA-N_1-lin are better 5,5% and 4,9% than for Instance1. Therefore, increasing WSN size with random distribution allows improve Lifetime(q) against with increased number of sensors with the same localization.

4.3 Coverage ratio requirement and lifetime of WSN

In this experiment, we would like to study relation between coverage ratio q requirement and lifetime of the network provided by SA algorithm. Maximal, average with standard deviation values of Lifetime(q), a number of times of maximum achieved and average performance time per run obtained by SA-K_1-lin, where $q \in \{0.75, 0.8, 0.85, 0.9, 0.95\}$ for Instance 1 and $b = 20$ are presented in Table 3. From Table 3 a linear decrease of Lifetime(q) with growth q can be observed, where reducing coverage requirement by 5% allows to improve network lifetime by 20 time units.

4.4 Impact of a randomness of WSN deployment on the lifetime of WSN

The task of these series of experiments is to study the influence of randomness of the WSN deployment on the quality of the SA algorithm performance. For this purpose, nine instances 1, 2, 3, 21, 22, 23 and 31, 32, 33 with random distribution of 100, 200 and 300 sensors were created.

Table 4 contains summarized results of Lifetime(0.9) obtained by SA-K_1-lin for three WSN instances with initial battery $b = 20$. The values are averaged over the 25 runs provided by the algorithm.

4.5 Impact of an initial energy charge of sensors on the lifetime of WSN

The aim of this experiment is to study the impact of the parameter b of the MLC problem on the results obtained by the SA algorithm. To achieve this goal, 25 runs of SA-K_1-lin for instance 1 with five values of battery capacity parameter $b \in \{20, 40, 60, 80, 100\}$ were made. The results are presented in Table 5.

Table 5 contains summarized results: maximal, average with standard deviation values of Lifetime(0.9) and number of times of maximum achieved by each of SA algorithms with linear cooling scheme.

It should be noticed, for some battery capacities, for example for $b = 80$ and $b = 100$ the Lifetime(0.9) values are better than linearly increased Lifetime(0.9) obtained for $b = 20$. So, if we increase average Lifetime(0.9) for $b = 20$ which is 76 four and five times respectively, we

Table 2: Maximal, average with standard deviation values of Lifetime(0.9), a number of times of maximum achieved and average performance time per run by SA-K_k-*log* and SA-K_k-*lin*, where $k \in \{1, 2, 3, 4, 5, 10, 15\}$ for Instances 1, 21 and 31 and $b = 20$.

WSN	Algorithm	Max	Avg $\pm$ σ	Times of Max	T_{exec} [s]
Instance 1	SA-K_1-*log*	81	76.0 $\pm$ 2.23	1	66
	SA-K_2-*log*	81	75.0 $\pm$ 2.23	1	68
	SA-K_3-*log*	81	76.0 $\pm$ 2.23	1	69
	SA-K_4-*log*	79	74.0 $\pm$ 2.23	1	46
	SA-K_5-*log*	78	74.0 $\pm$ 1.73	1	68
	SA-K_{10}-*log*	77	73.0 $\pm$ 2.0	5	69
	SA-K_{15}-*log*	76	72.0 $\pm$ 1.73	1	76
Instance 1	SA-K_1-*lin*	80	76.0 $\pm$ 2.0	1	66
	SA-K_2-*lin*	79	76.0 $\pm$ 1.41	2	102
	SA-K_3-*lin*	78	74.0 $\pm$ 2.0	2	104
	SA-K_4-*lin*	77	74.0 $\pm$ 1.41	2	79
	SA-K_5-*lin*	79	75.0 $\pm$ 1.73	1	68
	SA-K_{10}-*lin*	79	73.0 $\pm$ 2.0	1	66
	SA-K_{15}-*lin*	77	72.0 $\pm$ 2.0	1	68
Instance 21	SA-K_1-*log*	171	164.0 $\pm$ 2.82	1	305
	SA-K_2-*log*	168	162.0 $\pm$ 2.23	1	256
	SA-K_3-*log*	167	161.0 $\pm$ 2.44	1	402
	SA-K_4-*log*	163	159.0 $\pm$ 2.64	3	242
	SA-K_5-*log*	162	158.0 $\pm$ 2.64	1	241
	SA-K_{10}-*log*	161	154.0 $\pm$ 3.0	1	283
	SA-K_{15}-*log*	161	153.0 $\pm$ 3.60	1	242
Instance 21	SA-K_1-*lin*	167	163.0 $\pm$ 2.44	3	250
	SA-K_2-*lin*	166	161.0 $\pm$ 2.23	1	245
	SA-K_3-*lin*	166	161.0 $\pm$ 2.23	1	245
	SA-K_4-*lin*	163	159.0 $\pm$ 2.44	3	260
	SA-K_5-*lin*	163	158.0 $\pm$ 2.23	1	238
	SA-K_{10}-*lin*	161	154.0 $\pm$ 3.0	1	340
	SA-K_{15}-*lin*	157	152.0 $\pm$ 2.23	1	287
Instance 31	SA-K_1-*log*	255	248.0 $\pm$ 2.82	1	409
	SA-K_2-*log*	251	246.0 $\pm$ 3.46	3	519
	SA-K_3-*log*	249	245.0 $\pm$ 2.64	2	420
	SA-K_4-*log*	248	242.0 $\pm$ 3.16	2	414
	SA-K_5-*log*	247	240.0 $\pm$ 3.31	1	414
	SA-K_{10}-*log*	242	235.0 $\pm$ 2.82	1	586
	SA-K_{15}-*log*	237	231.0 $\pm$ 2.82	2	569
Instance 31	SA-K_1-*lin*	255	248.0 $\pm$ 3.31	1	548
	SA-K_2-*lin*	254	246.0 $\pm$ 2.82	1	538
	SA-K_3-*lin*	251	244.0 $\pm$ 3.16	2	561
	SA-K_4-*lin*	247	243.0 $\pm$ 2.64	6	755
	SA-K_5-*lin*	246	240.0 $\pm$ 3.60	2	545
	SA-K_{10}-*lin*	238	233.0 $\pm$ 2.64	1	550
	SA-K_{15}-*lin*	239	230.0 $\pm$ 4.24	1	656

Table 3: Maximal, average with standard deviation values of Lifetime(q), a number of times of maximum achieved and average performance time per run by SA-K_1-*log* and SA-K_1-*lin*, where $q \in \{0.75, 0.8, 0.85, 0.9, 0.95\}$ for Instance 1 and $b = 20$.

q	Algorithm	Max	Avg $\pm$ σ	Times of Max	T_{exec} [s]
0.95	SA-K_1-*lin*	59	54.0 $\pm$ 1.73	1	49
	SA-K_1-*log*	58	54.0 $\pm$ 1.73	1	51
0.9	SA-K_1-*lin*	80	76.0 $\pm$ 2.0	1	66
	SA-K_1-*log*	81	76.0 $\pm$ 2.23	1	66
0.85	SA-K_1-*lin*	100	95.0 $\pm$ 2.23	2	85
	SA-K_1-*log*	100	95.0 $\pm$ 2.0	1	86
0.8	SA-K_1-*lin*	119	115.0 $\pm$ 2.0	1	114
	SA-K_1-*log*	119	115.0 $\pm$ 1.41	1	105
0.75	SA-K_1-*lin*	140	135.0 $\pm$ 1.73	1	120
	SA-K_1-*log*	140	136.0 $\pm$ 2.23	3	119

would obtain 304 and 380 for $b = 80$ and $b = 100$. These values are smaller than those that provided by SA algorithm.

5 Conclusion

In this paper, we have proposed a novel simulated annealing algorithm to solve MLCP in WSN. The proposed algorithm is studied under different parameter setup assumptions and using problem specific knowledge about the current solution in order generates the next one in a effective way.

Results of experimental study of the algorithm and comparison of them with ones obtained by applying recently proposed centralized algorithms show that despite its simplicity and limited local information it is able to achieve similar results as centralized algorithms in terms of Lifetime(q) metric. The purpose of current and future studies is more detailed study of the proposed algorithm for different parameters of GCA and WSN densities and using it for different variants of MLCP problem.

References

[1] D. Abramson, M. Krishnamoorthy, and H. Dang. Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research*, 16:1–22, 1999.

[2] M. M. Atiqullah. An efficient simple cooling schedule for simulated annealing. *Lecture Notes in Computer Science*, 3045:396–404, 2004.

[3] T. Emden-Weinert and M. Proksch. Best practice simulated annealing for the airline crew scheduling problem. *Journal of Heuristics*, 5:419436, 1999.

[4] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and bayesian restoration of images. *Journal of Applied Statistics*, 20(5):721–741, 1984.

[5] Hui Huang, Yan Jin, Bo Huang, and Han-Guang Qiu. Mixed replenishment policy for ato supply chain based on hybrid genetic simulated annealing algorithm. In *Mathematical Problems in Engineering*, volume 574827, pages 420–437, 2014.

Table 4: Maximal, average with standard deviation values of Lifetime(0.9), a number of times of maximum achieved and average performance time per run by SA-K_1-*log* and SA-K_1-*lin* for nine instances and $b = 20$.

Instance	Algorithm	Max	Avg $\pm$ σ	Times of Max	$T_{exec}[s]$
			N=100		
1	SA-K_1-*lin*	80	76.0 $\pm$ 2.0	1	66
	SA-K_1-*log*	81	76.0 $\pm$ 2.23	1	66
2	SA-K_1-*lin*	74	70.0 $\pm$ 1.73	1	50
	SA-K_1-*log*	74	70.0 $\pm$ 1.41	1	139
3	SA-K_1-*lin*	89	83.0 $\pm$ 2.64	1	49
	SA-K_1-*log*	87	83.0 $\pm$ 2.23	3	144
			N=200		
	SA-K_1-*log*	171	164.0 $\pm$ 2.82	1	305
21	SA-K_1-*lin*	167	163.0 $\pm$ 2.44	3	250
	SA-K_1-*log*	160	154.0 $\pm$ 2.0	1	239
22	SA-K_1-*lin*	158	153.0 $\pm$ 3.16	2	238
	SA-K_1-*log*	171	165.0 $\pm$ 2.64	1	274
23	SA-K_1-*lin*	170	165.0 $\pm$ 2.44	2	237
			N=300		
	SA-K_1-*log*	255	248.0 $\pm$ 2.82	1	409
31	SA-K_1-*lin*	255	248.0 $\pm$ 3.31	1	548
	SA-K_1-*log*	233	226.0 $\pm$ 2.64	1	520
32	SA-K_1-*lin*	236	228.0 $\pm$ 3.0	1	390
	SA-K_1-*log*	252	246.0 $\pm$ 3.16	2	509
33	SA-K_1-*lin*	254	245.0 $\pm$ 4.0	1	511

[6] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing - an experimental evaluation; part 1, graph partitioning. *Operations Research*, 37:865892, 1989.

[7] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing - an experimental evaluation; part 2, graph-coloring and number partitioning. *Operations Research*, 39:378406, 1991.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[9] C. Koulamas, S. R. Antony, and R. Jaen. A survey of simulated annealing applications to operations-research problems. *OMEGA-International Journal of Management Science*, 22:41–56, 1994.

[10] Hui Miao and Yu-Chu Tian. Dynamic robot path planning using an enhanced simulated annealing approach. In *Applied Mathematics and Computation*, volume 222, pages 420–437, 2013.

[11] Y. A. B. Nourani. A comparison of simulated annealing cooling strategies. *Journal of Physics*,

Table 5: Maximal, average with standard deviation values of Lifetime(0.9), a number of times of maximum achieved and average performance time per run by SA-K_1-*log* and SA-K_1-*lin* for Instance 1 and $b \in \{20, 40, 60, 80, 100\}$.

b	Algorithm	Max	Avg $\pm$ σ	Times of Max	T_{exec} [s]
20	SA-K_1-*lin*	80	76.0 $\pm$ 2.0	1	66
	SA-K_1-*log*	81	76.0 $\pm$ 2.23	1	66
40	SA-K_1-*lin*	162	155.0 $\pm$ 2.82	1	242
	SA-K_1-*log*	159	155.0 $\pm$ 2.23	2	216
60	SA-K_1-*lin*	240	232.0 $\pm$ 3.16	1	451
	SA-K_1-*log*	237	232.0 $\pm$ 2.44	1	570
80	SA-K_1-*lin*	322	309.0 $\pm$ 4.47	1	764
	SA-K_1-*log*	320	310.0 $\pm$ 3.87	1	781
100	SA-K_1-*lin*	397	389.0 $\pm$ 3.74	1	1090
	SA-K_1-*log*	399	389.0 $\pm$ 4.24	1	1103

31(41):8373–8386, 1998.

[12] Precup R., David R.C., Petriu E.M., Preitl S., and Radac M. Fuzzy control systems with reduced parametric sensitivity based on simulated annealing. In *IEEE Transactions on Industrial Electronics*, volume 59 (8), pages 3049–3061, 2012.

[13] A. Tretyakova and F. Seredynski. Application of evolutionary algorithms to maximum lifetime coverage problem in wireless sensor networks. *27-th IEEE International Parallel and Distributed Processing Symposium, NIDISC Workshop, Boston, USA*, May 2013.

[14] L. Wang and S. S. Kulkarni. Sacrificing a little coverage can substantially increase network lifetime. *Elsevier Ad Hoc Networks*, 6(8):1281–1300, 2008.

[15] Fatos Xhafaa, Admir Barollib, Christian Sncheza, and Leonard Barollic. A simulated annealing algorithm for router nodes placement problem in wireless mesh networks. In *Simulation Modelling Practice and Theory*, volume 19 (10), pages 2276–2284, 2011.

[16] H. Zhang and J. Hou. On deriving the upper bound of alpha-lifetime for large sensor network. *Proceedings of the ACM International Simposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 1:121–132, 2004.

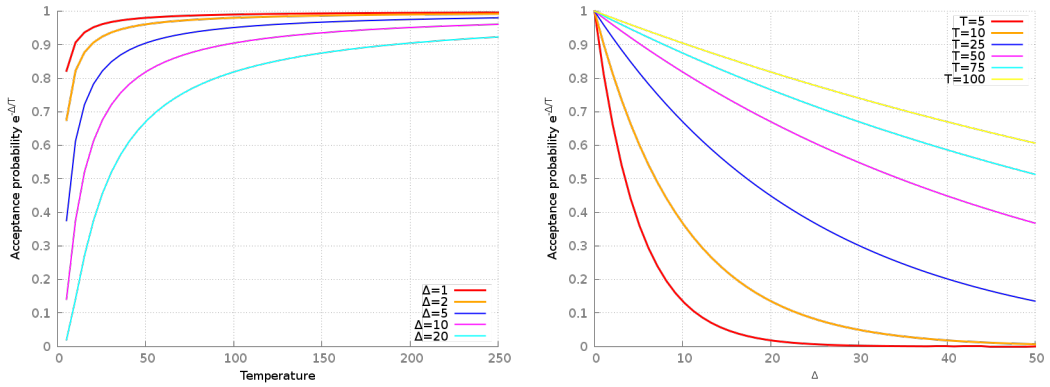
[17] H. Zhang and J. Hou. On the upper bound of alpha-lifetime for large sensor network. *ACM Transactions on Sensor Networks*, 1(2):272–300, 2005.

[18] Honghai Zhang and Jennifer Hou. Maintaining sensing coverage and connectivity in large sensor network. *Ad Hoc and Sensor Wireless Networks (AHSWN)*, 1:89–124, 2005.

A Appendix

A.1 Initial temperature setup

Temperature level has a high impact on acceptance the solution during each stage of SA. The higher temperature, the lower probability of acceptance the solution worse than the current one. With decreasing temperature level, the probability of acceptance of worse solution is increasing. Optimal value of initial temperature depends on the range of difference between evaluation function's values of the current solution and its neighbourhood in the problem. The initial temperature should be set under consideration of acceptance probability during the first



(a) (b)

Figure 1: Dynamics of acceptance probability $e^{-\frac{\Delta}{T}}$ with temperature level T for difference between current and neighbouring solutions $\Delta \in \{1, 2, 5, 10, 20\}$ (a) and with Δ for temperature $T \in \{5, 10, 25, 50, 75, 100\}$ (b)

stage of the algorithm performance.

A.2 Cooling scheme

We consider cooling schemes linear scheme and linear scheme with heating. Heating condition allows to prevent convergence to local minima and occurs when there is no improvement of solution for a predefined number of iterations. Temperature is arisen in case of all the following conditions are met:

- all steps in the internal cycle are performed L times, where L is a temperature cycle length,
- during the current temperature level at least one change of accepted solution was made,
- the termination condition is violated.

The temperature level is increased, if all the following statements is true:

- all steps in the internal cycle are performed L times, where L is a temperature cycle length,
- during the current temperature level no one change of accepted solution was made,
- the termination condition is violated.

In further experiments, we will study two different cooling schemes with heating: (1) linear with $\alpha = 0.5$ and initial temperature $T_0 = 100$ and (2) logarithmic initial temperature $T_0 = 50$. The first scheme is characterized by a high acceptance probability of worse solutions during the first stage of SA algorithm, quick fall of the acceptance probability in the second stage and stagnation process with zero acceptance probability during the third stage, when the new solution can be accepted only if it is strictly better than previous one. The second logarithmic scheme can be described by two phases. During the first phase lasted during the first 50 iterations, the worse new solution is accepted with high probability changing from 0.95 to 0.7. During the second phase the worse solution can be accepted with probability around 0.6. A number of iterations we restrict by 200 in order to expand the linear scheme by the third stage and observe the effects of dynamics of the solution under the neighbouring modification algorithm.

For termination condition we assume the following values:

1. exceeding a number of iterations accepted as minimal waiting value f_{min} times fifty.
2. achieving the temperature level around 0, given by a small value θ equal to 5,
3. a lack in solution improvement after a predefined number of iteration $L_{Term} = 50$ under the condition of the evaluation function is not less than minimal waiting value f_{min}

A.3 Impact of sensing range on the lifetime of WSN

As one can see from Table 6, the upper bound of timeline T_{max} in WSN schedules varies highly and depends on the sensing range value. The very low values of Lifetime(1.0) upper bound were obtained for small sensing ranges such as R_s equal to 1, 2 and 5. This fact can be observed in Figure 2 on the example of WSN Instance1 distribution with different sensing ranges $R_s \in \{1, 2.5, 10\}$ and distances between POIs equal to 10 (Figure 2). As one can observe from the Table, too small values of sensing range R_s relatively to the distance between POIs g follows the fact that there exist sensors which have no any POIs within their sensing ranges. The less sensing range of sensors is, the greater number of sensors covering no POIs appears, as a consequence, possible achieved coverage by WSN is low. However, the probability of that a greater number of POIs are covered is increased with increasing WSN density. For high sensing ranges such that each sensor covers all target field, the MLCP becomes trivial and leads to switching on one sensor during each time interval. In that case Lifetime(q) for all q is equal to a number of sensors times battery capacity, i.e.

$$Lifetime(q) = N \times b, \ \forall q \in (0, 1) \tag{6}$$

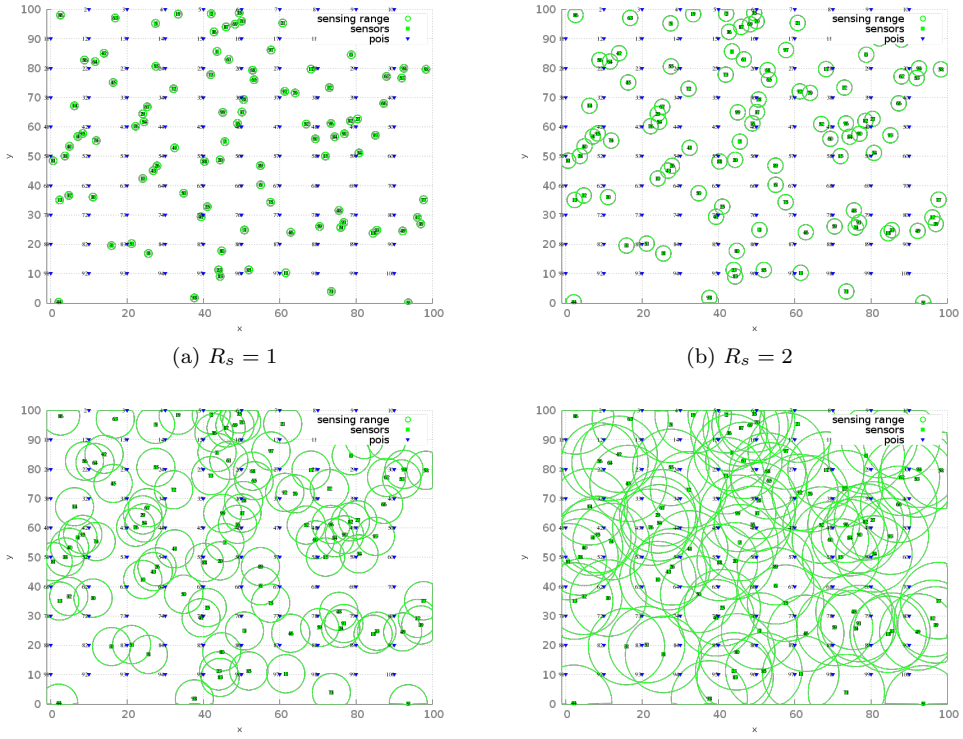
Due to the fact, the the high coverage, i.e coverage greater than 90%, is achievable by WSN instance with sensing range higher than 10, in the experiments we study WSN with $R_s \in \{10, 20\}$.

A.4 Impact of POIs density on the lifetime of WSN

Increasing resolution of target field, increasing a number of POIs in the target field, allows to improve possible achieved coverage by WSN under the condition that all sensors are switched on. Table 6 represents the percentage of covered POIs computed for 9 Instances, $N_{POIs} \in \{100, 400, 1000\}$, $R_s \in \{1, 2, 5, 10, 20\}$. Increasing POIs density in most cases follows a slight increase of percentage of covered POIs. As one can see from Table 6 this phenomena occurs in 39 from 45 cases. As an example of such exception is Instance 1 with $R_s \in \{5,10\}$, Instance 22 with $R_s \in \{1,2\}$ and Instance 32 with $R_s \in \{1, 5\}$. Because of that the percentage of covered POIs for different density of the target field does not vary highly, further we assume g equal to 5 as default value.

A.5 Impact of WSN density on the lifetime

As one can observe from Table 6, for small values of sensing range percentage of covered POIs grows with increasing WSN density. This is visible for $R_s \in \{1, 2, 5\}$, while this affect is not high for R_s equal to 10 and is not observed for R_s equal to 20. The impact of WSN density on Lifetime(q) will be study in the following chapter.



(a) $R_s = 1$ (b) $R_s = 2$

(c) $R_s = 5$ (d) $R_s = 10$

Figure 2: Instance1 of WSN distributed in the target field with distance between POIs g=10 and sensing range R_s is equal to 1 (a), 2 (b), 5 (c) and 10 (d).

Table 6: Percentage of covered POIs computed for 9 Instances, $N_{POIs} \in \{100, 400, 1000\}$, $R_s \in \{1, 2, 5, 10, 20\}$.

Instance	N_{POIs}	R_s 1	2	5	10	20
		N=100				
	100	0.01	0.07	0.45	0.92	1.0
1	400	0.0325	0.085	0.525	0.9175	1.0
	10000	0.0306	0.1183	0.5187	0.9295	1.0
	100	0.01	0.07	0.44	0.89	1.0
2	400	0.0325	0.1125	0.5025	0.92	1.0
	10000	0.0309	0.1129	0.51	0.9301	1.0
	100	0.02	0.11	0.45	0.92	1.0
3	400	0.0225	0.115	0.5	0.9225	1.0
	10000	0.0297	0.116	0.5115	0.9259	1.0
		N=200				
	100	0.03	0.14	0.65	0.98	1.0
21	400	0.055	0.2075	0.7425	0.9875	1.0
	10000	0.0603	0.2157	0.7553	0.9939	1.0
	100	0.07	0.18	0.76	0.95	1.0
22	400	0.0725	0.2275	0.7775	0.9675	1.0
	10000	0.0629	0.2186	0.7856	0.9789	1.0
	100	0.04	0.17	0.7	0.95	1.0
23	400	0.0525	0.2	0.755	0.9825	1.0
	10000	0.0601	0.2162	0.7579	0.9956	1.0
		N=300				
	100	0.07	0.25	0.78	0.99	1.0
31	400	0.085	0.3	0.855	0.9925	1.0
	10000	0.0904	0.3046	0.8726	0.9964	1.0
	100	0.09	0.23	0.86	0.96	1.0
32	400	0.0925	0.305	0.855	0.9775	1.0
	10000	0.0899	0.3015	0.872	0.9877	1.0
	100	0.07	0.25	0.84	0.98	1.0
33	400	0.0775	0.28	0.87	0.995	1.0
	10000	0.0889	0.3093	0.8782	0.9994	1.0