An Efficient ACO-SA Hybrid Metaheuristic for the Synchronization of Single Frequency Networks in Broadcasting

Akram Bedoui[1,2](✉), Philippe Debreux[2], and Thierry Schott[2]

[1] LORIA Laboratory, University of Lorraine, F-54506 Vandoeuvre-lés-Nancy, France
akram.bedoui@loria.fr
[2] TDF Company, F-57078 Metz, France
{akram.bedoui,philippe.debreux,thierry.schott}@tdf.fr

Abstract. The treasure of any radio communication network provider is the set of available frequencies and the challenge is to use the frequencies in the best possible way. Single Frequency Networks (SFNs) are broadcast networks where several transmitters send the same signal over the same frequency. They allow more efficient utilization of the radio spectrum in comparaison to traditional Multi Frequency Networks (MFNs) that use one different frequency per transmitter. SFN Synchronization Problem (SFNSP) is known to be a NP-hard problem. The aim of this paper is to present an original hybrid metaheuristic (ACO-SA) based on Ant Colony Optimization (ACO) and Simulated Annealing (SA) to solve SFNSP. Experimental results obtained with our hybrid ACO-SA on real-world benchmarks provided by the french telecommunication company named TDF[1], show drastic runtime improvement over existing approaches, and also quality improvement in comparison with existing SFN's synchronizations in the field of TV broadcasting in France.

Keywords: Ant Colony Optimisation · Simulated Annealing · Hybrid Metaheuristic · Single Frequency Network · Digital TV broadcasting

1 Introduction

Both the sectors of telecommunications and of broadcasting have to accommodate strong growth, with the sustained deployment of 3G and 4G networks, and the densification of TV networks. DVB-T, the current technical norm for Digital TV in Europe, offers the possibility to use the Single Frequency Network (SFN) technique, which consists in associating sets of synchronized transmitters. SFN's transmitters broadcast the same signal over one and only one frequency. The aim of SFN is to save utilization of the radio spectrum and allow a higher number of TV programs in comparaison to Multi Frequency Networks (MFNs)

[1] TDF is a french company, which provides radio and television services for telecom operators, and other multimedia services: digitization of content, encoding, storage, etc.http://www.tdf.fr.

P. Siarry et al. (Eds.): ICSIBO 2014, LNCS 8472, pp. 175–184, 2014.
DOI: 10.1007/978-3-319-12970-9_19

that use one different frequency per transmitter. The Quality of Service (QoS) of a SFN depends on the *extra-SFN jamming* and the *intra-SFN jamming*. The extra-SFN jamming depends on gaps in frequencies between transmitters which constitute the SFN and other transmitters not belonging to the considered SFN (i.e. transmitters on the same frequency not sufficiently far away and transmitters on adjacent channels in or in the vicinity of the SFN's coverage area). As for the intra-SFN jamming, it depends on the synchronization between the transmitters of the same SFN. In fact, the DVB-T technologies permit, in an interval of time called *Guard Interval (GI)*, to benefit from signals of the various co-channel transmitters constituting a SFN. Beyond GI, these signals are considered as interferers between them [2,12].

In this paper, we formulate the Single Frequency Network Synchronization Problem (SFNSP) as a combinatorial optimisation problem and we present an original hybrid metaheuristic based on Ant Colony Optimization algorithm (ACO) [4,9,11] and Simulated Annealing (SA) [1,3,5–7,10] to minimize the intra-SFN jamming of a SFN. We compare QoS of solutions calculated by our hybrid ACO-SA with operating SFN synchronizations in the field of TV broadcasting used nowadays in France.

This paper is organized as follows: in Section 2, we describe SFNSP. In Section 3, we present our hybrid ACO-SA metaheuristic. Experimental performance comparisons on real-world benchmarks provided by TDF Company are given in Section 4. Section 5 contains concluding remarks and further research aspects.

2 Single Frequency Network Synchronization Problem

SFN's transmitters are spread over the geographical area where broadcasters wish to provide the users with their services. Each transmitter covers a part of this geographical area called its *coverage area*. The area around a transmitter where transmission conditions are favourable enough to have a good reception of the signal is known as the *service area* of the transmitter. The service area is the portion of the coverage area that is not jammed by other transmitters.

The optimization of a SFN synchronization requires the adjustment of an initial transmitting delay on every transmitter so that all the signals transmitted by the SFN members fall within the *Guard Interval* (GI) on the maximum of the locations in the SFN's coverage area. If the delay spread is higher than the GI, according to the synchronization strategy of the receivers, the contributions outside the GI are considered as potential interferers and weighted with a co-channel protection ratio [2,12].

The formal definition of the considered SFNSP is given by: let S be a SFN. Let $T = \{t_i\}_{1 \leq i \leq n}$ be a set of n transmitters distributed across the geographical area of S. Let $D_i = \{d_{i,1}, d_{i,2}, ...d_{i,m}\}$ be the set of m valid delays, that can be assigned to the transmitter t_i.

If the coverage area of a transmitter t_i and the interference area of a transmitter t_j intersect, there is an intra-SFN jamming constraint $C_{t_i \leftarrow t_j}$ between the pair of transmitters (t_i, t_j). The constraint corresponds to the amount of jamming between the transmitters for different gaps in delay $d_{i,x} - d_{j,y}$ $(1 \leq x, y \leq m)$.

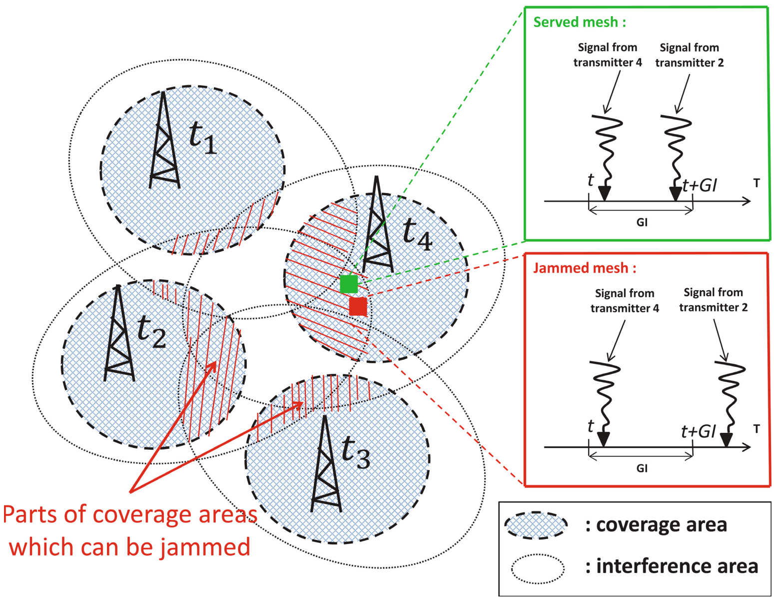


Fig. 1. Example of SFN network

A solution (i.e. synchronization) to the problem is obtained by assigning to each transmitter t_i one of the delays from D_i. It's henceforth denoted by $s \in D_1 \times D_2 \times ... \times D_n$ where $s(t_i) \in D_i$ is the delay assigned to the transmitter t_i. The optimal solution is the one which minimizes the objective function ϕ (see Formula 1).

$$Min\ \phi(s) = \sum_{t_i \in S}^{n} \sum_{t_j \in S \wedge t_j \neq t_i}^{p} \rho_{i,j} \times C_{t_i \leftarrow t_j}(s(t_i) - s(t_j)) \tag{1}$$

where p is the number of jammers of the transmitter t_i and $\rho_{i,j}$ is a weight of the constraint $C_{t_i \leftarrow t_j}$ Figure 1 shows an example of a SFN network constituted by four co-channel transmitters (t_1,t_2,t_2, and t_4). Between these transmitters, there are ten intra-SFN jamming constraints. For example, there is a constraint $C_{t_1 \leftarrow t_2}$ between t_1 and t_2 because the intersection of the interference area of t_2 with the coverage area of t_1.

A SFN network can be modelled by an oriented graph in which vertices represent transmitters and oriented edges represent intra-SFN jamming constraints. There is a strong link between graph coloring and delays synchronisation with binary interference constraints. The graph coloring problem is known to be NP-Hard [8], thus, consequently the SFNSP.

3 Principles of Our Hybrid ACO-SA Metaheuristic and Pseudo-Code

The idea of our hybrid ACO-SA (see Algorithm 1) consists in using a modified version of ACO algorithm inspired by [11] adapted to solve SFNSP combined

Algorithm 1. Pseudocode ACO-SA

```
Initialize S_0 ;                     /* according to the operational delays */
n ← |S_0|; m ← number of possible delays;
bestcost ← ∞; newcost ← 0 ;   /* initialization of the best and the new costs */
shortStagnation ← 0; longStagnation ← 0 ;   /* initialization of stagnation counters */
restartSAThreshold ← nb. of stagnation iterations allowed before we run SA;
stopThreshold ← nb. of stagnation iterations allowed before we stop ACO-SA;
R ← 2;
Initialize trace[n][m] ;             /* matrix which represents the memory */
Initialize sumTrace[n] ;   /* the vector which contains the sum of the values of each column of the memory */
parametersSA[] ← InitialisationParametresSA(S_0) ;   /* adaptive computation of SA's parameters */
while (longStagnation < stopThreshold) do
    for i ← 1 to n do
        sumTrace[i] ← 0 ;
    for i ← 1 to n do
        for i ← 1 to m do
            sumTrace[i] ← sumTrace[i] + trace[i][j];
    i_min ← index of the component of sumTrace which contains the minimal value;
    S_t ← GenerateNewSolution(i_min,S_{t-1}) ;   /* computation of a neighbor solution */
    if (shortStagnation = restartSAThreshold) then
        S'_t ← SimulatedAnnealing(S_t, parametersSA[]) ;   /* see Algorithm 2 */
        newcost ← φ(S'_t) ;                                 /* see Formula 1 */
        shortStagnation ← 0;
    else
        newcost ← φ(S_t) ;                                  /* see Formula 1 */
    if ( newcost < bestcost ) then
        bestcost ← newcost ;                  /* updating of the best cost */
        S_best ← S_t ;                        /* updating of the best solution */
        increment ← 1;
        for i ← 1 to n do
            for j ← 1 to m do
                trace[i][j] ← 1;
        shortStagnation ← 0; longStagnation ← 0;
    else
        UpdateTrace(S_t, S_best, increment, R) ;   /* see Algorithm 3 */
        shortStagnation ← shortStagnation + 1;
        longStagnation ← longStagnation + 1;
return S_best;
```

Algorithm 2. SimulatedAnnealing(S_t,$parameters[]$)

Data: a solution S_t and SA parameters (temperature and α stocked in $parameters[]$)

Result: S_{best} (i.e. improved S_t)

$stagnation \leftarrow 0$;
$stopThreshold \leftarrow$ number of stagnation iterations allowed before we stop the procedure SA ;
$n \leftarrow \mid S_t \mid$;
$S_0 \leftarrow S_t$;
$maxFail \leftarrow \frac{n*(n-1)}{2}$;
$nbFail \leftarrow 0$;
$tFound \leftarrow parameters[0]$;
$temperature \leftarrow parameters[0]$;
$\alpha \leftarrow parameters[1]$;
while *($stagnation < stopThreshold$)* **do**
  $temperature \leftarrow \frac{temperature}{1+\alpha*temperature}$;
  $oldCost \leftarrow \Phi(S_{t-1})$;
  $S_t \leftarrow GnrerNouvelleSolution(S_{t-1})$;
  $newCost \leftarrow \Phi(S_{t-1})$;
  $\Delta \leftarrow oldCost - newCost$;
  if $((\Delta > 0) \vee (rand(0,1) < e^{\frac{-\Delta}{temperature}}) \vee (maxFail == nbFail))$ **then**
    $S \leftarrow S_t$;
    $nbFail \leftarrow 0$;
  else
    $nbFail \leftarrow nbFail + 1$;
  $stagnation \leftarrow stagnation + 1$;
  if *($maxFail == nbFail$)* **then**
    $\alpha \leftarrow 0$;
    $temperature \leftarrow tfound$;
  if *($newCost <= bestCost$)* **then**
    $S_{best} \leftarrow S_t$;
    $bestCost \leftarrow newCost$;
    $tfound \leftarrow temperature$;
    $stagnation \leftarrow 0$;
return S_{best};

Algorithm 3. UpdateTrace(S_t,S_{best}, *increment*,R)

```
Data: current solution S_t, best solution until now S_best, increment, and R
Result: updated matrix trace
transmitter ← 1;
curentDelay ← S_t(transmitter);
bestDelay ← S_best(transmitter);
while ((transmitter ≤ n) ⋀(curentDelay == bestDelay)) do
    transmitter ← transmitter + 1;
    curentDelay ← S_t(transmitter);
    bestDelay ← S_best(transmitter);
if (transmitter = n) then
    increment ← increment + 1;
    for i ← 1 to n do
        for j ← 1 to m do
            trace[i][j] ← increment;
else
    for (i ← 1 to n) do
        curentDelay ← S_t(transmitter);
        bestDelay ← S_best(transmitter);
        trace[i][curentDelay] ← trace[i][curentDelay] + increment;
        trace[i][bestDelay] ← trace[i][bestDelay] + R;
```

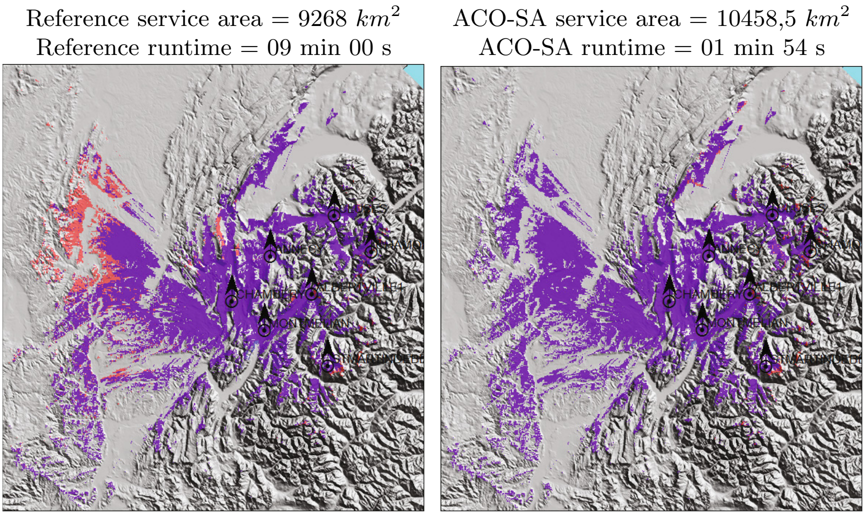
with a modified version of adaptive SA algorithm inspired by [3] also adapted to solve SFNSP. The goal of this hybridization is to improve the quality of ants using adaptive SA algorithm (see Algorithm 2) when the search stagnates in a local minimum. Our hybrid ACO-SA metaheuristic relies on the the following main components:

- **Representation of a solution:** a solution represents a possible synchronization of considered SFN's transmitters. We represent a solution by a vector. The indices of this vector represent the transmitters and the values of the components represent the delays affected to the transmitters.
- **Initial solution:** there exist three possibilities for generating the initial solution S_0: it can be a randomly generated synchronization, or a synchronization associating a delay equal to zero to all transmitter stations, or the operational synchronization used nowadays. In our ACO-SA metaheristic we use the later possibility (see line 1 of Algorithm 1);
- **Pheromone memory:** the pheromone memory is represented by a matrix (*trace*) of dimension $n \times m$, where n is the number of transmitters of the SFN to be synchronized, and m is the number of possible delays for each transmitter. Initially, all elements of the matrix are equal to 1 (see line 8 of Algorithm 1);

- **ACO stop criterion:** the stop criterion of ACO-SA is dynamic. If the costs of a sequence of ongoing solutions continues to grow during *stopThreshold* iterations, then ACO-SA stops (see line 11 of Algorithm 1);
- **SA initialization:** the initial step of the algorithm includes also the adaptive calculus of the parameters (initial temperature, attenuation coefficient) of SA. This calculus depends on the instance under consideration;
- **SA stop criterion:** the stop criterion of SA is also dynamic: if the costs of an ongoing sequence of solutions is larger than the cost of the best solution explored until now, then SA stops.
- As long as the stop criterion of ACO-SA has not been reached, the following set of operations is executed at each iteration:
  - **Update of the vector sumTrace:** all components of $sumTrace$ are reinitialized to 0, then the sum of the components of the i^th column of the $trace$ matrix is stored in i^{th} component of the vector $sumTrace$ (see lines 12-16 of Algorithm 1);
  - **Computation of neigbor solution based on pheromone memory:** to this matrix ($trace$), we associate a vector ($sumTrace$) of length n such as $\forall 1 \leq i \leq n, sumTrace[i] = trace[i][1] + trace[i][2] + \ldots + trace[i][m]$. Based on this vector, we calculate the neigbor solution. We look for the index i_{min} of the element of $sumTrace$ which contains the minimal value. Then we assign the best possible delay (i.e. delay which minimizes the number of jammed meshes in the coverage area of the i^{th}_{min} transmitter) to the i^{th}_{min} transmitter (see lines 17 and 18 of Algorithm 1);
  - **Run of SA with the neighbor solution as input:** if a stagnation of size *restartSAThreshold* is detected, then adaptive SA algorithm runs with the calculated neighbor solution and the stagnation counter short-Stagnation resets (see lines 19-22 of Algorithm 1).
  - **Update of current and best solutions:** if the cost of the neighbor solution is smaller than that of the current solution, the neighbor solution becomes the current one. The until now best cost becomes the cost of the neighbor solution. The memory matrix (*trace*) is reinitialized. The two counters *longStagnation* (this counter is in charge of stopping the hybrid procedure ACO-SA) and *shortStagnation* (this counter is in charge of restarting the procedure SA) are initialized to 0 (see lines 25-32 of Algorithm 1). If the cost of the neighbor solution is larger than the cost of the current solution, the memory matrix (*trace*) is updated according to the current solution, the neighbor solution and the two parameters *increment* and R (see Algorithm 3). The two counters *shortStagnation* and *longStagnation* are incremented (see lines 33-36 of Algorithm 1).

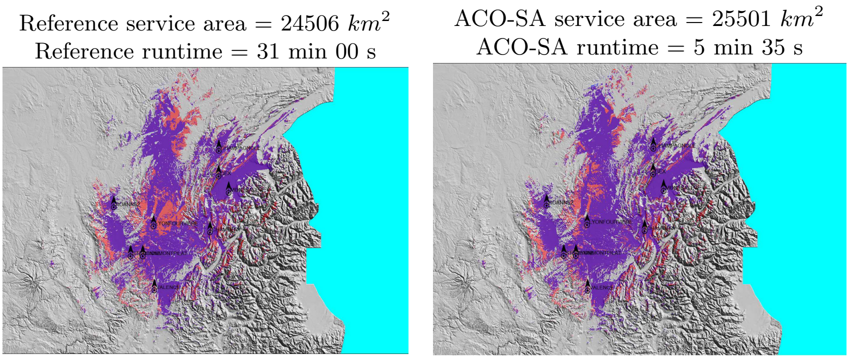
4 Experimental Results

We use real-world benchmarks provided by TDF and compare the experimental results obtained thanks to our ACO-SA metaheuristic with these currently obtained by TDF's software. In Figure 2, Figure 3 and Figure 4 red areas represent jammed areas, and purple areas represent service areas.



(a) (b)

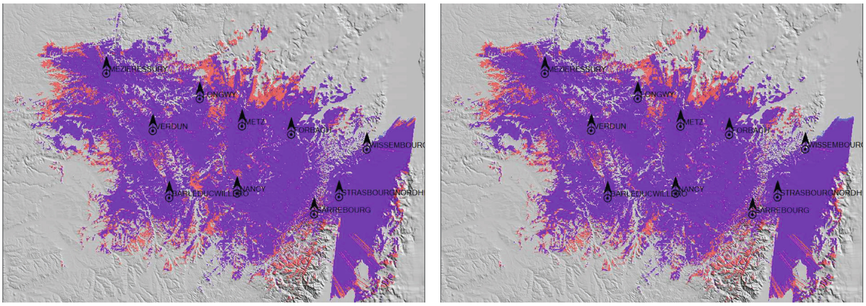
Fig. 2. QoS of reference solution (a) and ACO-SA solution (b) for Benchmark 1



(a) (b)

Fig. 3. QoS of reference solution (a) and ACO-SA solution (b) for Benchmark 2

Reference service area = 37167,8 km^2 ACO-SA service area = 39286,5 km^2
Reference runtime = 05 h 28 min 00 s ACO-SA runtime = 00 h 04 min 21



(a) (b)

Fig. 4. QoS of reference solution (a) and ACO-SA solution (b) for Benchmark 3

– Figure 2.(b) shows that jammed areas have disappeared in comparison with Figure 2.(a). ACO-SA runs 4.7 times faster than TDF's actual software and increases the service area by 13 %.
– Figure 3.(b) shows that jammed areas have been reduced in comparison with Figure 3.(a). ACO-SA runs 5.5 times faster than TDF's actual software and increases the service area by 3.9 %.
– Figure 4.(b) shows that jammed areas have been reduced in comparison with Figure 4.(a). ACO-SA runs 75.4 times faster than TDF's actual software and increases the service area by 5.4 %.

5 Concluding Remarks and Further Research Aspects

Our hybrid ACO-SA metaheuristic has good time performances and improves (or maintains) the quality of the solutions (by at most 13%). There is still room for improvement. We are planning to explore additional hybrid metaheuristics (based on Particle Swarm Optimization, for example) and to design a non trivial, distributed version of ACO-SA.

References

1. Aouad, M.I., Idoumghar, L., Schott, R., Zendra, O.: Sequential and distributed hybrid ga-sa algorithms for energy optimization in embedded systems. In: Proceedings of IADIS International Conference Applied Computing, pp. 167–174 (2010)
2. Brugger, R., Hemingway, D.: Ofdm receivers: impact on coverage of inter-symbol interference and fft window positioning. EBU Technical Review, pp. 19–30 (1996)
3. Connolly, D.T.: An improved annealing scheme for the qap. European Journal of Operational Research **46**, 93–100 (1990)
4. Dorigo, M.: Ant Colont Optimization. Scholarpedia (2007)

5. Granville, V., Kriválek, M., Rasson, J.P.: Simulated annealing: A proof of convergence. IEEE Transaction on Pattern Analysis Machine Intelligence **16**(6), 652–656 (1994)
6. Idoumghar, L., Chrin, N., Siarry, P., Roche, R., Miraoui, A.: Hybrid icapso algorithm for continuous optimization. Applied Mathematics and Computation **219**(24), 11149–11170 (2013)
7. Idoumghar, L., Debreux, P.: New modeling approach to the frequency assignment problem in broadcasting. IEEE Transactions Broadcasting **48**(4), 293–298 (2002)
8. Jensen, T.R., Toft, B.: Graph Coloring Problems. WILEY (1995)
9. Monmarché, N., Guinand, F., Siarry, P. (eds.) Artificial Ants. ISBN 978-1-84821-194-0. Hardback (2010)
10. Gelatt, C.D., Kirkpatrick, S., Vecchi, M.P.: Optimisation by simulated annealing. Science **220**(4598), 671–680 (1983)
11. Taillard, E.D.: Fant: Fast ant system. Technical report, Instituto Dalle Molle Di Studi Sull Intelligenza Artificiale (1998)
12. Weck, C.: Coverage aspects of digital terrestrial television broadcasting. EBU Technical Review, pp. 19–30 (1996)