# A Parallel Multi-objective Local Search for AEDB Protocol Tuning

Santiago Iturriaga
*Univ. de la República*
*Uruguay*
*siturria@fing.edu.uy*

Patricia Ruiz
*Univ. of Luxembourg*
*Luxembourg*
*patricia.ruiz@uni.lu*

Sergio Nesmachnow
*Univ. de la República*
*Uruguay*
*sergion@fing.edu.uy*

Bernabé Dorronsoro
*Univ. of Lille*
*France*
*bernabe.dorronsoro_diaz@inria.fr*

Pascal Bouvry
*Univ. of Luxembourg*
*Luxembourg*
*pascal.bouvry@uni.lu*

***Abstract*—Mobile ad hoc networks are infrastructureless communication networks that are spontaneously created by a number of mobile devices. Due to the highly fluctuating topology of such networks, finding the optimal configuration of communication protocols is a complex and crucial task. Additionally, different objectives must be usually considered. Small changes in the values of the parameters directly affects the performance of the protocol, promoting one objective while reducing another. Therefore, multi-objective optimisation is needed for fine tuning the protocol. In this work, we propose a novel parallel multi-objective local search that optimises an energy efficient broadcasting algorithm in terms of coverage, energy used, broadcasting time, and network resources. The proposed method looks for appropriate values for a set of 5 variables that markedly influence the behavior of the protocol to provide accurate tradeoff configurations in a reasonable short execution time. The new proposed algorithm is validated versus two efficient multi-objective evolutionary algorithms from the state of the art, offering comparable quality results in much shorter times.**

***Keywords*-multi-objective optimisation; local search; communication protocol; energy efficiency; mobile ad hoc networks**

## I. Introduction

Mobile ad hoc networks, also called MANETs, are self-organised networks spontaneously created between neighbouring devices without the need of any existing infrastructures. Due to the intrinsic broadcast nature of wireless networks, broadcasting is considered one of the most suitable protocols for disseminating messages. Indeed, many high level applications and even other protocols assume the existence of broadcasting as a low level operation and rely on its service. In wireless networks, these dissemination algorithms are generally associated with the broadcast storm problem [12], when all nodes receiving a message resend it.

However, due to the recently appearance of MANETs, and all the drawbacks inherited from them, the main problem in broadcasting is not only reducing the number of forwardings, but also overcoming all these undesirable aspects. Therefore, most of the existing dissemination protocols have different parameters for adapting to different network conditions and/or requirements. The performance of the algorithm highly depends on the setting used, that typically is chosen experimentally [2].

As already mentioned, the design of communication protocols for MANETs is a complex and critical task that directly impacts on the network performance. As a result of the unpredictable and highly changing topology, the behaviour of the protocol is highly sensitive to small changes in the set of configuration parameters. Therefore, fine tuning them for optimally configuring a communication protocol is a difficult task. Moreover, in these self-organised networks there is not a single goal to be satisfied but several (usually in conflict) like network resources, QoS, energy used, etc.

We present AEDB-MLS, a novel parallel multi-objective local search to look for the optimal configuration of an energy efficient dissemination algorithm, namely the adaptive enhanced distance based broadcasting algorithm (AEDB) [13]. AEDB is an energy-aware broadcasting algorithm that makes use of a cross-layer design to reduce the energy consumption. This protocol adapts its behaviour according to five different parameters. For finding optimal AEDB configurations, it was optimised in previous work [14] by two well known multi-objective evolutionary optimisation algorithms (MOEAs): CellDE [4] and NSGAII [3].

The optimisation process using those two MOEAs takes too long, thus, the utilisation of highly parallel and efficient techniques is required. Results confirm that AEBD-MLS highly speeds up the configuration process of the dissemination algorithm while finding competitive results. Moreover, the parallel local search is a useful technique to be included in any MOEA for obtaining even more precise solutions.

The contribution of this paper is threefold. First, a sensitivity analysis is done for designing efficient heuristic local search operators. Second, we design AEDB-MLS, the novel parallel multi-objective local search algorithm that can also be used within EAs or any other metaheuristics. Third, we confirm that AEDB-MLS allows speeding up the optimisation process, and finds competitive results.

The paper is organised as follows. Next section reviews the most relevant works applying metaheuristics for optimising protocols. Sect. III describes the AEDB protocol and the optimisation problem tackled in this work. The AEBD-MLS method is introduced in Sect. IV. The experimental analysis is reported in Sect. V, just before the presented results (Sect. VI). Finally, the conclusions and main lines for future work are formulated in Sect. VII.

## II. Related Work

We can find in the literature a few papers using metaheuristics for optimising protocols in MANETs. In all cases, the optimisation is an offline process that (usually) looks for the optimal configuration of the protocol to enhance some aspect of the network, such as QoS, the network use, or the energy used, as it is the case considered in our work. The first study in this line was probably the one proposed by Alba et al. [2], in which the DFCN broadcasting protocol for MANETs was optimised using multi-objective techniques.

Different metaheuristics have been applied to solve the minimum energy broadcast (MEB) problem in wireless ad hoc networks (PSO, EAs, ACO, hybrid EAs) [9], [20]. All of them are offline techniques that are limited to static networks.

Abdou et al. [1] optimised a probabilistic broadcasting algorithm in terms of the local density. The multi-objective optimisation focuses on minimising the channel utilisation as well as the broadcasting time.

In [6], a study on the optimisation of the AODV routing protocol for vehicular ad hoc networks is presented. And in [18], a parallel EA to optimise the energy used by the OLSR routing algorithm subject to acceptable QoS requirements is proposed also for VANETs. Both works deal with single-objective optimisation.

Ruiz et al. [14] optimise the performance of AEDB using two well known multi-objective algorithms, by maximising the coverage achieved in the dissemination process and minimising both the time and the energy used.

## III. AEDB Protocol Optimisation

The AEDB protocol [13] is a broadcasting algorithm that reduces the transmission power for disseminating a message, aimed at saving energy in both sparse and dense networks. A pseudocode is provided in Fig. 1. As any distance based broadcasting algorithm, nodes are candidates to forward the message if the distance to the source node is higher than a predefined threshold. Thus, there exists a forwarding area, and only nodes located in it are potential forwarders. In this case, a crosslayer technique is used to inform the upper layers about the signal strength of messages received. Therefore, the decision is not taken in terms of distance but power. This predefined value for the energy is called the *borders_threshold*. Before forwarding, the node sets a random delay.

AEDB tries to save energy by reducing the transmission power when forwarding the broadcasting message. The new transmission power is the one that reaches the furthest neighbour. The energy needed is estimated according to the reception energy detected in the beacons exchanged (every 1 second). In order to be aware of the nodes mobility, an extra fixed amount of energy, the *margin_threshold*, is added to the one estimated.

Figure 1. Pseudocode of the new Adaptive EDB.

```
Data: m: the incoming broadcast message.
Data: r: the node receiving broadcast message.
Data: s: the node that sent m.
Data: p: the received signal strength of m sent by s.
Data: pmin: the minimum signal strength received
from any s.
Data: potentialForwarders: # neighbors in the
forwarding area.
 1: if m is received for the first time then
 2:     calculate p;
 3:     update pmin;
 4:     if $pmin > borders\_Threshold$ then
 5:         r $\rightarrow$ drop message m;
 6:     else
 7:         waiting $\leftarrow$ true;
 8:         wait time rand $\in$ [delay interval];
 9:     end if
10: else if waiting then
11:     calculate p;
12:     if $p > pmin$ then
13:         update pmin;
14:     end if
15: end if
16: if $pmin > borders\_Threshold$ then
17:     r $\rightarrow$ drop message m;
18: else
19:     if # potentialForwarders $>$ neighbors_Threshold
        then
20:         estimate p to reach closest neighbor to
            borders_Threshold
21:     else
22:         discard s from the one hop neighbors list.
23:         estimate p to reach furthest neighbor
24:     end if
25:     transmit m;
26: end if
27: waiting $\leftarrow$ false;
```

In denser networks, the probability of having a node close to the transmission range limit is higher. This would highly reduce the energy saved in such networks. Indeed, when the network is very dense the connectivity is usually very high. Thus, reducing the transmission power allowing the loss of some one hop neighbours will save energy without any detriment in the performance of the broadcasting process.

Contrary, when the network is sparse, the node must maintain the network connectivity, as not doing so would make more difficult to spread a message through the whole network. AEDB is able to adapt its behaviour to the network density. If many nodes located in the forwarding area are detected (the *neighbours_threshold*), the transmission range is reduced and some one hop neighbours are discarded. The new furthest neighbor is the node located in the forwarding area that is the closest one to the source node. A more detailed explanation can be found in [13].

Next, we are describing the problem at hands.

## A. Problem Description

The performance of a broadcasting algorithm in MANETs is usually related to some standard metrics. We consider here the most common ones:

1) *coverage*: the number of devices that receive the broadcast message after the dissemination process;
2) *energy used* by the broadcast process: the sum of the energy every device consumes to forward the message;
3) number of *forwardings*: the amount of nodes that after receiving the broadcasting message decide to resend it;
4) *broadcast time*: the time needed to spread a message in the network, since the source node sends the message until the last node receives it.

From the point of view of the broadcasting algorithm designer, the higher the number of objectives the more complex the optimisation process and the decision making. Therefore, as it was previously done in [14], in this work AEDB is optimised in terms of three objectives: coverage, number of forwardings, and energy used. The broadcasting time is included as a constraint: any solution that takes longer than 2 seconds is no longer valid.

The main goal of this work is to tune the main AEDB parameters (*borders_threshold*, *margin_forwarding*, *delay*, and *neighbors_threshold*) using multi-objective techniques based on Pareto dominance in order to obtain the best possible protocol behavior, considering the three objectives explained and the constraint. The parameters are explained next:

- *borders_threshold* sets the size of the forwarding area. The higher the threshold, the higher the number of potential forwarders, the coverage, the network resources and the number of collisions;
- *margin_forwarding* is related to both the energy saved and the coverage achieved. The higher the margin value, the higher the coverage reached as well as the energy used;
- the *delay* interval sets the waiting time and also affects the behavior of the protocol. If the delay is very high, the time used to spread the message will be high, but if it is very small, the number of collisions will probably increase;
- *neighbors_threshold* fixes the minimum number of neighbors in the forwarding area needed to discard some nodes. It affects the use of the network and the energy used: the lower the value, the lower the energy used and the higher number of forwardings.

The optimisation problem is defined by function $F$ in Eq. 1, where $s$ is an AEDB configuration, simulated using the ns3 network simulator on 10 different networks, and $e$, $c$, $f$, and $bt$ stand for the average energy saved, coverage, number of forwardings, and broadcasting time out of the 10 simulations, respectively.

$$F(s) = \begin{cases} min\,\{e\} \\ max\,\{c\} \\ min\,\{f\} \end{cases} ;\text{s. t. } bt < 2 \qquad (1)$$

## B. Sensitivity analysis

In order to better understand the relationship between the AEDB parameters and the objective function values, we carried out a sensitivity analysis. This method is based on decomposing the variance of the output, as introduced in [15]. The Fourier Amplitude Sensitivity Test Fast99 [16] is used to compute the first order effects and interactions for each parameter. Parameters interaction occurs when the effect of the parameters on the output is not a sum of their linear effects. We considered a wide range of values for every parameter in the sensitivity analysis: $min\_delay \in [0, 5]$, $max\_delay \in [0, 5]$, $border\_threshold \in [0.0, 95.0]$, $margin\_threshold \in [0.0, 16.2]$, and $neighbor\_threshold \in [0, 100]$.

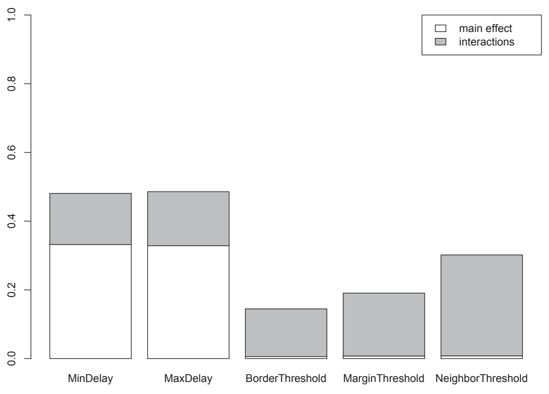
An example of the results obtained from the sensitivity analysis for the 300 devices network is shown in Figure 2. The influences of the variables on the different objectives are presented. We can see that the broadcast time is mainly influenced by $max\_delay$ and $min\_delay$. The coverage achieved is markedly affected by the $neighbor\_threshold$. While both $neighbor\_threshold$ and $border\_threshold$ are the parameters that influence most in the energy used and the number of forwardings.

Generally speaking, we notice that the $margin\_threshold$ has the lowest direct influence on any objective or density.The $delay$ interval strongly affects the broadcast time in any density. For the number of forwardings, the $border\_threshold$ and the $neighbor\_threshold$ show the highest direct influence. The energy used is affected mainly by the $border\_threshold$ and the $neighbor\_threshold$ and then by the $delay$, in that order, but the importance of the $border\_threshold$ decreases with density, while $neighbor\_threshold$ becomes more prominent. The same behavior is shown in case of the coverage, but for the densest network it is mainly affected by the $neighbor\_threshold$.

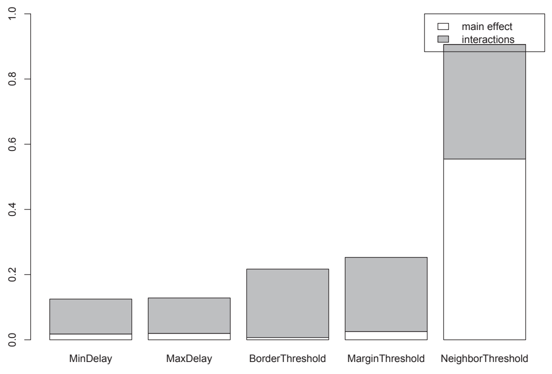
Table I summarises our main findings (symbols $\triangle$ and $\bigtriangledown$ indicate whether the variable should be increased or decreased, respectively, to optimise the corresponding objective; $\square$ stands for no interaction found).

## IV. Local Search

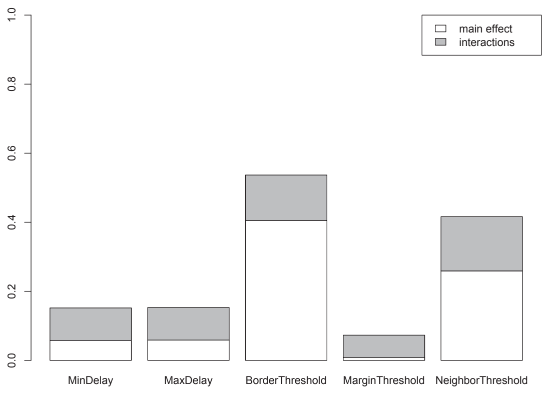
AEDB-MLS is a multi-start population-based local search algorithm that maintains several distributed populations. It



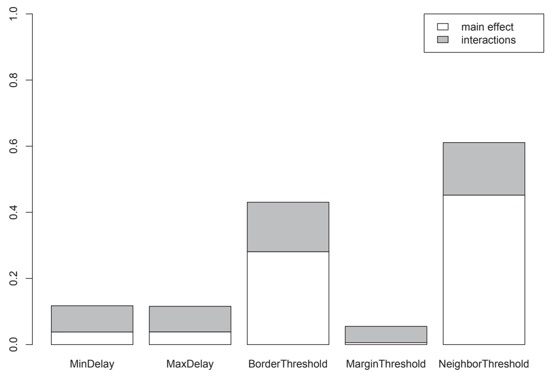
(a) Influence on the Broadcast Time



(b) Influence on the coverage



(c) Influence on the forwarding



(d) Influence on the energy

Figure 2. Influence of the parameters on the different objectives for the 300 devices network

Table I
SUMMARY OF THE PARAMETER SENSITIVITY ANALYSIS

| | **objective** | | | |
|---|---|---|---|---|
| **parameter** | coverage *maximise* | forwardings *minimise* | energy used *minimise* | broadcast time *constraint* |
| *border threshold* | △ yes | △ **yes** | △ **yes** | □ few |
| *delay* | ▽ few | △ few | ▽ few | △▽ **yes** |
| *margin threshold* | △ very few | △ no | △ no | □ no |
| *neighbors threshold* | △ **yes** | △ **yes** | △ **yes** | ▽ few |

is a massively parallel algorithm in which every solution in every population is simultaneously improved by the parallel application of an iterative local search procedure. When improving a solution in a given population, each local search procedure makes use of the other solutions in the same population in order to guide the search. The best solutions found by all the local search procedures are stored in a distributed external archive of non-dominated solutions (see Sect. IV-A). After each iteration, the best solutions in every population are stored into the external archive; once in the archive, each solution will be preserved only if it is a non-dominated solution.

The solutions in each population are improved in an isolated fashion for a fixed number of iterations, without collaborating with other populations. After a given number of iterations, each population is reinitialised by using randomly selected solutions from the external archive. This mechanism generates population diversity for the local search procedure, provides a collaboration mechanism between the distributed populations, and helps to avoid stagnation conditions.

The algorithm follows a non-hierarchical schema in which all parallel local search procedures are peers and no procedure performs a master role. Figure 3 presents the logic of the parallel local search procedure that iteratively improves each solution in the AEDB-MLS algorithm.

Figure 3. Pseudocode of the local search procedure in the AEDB-MLS algorithm

```
 1: $s \leftarrow$ **initialise_solution**()
 2: **evaluate**($s$)
 3: **store_in_archive**($s$)
 4: **synchronise_threads**()
 5: **while** *stopping condition* is not met **do**
 6:    $t \leftarrow$ **random_solution**(*population*)
 7:    $\hat{s} \leftarrow$ **local_search_operator**($s$,$t$)
 8:    **evaluate**($\hat{s}$)
 9:    **if** $\hat{s}$ is feasible **then**
10:       **store_in_archive**($\hat{s}$)
11:       $s = \hat{s}$
12:    **end if**
13:    **if** *reinitialise condition* is met **then**
14:       $s \leftarrow$ **receive_from_archive**()
15:       **synchronise_threads**()
16:    **end if**
17: **end while**
```

Initial feasible solutions are randomly initialised in each population. Each local search initialises its assigned starting solution $s$ (line 1), evaluates it, and stores it to the external archive (lines 2–3). Then, all the local search procedures working on the same population are synchronised with each other in order to wait until the local population is fully initialised. Once this happens, the main loop is repeated until the *stopping condition* is met. In it, the local search operator is iteratively applied to the currently assigned solution $s$. In order to apply the operator to the solution $s$, another solution $t$ is randomly selected from the local population and used as reference to quantify the perturbation applied during the local search (lines 6–8), as explained in Sect. IV-B. If the perturbed solution $\hat{s}$ is feasible (i.e., it complies with the time constraint), then $s$ is replaced with $\hat{s}$ and the new solution is stored in the external archive (lines 9–12). When the *reinitialise condition* is met, the whole local population is discarded and replaced with randomly selected solutions form the external archive, restarting the local search procedures from a new location in the search space (lines 13–15). Figure 4 shows a general overview of the AEDB-MLS algorithm.

The AEDB-MLS algorithm follows a hybrid parallel model: message-passing is used for the collaboration between the distributed populations and the external archive, and shared-memory is used in the collaboration between solutions in the same population.

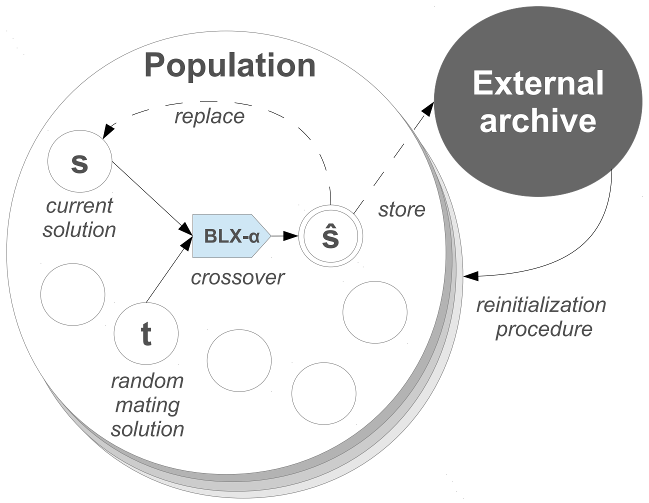


Figure 4. General overview of the AEDB-MLS algorithm

## A. External archive

In this work, we use the Adaptive Grid Archiving (*AGA*) algorithm as the archiving method. The elite population in AEDB-MLS is limited in size, so an archiving technique must be applied to discard solutions when the maximum size of the archive is exceeded. The AGA algorithm was initially proposed as the density estimator for the Pareto Archived Evolution Strategy (*PAES*) algorithm [10]. It consists in dividing up the objective space into hypercubes with the goal of balancing the density of non-dominated solutions in the hypercubes. Then, when inserting a non-dominated solution in the Pareto front, its grid location in the solution space is determined. If the Pareto front is already full and the grid location of the new solution does not match with the most crowded hypercube, a solution belonging to that most crowded hypercube is removed before inserting the new one. The AGA strategy guarantees three very desirable properties for multi-objective optimisation algorithms: i) it maintains solutions at the extremes of all objectives; ii) it maintains solutions in all of the Pareto occupied regions, and iii) it distributes the remaining solutions evenly among the Pareto regions.

## B. Local search operators

The local search operators were designed based on the sensitivity analysis study presented in Sect. III-A. From this study we conclude that there are three different search criteria that can be applied when modifying a solution, depending on the objective to be improved: i) if the *energy used* objective or *forwardings* objective are targeted for improvement, then the *border_threshold* and *neighbors_threshold* parameters should be modified; ii) if the *coverage* objective is to be improved, then the *neighbors_threshold* parameter should be tuned; and iii) if the *broadcast time* constraint is to

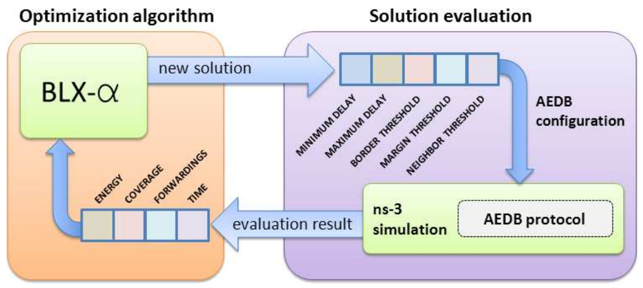


Figure 5. Search and solution evaluation procedure in AEDB-MLS

be improved, then the *min_delay* and *max_delay* parameters should be adjusted.

The local search operator uses the BLX-$\alpha$ operator, a recombination operator for real-coded EAs [5], which has been successfully used on a wide range of problems [8], [17]. Each iteration, one of the three aforementioned search criteria is randomly selected and it is applied to the current solution $s$. The BLX-$\alpha$ operator is independently applied to each of the corresponding parameters $P$ defined by the selected search criterion. The value of the parameter $p \in P$ in solution $s$ is modified by BLX-$\alpha$ as shown in Equation 2.

$$\begin{gathered} \hat{s_p} = s_p + \phi \times [(3 \times \rho) - 2] \\ \phi = \alpha \times |s_p - t_p| \end{gathered} \tag{2}$$

where $t$ is an auxiliary solution randomly selected from the current population, $\rho \in [0, 1)$ is a randomly selected number, $s_p$ is the value of the parameter $p$ in the solution $s$, and $\hat{s_p}$ is the value of the parameter $p$ in the solution $\hat{s}$. The additional parameter $\alpha \in (0, 1)$ represents the perturbation magnitude of the BLX-$\alpha$ operator, the higher the $\alpha$-value the more perturbed is the $s_p$-value during the crossover. The parameter $\alpha$ was empirically tunned during the experimental analysis presented in Sect. V.

Figure 5 shows a general overview of the iterative search AEDB-MLS performs, and the evaluation procedure.

## V. Experimental analysis

The quality of each solution found by the AEDB-MLS is measured in terms of the different objectives explained before: (1) coverage achieved, (2) energy used and (3) network resources. The value of the broadcast time is also needed to verify the validity of the solution. For evaluating the solutions, we rely on the well-known ns3 [11] network simulator. It is an event driven simulator written in C++, highly realistic for wireless networks.

In order to have confident results, the quality of the solution is not tested in one single network but in 10 different networks, and the fitness value of each objective is defined as the average value of the 10 runs. These 10 networks are always the same for evaluating every solution.

Regarding the configuration of ns3 for the simulation of the broadcasting algorithm, the mobility model used is the random walk [7]. The simulation environment used is a square area of 500 m side. The speed of the nodes can vary from 0 to 2m/s (i.e., between 0 and 7.2km/h). We study three different network densities in the optimisation process, with 100, 200, and 300 devices/km$^2$. All the parameters are summarised in Table II.

Table II
Configuration of ns3

| | |
|---|---|
| *Devices/km$^2$* | 100, 200, 300 |
| *Speed* | [0, 2] m/s |
| *Size of the area* | 500 m $\times$ 500 m |
| *Default trans. power* | 16.02 dBm |
| *Dir. & speed change* | every 20 s |

In the simulations, the network evolves for 30 seconds in order to have the nodes uniformly distributed in the area. Then, after these 30 seconds, a node starts the broadcasting process. The simulation stops after 40 seconds.

In order to limit the search space, we defined reasonably large intervals for each of the parameters we are optimising. They are shown in Table III.

Table III
Domain of the variables

| | |
|---|---|
| *minimum delay* | [0, 1] s |
| *maximum delay* | [0, 5] s |
| *border_Threshold* | [-95, -70] dBm |
| *margin_Threshold* | [0, 3] dBm |
| *neighbors_Threshold* | [0, 50] devices |

The goal of this work is to *efficiently* solve the AEDB tuning problem, thus a fixed limit of 250 solution evaluations per thread is used as a stopping criterion for the AEDB-MLS algorithm. Each execution is performed using 8 distributed populations with 12 threads per population, the maximum number of cores per computing node available in the computing platform. This gives us a total of 24000 evaluations per algorithm execution.

A configuration analysis was performed using the less dense network in order to find the best values for the $\alpha$ parameter used in the BLX-$\alpha$ operator, and the number of iterations needed to meet the *reset condition* in the AEDB-MLS algorithm. The candidate values for the parameter settings study were: $\alpha \in \{0.1, 0.2, 0.3\}$, and *reset condition* $\in \{15, 25, 50\}$. The best results were obtained using $\alpha = 0.2$, and *reset condition* $= 50$. These are the values we adopted for AEDB-MLS.

## VI. Results

We proceed to analyse the solutions obtained by the AEDB-MLS algorithm, and compare them to those reported in [14], found with the NSGAII [3] and CellDE [4] MOEAs. We build the Pareto front approximation of AEDB-MLS with the best non-dominated solutions found in 30 independent executions (according to the AGA method) for every network density. They are displayed in Figure 6, and compared versus a *Reference* Pareto front approximation built from the best results found by the two MOEAs in 30 independent runs for every density (AGA was used in this case too). For a more detailed information about the configuration of the evolutionary algorithms, please refer to [14].

Results show that the MOEAs and AEDB-MLS obtain similar Pareto front shapes. As stated in [14], there exist two clearly differentiated set of solutions. One with very low values of energy approximately between $[-20, 20]$ dBm and similar values for the coverage and the number of forwardings, and another with higher energy values and coverage growing much faster than the number of forwardings. The latter region of the front is the one in which we are more interested, since it is providing high coverage at a reasonable number of forwardings and energy requirements.

We can see that AEDB-MLS solutions are very close to the best solutions found by the MOEAs. Additionally, it can be seen that AEDB-MLS provides a set of diverse solutions, well spread along the Pareto front approximation. We compared the Pareto fronts obtained by the three algorithms in terms of *spread* (to quantify the diversity of solutions), *inverted generational distance* (that measures the accuracy of solutions), and *hypervolume* (accounting for both, accuracy and diversity). They are defined next:

- *Inverted generational distance*. It measures the average euclidean distance from the found solutions to the Pareto-front. It was presented in [19] and defined in Eq. 3:

$$IGD = \frac{\sqrt{\sum_{i=1}^{k} d_i^2}}{n} \quad , \tag{3}$$

  where $d_i$ is the Euclidean distance from point *i* in the Pareto front approximation found to the closest one in the optimal Pareto front, and *n* is the number of solutions in the front.
  Fronts with small *inverted generational distance* values are desirable. It takes value $0$ when all solutions are actually on the Pareto front.
- *Spread*. It quantifies the diversity of solutions in the front by means of how well they are spread along the front. It is defined as:

$$I_{\triangle} = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \quad , \tag{4}$$

  where $d_i$ is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and $d_f$ and $d_l$ are the Euclidean distances to the *extreme* solutions of the optimal Pareto front in the objective space. This indicator takes value zero for an ideal distribution, which has a perfect spread of the solutions in the Pareto front.

Table IV
COMPARISON OF THE ALGORITHMS ACCORDING TO WILCOXON TEST

| | NSGAII | AEDB-MLS |
|---|---|---|
| *Spread* | | |
| CellDE | ▲▲▲ | ▲ – – |
| NSGAII | | – ▽▽ |
| *Inverted generational distance* | | |
| CellDE | ▽▽ – | ▲▲▲ |
| NSGAII | | ▲▲▲ |
| *Hypervolume* | | |
| CellDE | ▽▽▽ | ▲▲▲ |
| NSGAII | | ▲▲▲ |

- *Hypervolume*. This indicator calculates the volume, in the objective space, covered by members of a non-dominated set of solutions $Q$, for problems where all objectives are to be minimised [21]. Mathematically, for each solution $i \in Q$, a hypercube $v_i$ is constructed with a reference point $W$ and the solution $i$ as the diagonal corners of the hypercube. The reference point can simply be found by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume ($HV$) is calculated as:

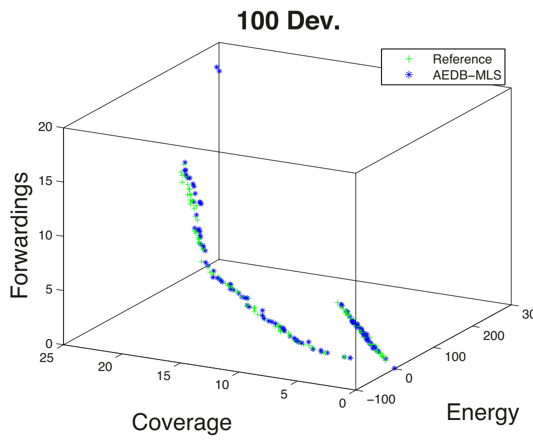
$$I_{HV} = \text{volume}\left( \bigcup_{i=1}^{|Q|} v_i \right) . \tag{5}$$

  The higher the value of *hypervolume*, the better the approximated Pareto front is.

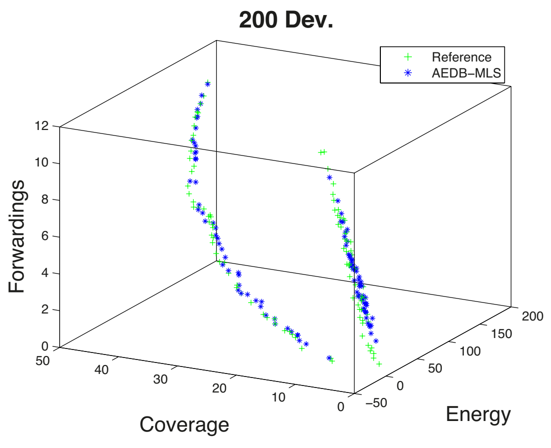
Before applying these metrics, all fronts were normalised because these indicators are not free from arbitrary scaling of the objectives. An approximation of the true Pareto front built from the best solutions found by the three considered algorithms (after 30 independent executions) was used in the normalisation process.

The results of the pairwise comparison of all the three algorithms according to the three metrics is summarised in Table IV, where symbol '▲' indicates that the algorithm in the row is better than the algorithm in the corresponding column, with $95\%$ statistical confidence according to Wilcoxon unpaired signed rank test. On the contrary, '▽' means that it is worse, and '–' represents those cases when no statistical significance was found. The three symbols in every column represent the result of the comparison of the corresponding algorithms for 100, 200, and 300 devices/km$^2$ instances, in that order.

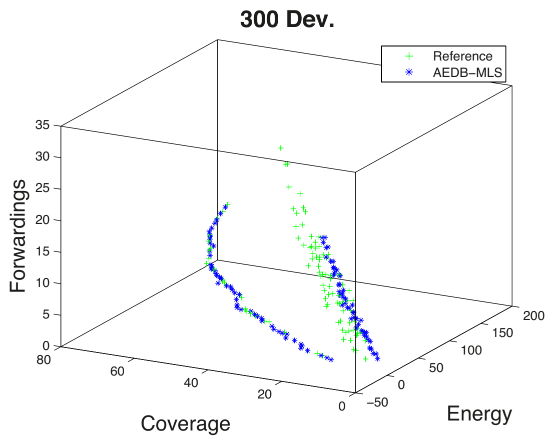
According to *spread*, CellDE outperformed NSGAII for the three instances (with statistical confidence), while it can only outperform AEDB-MLS for the sparsest density. AEDB-MLS is significantly better than NSGAII for the two biggest problem instances (no statistical confidence was found for the sparse networks). Regarding both *inverted generational distance* and *hypervolume*, AEDB-MLS is outperformed by the two MOEAs for the three instances.



(a) 100 Devices/km$^2$



(b) 200 Devices/km$^2$



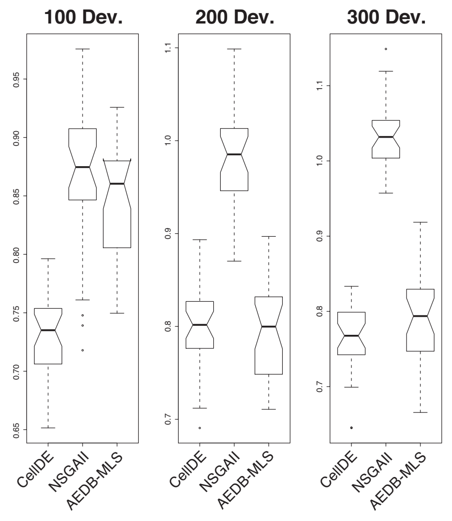
(c) 300 Devices/km$^2$

Figure 6. The *Reference* Pareto fronts obtained for the studied densities and the Pareto fronts obtained with the local search

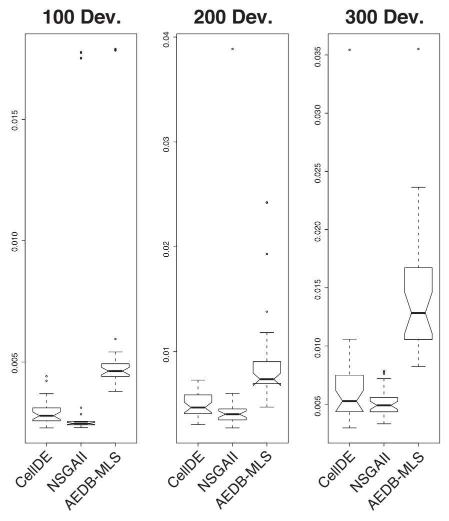
We graphically show in Fig. 7 the boxplots of the values obtained by the three algorithms in the 30 runs for the three considered metrics. We can see that, as mentioned before, AEDB-MLS is highly competitive with the MOEAs according to the *spread* metric for the three networks. The good *spread* values provided by the proposed local search algorithm, better than NSGAII, show its capability to effectively explore the search space to provide a diversified set of solutions. However, regarding the accuracy of the algorithms, it is visible that AEDB-MLS is not so competitive with the MOEAs.

We also checked how many solutions from the *Reference* Pareto front (built from the solutions of the two MOEAs) are dominated by at least one solution of the AEDB-MLS Pareto approximation and vice versa. We found out that AEDB-MLS dominates 13 solutions of the *Reference* Pareto front for the 100 devices density, while its solutions are dominated by 54 solutions of the *Reference* Pareto front. For the 200 devices network, AEDB-MLS dominates 11 solutions and is dominated 40 times. Finally, for the densest configuration, AEDB-MLS dominates 15 solutions and is dominated 17 times. The results obtained demonstrate that the quality of the solutions found using the local search is competitive compared to the MOEAs. Please, notice that we are comparing here our local search versus the best results of the two MOEAs. Therefore, we foresee that enriching the MOEAs with the proposed local search algorithm could significantly improve the quality of the obtained results.

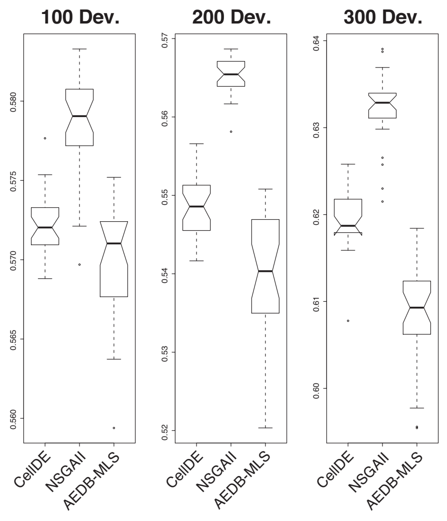
Regarding the execution time, AEDB-MLS requires, in average, 48, 188, and 417 minutes to find the Pareto front approximations for the three network densities:100, 200, and 300 devices/km$^2$, respectively. The evolutionary algorithms take 32, 123, and 264 hours on the same server (Intel Xeon L5640 under Debian 6.0). It means that the multi-objective local research presented here is over 38 times faster than any of the evolutionary algorithms, and it performs 2.4 times more evaluations than the EAs.



(a) *Spread*



(b) *Inverse generational distance*



(c) *Hypervolume*

Figure 7. Boxplot comparison of the quality of the Pareto fronts obtained for the studied densities

## VII. Conclusions and Future Work

We present a novel parallel multi-objective local search for optimising an energy aware broadcasting algorithm, called AEDB-MLS. A sensitivity analysis was performed in order to learn how the different parameters influence the performance of the protocol. The conclusions obtained from such analysis where used in the design of the operators applied in AEDB-MLS.

The protocol is optimised in terms of three different objectives: (1) coverage achieved, (2) number of forwardings, and (3) energy used. Additionally, we restricted the solutions according to the brocasting time. A solution is no longer valid if the broadcast time is longer than two seconds.

The solutions obtained by the AEDB-MLS were compared to those reported by two well known evolutionary algorithms: CellDE and NSGAII. Results show that the multi-objective local search finds similar solutions to the ones by the EAs, even outperforming some of them. Additionally, we compared the obtained Pareto fronts in terms of the *spread*, *hypervolume*, and *inverted generational distance*. The good values in the *spread* metric of the AEDB-MLS shows its capability for exploring the search space.

Moreover, we must highlight that we are obtaining competitive results in much shorter time. The proposed local search is 38 times faster than any of the evolutionary algorithms compared, even when it performs 2.4 times more evaluations.

As future work, we plan to parallelise the cellular multi-objective evolutionary algorithm, and include AEDB-MLS in it as a local search for fine tuning the solutions generated by CellDE.

## References

[1] W. Abdou, A. Henriet, C. Bloch, D. Dhoutaut, D. Charlet, and F. Spies. Using an evolutionary algorithm to optimize the broadcasting methods in mobile ad hoc networks. *Journal of Network and Computer Applications*, 34:1794–1804, 2011.

[2] E. Alba, P. Bouvry, B. Dorronsoro, F. Luna, and A.J. Nebro. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. In *Nature Inspired Distributed Computing (NIDISC)*, page 192b, 2005.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evol. Comp.*, 6(2):182–197, 2002.

[4] J.J. Durillo, A.J. Nebro, F. Luna, and E. Alba. Solving three-objective optimization problems using a new hybrid cellular genetic algorithm. In *Parallel Problem Solving from Nature (PPSN X)*, volume 5199 of *LNCS*, pages 661–670. Springer, 2008.

[5] Larry J. Eshelman and J. David Schaffer. Real-coded genetic algorithms and interval-schemata. In L. Darrell Whitley, editor, *FOGA*, pages 187–202, 1992.

[6] J. García-Nieto and E. Alba. Automatic parameter tuning with metaheuristics of the AODV routing protocol for vehicular ad-hoc networks. In *Applications of Evolutionary Computation*, volume 6025 of *LNCS*, pages 21–30, 2010.

[7] R. Groenevelt, E. Altman, and P. Nain. Relaying in mobile ad hoc networks: The brownian motion mobility model. *J. of Wireless Networks*, pages 561–571, 2006.

[8] F. Herrera, M. Lozano, and A. M. Snchez. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, 18(3):309–338, 2003.

[9] P.-C. Hsiao, T.-C. Chiang, and L.-C. Fu. Particle swarm optimization for the minimum energy broadcast problem in wireless ad-hoc networks. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2012.

[10] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evol. Comput.*, 8(2):149–172, 2000.

[11] M. Lacage and T. Henderson. Yet another network simulator. In *Proceeding of the workshop on ns-2: the IP network simulator*, page 12, 2006.

[12] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Conf. on Mobile Comp. and Networking*, pages 151–162, 1999.

[13] P. Ruiz and P. Bouvry. Distributed energy self-adaptation in ad hoc networks. In *proc. of IEEE Int. workshop on Management of Emerging Networks and Services (MENS), in conjunction with IEEE Globecom*, pages 539–543, 2010.

[14] P. Ruiz, B. Dorronsoro, and P. Bouvry. Finding scalable configurations for AEDB broadcasting protocol using multi-objective evolutionary algorithms. *Cluster Computing*. Online First, 2012. DOI: 10.1007/s10586-012-0220-0.

[15] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. Wiley, 2004.

[16] A. Saltelli, S. Tarantola, and S. Chan. A quantitative, model independent method for global sensitivity analysis of model output. *Technometrics*, 41(39-56), 1999.

[17] M. Takahashi and H. Kita. A crossover operator using independent component analysis for real-coded genetic algorithms. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 643 –649 vol. 1, 2001.

[18] J. Toutouh, S. Nesmachnow, and E. Alba. Fast energy-aware OLSR routing in VANETs by means of a parallel evolutionary algorithm. *Cluster Computing*. Online First, 2012. DOI: 10.1007/s10586-012-0208-9.

[19] David Allen Van Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. PhD thesis, 1999.

[20] S. Wolf and P. Merz. Evolutionary local search for the minimum energy broadcast problem. In *Evol. Comp. in Comb. Opt.*, volume 4972 of *LNCS*, pages 61–72. 2008.

[21] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.