Optimizing the DFCN Broadcast Protocol with a Parallel Cooperative Strategy of Multi-Objective Evolutionary Algorithms

Carlos Segura[1], Alejandro Cervantes[2], Antonio J. Nebro[3], María Dolores Jaraíz-Simón[4], Eduardo Segredo[1], Sandra García[2], Francisco Luna[3], Juan Antonio Gómez-Pulido[4], Gara Miranda[1], Cristóbal Luque[2], Enrique Alba[3], Miguel Ángel Vega-Rodríguez[4], Coromoto León[1], and Inés M. Galván[2,⋆]

[1] Department of Statistics, O.R. and Computation, University of La Laguna
csegura@ull.es
[2] Computer Science Department, University Carlos III of Madrid
[3] Computer Science Department, University of Málaga
[4] Department of Technologies of Computers and Communications, University of Extremadura

Abstract. This work presents the application of a parallel cooperative optimization approach to the broadcast operation in mobile ad-hoc networks (MANETs). The optimization of the broadcast operation implies satisfying several objectives simultaneously, so a multi-objective approach has been designed. The optimization lies on searching the best configurations of the DFCN broadcast protocol for a given MANET scenario. The cooperation of a team of multi-objective evolutionary algorithms has been performed with a novel optimization model. Such model is a hybrid parallel algorithm that combines a parallel island-based scheme with a hyperheuristic approach. Results achieved by the algorithms in different stages of the search process are analyzed in order to grant more computational resources to the most suitable algorithms. The obtained results for a MANETs scenario, representing a mall, demonstrate the validity of the new proposed approach.

1 Introduction

Mobile ad-hoc networks (MANETs) [1] are fluctuating, self-configuring networks of mobile hosts, called *nodes* or *devices*, connected by wireless links. This kind of network has numerous applications because of its capacity of auto-configuration and its possibilities of working autonomously or connected to a larger network. No static network infrastructure is needed to support the communications between nodes, which are free to move arbitrarily. Devices in MANETs are usually

⋆ This work has been supported by the EC (FEDER) and the Spanish Ministry of Education and Science inside the 'Plan Nacional de I+D+i' (TIN2005-08818-C04) and (TIN2008-06491-C04-02). The work of Gara Miranda has been developed under grant FPU-AP2004-2290.

laptops, PDAs, or mobile phones, equipped with network cards featuring wireless technologies. This implies that devices communicate within a limited range and also that they can move while communicating.

Broadcasting is a common operation at the application level and also widely used for solving many network layer problems. It is expected to be performed very frequently, serving also as a last resort to provide multicast services. Hence, having a well-tuned broadcast strategy results in a major impact in network performance. The optimization implies satisfying several objectives simultaneously: the number of reached devices (*coverage*) must be maximized, a minimum usage of the network (*bandwidth*) is desirable, and the process must take a time as short as possible (*duration*). These objectives are conflicting among them, so we are dealing with a multi-objective optimization problem (MOP).

Since exact approaches are practically unaffordable for real world MOPs, a wide variety of approximated algorithms have been designed. Among them, metaheuristics are a family of techniques which have become popular to solve both single and multi-objective problems. They can be considered as high-level strategies that guide a set of simpler heuristic techniques in the search of an optimum [2]. Among these techniques, evolutionary algorithms for solving MOPs are very popular [3] giving raise to a wide variety of algorithms, such as NSGA-II [4] and SPEA2 [5]. Other family of metaheuristics widely applied in multi-objective optimization is particle swarm optimization or PSO [6].

This work presents an optimization of the broadcast operation for a real MANET instance. In order to provide an efficient, and robust approach, applicable to a wide range of problem instances, a new parallel evolutionary model has been applied[1]. The model is based on the hybridization of parallel island-based evolutionary algorithms and hyperheuristics. In particular, eight different multi-objective algorithms comprising genetic algorithms, differential evolution, evolutionary strategies, and PSO have been combined in the island scheme.

The remaining content is structured in the following way: Section 2 presents the broadcast optimization problem in MANETs. The sequential approaches applied in this work are presented in section 3. The proposed parallel model for multi-objective optimization is described in detail in section 4. The computational study is presented in section 5. Finally, the conclusions and some lines of future work are given in section 6.

2 Broadcast Operation in MANETS

This work focuses on the study of the broadcast operation in a particular kind of MANETs, the *metropolitan* MANETs. These MANETs have some specific features that hinders the testing in real environments: the network density is heterogeneous and it is continuously changing because devices in a metropolitan area move and/or appear/disappear from the environment. For this reason, many simulation tools have been developed [7]. In this work the *Madhoc* simulator [8]

[1] We will use the most familiar term *evolutionary algorithm* instead of *metaheuristic* throughout the paper, although in the algorithms we study there is a PSO.

was the choice. This tool provides a simulation environment for several levels of services based on different types of MANETs technologies and for a wide range of MANET real environments. It also provides implementations of several broadcast algorithms [9]. From the existing broadcast protocols, the *Delayed Flooding with Cumulative Neighbourhood* (DFCN) [10] has been selected because it was specifically designed to deal with metropolitan MANETs.

DFCN is a deterministic and totally localized algorithm. It uses heuristics based on the information from one hop. Thus, it achieves a high scalability. The behaviour of each device when using DFCN is driven by three events: the reception of a message (*reactive behaviour*), the expiration of the *random delay for rebroadcasting* (RAD) of a message, and the arrival of a new neighbour to its covered area (*proactive behaviour*). Although DFCN has shown good behaviour with metropolitan MANETs, the task of configuring such parameters is not trivial, and the proper operation of the protocol is sensitive to such configuration. The set of parameters that must be configured is:

- *minG*: minimum gain for forwarding a message.
- [*lowerRAD*, *upperRAD*]: range values for the RAD.
- *proD*: maximum density for which it is still necessary to use proactive behaviour for complementing the reactive behaviour.
- *safeDensity*: maximum density below which DFCN always rebroadcasts.

Given the values for the five DFCN configuration parameters and a MANET scenario, the *Madhoc* tool does the corresponding simulation and provides an estimate for the three objectives: duration, coverage, and bandwidth. One possibility to find the most suitable configuration is to systematically vary each of the five DFCN parameters. However, the possible parameter combinations are too large and evaluations in the simulator are computationally expensive. So, such technique is unable to obtain good quality solutions in a reasonable time. Other alternative relies on deeply analysing the problem to extract information to define a heuristic strategy, but the complexity and stochastic behaviour of the given problem hinders it. For these reasons, one usual way of affording this problem is through evolutionary techniques [11].

3 Applied Sequential Approaches

The aim of this section is to present the sequential algorithms used in this work for solving the proposed broadcast optimization problem. Algorithms previously used in [12], as well as other new alternatives has been applied. In this work, all these algorithms were also used in parallel, following some standard island-based models and applying the method explained in section 4.

3.1 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II [4] is a non-dominated sorting based multi-objective evolutionary algorithm. Two of the most important characteristics which differentiates NSGA-II of

NSGA and other non-dominated sorting based approaches are the following. First, a fast non-dominated sorting approach with reduced computational complexity ($O(mN^2)$). Second, a selection operator which combines previous populations with new generated child populations, ensuring elitism in the approach.

The procedure is as follows: the two populations are sorted according to their rank, and the best solutions are chosen to create a new population. In the case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals belonging to the same rank is used to get the most promising solutions.

Algorithm 1. NSGA-II Pseudocode

```
Initialization: Generate an initial population P_0 with N individuals. Assign t = 0.
while (not stopping criterion) do
    Fitness assignment: Calculate fitness values of individuals in P_t. Use the non-domination rank
    in the first generation, and the crowded comparison operator in other generations.
    Mating selection: Perform binary tournament selection on P_t in order to fill the mating pool.
    Variation: Apply genetic operators to the mating pool to create a child population CP.
    Combine P_t and CP selecting the best individuals using the crowding operator to constitute
    P_{t+1}.
    t = t + 1
end while
```

3.2 Evolution Strategy with NSGA-II (ESN)

The ESN algorithm is based on the hybridization of Evolution Strategies and NSGA-II. The algorithm uses the standard Evolution Strategies' steps [13], replacing the selection process by the NSGA-II [4] selection process. The main difference between Evolution Strategies and Genetic Algorithms is that crossover operators are not used in ES, and each parent produces one offspring only by mutation. The mutation process implemented was the standard $(\mu + \lambda)$ process explained in [14], although in our case, $\lambda = \mu$.

Algorithm 2. ESN Pseudocode

```
Initialize population P of μ individuals
Initialize variance σ for each individual I ∈ P
while (not stopping criterion) do
    P' = ∅
    for each I = (x_1, ..., x_n, σ) ∈ P do
        σ' = σ e^{N(0,Δ)}
        Create I' = (N(x_1, σ'), N(x_2, σ'), ..., N(x_n, σ'), σ')
        P' = P' ∪ {I'}
    end for
    P = P ∪ P'
    Calculate front F_1 as Non-dominated individuals of P
    for i = 2 to n do
        Generate fronts F_i as Non-dominated individuals of P \ (F_1 ∪ ... ∪ F_{i-1})
    end for
    Sort solutions in each F_i (i = 1, ..., n) using the crowding distance
    Delete the worst μ individuals in population P
end while
```

3.3 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 was proposed by Zitzler *et al.* [5]. This algorithm uses a population and an archive. It assigns to each individual a fitness value that is the sum of its strength raw fitness plus a density estimation. In each generation the non-dominated individuals of both the original population and the archive are used to update the archive; if the number of non-dominated individuals is greater than the population size, a truncation operator based on calculating the distances to the k-th nearest neighbor is used. All this procedure is known as *Environmental Selection*. Then, the algorithm applies the selection, crossover, and mutation operators to members of the archive in order to create a new population of offsprings which becomes the population of the next generation.

Algorithm 3. SPEA2 Pseudocode

1: Initialization: Generate an initial population P_0 and create the empty archive $\overline{P}_0$.
2: **while** (not stopping criterion) **do**
3: Fitness assignment: Calculate fitness values of individuals in P_t and $\overline{P}_t$.
4: Environmental selection: Copy nondominated individuals in P_t and $\overline{P}_t$ to $\overline{P}_{t+1}$. if $|\overline{P}_{t+1}| > \overline{N}$ reduce $\overline{P}_{t+1}$; otherwise, fill $\overline{P}_{t+1}$ with dominated individuals in P_t and P_{t+1}.
5: Mating selection: Perform binary tournament selection on $\overline{P}_{t+1}$.
6: Variation: Apply crossover and mutation operators to the mating pool and set $\overline{P}_{t+1}$ to the resulting population.
7: **end while**

3.4 Indicator-Based Evolutionary Algorithm (IBEA)

The IBEA algorithm [15] allows to define the optimization goal in terms of a performance measure or *quality indicator*. This measure is used directly for fitness calculation. The IBEA algorithm allows the use of different binary quality indicators. In this work the binary multiplicative ϵ-indicator [16] was used. There exists two versions of IBEA, the basic one and a more robust version known as adaptive. In the adaptive version, objectives values are normalized, and the indicator values are adaptively scaled. Both versions of the algorithm have been implemented.

Algorithm 4. IBEA Pseudocode (Adaptive Version)

1: Initialization: Generate an initial population P with N individuals.
2: **while** (not stopping criterion) **do**
3: Fitness assignment: calculate the fitness values using the quality indicator.
1. Calculate indicator values $I(x^1, x^2)$ using the normalized objective values f_i' and determine the maximum absolute indicator value $c = max_{x^1,x^2 \in P}|I(x^1, x^2)|$.
2. $\forall x^1 \in P\,, F(x^1) = \sum_{x^2 \in P \setminus \{x^1\}} -e^{-I(\{x^2\},\{x^1\})/(c \cdot k)}$.
4: Enviromental selection: until the size of P does not exceed N, remove the individual with the smallest fitness value, and recalculate the fitness value of the remaining individuals.
5: Mating selection: Perform binary tournament selection with replacement on P in order to fill the temporary mating pool P'.
6: Variation: Apply recombination and mutation operators to the mating pool P' and add the resulting offspring to P.
7: **end while**

3.5 Multiple Objective Particle Swarm Optimization (MOPSO)

MOPSO [17] is an adapted version of Particle Swarm Optimization (PSO) to multi-objective optimization problems. MOPSO combines PSO [18] with the archiving strategy of PAES [19]. In this work, we use the version available in the EMOO repository [20].

The algorithm uses an external repository that stores non-dominated solutions in the swarm. The velocity and position of particles are updated using the standard equations of PSO but using a leader particle selected from the repository, instead of best neighbor as it is usual. The mechanism of leader selection is as follows: the fitness space is divided in hypercubic sectors, and one of the positions in the repository is randomly selected using a roulette algorithm that favors the sectors that are less populated. Therefore, particles are attracted by leaders located in the areas where fewer non-dominated positions have been found. The mechanism for inclusion in the repository ensures that it only stores non-dominated solutions. If the new solution dominates some of the solutions in the repository, those solutions are removed. The maximum number of particles in the repository is fixed, so when this limit is reached, before the insertion, a particle from the most-populated sector of the repository is removed. Upon completion of the specified number of iterations, the set of solutions in the repository is reported as the Pareto front.

Algorithm 5. MOPSO Pseudocode

```
1: Initialize the swarm
2: Calculate fitness for each particle, store fitness and position as PBestFitness and PBestPosition
3: Store the position and fitness of non-dominated particles in the Repository
4: while (not stopping criterion) do
5:     for Particle do
6:         Select Leader from the repository
7:         Update NewVelocity using standard equations and Leader as neighbor.
8:         Update NewPosition using standard equations of PSO
9:         Calculate NewFitness
10:        if NewFitness dominates PBestFitness then
11:            PBestFitness ← NewFitness and PBestPosition ← NewPosition
12:            if NewFitness is not dominated by solutions in the Repository then
13:                if Repository is full then
14:                    Remove one solution
15:                end if
16:                Insert NewPosition and NewFitness in the Repository
17:                Remove from the Repository solutions dominated by the one just inserted
18:            end if
19:        end if
20:    end for
21: end while
```

3.6 Multi-Objective Cellular Genetic Algorithm (MOCell)

MOCell [21] is a cellular genetic algorithm (cGA). As other multi-objective metaheuristics, it includes an external archive to store the non-dominated solutions found so far. This archive is bounded and uses the crowding distance of NSGA-II to keep diversity in the Pareto front.

We have used here an asynchronous version of MOCell, similar to the one called aMOCell4 in [22], in which the cells are explored sequentially (asynchronously). The selection is based on taking an individual from the neighborhood of the current cell and another one chosen from the archive. After applying the genetic crossover and mutation operators, the new offspring is compared with the current one, replacing it if better; in the case of both solution be non-dominated, the worst individual in the neighborhood is replaced by the current one. In this two cases, the new individual is inserted into the archive.

Algorithm 6. MOCell Pseudocode

```
1: population ← initialize()
2: archive ← NULL
3: while (not stopping criterion) do
4:    for individual ← 1 to population.size() do
5:       neighbours ←getNeighborhood(population, position(individual));
6:       neighbours.add(position(individual));
7:       parent1 ←selection(neighbours);
8:       parent2 ←selection(archive);
9:       offspring←recombination(Pc, parent1, parent2);
10:       offspring←mutation(Pm, offspring);
11:       evaluateFitness(offspring);
12:       replacement(position(individual), offspring);
13:       insertIntoArchive(offspring);
14:    end for
15: end while
```

3.7 Non-dominated Sorting Differential Evolution (NSDEMO)

Differential Evolution (DE) [23,24,25] is an evolutionary algorithm introduced by Storn and Price in 1995. DE was designed to optimize (single-objective) problems over continuous domains. In this paper, we extend DE to solve multiobjective optimization problems. Like NSDE [26], our approach is a multiobjective differential evolution algorithm based on NSGA-II [4]. We call it NSDEMO: Non-dominated Sorting Differential Evolution for Multiobjective Optimization.

This algorithm replaces the crossover and mutation operators of the NSGA-II with the DE scheme. In particular, the DE/*rand*/1/*bin* strategy has been applied. The mutation is scaled using the factor F, while the crossover is controled with the parameter CR.

Algorithm 7. NSDEMO Pseudocode

```
1: initializeParameters
2: createPopulation
3: evaluate
4: assignRank
5: while (not stopping criterion) do
6:    for i = 0 to PopulationSize − 1 do
7:       selectThreeVectorsParents
8:       mutationDE and crossoverDE
9:       addChildToPopulation
10:       evaluate
11:    end for
12:    assignRankCrowding
13: end while
```

4 Proposed Parallel Approach

Multi-objective metaheuristic approaches in general, and evolutionary algorithms (MOEAs) in particular [3] are proven to be effective when solving multi-objective optimization problems, but they can be time and domain knowledge intensive when applied to solve real world instances. Several studies have been performed in order to reduce the resource expenditure when using MOEAs. These studies naturally lead to considering the MOEAs parallelization. In the parallel MOEA (pMOEA) island-based model [27] the population is divided into a number of independent subpopulations. Each subpopulation is associated to an island and a MOEA configuration is executed over each of them. Usually, each available processor constitutes an island. Each island evolves in isolation, but occasionally some solutions can be migrated between neighbour islands. Island-based models have shown good performance and scalability in many areas. Four basic island variants are seen to exist: all islands execute identical MOEAs/parameters (homogeneous), all islands execute different MOEAs/parameters (heterogeneous), each island evaluates different objective functions subsets, or each island represents a different region of the genotype/phenotype domains.

If we were able of finding a particular MOEA that clearly outperformed the other ones in solving a given MOP, the homogeneous island-based model using such MOEA would be the choice to consider. However, it is difficult to know a priori which MOEA is the most appropriate to solve a problem. If we consider that, when dealing with a real world problem, its objective functions can require a significant amount of computing time to be evaluated, it can be hard to choose the MOEA to be the basis of the homogeneous approach. As an alternative, the heterogeneous models allow to execute different MOEAs and/or parameters on the islands. By using heterogeneous models, the user avoids the selection of a specific MOEA to solve the problem. However, if some of the used MOEAs are not suitable to optimize the problem, the consequence can be a waste of resources.

The existence of a wide variety of MOEAs in the literature and the dependence on the problem domain and instance in the performance of the approaches hinders the user decision about the algorithm to be applied. For this reason, a promising approach appears in the application of hyperheuristics [28]. The underlying principle in using a hyperheuristic approach is that different algorithms have different strengths and weaknesses and it makes sense to combine them in an intelligent manner. A hyperheuristic solves the problem indirectly by recommending which solution method to apply at which stage of the solution process. One of the motivations is that the same hyperheuristic method can be applied to a wide range of problems. The goal is to raise the level of generality of decision support methodology perhaps at the expense of reduced - but still acceptable - solution quality when compared to tailor-made evolutionary approaches.

The proposed pMOEA model [29,30] breaks from the island-based model adding an adaptive property behaviour to it. Such property allows, by applying a hyperheuristic, to change in an automatic and dynamic way the MOEAs and/or parameters that are used in the islands along the pMOEA run. To the best of our knowledge, the application of hyperheuristics into parallel schemes aimed

at multi-objective optimization is a novelty. The architecture of the new hybrid model is similar to the island-based model, i.e., it is constituted by a set of *slave islands* that evolves in isolation by applying an evolutionary algorithm to a given population. The number of islands and the different MOEAs to execute over the local populations are defined by the user. Also, as in the island-based model, a tunable migration scheme allows the exchange of solutions between neighbour islands. However, a new special island, the *master island*, is introduced into the scheme. It is in charge of maintaining the global solution achieved by the pMOEA and selecting the MOEA configurations that are executed on the slave islands. The *global solution* is obtained by selecting the non-dominated solutions from the ones locally achieved by the slave islands. Usually, it is not desirable to manage a global solution with unlimited size, so the NSGA-II crowding operator [4] is proposed as the way to limit the size of the global solution set.

In standard island models, only a global stop criterion is defined. However, in the proposed model, local stop criteria are also defined for the execution of the MOEAs on the islands. When a local stop criterion is reached, the island execution is stopped and the local results are sent to the master island. The master scores, according to a quality indicator, the different configurations defined by the user taking into account their obtained results. A configuration is a MOEA together with the set of parameters that define such MOEA, e.g. the mutation and crossover rate, the population size, the archive size, etc. Based on such score, the hyperheuristic is applied and the master selects the configuration that will continue executing on the idle island. If the new selected configuration is the same as the island current configuration, the local stop criterion is updated and the execution continues. Otherwise, the configuration is changed and the new selected MOEA begins its execution by randomly taking the initial population individuals from the current global solution. Finally, when the global stop criterion is reached, every island sends its local solution to the master and all the local solution sets are considered to generate the global final solution.

One crucial point for the correct operation of the model is the selection process performed by the hyperheuristic. Considering the results obtained through the executions, it is beneficial to grant more opportunities to the configurations with better expectations. The decision process must be light in order to avoid having idle processes. One possibility to predict the behaviour of the configurations in a fast way is to pay attention to the contribution [31] of every configuration to the global solution. In this work, the score of each configuration is calculated as the contribution metric of such configuration - considering the current global solution as the reference front - divided by the number of evaluations that it has performed. A probabilistic selection, based on the score of each configuration, is used to decide the next configuration to execute. It is important to note that the behaviour of the configurations can change along the execution. Moreover, the stochastic behaviour of the evolutionary approaches may lead to variations in the results achieved by each algorithm. Therefore, a non-promising configuration must have a low probability of being selected, but not a zero probability. Probabilistic selection methods are more conservative than elitist ones, tending to

distribute the resources in a more uniform way. Probabilistic selection methods reduce the negative impact that a not accurate scoring method can introduce in the results. Even when the scoring method fails, some resources will be granted to good-behaved configurations, speeding up the optimization process. On the other hand, when the scoring method works correctly, some resources will be granted to non-promising configurations, slowing down the optimization. Preliminary studies shows that, in general, the usage of probabilistic methods is more suitable than the usage of elitist methods.

5 Experimental Evaluation

Initial experiments showed an irregular behaviour of different MOEAs when dealing with different MANET scenarios. In order to avoid the testing of many algorithms for solving each instance the adaptive model can been applied. Results for a mall MANET scenario demonstrate the validity of the approach. All the MOEAs presented in section 3 have been used inside the model. Tests have been run on a Debian GNU/Linux cluster of 8 Intel® Xeon™ 3.20 Ghz bi-processor nodes with 1Gb RAM. The interconnection network is a Gigabit Ethernet. The compiler and MPI implementation used were *gcc 3.3* and MPICH *1.2.7*.

The new model has been compared with sequential MOEAs and with other standard pMOEAs. For each implemented MOEA a homogeneous scheme is considered: "*homo*-SPEA2", "*homo*-NSGA-II", "*homo*-IBEA", "*homo-adap*-IBEA", "*homo*-ESN", "*homo*-MOPSO", "*homo*-MOCell", and "*homo*-NSDEMO". Also, a heterogeneous scheme constituted by the eight implemented MOEAs is considered: "*heterogeneous*". The new proposal, labelled as "*8-adaptive*", also uses the eight different MOEAs but following the adaptive behaviour.

Each tested pMOEA is constituted by eight slave islands. The subpopulation size on each island has been fixed to 15 individuals, while the population size for every sequential execution has been fixed to 100 individuals. The maximum size of the external set for those algorithms maintaining an archive was fixed to 100 individuals in the sequential experiments and to 15 individuals in the parallel ones. The remaining parameterization of each MOEA was as follows:

- SPEA2: $p_m = 0.2$, $p_c = 0.9$
- NSGA-II: $p_m = 0.2$, $p_c = 0.9$
- IBEA, adaptive-IBEA: $p_m = 0.2$, $p_c = 0.9$, k = 0.002
- ESN: $\sigma = 0.1$
- MOPSO: $p_m = 0.2$, divisions in archive = 30
- MOCell: $p_m = 0.2$, $p_c = 0.9$
- NSDEMO: $F = 0.5, CR = 1.0$

For every algorithm not doing special emphasis on the mutation or crossover operators, a polynomial mutation [32] ($\eta_m = 20$) and a simulated binary crossover (SBX) [33] ($\eta_c = 20$) were applied. In every execution the same migration scheme is specified. It consists in an unrestricted topology where the migration is performed from a slave to a randomly selected partner. The migration probability has been fixed to 0.05 and the number of individuals to migrate was limited to 4 each time. The global stopping criterion for every execution was 25000 evaluations. The

Table 1. Average hypervolume achieved by the different pMOEAs

Parallel Model	Evaluations limit 5000	10000	25000
8-adaptive	0.723	0.735	0.742
heterogeneous	**0.711**	**0.721**	**0.729**
homo-SPEA2	0.713	**0.724**	0.738
homo-NSGA-II	**0.712**	**0.723**	0.739
homo-IBEA	0.715	**0.724**	0.733
homo-adap-IBEA	0.716	0.725	0.733
homo-ESN	**0.707**	**0.718**	**0.726**
homo-MOPSO	**0.682**	**0.683**	**0.685**
homo-MOCell	**0.690**	**0.702**	**0.712**
homo-NSDEMO	**0.713**	**0.720**	**0.727**

local stopping criterion in 8-adaptive executions was fixed to 15 generations. In all cases, the final solution was limited to 100 elements.

The first experiment compares the different aforementioned pMOEAs among them. For each type of execution, 30 repetitions have been performed and average values considered. In order to detect differences between the algorithms within short and long time ranges, three different number of individual evaluation limits have been considered: 5000, 10000 and 25000 evaluations. The computing time of a sequential execution with 25000 evaluations is approximately 50 hours. Table 1 shows the average hypervolume [34] achieved by each parallel model at the given limits. The hypervolume indicator makes possible to combine the quality information of convergence and diversity in a single value. The 8-adaptive configuration achieves the best results in every case. The dynamic mapping of the MOEAs into the islands allows to give more computational resources to the most suitable algorithms, thus improving the results of the heterogeneous model. Moreover, the simultaneous usage of different evolutionary algorithms makes possible to combine the benefits of each one, so that, the results of every homogeneous island-based model is improved.

In order to provide the results with confidence, the following statistical analysis has been performed [35,36]. First, a Kolmogorov-Smirnov test is performed in order to check whether the results follow a normal (gaussian) distribution. Every sample passes the normality test. The homogeneity of the variances for each pair of samples is ensured through the Levene test. Finally, the ANOVA test is passed to check the confidence levels. As our interest is focused in the new model, it has been statistically compared with the remaining pMOEAs. Table 1 shows data in bold when differences between such model and the new proposed model are significant. The new model achieves a better hypervolume in every case. In the case of 10000 evaluations, all differences are significant, except with *homo-adap-*IBEA, showing the good performance of the approach. Figure 1 presents, for each model, the average hypervolume achieved along the executions.

The second experiment analyzes the run-time behaviour of the sequential and adaptive models. The ideas presented in [37] were followed. Each MOEA, as well

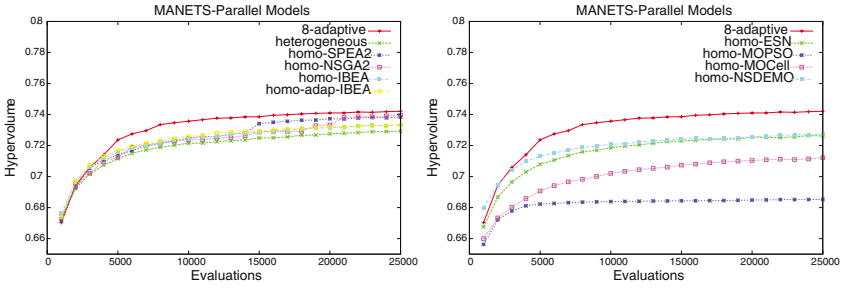

Fig. 1. Hypervolume achieved by the parallel models

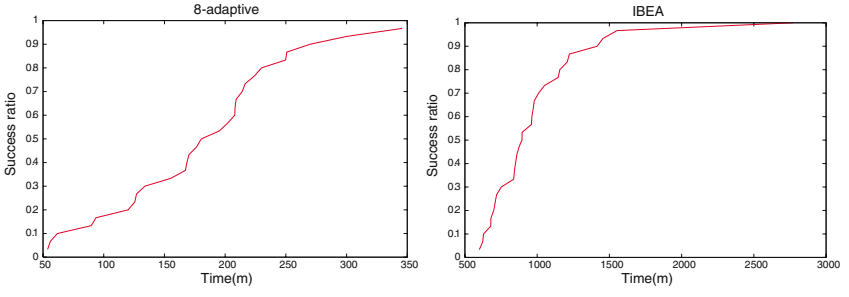

Fig. 2. Run length distributions

Table 2. Speedup of the proposed model and success ratio for sequential models

Sequential Models	8-adaptive Speedup	Success Ratio
SPEA2	7.25	80
NSGA-II	8.8	86.6
IBEA	5.57	100
adap-IBEA	5.09	86.6
ESN	9.75	83.3
MOPSO	-	0
MOCell	-	0
NSDEMO	11.59	36.6

as the new model, were executed using as finalization condition the achievement of a certain level of hypervolume quality: the 95% of the average achieved by the adaptive model in the first experiment. A second stopping criterion, consisting in executing a maximum number of 25000 evaluations was also considered. Figure 2 shows the run length distribution for the adaptive pMOEA and for the best behaved sequential MOEA (IBEA). For the remaining models a summary of the obtained information is presented in Table 2. For each sequential execution, the table shows the average speedup of the new model when the required quality is achieved, together with the success ratio, i.e. the probability of achieving the required hypervolume value, considering a maximum of 25000 evaluations. The

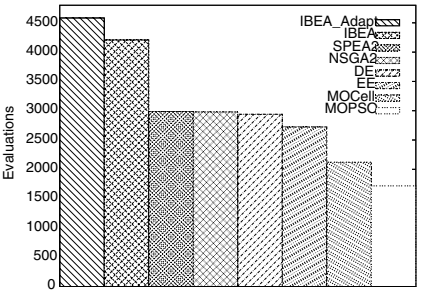

Fig. 3. Distribution of evaluations for the 8-adaptive model

parallel model obtains an almost linear speedup even when compared with the best MOEA, and a very high ratio of achieving the desired quality (96.6%).

Figure 3 shows, for the 30 executions of the 8-adaptive model, the average number of evaluations executed by each configuration. By comparing it with results in Table 2, it is clear that more computational resources are granted to the algorithms with best behaviour. IBEA and adap-IBEA, the best-behaved sequential algorithms, are the most used configurations, while MOPSO and MOCell, the worst-behaved sequential algorithms, are the least used configurations.

6 Conclusions and Future Work

An optimization approach for the broadcast operation in MANETs based on the DFCN protocol has been presented. In order to increase the level of generality of the proposed solution - so it can be applied to any MANET scenario - a new hybrid model, which adds an adaptive property to the well known island-based models by applying the operation principles of the hyperheuristics, was applied. The adaptive property allows to dynamically grant more computational resources to the most promising algorithms. Results achieved for a mall MANET scenario demonstrate the positive effect introduced by the hybridization. The new model provides high-quality solutions without forcing the user to have a prior knowledge about each MOEA behaviour when applied to any considered problem instance. The DFCN configurations obtained by the new parallel model clearly improve the ones obtained sequentially.

Future work is related to two different fields: the improvement of the broadcast in MANETs and the improvement of the designed optimization model. Other broadcast protocols can be considered to solve the same problem instances. In relation to the hybrid model, other MOEAs and even other kind of multi-objective optimization algorithms could be incorporated to the scheme. Some further studies concerning the model self-adaptation can be performed. In particular, other ways to measure the algorithms quality can be tested.

References

1. Macker, J., Corson, M.: Mobile Ad Hoc Networking and the IETF. ACM Mobile Computing and Communications Review 2(1) (1998)
2. Blum, C., Roli, A.: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys 35(3), 268–308 (2003)

3. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer, New York (2006)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
5. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. Evolutionary Methods for Design, Optimization and Control, 19–26 (2002)
6. Reyes-Sierra, M., Coello, C.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
7. Hogie, L., Bouvry, P., Guinand, F.: An Overview of MANETs Simulation. Electronics Notes in Theorical Computer Science 150(1), 81–101 (2006)
8. Hogie, L.: Mobile Ad Hoc networks: modelling, simulation and broadcast-based applications. PhD thesis, Le Havre University and Luxembourg University (2007)
9. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 194–205 (2002)
10. Hogie, L., Seredynski, M., Guinand, F., Bouvry, P.: A Bandwidth-Efficient Broadcasting Protocol for Mobile Multi-hop Ad hoc Networks. In: 5th International Conference on Networking (ICN 2006). IEEE, Los Alamitos (2006)
11. Alba, E., Dorronso, B., Luna, F., Nebro, A.J., Bouvry, P., Hogie, L.: A Cellular Multi-Objective Genetic Algorithm for Optimal Broadcasting Strategy in Metropolitan MANETs. Computer Communications 30(4), 685–697 (2007)
12. Alba, E., Cervantes, A., Gómez, J., Isasi, P., Jaraíz, M., León, C., Luque, C., Luna, F., Miranda, G., Nebro, A., Pérez, R., Segura, C.: Metaheuristic approaches for optimal broadcasting design in metropolitan mANETs. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2007. LNCS, vol. 4739, pp. 755–763. Springer, Heidelberg (2007)
13. Bäck, T., Schwefel, H.: Evolutionary algorithms: Some very old strategies for optimization and adaptation. In: New Computing Techniques in Physics Research II: Proceedings of the Second International Workshop on Software Engineering, Artificial Intelligence, and Expert Systems for High Energy and Nuclear Physics, pp. 247–254 (1992)
14. Bäck, T., Rüdolph, G., Schwefel, H.: A survey of evolution strategies. In: Proceedings of the 4th International Conference on Genetic Algorithms, pp. 2–9 (1991)
15. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
16. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)
17. Coello, C.A., Toscano, G., Salazar, M.: Handling multiple objectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation 8(3), 256–279 (2004)
18. Kennedy, J., Eberhart, R., Shi, Y.: Swarm intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
19. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)

20. Coello, C.A., et al.: EMOO Repository, http://www.lania.mx/~ccoello/EMOO
21. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: A cellular genetic algorithm for multiobjective optimization. In: Pelta, D.A., Krasnogor, N. (eds.) Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006), Granada, Spain, pp. 25–36 (2006)
22. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: Design issues in a multiobjective cellular genetic algorithm. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 126–140. Springer, Heidelberg (2007)
23. Price, K., Storn, R., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2006)
24. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. of Global Optimization 11(4), 341–359 (1997)
25. Storn, R.: System design by constraint adaptation and Differential Evolution. IEEE Transactions on Evolutionary Computation 1(3), 22–34 (1999)
26. Iorio, A.W., Li, X.: Solving rotated multi-objective optimization problems using differential evolution. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 861–872. Springer, Heidelberg (2004)
27. Van Veldhuizen, D.A., Zydallis, J.B., Lamont, G.B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Trans. Evolutionary Computation 7(2), 144–173 (2003)
28. Burke, E.K., Landa, J.D., Soubeiga, E.: Hyperheuristic Approaches for Multiobjective Optimisation. In: Metaheuristics International Conference, pp. 11.1–11.6 (2003)
29. León, C., Miranda, G., Segura, C.: Parallel Hyperheuristic: A Self-Adaptive Island-Based Model for Multi-Objective Optimization. In: Genetic and Evolutionary Computation Conference, pp. 757–758. ACM, New York (2008)
30. León, C., Miranda, G., Segura, C.: A Parallel Plugin-Based Framework for Multiobjective Optimization. In: International Symposium on Distributed Computing and Artificial Intelligence, vol. 50/2009, pp. 142–151. Springer, Heidelberg (2008)
31. Meunier, H., Talbi, E.G., Reininger, P.: A multiobjective genetic algorithm for radio network optimization. In: Congress on Evolutionary Computation (CEC 2000), La Jolla Marriott Hotel La Jolla, California, USA, pp. 317–324. IEEE Press, Los Alamitos (2000)
32. Deb, K., Goyal, M.: A combined genetic adaptive search (geneAS) for engineering design. Computer Science and Informatics 26(4), 30–45 (1996)
33. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
34. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
35. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
36. Sheskin, D.: The handbook of parametric and nonparametric statistical procedures. CRC Press, Boca Raton (2003)
37. Hoos, H., Informatik, F., Hoos, H.H., Stutzle, T., Stutzle, T., Intellektik, F., Intellektik, F.: On the run-time behavior of stochastic local search algorithms for sat. In: Proceedings AAAI 1999, pp. 661–666 (1999)