Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia
Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 19-21, 2019, Halifax, Canada

A Quantum-Annealing-Based Approach to Optimize the Deployment Cost of a Multi-Sink Multi-Controller WSN

Reihaneh Nikouei[a], Nayereh Rasouli[b], Shirin Tahmasebi[c], Somayeh Zolfi[d], Hamid Faragardi[e,], Hossein Fotouhi[f]

[a]*Shahid Bahonar University of Kerman, Iran*
[b]*Technical and Vocational University, Karaj Branch, Alborz, Iran*
[c]*Sharif University of Technology, Tehran, Iran*
[d]*University of Science and Technology, Tehran, Iran*
[e]*KTH Royal Institute of Technology, Stockholm, Sweden*
[f]*Mälardalen University, Västerås, Sweden*

Abstract

Software Defined Networking (SDN) provides network significant reconfiguration capability to Wireless Sensor Networks (WSNs). SDN is a promising technique for WSNs with high scalability and high reliability requirements. In SDN, a set of controller nodes are integrated into the network to advertise routing rules dynamically based on network and link changes. Determining the number and location of both sinks (are in charge of collecting the sensors data) and controller nodes in a WSN subject to both reliability and performance constraints is an important research challenge. In this paper, to address this research challenge, we propose a Quantum Annealing approach that improves the deployment cost of the system by minimizing the number of required sinks and SDN controller nodes. The experiments show that our approach improves the deployment cost of the network against the state-of-the-art by 10.7% on average.

Keywords: Quantum Annealing; Wireless Sensor Network; Software Define Networking; Node Placement.

1. Introduction

Software Defined Networking (SDN) controllers are introduced to provide more flexible network architecture. On-the-fly programming is the paradigm exploited by SDN to provide network reconfiguration. To realize SND, controller nodes are deployed in the network. Having only a single SDN-controller not only makes a single point of

[*] Corresponding author. Tel.: +46-769-225471.
E-mail address: hrfa@kth.se

failure but also may not be efficient for large scale networks in terms of network performance. Additionally, since controllers are responsible for updating routing strategies, controller failure will result in performance degradation. A multi-controller design is a promising solution to have a more reliable network that yields higher SDN availability and system responsiveness. A similar idea has been widely discussed for sink nodes in WSNs. The sink nodes are in charge of collecting sensors data and send them to the gateway or directly to the cloud. As demonstrated before [1], using multiple sinks in a sensor network ensures better network performance by creating the opportunity of collecting measurements at the sink(s) closer to sensors [2].

Besides both reliability and performance advantages of the multi-sink multi-controller WSN deployment, such a network design leads to a higher deployment cost of the network. Sink and controller entities are considered as the most costly elements of a network since they are equipped with high storage capacity to keep all measurements – sinks – or with high processing capability that provides network brain – controllers. Thus, it is important to reduce the cost of the network design by reducing the number of sinks and controllers.

Finding optimal sink/controller placement is an NP-hard problem, hence it cannot be addressed in polynomial time. Heuristic and meta-heuristic algorithms are promising approaches to deal with the node placement problem. Although multiple heuristic and meta-heuristic algorithms have been proposed in the literature to address the problem, new techniques and algorithms are highly desirable to push forward the state of the art in this context.

Contributions. The main contributions of this paper is to propose a Quantum-Annealing-based algorithm to tackle the multi-sink multi-controller node placement in WSNs. The performance of the proposed method has been compared not only with the original Simulated Annealing (SA) but also with the state of the art algorithm in the context of node placement in WSNs.

Paper structure. In Section 2 a brief review of related work is presented. The problem is described in detail and assumptions are defined in Section 3. In Section 4, the proposed method is introduced. In Section 5 the performance of the proposed solution is compared with the prior art. Finally, the main results are concisely summarized along with directions for future work in Section 6.

2. Related Work

There are a huge number of studies focusing on node placement in WSNs. They concentrate on different types of nodes (such as sensors, sinks, relays, and controllers) and different performance metrics in their node placement strategies. For example, in [3], the authors targeted coverage and connectivity metrics by solving the sensor placement problem. This work formulated the sensor placement problem as a constrained optimization problem.

In [4], a Joint Sink Deployment and Association (JSDA) in a multiple sink wireless camera network is formulated as a mixed integer linear programming problem.

In [5], four algorithms are proposed, namely, GreedyMSP and GRASP-MSP to solve the problem of multiple sink placement, and Greedy-MSRP and GRASP-MSRP for the problem of multi-sink and relay placement. In [6], a Genetic Algorithm (GA) is proposed to deploy multiple sinks in order to minimize the worst-case delay in WSNs, while in [7], a Particle Swarm Optimization (PSO) is employed to calculate the number of sinks and their locations in order to reduce the path length between sensors and sinks. In the same context, in [8], a PSO-based algorithm is proposed to prolong the network lifetime by considering the Euclidean distance and hop count. In [9], the placement of multiple sinks in order to minimize worst-case delay is taken into account and an approximation algorithm called GREEDY-k-SPP is proposed while [8] focuses on reducing energy consumption in WSNs in a random deployment of a grid topology network.

The controller placement problem for wired networks has been addressed in [10]. The authors explore the trade-offs when optimizing for minimum latency between nodes and controllers. [11] is an extension of [10] which is referred to as Pareto-Optimal Controller placement (POCO) considering additional aspects other than network latency. In WSNs, [12] presents a deployment strategy for multiple controller nodes that minimizes the transmission delay of the network while satisfying different critical requirements such as deployment cost and reliability.

Multi-sink multi-controller node placement in WSNs is already addressed by a few research works such as [1, 13] where, similar to our paper, they focus on the deployment cost of the network. Indeed, our proposed method extends these works by proposing a novel method.

3. Problem Modeling

In this section, the problem is modeled as an optimization problem, while we introduce the main assumptions.

A WSN is represented by an undirected graph, where vertices are partitioned into a set of sensors T, sinks S and controllers C. An edge of the graph indicates a wireless connection between a pair of nodes. The total number of sensors is N; i.e., $N = |T|$. A pair of nodes connected with an edge are called, *neighbor*. Furthermore, to represent the placement problem, we define A_S and A_C, denoting a set of candidate sinks and a set of candidate controllers respectively. Hence, $S \subseteq A_S$ and $C \subseteq A_C$.

A sensor is *k-sink-covered* if and only if it has at least k paths of length $\leq l_{max}$ to k sinks in S (k is an integer number ≥ 1). We define a WSN as k-sink-covered if each sensor $v \in T$ is k-sink-covered. A similar definition can be applied to indicate the relation between the sensors and controllers, meaning that a sensor is k'-controller-covered if and only if it has at least k' paths of length $\leq l_{max}$ to k' controllers in C.

Besides the k-covered constraints, to satisfy timing requirements in WSNs, the amount of data coming from sensors to a particular sink should be restricted to avoid too much workload on a sink. We define the load of sensor i as the rate of data which is sent to sinks per second, denoted by ω_i. If a sensor covered by n sinks (i.e., the distance between those n sinks and the sensor is shorter than l_{max}), the load of the ith sensor on the jth sink covering the sensor is denoted by $w_{i,j}$, which is equal to ω_i/n (load balancing).

Assuming a heterogeneous WSN, where sensors generate data with different periods. A similar constraint can be applied to controllers to manage the maximum workload on each controller, which is called controller load constraint. ω'_i implies the rate of data exchange between a sensor and controllers per second. Here again, $w'_{i,j}$ denotes the load of ith sensor on the jth controller.

3.1. Optimization problem

First of all, we introduce the notations used in the problem formulation. The binary array X determines if a candidate sink is chosen or not. If the candidate sink i is chosen, $X_i = 1$, otherwise, it is equal to zero. Furthermore, in order to represent the k-sink-covered constraint, we define the binary matrix Y that determines if the shortest distance between the ith sensor and the jth candidate sink is shorter than l_{max} or not. To calculate the shortest path between node v_i and sink s_j, the Dijkstra algorithm is used. Accordingly, the k-sink-covered network constraint can be formulated by

$$\sum_{\forall s_j \in A_S} Y_{i,j} X_j \geq K \qquad \forall v_i \in T \tag{1}$$

Let us assume that W shows the maximum workload that can be handled by each sink within an acceptable time, then the workload constraint can be reflected by

$$\sum_{\forall v_i \in T} Y_{i,j} X_j \frac{\omega_i}{\sum_{\forall s_t \in A_S} Y_{i,t} X_t} \leq W \qquad \forall s_j \in A_S \tag{2}$$

where $\sum_{\forall s_t \in A_S} Y_{i,t} X_t$ returns the number of selected sinks covering the ith sensor. Similarly, the aforementioned constraints for sink nodes are defined for controller nodes. Indeed, the binary array X' determines whether a candidate controller has been selected or not. Here, we use binary matrix Y' to show whether the sensor v_i is covered by the controller c_j, or not.

$$\sum_{\forall c_j \in A_C} Y'_{i,j} X'_i \geq K' \qquad \forall v_i \in T \tag{3}$$

The controller load constraint is formulated as follows:

$$\sum_{\forall v_i \in T} Y'_{i,j} X'_j \frac{\omega'_i}{\sum_{\forall c_t \in A_C} Y'_{i,t} X'_t} \leq W' \qquad \forall c_j \in A_C \tag{4}$$

where W' shows the maximum workload that can be handled by a controller within an acceptable time.

We define the cost of deployment of such a WSN as follows:

$$\zeta_{s_i,c_j} = \begin{cases} cost(s_i, c_j) & \mathrm{loc(s_i) = loc(c_j)} \\ cost(s_i) + cost(c_j) & \mathrm{else} \end{cases} \tag{5}$$

where $cost(s_i)$ and $cost(c_j)$ is the cost of purchasing, installation and maintenance of the ith sink and the jth controller respectively. Additionally, $cost(s_i, c_j)$ indicates the cost of one single chip (possibly a multi-core chip) including both the ith sink and the jth controller. We apparently expect that $cost(s_i, c_j)$ is less than $cost(s_i) + cost(c_j)$, since we merge two functions into a single chip. Thus, $cost(s_i, c_j) = \alpha(cost(s_i) + cost(c_j))$, where α is a real number between 0 and one.

The optimization problem is formulated as follows:

$$\textbf{Minimize}: \quad TotalCost(X, X') = \sum\nolimits_{\forall s_i \in A_S} \sum\nolimits_{\forall c_i \in A_C} \zeta_{s_i,c_j} X_{s_i} X'_{c_j}, \tag{6a}$$

$$\textbf{Subject to}: (1), (2), (3), (4). \tag{6b}$$

4. Solution Framework

4.1. Simulated Annealing

Simulated annealing is a meta-heuristic algorithm used to approximate the global optimum of a function in a large and often discrete search space. It is based on principles of statistical mechanics but its application in optimization problems emerges from [14]. It is a memory-less and iterative algorithm. In SA, first, an initial solution is generated. Then, in each iteration, a random neighbor is generated. If a solution is better in terms of energy or cost than the current state of the system, it is called an improving solution. Otherwise, it is called a non-improving solution. An improving generated neighbor is always accepted. But a non-improving generated neighbor is accepted with a given probability. This probability depends on both the current temperature of the system and the difference between the current solution and the generated neighbor energy. At the end of each iteration, when the system has reached an equilibrium state, the temperature will be decreased based on a particular cooling schedule. The cooling schedule has a great impact on the performance of the algorithm. In other words, there is always a compromise between the quality of the final solution and the cooling rate. This process will continue until the system reaches the final temperature. [15]

4.2. SA Disadvantages

As mentioned, in SA the probability to accept a non-improving solution depends on the current temperature and the difference between the energy of the current solution and that of the generated neighbor. Consequently, if the target function contains several high barriers, it is likely to get stuck in local optimum since the thermal transition in SA depends on the height of barriers. To overcome this problem, as shown in fig.1, we can use an alternative approach called Quantum Annealing which uses quantum tunneling (quantum jumps) to pass through high and thin barriers [16].

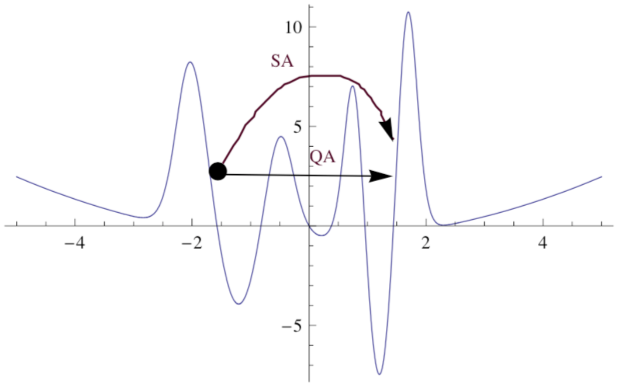

Fig. 1. Thermal Transition vs Quantum Tunneling [16]

4.3. Quantum Annealing

Quantum annealing is a meta-heuristic algorithm used to approximate global optimum of a function. It is based on principles of quantum mechanics. In quantum annealing the cost function to be minimized is: $H_q = H_{potential} + H_{kinetic}$ So H_q is composed of two components:

- $H_{potential}$: potential energy which is the same as cost function in SA.
- $H_{kinetic}$: kinetic energy which is used to take advantage of quantum fluctuation in order to escape from local minima. So $H_{potential}$ is minimized as a side effect of minimizing H_q.

One of the implementations of quantum annealing is Path Integral Monte Carlo Quantum Annealing (PIMC-QA). This implementation has been successfully used in solving several optimization problems e.g. graph coloring [17, 18] and traveling salesman problem (TSP) [19]. In this implementation there are P replicas of problem configuration called $\{\rho_1, \rho_2, ..., \rho_P\}$ and kinetic energy of system depends on the interactions between these replicas.

4.4. Applying QA to the Problem

As the problem described in [1] is NP-complete, we use PIMC-QA to solve it. To define $H_{potential}$, $H_{kinetic}$ and use QA algorithm we should first encode the problem and its solution using Ising model. Then we can formulate $H_{potential}$ and $H_{kinetic}$ and finally seek to minimize H_q.

4.4.1. Ising Model

In an Ising model, the system configuration is represented by a set of spin variables which can have only two states (Boolean variables). In our considered problem, we can divide our spin variables into two groups: problem space spin variables and solution spin variables. To define problem space spin variables, we use Y and Y'. Thus:

$$Problem\ Spin\ Variables = \begin{cases} S_Y & distance(sensor_i, sink_j) < l_{max} \\ S_{Y'} & distance(sensor_i, controller_j) < l'_{max} \end{cases} \tag{7}$$

To define solution spin variables, we use X and X'. Thus:

$$Solution\ Spin\ Variables = \begin{cases} S_X & sink_i\ is\ selected\ or\ not \\ S_{X'} & controller_i\ is\ selected\ or\ not \end{cases} \tag{8}$$

Therefore, each configuration of system consists of a complete set of spin variables (S_X, $S_{X'}$, S_Y, $S_{Y'}$) which will be represented by $\{S_i\}$ or ϖ.

4.4.2. Potential Hamiltonian

In our considered problem, $H_{potential}$ consists of three parts: $H_{reliability}$, $H_{load\ balancing}$ and H_{cost}:

$$H_{reliability} = \left(k\,|T| - \sum_{\forall v_i \in T} \min\left\{k, \sum_{\forall s_j \in A_s} S_{X_j} S_{Y_{ij}}\right\}\right) + \left(k'\,|T| - \sum_{\forall v_i \in T} \min\left\{k, \sum_{\forall c_j \in A_c} S_{X'_j} S_{Y'_{ij}}\right\}\right) \tag{9}$$

$$\begin{aligned} H_{load\ balancing} = &\left(\sum_{\forall s_j \in A_s} \max\left\{0, \left[\sum_{\forall v_i \in T} S_{X_j} S_{Y_{ij}} \frac{w_i}{\sum_{\forall s_t \in A_s} S_{X_t} S_{Y_{it}}} - \frac{W}{k-1}\right]\right\}\right) \\ &+ \left(\sum_{\forall c_j \in A_c} \max\left\{0, \left[\sum_{\forall v_i \in T} S_{X'_j} S_{Y'_{ij}} \frac{w'_i}{\sum_{\forall c_t \in A_c} S_{X'_t} S_{Y'_{it}}} - \frac{W'}{k'-1}\right]\right\}\right) \end{aligned} \tag{10}$$

$$H_{cost} = \sum_{\forall s_i \in A_s} \sum_{\forall c_j \in A_c} \zeta_{s_i,c_j} S_{X_{s_i}} S_{X'_{c_j}} \tag{11}$$

Hence, for each replica: $H_{potential} = H_{reliability} + H_{load\ balancing} + H_{cost}$.

4.5. Kinetic Hamiltonian

In [20] a formula is given to calculate $H_{kinetic}$ in the 3SAT problem. As our considered problem is NP-complete and based on Cook-Levin theorem, any NP-complete problem can be reduced to SAT in polynomial time. Thus, we can use the same formula here to calculate $H_{kinetic}$:

$$H_{kinetic} = -J_\Gamma \sum_i S_{i,\rho} S_{i,\rho+1} \; : \; \forall \rho \in P \tag{12}$$

$$J_\Gamma = -\frac{T_q}{2} \ln \tanh\left(\frac{\Gamma}{PT_q}\right) \tag{13}$$

Total solution energy is:

$$H = \frac{1}{P} \sum_{\rho=1}^{P} H_{potential}(\{S_i\}) - J_\Gamma \sum_{\rho=1}^{P} \sum_i S_{i,\rho} S_{i,\rho+1} \tag{14}$$

The pseudo-code of the proposed algorithm is presented in Algorithm 1.

Algorithm 1: Path Integral Monte Carlo Quantum Annealing

```
   Result: Return a configuration with low cost
 1 Initialize $T = T_0$, $\Gamma = \Gamma_0$;
 2 Generate P configurations: $\{S_i\}$ or $\varpi$;
 3 do
 4     foreach replica $\rho \in P$ do
 5         repeat
 6             $\varpi'$ = Generate neighbor of configuration $\varpi$;
 7             $H_{potential}$ = Calculate potential energy of configuration $\varpi'$;
 8             $H_{kinetic}$ = Calculate kinetic energy of configuration $\varpi'$;
 9             $H = H_{potential} + H_{kinetic}$;
10             $\Delta H_{potential} = H_{potential} - H'_{potential}$;
11             $\Delta H = H - H'$;
12             if $\Delta H_{potential} < 0$ or $\Delta H < 0$ then
13                 $\varpi = \varpi'$;
14             else
15                 With probability $e^{\frac{-\Delta H}{T}}$ : $\varpi = \varpi'$;
16             end
17         until Number of iterations $< N_{QA}$;
18     end
19     Update quantum tunneling field: $\Gamma = \Gamma \times \sigma$;
20     Update temperature: $T = T \times \mu$;
21 while $\Gamma > \Gamma_f$;
```

5. Performance Evaluation

To conduct the experiments, we investigate three different sizes of the system where the first benchmark includes 50 sensors, the second benchmark 100 sensors, and the last benchmark includes 150 sensors. To evaluate the proposed algorithm, we not only implemented quantum annealing, but also classic SA and the recently proposed method in the literature (dubbed as PACSA) published in 2018 by Faragardi et. al. [13] were implemented.

For each size of the network (50 sensors, 100 sensors, and 150 sensors), all the three methods (QA, SA, and PACSA) were run 20 times to reach a 95% confidence interval. All algorithms are implemented in Java and running on a laptop with Mac OS, Core i5 2.9 GHz, and 8GB memory. The source code is publicly available on Github [21].

Table 1. Algorithm Parameters

Parameter	Description	Value	Parameter	Description	Value
$Cost(S_i)$	**Cost of Sinks**	[1,6] \$	T_0	**Initial Temperature**	100
$Cost(C_i)$	**Cost of Controllers**	[3,10] \$	T_f	**Final Temperature**	1
α	**Cost Reduction Factor**	0.75	μ	**Temperature Cooling Rate**	0.9
W	**Workload Capacity of Sinks**	30 Bytes/sec	N_{SA}	**SA Monte Carlo Steps**	50
W'	**Workload Capacity of Controllers**	30 Bytes/sec	tr	**QA Trotter Replicas**	50
$lmax$	**Maximum Length of Sinks**	3	$\Gamma 0$	**QA Initial Tunneling Field**	1
$l'max$	**Maximum Length of Controllers**	2	Γf	**QA Final Tunneling Field**	0.5
k	**k Sink Covered**	6	σ	**Tunneling Field Decreasing Rate**	0.95
k'	**k Controller Covered**	6	N_{QA}	**QA Monte Carlo Steps**	100

Table 2. Average execution time of the algorithms in 20 runs

Sensors	Candidate Controllers	Candidate Sinks	SA Execution Time	PACSA Execution Time	QA Execution Time
50	**5**	**10**	**48 Sec**	**14.22 Sec**	**72 Sec**
100	**10**	**20**	**150 Sec**	**108.3 Sec**	**225 Sec**
150	**15**	**30**	**216 Sec**	**676.5 Sec**	**324 Sec**

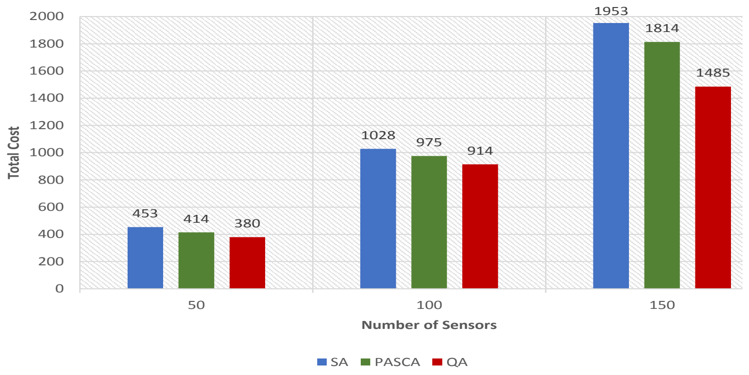

Fig. 2. Total deployment cost of the network for the considered benchmarks.

Table 3. Cost improvement percentage against QA

Sensors	Candidate controllers	Candidate sinks	SA	PACSA
50	5	10	16%	8%
100	10	20	11%	6%
150	15	30	23%	18%

The algorithm parameters have a substantial impact on the performance of meta-heuristic algorithms. To select the best value for the parameters, we checked all possible values in a reasonable range for each parameter. The algorithm parameters are listed in Table 1. The results achieved by all three methods in terms of the total deployment cost of the network for the considered benchmarks are illustrated in Fig. 2. Furthermore, to clearly demonstrate the performance enhancement of our proposed method against other baselines (i.e., SA and PACSA), we present Table 3 listing the percentage of improvement in the deployment cost of the network achieved by our method against SA and PACSA for the considered benchmarks. For example, for the first benchmark (i.e., the benchmark with 50 sensors), QA outperforms SA by 16% while outperforms PACSA by 8%. The execution time of the algorithms are listed in Table 2.

6. Conclusion

In this paper, we have looked into the placement problem for both multiple sinks and multiple controllers in WSNs to reduce deployment cost, subject to both reliability and timeliness. We formulated the problem as an optimization problem. A Quantum-Annealing-based algorithm was proposed to address the problem. The results indicated that on average, the proposed method outperforms the classic SA and PACSA by 17.7% and 10.7% respectively. For future work, we plan to investigate the use of multiple mobile sink nodes to collect the sensors data.

Acknowledgements

The work presented in this paper is supported by the Swedish Foundation for Strategic Research via the project Future Factories in the Cloud (FiC) project, and by the Swedish Research Council (Vetenskapsrådet), through starting grant no. 2018-04582 via the MobiFog project, and by the Swedish Knowledge Foundation (KKS), through Prospekt grant via the FlexiHealth project.

References

[1] Hamid Reza Faragardi, Hossein Fotouhi, Thomas Nolte, and Rahim Rahmani. A cost efficient design of a multi-sink multi-controller wsn in a smart factory. In *High Performance Computing and Communications; IEEE 19th International Conference on*, pages 594–602. IEEE, 2017.

[2] Chuanxin Zhao, Changzhi Wu, Xiangyu Wang, Bingo Wing-Kuen Ling, Kok Lay Teo, Jae-Myung Lee, and Kwang-Hyo Jung. Maximizing lifetime of a wireless sensor network via joint optimizing sink placement and sensor-to-sink routing. *Applied Mathematical Modelling*, 2017.

[3] Huping Xu, Jiajun Zhu, and Bang Wang. On the deployment of a connected sensor network for confident information coverage. *Sensors*, 15(5):11277–11294, 2015.

[4] Michael Chien-Chun Hung and Kate Ching-Ju Lin. Joint sink deployment and association for multi-sink wireless camera networks. *Wireless Communications and Mobile Computing*, 16(2):209–222, 2016.

[5] Lanny Sitanayah, Kenneth N Brown, and Cormac J Sreenan. Planning the deployment of multiple sinks and relays in wireless sensor networks. *Journal of Heuristics*, 21(2):197–232, 2015.

[6] Wint Yi Poe and Jens B Schmitt. Placing multiple sinks in time-sensitive wireless sensor networks using a genetic algorithm. In *MMB*, pages 1–15. VDE, 2008.

[7] Haidar Safa, Wassim El-Hajj, and Hanan Zoubian. A robust topology control solution for the sink placement problem in wsns. *Journal of Network and Computer Applications*, 39:70–82, 2014.

[8] Sirnivas Rao, Haider Banka, and Prasanta Jana. PSO-based multiple-sink placement algorithm for protracting the lifetime of wireless sensor networks. In *Proceedings of the Second International Conference on Computer and Communication Technologies*, pages 605–616. Springer, 2016.

[9] Donghyun Kim, Wei Wang, Nassim Sohaee, Changcun Ma, Weili Wu, Wonjun Lee, and Ding-Zhu Du. Minimum data-latency-bound k-sink placement problem in wireless sensor networks. *TON*, 19(5):1344–1353, 2011.

[10] Brandon Heller, Rob Sherwood, and Nick McKeown. The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 7–12. ACM, 2012.

[11] David Hock, Steffen Gebert, Matthias Hartmann, Thomas Zinner, and Phuoc Tran-Gia. Poco-framework for pareto-optimal resilient controller placement in sdn-based core networks. In *Network Operations and Management Symposium (NOMS)*, pages 1–2. IEEE/IFIP, 2014.

[12] K. Mousavi, S. Fazliahmadi, N. Rasouli, H. R. Faragardi, H. Fotouhi, and T. Fahringer. A budget-constrained placement of controller nodes for maximizing the network performance in sdn-enabled wsns. In *International Conference on Communication, Management and Information Technology (ICCMIT)*, 2019.

[13] Hamid Reza Faragardi, Maryam Vahabi, Hossein Fotouhi, Thomas Nolte, and Thomas Fahringer. An efficient placement of sinks and sdn controller nodes for optimizing the design cost of industrial iot systems. *Software: Practice and Experience*, 48(10):1893–1919, 2018.

[14] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[15] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[16] Diego de Falco and Dario Tamascelli. An introduction to quantum annealing. *RAIRO - Theoretical Informatics and Applications*, 45, 07 2011.

[17] Olawale Titiloye and Alan Crispin. Quantum annealing of the graph coloring problem. *Discrete Optimization*, 8(2):376–384, 2011.

[18] Kazue Kudo. Constrained quantum annealing of graph coloring. *Physical Review A*, 98(2):022301, 2018.

[19] Roman Martoňák, Giuseppe E Santoro, and Erio Tosatti. Quantum annealing of the traveling-salesman problem. *Physical Review E*, 70(5):057701, 2004.

[20] Demian A Battaglia, Giuseppe E Santoro, and Erio Tosatti. Optimization by quantum annealing: Lessons from hard satisfiability problems. *Physical Review E*, 71(6):066707, 2005.

[21] Quantum annealing based algorithm to node placement in wsns.