Search-Based Parameter Tuning on Application-Level Load Balancing For Distributed Embedded Systems

Ipek Caliskanelli, Leandro Soares Indrusiak
Department of Computer Science
University of York, UK
{ic539,leandro.indrusiak}@york.ac.uk

***Abstract*—Load balancing techniques for distributed embedded systems tend to be very parameter-rich, and finding adequate parameters for a given scenario is not trivial. Furthermore, the parameter values that are adequate for one scenario are rarely applicable to another that has, for instance, a different application profile or network topology. In this paper, we present a search-based parameter tuning approach that aims to automate the process of configuring a parameter-rich load balancing technique. It considers the service availability and energy dissipation figures obtained by each configuration of the load balancing technique, and uses those values to explore the parameter space towards optimised solutions. To accelerate the search, we also present a number of improvements to the simulator used to evaluate each configuration. The proposed parameter tuning approach is then evaluated by analysing the best configurations it can find for several scenarios, and we use Principle Component Analysis to identify which of the parameters have the most critical effect on the quality of the solutions.**

I. Introduction

Wireless Sensor Networks (WSNs) consist of sensor nodes that have limited energy supply, sensor devices, a short-range radio and on-board processing capabilities. Techniques for low power design with high quality of service (QoS) have been attracting a great deal of attention in the last decade [1] to manage the limited resources of the WSNs in the most effective manner.

One way to cope with the energy limitations and processing capabilities of WSNs is to balance the processing load over the entire network. This, therefore, requires us to distribute responsibility for performing work across the entities of a distributed system such as a sensor network. In the past, many static techniques have been proposed to identify and distribute processing responsibility for load balancing. However, static techniques are poor and inefficient due to their lack of knowledge of the network. On the other hand, dynamic techniques, either at runtime or on-demand, have better control over the networks. This is highly desirable for load balancing procedures that can extend the network lifetime, as well as increase both service availability and QoS.

In our initial work [2],[3], we have proposed a highly dynamic and robust load balancing technique which can cope with the adaptability and self-organisation issues of WSNs. Our *pheromone signalling based load balancing technique*, *PS*, is inspired by the pheromone stimulation process that is responsible for queen selection in beehives. By adapting that process into a distributed and asynchronous resource management protocol, we have enabled each node to decide whether it should be responsible for a given service request using only information available locally. Ideally, such a distributed decision making process would enable all services to be available when requested, and balance the service load across the network to avoid excessive energy consumption by a few overloaded nodes without significant sacrifice on service availability. In adaptive algorithms like *PS*, performance metrics highly depend on the set of selected parameters and as a result finding a good set of parameters is of key importance. However, tuning the PS's parameters is not easy and requires time due to its complicated biological background.

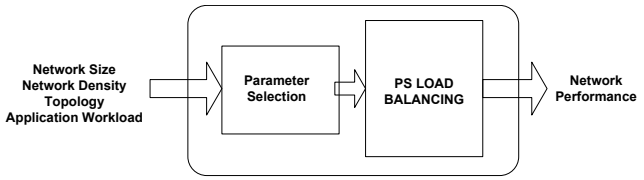


Fig. 1: Automating the parameter selection for the PS.

The main goal of this research is to automate the PS parameter selection process. For this purpose, we have created a parameter selection tool to find a set of parameters which will provide the highest possible network performance (maximum service availability and minimum energy consumption) for the *PS load balancing technique* for *any* given network configuration regardless of the network size, density, topology, or the application workload. In our initial work, we have illustrated the effects of the important parameters on *PS* using a variety of different sets of parameters. Intuitive parameter selection (based on trial and error) achieved a high level of service availability whilst minimising energy consumption for a fixed network schema (mesh network). However, the set of parameters used for the PS algorithm is biologically inspired, and trying to understand the effects of each parameter requires time and effort. Moreover, as the network configuration changes the set of parameters that is feed to the PS algorithm also has to change. This makes the PS algorithm

less likely to be used by end-users even though it is an effective technique. As a result, we have decided to implement a systematic technique to tune the parameters for the given network schema. As Fig. 1 illustrates, the parameter selection tool accepts as inputs the network configurations and outputs the set of parameters that is used by the PS algorithm for the given network schema. The parameter selection tool applies a well-known search metaheuristic *Simulated Annealing (SA)* to remove the complexity of the PS technique for the end-user.

The second goal of this research is to accelerate the search metaheuristic towards finding fit solutions quicker by improving the simulation infrastructure. We use *SA* with a fitness function that considers service availability as well as network lifetime, in order to measure of the "goodness" of the parameters. Finding the best way to investigate the fitness of the search and demonstrate the benefits of improved simulation infrastructure is a key part of this research. For this purpose, the fitness evaluation of the proposed global search uses a fast, system-level simulation model, to exploit the beneficial points of the proposed simulation infrastructure like time efficiency and accuracy. Advantages of our improved fast simulator versus the initial simulator proposed in [2], [3] (and vice versa) are discussed and analysed for the *SA* search metaheuristic.

II. Related Work

A. Metaheuristic Search

Metaheuristic search techniques are a set of generic algorithms that are concerned with searching for (near) optimal solutions to a problem within a large multi-modal search space and have been used to find solutions to many NP-complete problems [4]. Clarke et al [4] divide search metaheuristic techniques into two: local search techniques and evolutionary search using genetic algorithms. Hill-climbing, simulated annealing and tabu search is categorised as local search, whereas evolutionary search using genetic algorithms (GA) is categorised separately as population based search metaheuristics. All the moves applied in *SA* are based on relative desirability/ undesirability of particular local information. They are easy to program and time efficient due to their low level of complexity. Unlike *GAs*, *SA* is applied to single individuals rather than a population. Since the solution domain is sampled through all the population in GA, evaluation of the fitness function might be costly and take a long time without converging to the global optima. Although *exhaustive search* algorithms cover the whole search space, they are impractical, computationally unaffordable and time inefficient for multi criteria metrics with several parameters. Therefore, we propose search-based parameter tuning based upon *SA*, which is known as practical, time and cost efficient.

SA is a stochastic optimization procedure for obtaining approximate solutions to combinational optimisation problems. *SA* is an iterative process where the system starts rearranging itself until an improved configuration (the particular solution) is found. Once the solution is found, then that particular solution becomes the new starting point for the further rearrangements. This will continue until the system achieves the stopping criteria, where no further improvements can be found. This simple idea is often used to find feasible solutions which can converge to an optimal solution. Kirkpatrick et al [5] took the annealing concept and applied it to optimisation problems, and since then *SA* has been successfully used in many diverse fields of computer science; artificial neural networks [6], pattern detection [7], NoCs [8], mobile ad hoc networks [9], model-driven engineering [10] and software verification [11], as well as WSNs. Some of the applied work on WSNs in this section is as follows. Kannan et al [12], [13] present a simulated annealing metaheuristic for WSNs that aims to localise network nodes accurately for centralised architectures. Their technique reduces large scale localisation errors (flip ambiguity) significantly by applying static accurate position determination by SA. The fitness function measures the sum of squared distance over all pairs. Slijepcevic and Potkonjak [1] present a search heuristic which aims to find the optimal number of network nodes. Deterministic node placement in clusters is also implemented to maintain high network coverage with minimum energy consumption. The fitness function subtracts an approximate measurement of the minimally constraining heuristic from the most constrained heuristic in terms of number of nodes. PSO [14] is an evolutionary programming technique inspired from swarm optimization to minimise energy consumption while maximising the total data gathering of the group of sensors and equalising the number of the nodes on each cluster is the objective of the required technique. Wang at el [15] exploits a novel fault-tolerant distributed multiclass classification fusion approach using error correcting codes (DCFECC) that provides excellent fault-tolerance in WSNs. Park and Srivastava [16] present a centralised task decomposition, transformation and assignment solution using simulated annealing to maximize the lifetime of the network and/or minimize the latency. Their work also includes a distributed task migration algorithm at run-time which occurs based on the results of the SA search. The fitness function measures total energy consumption (which is the sum of communication and computation consumption for all tasks) weight of latency, weight of maximum energy consumption and penalty. Montemanni et al [17] combines mixed integer programming with a simulated annealing heuristic to achieve minimum power consumption for broadcasts. The Euclidean distance between nodes, channel loss exponent, and power required to transmit from source to destination are calculated and used as a parameter to measure the sum of the transmission powers of all the nodes.

B. Load Balancing

The concept of load balancing in WSNs refers to distributing work load over the network components. This concept has been applied at both the network level, and the application level, and it has significant impact on low power consumption. At the network level, work load refers to packet transfer and communication, whereas at the application level it refers to execution and processing the data (e.g. monitoring the environment, sensing the temperature). Both network and

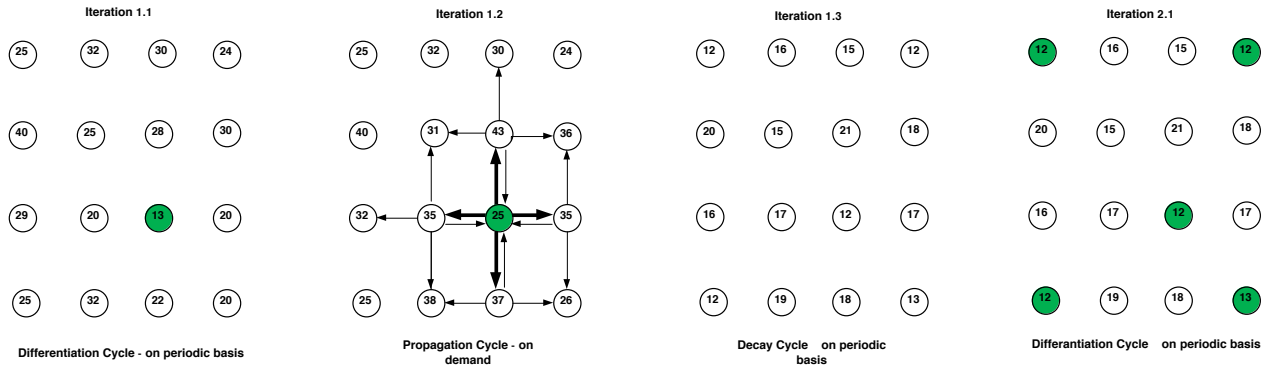


Fig. 2: Three important stages of PS algorithm: Differentiation, Propagation and Decay Cycles. *Green Indicates Queen Node

application level load balancing target efficient utilisation of resources to extend the network lifetime. Research that has focused on network level load balancing considers successful packet delivery ratio, error rate, latency, link failures and bandwidth usage. In [18] a reliability-based distributed routing algorithm is implemented to achieve low-power consumption. This dynamic approach keeps the network responsive to link dynamics by sending broadcast beacon packets periodically and updating route information. Trumler et al [19] present *AMUN* – self-organising middleware that distributes the work load of service-oriented applications. Their proposed technique is inspired by the human hormone system and provides near optimal resource usage for large-scale systems. The distributed algorithm uses organisational information about hormones to piggy-back on top of the messages that are exchanged between nodes. In the Beehive protocol [20] packets search for efficient routes through an IP network in a process modelled after the foraging behaviour of bees. Research that has focused on application level load balancing distributes the work load by deciding which node should execute the requested application requirement among network entities. A static technique presented by Zeng et al [21] aims to improve network response time and limit energy usage. However, this approach results in overloading the network, as the mapping cannot adapt effectively to network conditions. Miorandi et al [22] present a genetic approach, involving genome mutation and crossover. DNRS [23] limits energy consumption while improving reliable event detection. BTMS [24] uses zygote differentiation to extend the network lifetime whilst speeding up task mapping and scheduling. Homogeneous nodes begin in a default state and within time nodes differentiate themselves dynamically to perform distinct tasks according to their location.

In our initial work, PS, we presented a dynamic load balancing technique that is applied on the application level at runtime. PS is inspired by the pheromone signalling mechanism found in bees and provides distributed WSN control that depends on local information only. PS is unique; unlike most of the MAC layer approaches [25], [26], [20], PS is applied at the application level. Briefly, PS is applied in three steps: a differentiation cycle, a propagation cycle and a decay cycle as Fig. 2 shows. The differentiation cycle occurs on a periodic basis at every T_{QN} time unit on each node of the network. Each node decides to volunteer to be the node responsible for the task/event execution process; these are called *queen nodes* (QN). QN selection is based on each node's pheromone level, h_i, and nodes who have a lower h_i than a pre-defined threshold, $threshold_{QN}$, become QNs who then run the execution process. The second step of the algorithm is the propagation cycle and occurs on demand just after the differentiation cycle. When nodes become QNs, they start propagating some level of pheromone to the environment which indicates the resource usage in that part of the sensor field. A high pheromone level indicates high resource usage in that certain part of the network accordingly. The third step of the PS is called the decay cycle. This cycle occurs on a periodic basis on every node at every T_{DECAY} time unit and it indicates the elapsed time in the environment. As pheromones disappear (decay) over time, the load balancing algorithm reflects that process by decreasing each node's pheromone level. In Fig. 2 nodes are represented with circles, and the numbers in the circles represents the pheromone level of each node. QN's coloured into green, whereas pheromone propagation is illustrated by arrows. Darker arrows indicates high levels of pheromone propagation, whereas thinner arrows represent low level. For a more detailed explanation, see [2], [3].

III. Problem Statement

This paper considers two main challenges. The first objective of this research is to automatically tune the parameters of the *PS* algorithm for a given network configuration in order to maximise service availability and minimising energy consumption. The second challenge is to accelerate the evaluation of the search technique towards finding optimal parameter configurations quickly by speeding up the simulation infrastructure. We now define the fitness function we will use to evaluate parameter selection throughout the paper. Our performance metrics have been defined as:

1) *service availability*: the number of services that are successfully completed divided by the total number of requested services within a period of time;

2) *total energy consumption*: the sum of communication and computation energy consumption within a period of time.

In this research, a *service* is defined as the composition of a number of inter-communicating tasks, and therefore a service is considered to be successfully *detected* only if all of its tasks are executed by the nodes. Accordingly, our fitness function considers both the service availability and the number of alive nodes. We target high service availability and aim to minimise the number of time intervals (weeks) without service (based on the number of the alive nodes) whilst also minimising the total energy cost.

IV. Search-Based Parameter Tuning for PS

In SA algorithms positive improvements are always accepted, whereas negative improvements may be accepted probabilistically, depending on the temperature T. According to SA theory, the worse the move the less likely it is to be accepted. Similarly, a negative move is less likely to be accepted the cooler the temperature is. The temperature T starts with a high value and gradually cools as the algorithm progresses. Certain design decisions must be taken in order to run the simulated annealing algorithms: 1) SA configuration values; 2) Initial solution; 3) Neighbourhood generation; 4) Fitness function. SA configuration values are temperature-related control parameters like the initial temperature T_0 and the temperature at any time during SA process T_k. The *cooling rate*, which is often selected between 0 and 1, indicates how quickly the temperature decreases. The frequency at which cooling is applied also affects the search. In some SA algorithms cooling is applied once in every neighbourhood generation, whereas researchers who prefer to explore larger search space applies the cooling rate more than one neighbourhood generation. Often in SA algorithms, the parameters for the initial solution are set to default values, such as minimum values within the search space. Neighbourhood selection varies depending on size of the search space based on number of parameters and user preference. It is more likely that SA will find a good set of parameters with highly populated neighbourhood set, however a highly populated neighbourhood set will also increase the cost of the search. The definition of the fitness function varies with the application. For our SA search algorithm, our design decisions are as follows. We set the initial temperature T_0 to 100, and define the stopping criteria as $T_k < 1$. The cooling rate is set to 0.9 and is applied once in every twenty neighbourhood generations in order to explore the search space wider. Each solution is a set of assignments to the four key parameters of the PS technique (as explained in the II-B) together with temperature which is represented as a tuple $S = \{T_{DECAY}, T_{QN}, threshold_{QN}, QN_{INITIAL}, T_k\}$. The parameters of each solution $S_i \in S$ are tuned within the given range in Table I with the provided step values.

The parameters of the initial solution are set to minimum default values presented in Table I. Depending on the scenarios described in Section V, 8-16 neighbours are generated in every neighbourhood generation. Once the fitness evaluation of the entire neighbourhood is complete, candidate solutions are ranked based on their fitness. Once again, depending on the scenario, one of the candidate solutions is then accepted and becomes the new starting point for the further rearrangements. To reduce the complexity of the fitness function evaluation, normalisation is applied to the performance metrics. We defined our fitness function as the combination of total service availability and the minimum number of intervals without service detections. For a better understanding, we give an example to visualise our performance metrics first. Simulation results of one solution are shown in Fig. 3. Simulation time is set to 15 weeks and we evaluate the percentage of detected events and percentage of alive nodes weekly for each solution. Illustrated in Fig. 3, the definition of the fitness function is the combination of total service availability (the sum of percentage of detected events), which is 917 and the number of intervals without service detections, 3. The best solution is the one who has the highest total service availability and lowest number of intervals without service detections. Demonstrated by the given example in Fig. 3, our proposed algorithm targets maximum service availability and minimum energy consumption by defining a simple, but clever, fitness function.

TABLE I: Parameters setting for the SA

Parameters	Range	Step Value
T_{DECAY} (seconds)	1000-15000	1000-4000
T_{QN} (seconds)	2000-60000	1000-4000
$threshold_{QN}$	3-40	1-3
$QN_{initial}$	2-50	1-3

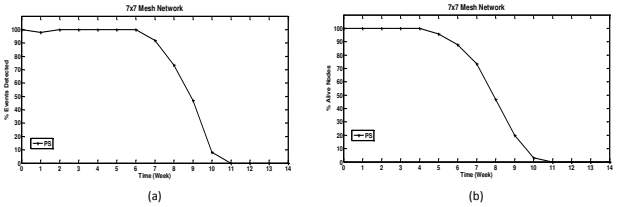


Fig. 3: % events detected and % alive nodes.

The performance of metaheuristic search algorithms is often highly dependent on the amount of the search space analysed by the algorithm. In SA algorithms, SA configuration values (mainly temperature-related parameters, the size of the generated neighbourhood set, and the metaheuristic-related parameters, such as selection range and step size) determine how much of the search space will be covered by the search metaheuristic. Covering a large amount of the search space increases the possibility of finding a good and accurate set of parameters, however it will also increase the evaluation time and cost of the search. By reducing the evaluation time of the search metaheuristic, computational cost will also be decreased since time and cost proportionally affect each other. In order to decrease the evaluation time and computational cost of our search metaheuristic, we developed the *SuperFast* simulator, which is explained in the next section.

V. Evaluation Environment

We have developed a fast, abstract simulator which allows us to cover quite a large search space within a short time. By reducing the evaluation time, we also avoid unnecessary computational cost, which is mentioned earlier. Our SuperFast simulator has been developed in Java. Accuracy and time evaluation tests of SuperFast simulator show that it is comparable with our initial system-level simulator, *Fast* [2], [3].

For our initial simulator, Fast, our design objective was a time and performance efficient simulator that has high level of accuracy. Detailed discussion on performance evaluation techniques and the reason behind on our choice of working on system-level simulator is explained in our previous work as well as the architecture of the simulator. However, we would like explain our initial design briefly to show the differences between Fast and SuperFast.

Fast is an event-driven simulator that uses the JavaSim library [27] to synchronise the events of the multi-threaded simulation engine. It has been developed in Java to use the advantage of the encapsulation of object-oriented programming. The multi-threaded nature of the simulator allow us to see multi-hop relations between nodes, task scheduling queues and energy consumption in terms of idle, processing and communication of the nodes. Contention on wireless channels was not considered during development due to the level of complexity it would add.

We started working on SA based parametric analysis using Fast. However, due to the complexity of the search process, required memory space for the hardware components and most importantly the time factor, we decided to create a more abstract simulator than Fast which will allow us to work faster. Removing the complexity of concurrent programming, and lowering the accuracy in certain limits reduced the computational time and the cost of the work, without sacrificing significant accuracy. In Table II features of simulators are analysed from component considerations and search engine

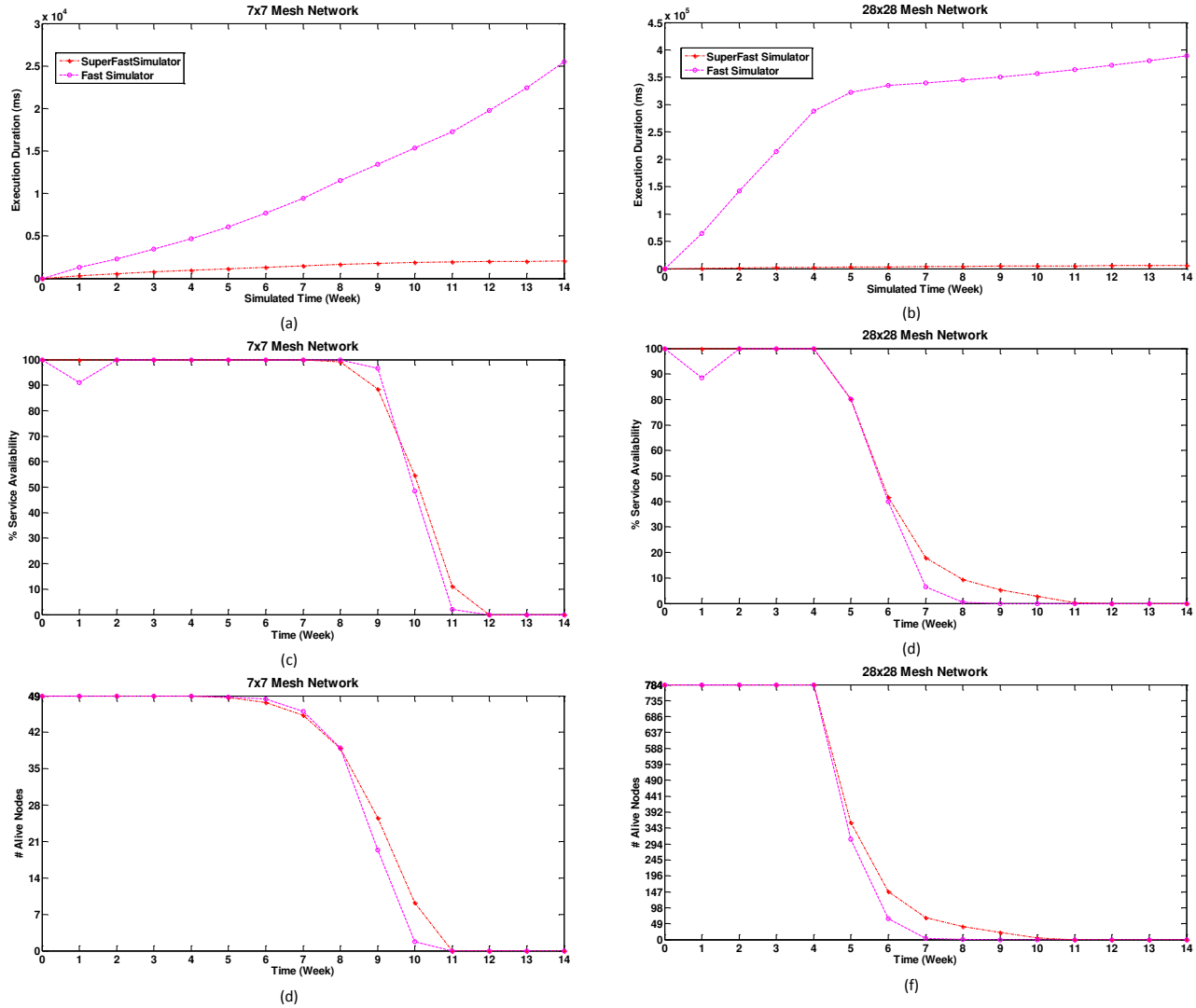


Fig. 4: Simulator comparison of execution duration, % service availability and # active nodes.

TABLE II: Comparison of Simulator

	Features	Fast	SuperFast
Consideration	multi-hop	✓	X
	scheduling queues	✓	X
	contention on wireless channel	X	X
	energy consumption while idle	✓	✓
	energy consumption while processing	✓	✓
	energy consumption while remapping tasks	✓	✓
	energy consumption for communication with sink	✓	X
SE	multi-threaded	✓	X
	JavaSim library	✓	X

points of the simulators. Disabling multi-threaded nature of the Fast simulator and implementing sequential programming instead of concurrent make SuperFast simulator time efficient. Since most of the energy consumption kept and taken under consideration in SuperFast, level of accuracy is preserved.

In Fig. 4, a detailed comparison between the Fast and SuperFast simulators in terms of time and accuracy is shown on 7x7 in Fig. 4 (a), (c), (d) and 28x28 in Fig. 4 (d), (e), (f). The PS algorithm behaves the same in terms of service availability and the number of the alive nodes in both small and large network for both Fast and SuperFast simulators. The differences occur due to the level of accuracy, where Fast is more realistic and accurate. Fig. 4 (a) and (b) illustrates the execution duration of both simulators, where SuperFast outperforms.

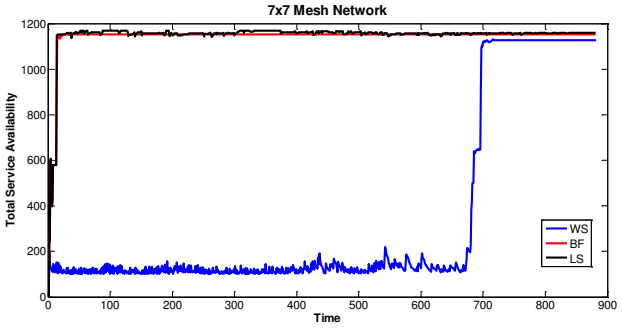
VI. Experimental Results

A. Verifying the Effectiveness of Parameter Selection Tool on a Known Schema

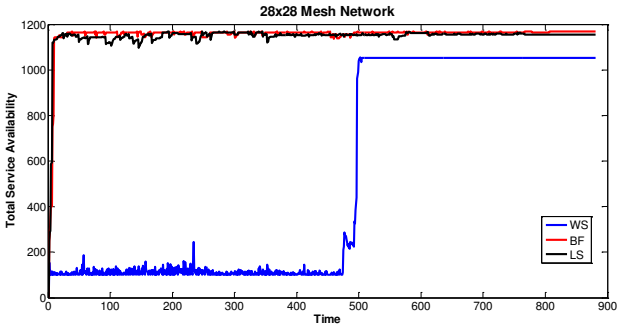
This section presents the initial set of experimental results. The goal of the first set of experiments is to demonstrate the behaviour of the SA algorithm towards selecting parameters that achieve high service availability and low energy consumption, and analyse how good our intuitive parameter search is based on small (7x7) and large (28x28) mesh network topologies. Three different scenarios have been prepared for this purpose.

1) *WS*: Represents the behaviour of the SA when the search uses a fixed small step size for the parameter values and the selection of the first encountered fitter neighbour;
2) *BF*: Represents the behaviour of the SA when a search changes step size dynamically (between large and small) depending on the improvement on fitness and selects randomly among fit neighbours;
3) *LS*: Represents the behaviour of the SA when a search changes step size dynamically (between large and small) depending on the improvement on fitness, ranks the fit neighbours and selects the fittest neighbour;

Initially the WS scenario is implemented and we found out that there is not only one optimal and fittest solution for the PS algorithm. As a result, we have decided to explore the larger search space within the shortest time as much as possible. Fig. 5 (a) on 7x7 Mesh Network, (b) on 28x28 Mesh Network show improvements on cumulative total service availability



(a)



(b)

Fig. 5: Improvements on total service availability.

over time for WS, BF and LS. In the 7x7 Mesh Network initial implementation, WS achieves high total service availability in very low temperatures and the algorithm does not converge fast due to small step sizes and neighbour selection. Whereas, LS and BF search algorithms converge much faster as these scenarios allow larger step sizes when there is no improvement. In both networks, the LS scenario performs the best in terms of achieving higher service availability in a shorter time, since the algorithm ranks neighbours and picks the fittest neighbour. Although the WS scenario converges faster in a large network in comparison to a small network, all three scenarios behaves similarly on both networks in terms of achieving higher service availability and time consumption.

In Fig. 6 shows three sample solutions for (a), (c) on 7x7 mesh network, and (b), (d) on 28x28 mesh network. For both networks, S1 and S4 represent the intuitive parameters, where S2, S3, S5, S6 are encountered by LS scenario on SA algorithm. Although the encountered parameter sets do not outperform S1, the experiments show that 1) SA algorithm works; 2) our intuitive parameter selection is near-optimal.

Knowing the biological background and how the PS algorithm works, we suspect that not all the parameters have the same effect on the results. In order to understand the importance of each parameter, we have decided to apply *Principle Component Analysis (PCA)*. The main purpose of PCA is to maximise the variance of a linear combination of the variables in order to scale and rank them. PCA is an effective tool that performs dimensionality reduction in which

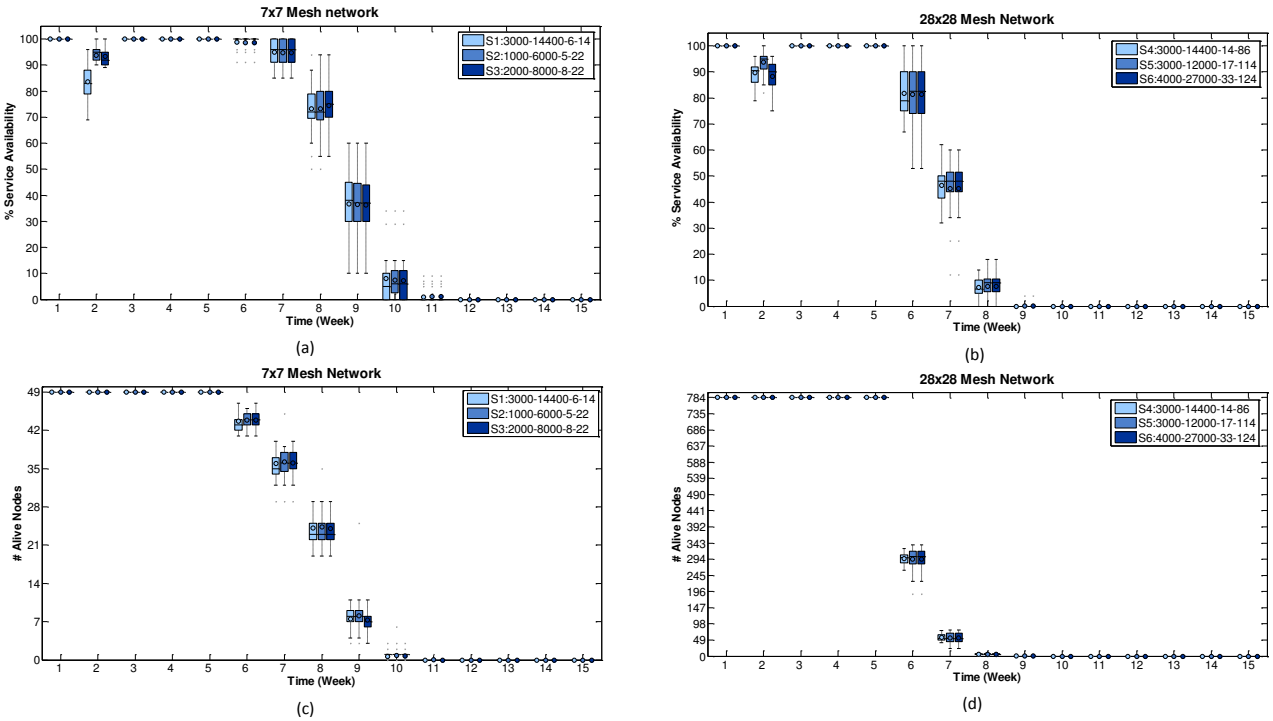


Fig. 6: Visualisation of sample solutions on mesh networks.

the original data is projected to the lower dimension spanned by leading eigenvectors of the covariance matrix of data [28]. We used the data that we gather from SA, and together with the results of randomly selected parameters to ensure that the results of PCA are not only based on the guided search metaheuristics but also contains some samples to cover the entire search space. Table III illustrates the impact of the four parameters of PS and highlights that only T_{DECAY} and T_{QN} play a role in determining the outcome of PS. The other two parameters depend on T_{DECAY} and T_{QN}.

TABLE III: PCA on PS.

Parameters	PCA results
T_{DECAY}	4.1285
T_{QN}	0.2564
$threshold_{QN}$	0.0000
$QN_{initial}$	0.0000

B. Applying Parameter Selection Tool Onto a New Schema

This section presents the second set of experimental results. The goal of the this set of experiments is to show the effectiveness of the parameter selection onto a new schema.

For our second set of experiments, we inspect the PS algorithm on a sparse network with 70 nodes. For this purpose, we have implemented an Eclipse-based (www.eclipse.org) graphical editor to allow end-users to easily design the network topology. Fig.7 illustrates the experimental results on the new network configurations and compares the intuitive parameter set, S7, with SA suggestions, S8 and S9. Although S7 performs well with a known topology (as shown in the previous section), it performs badly with the new network configuration as Fig.7 illustrates. On the other side, both S8 and S9 deliver some service availability until the 11th week. In Fig.7(b), the number of alive nodes for S8 outperforms, where the S9 does not improve the energy consumption as much, although both of their weekly service availability is almost the same. This shows that the S8 parameter set balances network load better than S9 parameter set. However, finding how and why is not easy since sparse networks are not uniform like mesh network topologies, and so we leave this as future work.

VII. Conclusion

This paper had two major goals: 1) automating the parameter tuning for the PS load balancing algorithm to address the trade-off between service availability and energy consumption; 2) accelerating the evaluation method towards finding fit solutions quicker by evolving the simulation infrastructure. We implemented a less accurate but very fast system level simulator, SuperFast, and analysed its accuracy and execution time as compared to our previous simulator. The advantages and disadvantages of these two simulators have been highlighted with the experimental results.

The need for systematic parameter selection on our load balancing technique to solve the trade-off between service availability and energy consumption in WSNs leads us to inspect search metaheuristics. We presented a search-based technique that uses *SA* to automate parameter tuning for our PS algorithm. In the first set of experiments, we have analysed the effectiveness of *SA* by creating three different scenarios on

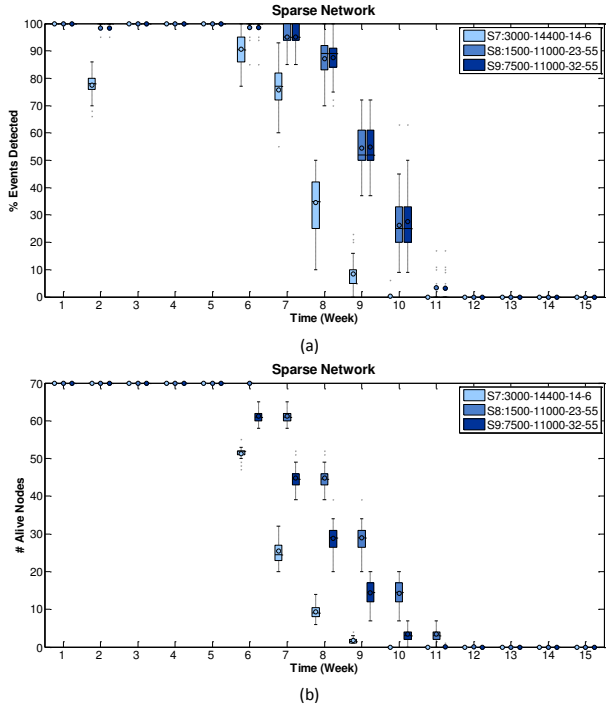


Fig. 7: Visualisation of sample solutions on sparse network.

the known network topology. The experimental results verify that there is more than one near-optimal solution. Based on the first set of experimental results, *PCA* has been applied to inspect the importance of each parameter of the parameter set. The results show that only two of the parameters are important for the PS technique. The search technique has been modified accordingly so that *SA* converges faster and is applied onto an untested network topology. The second set of experimental results compare our intuitive parameter set and sets of parameters found out by the *SA* on sparse network. Tuned parameters by *SA* outperform compared to our manual 'intuitive' set of parameters and consequently increase the network performance.

REFERENCES

[1] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Communications, 2001. IEEE Int. Conf. on*, vol. 2, 2001, pp. 472 –476 vol.2.

[2] I. Caliskanelli, J. Harbin, L. Soares Indrusiak, P. Mitchell, F. Polack, and D. Chesmore, "Runtime optimisation in wsns for load balancing using pheromone signalling," in *NESEA, 2012. 3rd IEEE Int. Conf. on*, dec. 2012.

[3] ——, "Bio-inspired load balancing in large-scale wsns using pheromone signalling," *Int. Journal of Distributed Sensor Networks*, May 2013.

[4] J. Clarke, J. Dolado, M. Harman, R. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd, "Reformulating software engineering as a search problem," *Software, IEEE Proc. -*, vol. 150, no. 3, pp. 161 – 175, june 2003.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[6] M. Gao and J. Tian, "Path planning for mobile robot based on improved simulated annealing artificial neural network," in *Natural Computation, 2007. 3rd Int. Conf. on*, vol. 3, aug. 2007, pp. 8 –12.

[7] K.-Y. Huang and Y.-H. Hsieh, "Very fast simulated annealing for pattern detection and seismic applications," in *Geoscience and Remote Sensing Symposium,2011. IEEE Int. Conf.*, july 2011, pp. 499 –502.

[8] Z. Lu, L. Xia, and A. Jantsch, "Cluster-based simulated annealing for mapping cores onto 2d mesh networks on chip," in *Design and Diagnostics of Electronic Circuits and Systems, 2008. 11th IEEE Workshop on*, april 2008, pp. 1 –6.

[9] D. Turgut, B. Turgut, R. Elmasri, and T. Le, "Optimizing clustering algorithm in mobile ad hoc networks using simulated annealing," in *Wireless Comm. and Networking, 2003. 2003 IEEE*, vol. 3, march 2003, pp. 1492 –1497 vol.3.

[10] J. R. Williams, F. R. Burton, R. F. Paige, and F. A. C. Polack, "Sensitivity analysis in model-driven engineering," in *Proc. 15th int. conf. on MODELS*. Springer, 2012, pp. 743–758.

[11] N. Tracey, J. Clark, and K. Mander, "Automated program flaw finding using simulated annealing," *SIGSOFT Softw. Eng. Notes*, vol. 23, no. 2, pp. 73–81, Mar. 1998.

[12] A. Kannan, G. Mao, and B. Vucetic, "Simulated annealing based localization in wireless sensor network," in *Local Computer Networks, 2005. IEEE Conf on*, nov. 2005, pp. 2 pp. –514.

[13] A. A. Kannan, G. Mao, and V. Branka, "Simulated Annealing based Wireless Sensor Network Localization," *Journal of Computers*, vol. 1, no. 2, pp. 15–22, May 2006.

[14] J. Tillett, R. Rao, and F. Sahin, "Cluster-head identification in ad hoc sensor networks using particle swarm optimization," in *Personal Wireless Communications, 2002 IEEE Inter. Conf. on*, dec. 2002, pp. 201 – 205.

[15] T.-Y. Wang, Y. Han, P. Varshney, and P.-N. Chen, "Distributed fault-tolerant classification in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 724 – 734, april 2005.

[16] H. Park and M. B. Srivastava, "Energy-efficient task assignment framework for wireless sensor networks," UC Los Angeles, Tech. Rep., 2003.

[17] R. Montemanni, L. Maria, G. Arindam, and K. Das, "The minimum power broadcast problem in wireless networks: a simulated annealing approach," in *In Proc. IEEE WCNC*, 2005.

[18] K. Daabaj, M. Dixon, and T. Koziniec, "Reliable load-balancing routing algorithm for wireless sensor networks," in *19th IEEE Int. Conf. on ICCCN*. IEEE, 2010.

[19] W. Trumler, T. Thiemann, and T. Ungerer, "An artificial hormone system for self-organization of networked nodes," in *Biologically Inspired Cooperative Computing*. Springer, US, 2006, vol. 216, pp. 85–94.

[20] H. Wedde, M. Farooq, and Y. Zhang, "Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior," in *Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3172, pp. 83–94.

[21] Z. Zeng, A. Liu, D. Li, and J. Long, "A highly efficient dag task scheduling algorithm for wireless sensor networks," in *Proc. of the 2008 The 9th Int. Conf. for Young Computer Scientists*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 570–575.

[22] D. Miorandi, L. Yamamoto, and P. Dini, "Service evolution in a bio-inspired communication system," 2006.

[23] B. Atakan and O. B. Akan, "Immune system based distributed node and rate selection in wireless sensor networks," in *Proc. BIONETICS*. ACM, 2006.

[24] Y. Hamouda and C. Phillips, "Biological task mapping and scheduling in wireless sensor networks," in *Communications Technology and Applications, 2009. IEEE Int. Conf. on*, oct. 2009, pp. 914 –919.

[25] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, vol. 2, Jan. 2000, p. 10.

[26] M. G. Gouda and T. M. McGuire, "Accelerated heartbeat protocols," in *18th Int. Conf. on Distributed Computing Systems*, 1998, pp. 202–209.

[27] M. C. Little. (2009) JavaSim. University of Newcastle upon Tyne.

[28] C. R. Rao, "The use and interpretation of principal component analysis in applied research," vol. 26, no. 4, pp. pp. 329–358, 1964.