A New Hybrid GA-MDP Algorithm For The Frequency Assignment Problem

Lhassane IDOUMGHAR
LMIA-MAGE,
Faculté des Sciences et Techniques
4 rue des Frères Lumière,
68093 Mulhouse, France
lhassane.idoumghar@uha.fr

René Schott
IECN and LORIA,
Université Henri Poincaré-Nancy 1,
54506 Vandoeuvre-lès-Nancy, France
schott@loria.fr

Abstract

We propose a novel algorithm called GA-MDP for solving the frequency assigment problem. GA-MDP inherits the spirit of genetic algorithms with an adaptation of Markov Decision Processes (MDPs). More precisely Policy Iteration (PI) and Value Iteration (VI) are used as mutation operators. Experimental results show that for our application, GA-MDP that uses PI as a mutation operator improves the quality and time performances of the hybrid algorithms and hybrid MDP ([1, 2]) designed previously by the authors for solving the same problem.

1. Introduction

In order to perform suitable geographic and/or demographic coverages for radiobroadcasting or radiocommunications, it is necessary to distribute large numbers of transmitters over the geographic areas. To each transmitter is assigned one frequency. Unfortunately, to each diffusion system is assigned a finite frequency spectrum, which makes it necessary to reuse frequencies. This problem generates interference phenomena due to same or adjacent frequencies allocated to neighbouring transmitters. Here, it is of major importance to minimize the interference while at the same time to use the spectrum efficiently. This problem, known to be NP-Hard, can be stated as finding an optimal assignment of frequencies to a set of transmitters, under various constraints. The frequency assignment problem can be represented as a generalised graph colouring problem. Transmitters are represented by the vertices of a constraints graph [3]. Each edge of this graph represents a constraint on the frequencies assigned to the two transmitters connected by this edge.

The graph colouring problem is one of the most studied NP-Hard problems, and can be defined informally as follows. Given an undirected graph, one wishes to colour, with a minimal set of colours, the vertices of the graph in such a way that two adjacent vertices are assigned different colours. Due to the NP-Hardness of this problem [4] heuristic methods must be used for large graphs. Many heuristic methods have been developed: constructive methods, local search meta-heuristics [5, 6, 7], hybrid algorithms [1, 2] and ant colony paradigms [8].

In this paper, we present an original algorithm, called GA-MDP, for solving the frequency assignment problem. An application of this new algorithm in the field of radiobroadcasting leads to enhanced performances (in terms of quality of the solutions and time complexity) compared to several other approaches like hybrid genetic algorithms.

This paper is organized as follows: Section 2 describes the radio network planning process. Section 3 recalls the graph colouring problem. Section 4 gives a description of genetic and MDPs algoritms. Section 5 describes our approach. In Section 6 we present some experimental results. Concluding remarks and further research aspects are contained in the last section.

2 Radio Network Planning

As the demand for telecommunication services increases (interactive TV, mobile phones, FM radio, etc.), providers need fast and efficient design tools to help guarantee their commercial success. By using such tools the intent is to reduce the planning costs and shorten the duration of network deployment by employing various optimisations. Keeping the required hardware investments to a minimal level while achieving a high quality of service is the basic principle of network planning.

The planning process can be split into two main steps. The first one, called *dimensionning*, consists in spreading an optimal number of transmitters over an area in order for this area to be efficiently covered. This number should be as small as possible for obvious economic reasons. Transmitters are spread over the geographical area where providers

wish to provide the users with their services, each transmitter covers a part of this geographical area called its *coverage area* (Figure 1). The area around a transmitter where transmission conditions are favourable enough to have a good receipt 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 transmission quality and thus the shapes of these regions will depend heavily on the propagation conditions and current interferences produced by the other transmitters. Therefore, coverage areas are frequently of highly irregular shape.

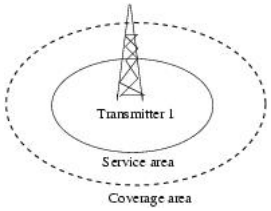

Figure 1. Coverage and service area of a transmitter

Let W be a set of transmitters distributed accross the geographical area. We build a constraints graph as follows:

1. apply a propagation routine to each element of W,
2. for each pair of elements in W, compute the intersection of the two corresponding coverage areas,
3. each element in W defines a vertex of the graph.

There exists an edge between two vertices if the overlapping of the two corresponding coverage areas is too much important. In Figure 2 at interference area the signal sent by transmitter 2 interfers with the one sent by transmitter 1. An example of a constraints graph of the radio network given in Figure 2 is given in Figure 3.

The weight of each edge depends on the kind of radio network. For example, in radiobroadcasting the weight assigned to an edge stands for the gap between the frequencies assigned to the vertices at the two endpoints. For example, it is represented by an integer value ranging from 1 to 5. Consequently, at the end of the dimensionning step we are provided with a weighted graph.

In the second step, which is the *frequency assignment* itself, we look for a best frequency to be assigned to each transmitter. This problem is the same as the well known graph colouring problem that we recall in the next section.

3 The Graph Colouring Problem

Let *G=(W, E)* be an undirected graph, where $W = \{w_1, ..., w_n\}$ is the set of vertices and $E = \{(w_i, w_j)/w_i, w_j \in W\}$ is the set of edges.

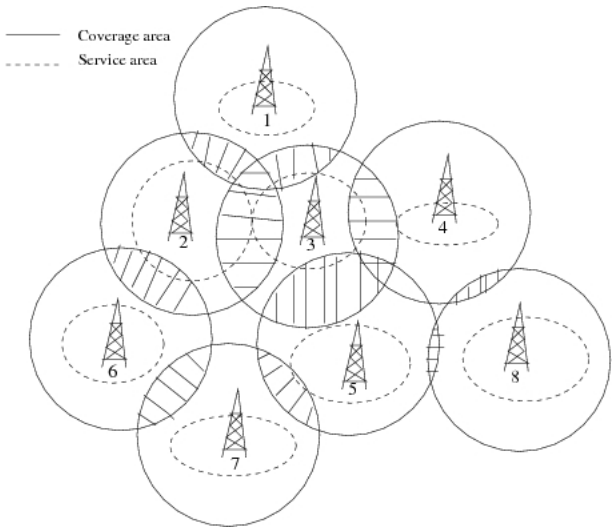

Figure 2. Influence of a transmitter beyond of its service area

Let $M = (m_{i,j})^{n*n}$ be a symmetric compatibility matrix defined by:

- n represents the number of vertices of the graph,
- $m_{i,j}$ with $i \neq j$ represents the minimum colour separation required to satisfy the constraint between vertices w_i and w_j,

Informally, a graph colouring assignment is a mapping $f : W \rightarrow C$ (where C is a set of consecutive integers 0,..,k representing colours) such that the constraints $\mid f(w_i) - f(w_j) \mid > m_{i,j}$ are satisfied for all $w_i, w_j \in E$. A feasible solution of this problem is represented by a vector $\overrightarrow{C}(c_0, c_2, ..., c_k)$ where $c_i \in C$ for all $i \in [0, k]$. The components of $\overrightarrow{C}$ are essentially the colours assigned to the vertices of G. This solution must satisfy all constraints if possible, otherwise the number of violated constraints must be minimized. In this case, our priority is to satisfy all highest level constraints.

4 Background

4.1 Genetic algorithms

Genetic algorithms were developed by Holland [9] to study the adaptative process of natural systems and to develop artificial systems that mimic the adaptative mechanism of natural systems. A review of their theoretical and practical aspects can be found in [10].

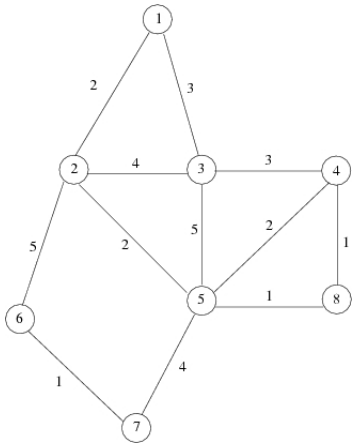

Figure 3. Constraints graph.

Recently, genetic algorithms have been successfully applied to various optimization problems, such as the frequency assignment problem [1]. Genetic algorithms differ from traditionnal optimization methods in the following ways.

1. Genetic algorithms use a coding of the parameter set rather than the parameters themselves.
2. Genetic algorithms search from a population of search solutions instead from a single one.
3. Genetic algorithms use probabilistic transition rules.

A genetic algorithm consists of a string representation ("genes") of the solutions (called individuals) in the search space, a set of genetic operators for generating new search individuals, and a stochastic assignment to control the genetic operators.

Tipically, a genetic algorithm consists of the following steps.

1. Initialization - an initial population of the search solutions is randomly generated.
2. Evaluation of the fitness function - the fitness value of each individual is calculated according to the fitness function (objective function).
3. Genetic operators - new individuals are generated randomly by examining the fitness value of the search individuals and applying the genetic operators to the individuals.
4. Repeat steps 2 and 3 until the algorithm converges.

From the above description, we can see that genetic algorithms use the notion of survival of the fittest by passing "good" genes to the next generation of strings and combining different strings to explore new search solutions. The construction of genetic algorithm for any problem can be separated into four distinct and yet related tasks.

1. The choice of the representation of the strings,
2. The design of the genetic operators,
3. The determination of the fitness function, and
4. The determination of the probabilities controlling the genetic operators.

Each of the above four components greatly affects the solution obtained as well as the performance of the genetic algorithm.

In Section 5, we examine each of them for the frequency assignment problem.

4.2 Markov Decision Processes

Markov Decision Processes [11] are one tool of Artificial Intelligence that can be used to get optimal action policies under a stochastic domain. Markov decision processes are utilized as a stochastic model of an agent that is interacting with the world. Given an action executed in a known state of the world, it is possible to calculate the probability of the next state in the world. The probability of reaching a state s' when the action a has occurred is calculated by the summation of the conditional probability of reaching a state s' from every possible state s_i of the world given the action a.

A formal description of an MDP is the tuple (S, A, T, R) where:

- S is a finite set of states of the world,
- A is a finite set of actions,
- $T : S$ x $A \rightarrow P$ is the transition function of states and P represents the transition matrix. For each action and state of the world, there is a probabilistic distribution over states of the world that can be reached by executing this action. The function $T(s, a, s')$ is defined as the probability $p_{s,s'}$ of reaching state s' starting in state s and given the action a.
- $R : S$ x $A \rightarrow \Re$ is a reward function. To each action in each state of the world, is assigned a real number. The function $R(s, a)$ is defined as the reward of executing action a in the state s.

MDP techniques provide policy actions such that the sum of rewards obtained when the actions are executed is maximal. MDPs provide an optimal policy for solving a specific problem if the state of the world is known in every step of the policy.

GA-PI algorithm

- Intitialize p_m, $p_c \in$]0,1], K > 0, maxGen > 0, cpt := 0 and i := 1
- Construct constraints graph
- Generate population P_0
- Evaluate P_0 and find the best solution π^*
- $\pi_{Elite} \leftarrow \pi^*$
- **Repeat**
  - $P_i \leftarrow \emptyset$
  - **For** j:= 1 to $PopSize/2$ **do**
    - Select two parents p_1 and p_2 from P_{i-1} offspring $\leftarrow (p_1, p_2)$
    - With a probability p_c, perform offspring := crossover(p_1,p_2)
    - With a probability p_m, mutate offspring by using the *Policy Iteration* algorithm
    - Evaluate offspring and add it to P_i
  - Add P_{i-1} to P_i
  - Sort P_i
  - Delete the $PopSize$ worst offspring in P_i
  - Stop criteria
    - if $\pi_{Elite} = \pi^*$ then $cpt \leftarrow cpt + 1$
    - π^* is better than π_{Elite} then cpt := 0 and $\pi_{Elite} \leftarrow \pi^*$
    - if $fitness(\pi_{Elite}) = 0$ OR $cpt = K$ OR $i = maxGen$ then terminate GA-PI
- Return π_{Elite}

Figure 4. GA-MDP Algorithm that uses PI approach as a mutation operator

Two main algorithms are usually used to compute such an optimal policy for an MDP, *Policy Iteration* [12] and *Value Iteration* [11] (more details are given in section 5) .

5 GA-MDP Algorithm

As indicated above, genetic algorithms have been introduced by Holland [9] to mimic some of the processes of natural evolution and selection. A genetic algorithm can be implemented in several different ways.

In this section we describe two approaches: GA-PI (respectively GA-VI) whose principle consists in using in the genetic algorithm a *Policy Iteration* (respectively *Value Iteration*) algorithm as mutation opertor. Figure 4 shows a high-level description of GA-PI, where some steps are described at a conceptual level with details provided in the following subsections.

5.1 Description of an individual

Each individual represents a possible solution to the problem and is composed of a string of genes. In our implementation a gene is coded by an integer array. A possible solution represents one graph colouring. i^{th} value corresponds to the colour assigned to vertex w_i. In such graph, some vertices would have fixed colours. These vertices are placed at the end of the string. The size of each individual is $\mid W \mid$.

5.2 Initial Population

Each genetic algorithm requires an initial population P_0 to serve as the starting point. This initial population is created randomly. In figure 4, *PopSize* is the size of every population P_i. During each of the *maxGen* generations, *PopSize* offsprings are generated through the crossover of parents selected from the population.

5.3 Fitness evaluation

The fitness function in genetic algorithms is typically the objective function that we want to optimize in the problem. It serves for each individual to be tested for suitability to the environment under consideration. Our objective function is defined as follows: $F = \sum_{i,j} \rho_{i,j}\delta_{i,j}$, where $\rho_{i,j}$ is a weight associated with constraint $m_{i,j}$ ($\rho_{i,j}$ and $m_{i,j}$ are positive integers). $\delta_{i,j}$ is defined by:

$$\delta_{i,j} = \begin{cases} 0 & \text{if } \mid f(w_i) - f(w_j) \mid \geq m_{i,j} \\ m_{i,j} - \mid f(w_i) - f(w_j) \mid & \text{otherwise} \end{cases}$$

5.4 Selection

The tournament selection principle increases the chances for poor quality individuals to take part in the improvement of the population. Several individuals are randomly chosen in the population, the winner of the tournament is granted with the highest quality.

5.5 Crossover operator

The crossover is a random process defined by a probability p_c and applied sequentially to pairs of parents chosen randomly in the population. It consists in exchanging parts of the genetic material of the parents in order to create two childs (offspring).

In our approach we have used the uniform crossover [13] obtained from a binary mask which possesses a number of genes equal to the number of genes of the individuals of the population. This mask is usually uniformly randomly generated: each bit has value 0 or 1 with equal probability. The used uniform crossover is parameterized by a function

of the probability $p_0 = 0.5$ [14], corresponding to appearance of the values (0 and 1) in each bit of the mask. The first child is created by taking the genes of the first parent corresponding to value 1 in the mask. The second child is obtained similarly but the complementary part of the mask is used.

5.6 Mutation operators

5.6.1 PI-mutation operator

The first mutation operator that we have used is based on Policy Iteration algorithm.

Policy Iteration iteratively maximizes (in our approach) the value function, which is formulated as follows:

$$V_\pi(s) = R(s, \pi_t(s)) - \gamma \sum_{s' \in S} T(s, \pi_t(s), s') V_{\pi_{t-1}}(s') \quad \textbf{(1)}$$

where γ ($0 \leq \gamma < 1$) is a discount factor that is used to give more or less importance to the future rewards. This factor is required to consider an infinite-horizon discounted model [15] in wich the number of iterations is in principle infinite. However, the discount factor γ provokes the convergence in a polynomial number of steps [16]. $\pi_t(s)$ is the action executed at state s in step t in the policy π. To find a policy, a set of $|S|$ linear equations in $|S|$ unknowns has to be solved. The initial policy is successively improved until an optimal one is found. The principle of PI is summarized in Figure 5.

- π':= any policy
- While $\pi \neq \pi'$
  - $\pi := \pi'$
  - For all $s \in S$
    - Compute $V_\pi(s)$ by solving the system of $|S|$ equations in $|S|$ unknowns given by equations (1)
  - For all $s \in S$
    - If there exist an action $a \in A$ such that: $R(s,a) - \gamma \sum_{s' \in S} T(s,a,s') V_\pi(s') > V_\pi(s)$
    - Then $\pi'(s) := a$
    - Else $\pi'(s) := \pi(s)$
- Return π

Figure 5. Policy Iteration Algorithm

Each iteration of the algorithm consists of two steps: solving the linear equations system, which can be done in $O(|S^3|)$ operations, and the improvement of the current policy, wich needs $O(|A||S^2|)$ operations [17]. The algorithm is guaranted to converge [12], and generally tends to do so over a few number of iterations [18].

The reward function used in this algorithm is defined as follows: $R(s,a) = -\sum_{s,s'} \rho_{s,s'} \delta_{s,s'}$, where $\rho_{s,s'}$ is a weight associated to constraint $m_{s,s'}$ ($\rho_{s,s'}$ and $m_{s,s'}$ are positive integers). $\delta_{s,s'}$ is defined by:

$$\delta_{s,s'} = \begin{cases} 0 & \text{if } \mid a - \pi(s') \mid \geq m_{s,s'} \\ m_{s,s'} - \mid a - \pi(s') \mid & \text{otherwise} \end{cases}$$

An optimal policy π defines for each transmitter of S an optimal frequency to be assigned. Several optimality criteria can be used; here we will focus on infinite horizon discounted decision problems. The optimal policy maximizes the gain of each transmitter ($R(s, \pi(s)) = 0$ for all $s \in S$). This means that we satisfy all constraints.

5.6.2 VI-mutation operator

A second mutation operator that we used is the Value Iteration [11] algorithm. The principle of this mutation operator is given in Figure 6.

t := 0
// Initialize state value at 0
For all $s \in S$
  $V_0(s) := 0$
Repeat
  t ← t + 1
  For all $s \in S$
    $V_t(s) = max_a\{R(s,a) - \gamma \sum_{s' \in S} T(s'/s,a) V_{t-1}(s')\}$
    $\pi(s) = argmax_a\{R(s,a) - \gamma \sum_{s' \in S} T(s'/s,a) V_{t-1}(s')\}$
  End For
until $max_s |V_t(s) - V_{t-1}(s)| < \varepsilon$

Figure 6. Value Iteration Algorithm

This algorithm works by computing the optimal value function assuming first a one-stage finite horizon, then a two-stage finite-horizon, and so on. The values functions so computed are guaranteed to converge in the limit to the value function V^*. In addition, the policy associated with the successive values functions will converge to the optimal policy π^* in a finite number of iterations [19], and in practice this convergence can be quite rapid.

6 Experimental results

This section presents the main results obtained after several runs performed over two real instances of frequency assignment problems provided by TDF-C2R Broadcasting and Wireless Research Center:

- *instance1*: more than 970 transmitters and more than 12900 constraints,
- *instance2*: more than 970 transmitters and more than 25550 constraints,

	HGA	HMDP	ANT
Total violated constraints	68	68	70
constraints 1	23	23	17
constraints 2	38	38	41
constraints 3	6	3	9
constraints 4	1	4	3
constraints 5	0	0	0
run time	37 s	8 s	71s

	GA-VI	GA-PI	CSA
Total violated constraints	69	66	142
constraints 1	22	16	37
constraints 2	39	38	68
constraints 3	4	7	15
constraints 4	3	5	6
constraints 5	1	0	16
run time	11s83	6s27	-

Table 1. Comparison of the results obtained by different methods used to solve *instance1*.

	HGA	HMDP	ANT
Total violated constraints	93	93	99
constraints 1	44	47	47
constraints 2	45	39	42
constraints 3	2	2	8
constraints 4	2	5	2
constraints 5	0	0	0
run time	92s	27s	408s

	GA-VI	GA-PI	CSA
Total violated constraints	89	87	271
constraints 1	38	44	92
constraints 2	44	35	117
constraints 3	4	5	27
constraints 4	3	3	10
constraints 5	0	0	25
run time	18s46	13s77	-

Table 2. Comparison of the results obtained by different methods used to solve *instance2*.

Each vertex of the contraints graph may take $m = 200$ possible values (bandwidth 87.5-107.5 MHz, | 107.5 - 87.5 MHz | = 20 MHz = m x 100 kHz).

We recall that the main objective is to satisfy all constraints (interference constraints) if it is possible, otherwise high level constraints would have to be satisfied first.

The results produced by the approach presented in this paper and those obtained by other approaches [1, 2, 8] compared to the best operating solution in the field of radiobroadcasting given by CSA [20] in France are given in Table 1, Table 2, Table 3 and Table 4.

In these tables:

- HGA: means that we use a hybrid genetic algorithm [1], this aghorithm uses a probabilistic tabu search algorithm as a mutation operator,
- HMDP: means that we use hybrid MDP algorithm [2] that uses a probabilistic tabu search to generate an initial policy, and next we use a Policy Iteration algorithm to find the best solution.
- ANT: means that we use an ant algorithm [8]. This algorithm is a multiagent system based on the idea of parallel search. In this algorithm, a given number of ants move around the vertices of the graph and change the colour of each visited vertex according to a local criterion. At each iteration each ant k moves from vertex i to vertex j and changes its colour, it remains there until the next iteration, when it will move again towards one of the vertices of j's neighbourhood. Each ant can "remember" at each step the former changes performed by the algorithm, and takes into account that these changes may have modified the cost function of the neighbourhood of j. Therefore, at the next step, the ant k will normally try to arrange the colouring of the worst adjacent vertex to j. Any single action depends strongly on the last move of each ant; this dependence reinforces the results of recent modifications. At a given iteration each ant moves from the current vertex to the adjacent vertex with the highest level of violations, and replaces the old colour of the vertex with a new colour that minimizes these violations. For a given vertex, the highest level of violations is computed by using the *Cost* function described in Section 4. This action is randomly repeated for each ant: the ant moves to the worst adjacent vertex with a certain probability p_n (otherwise it moves to any other adjacent vertex randomly chosen), and assigns the best possible colour with a probability p_c (otherwise any colour is assigned at random). The probabilistic nature of the algorithm allows the ants to escape from local minima and to obtain bounds close to the absolute minimum. This process, which is carried out simultaneously by the set of ants, is repeated until the optimal solution is found or the algorithm converges. The num-

ber of ants that move along the graph is an adjustable parameter and increases with the order of the graph.

- GA-VI: means that the genetic algoritm uses a Value Iteration (VI) algorithm as mutation operator.
- GA-PI: means that the genetic algoritm uses a Policy Iteration (PI) algorithm as mutation operator.
- CSA: corresponds to the best operating solution in the field of radiobroadcasting given by the CSA [20] in France,
- The second row gives the total number of constraints that are violated by each approach,
- The next five rows give the details of the number of total violated constraints for each constraint type,
- The last row gives the run time of each algorithm.

By analysing the results given in these tables, several observations can be made:

- First, by comparing the run time of all algorithms we can observe that by using *GA-PI* the best results are obtained quicker.
- Second, the results obtained by *GA-VI* approach are still competitive with those obtained by *GA-PI* algorithm but GA-PI ,nevertheless, outperforms GA-VI.
- Third, we observe that both algorithms produce a better solution than the best existing operating solution in the field of radiobroadcasting in France,
- Another interesting observation is that the results obtained by *GA-PI* are competitive with those obtained by *HMDP* and constraints with high level are firstly satisfied,

7 Concluding remarks

This paper has described a new hybrid GA-MDP algorithm applied to solve the frequency assignment problem in the radiobroadcasting domain. This algorithm uses two mutation operators. The first mutation operator uses a *Policy Iteration* algorithm (approach called GA-PI). The second mutation operator is based on *Value Iteration* algorithm (approach called GA-VI).

A comparison of the results obtained by using the the designed GA-PI algorithm shows that we improve the quality of the solution and time performances of the hybrid algorithms.

Currently, we are studying more hybridization possibilities for solving the frequency assignment problem. We are also working on the implementation and experimentation of the algorithms proposed by Chang et al. ([21, 22]) on our application.

References

[1] M. Alabau and L. Idoumghar and R. Schott, New Hybrid Genetic Algorithms for the Frequency Assignment Problem, *Journal of IEEE Transactions on Broadcasting*, March 2002, Vol.48, N.1, 27-34.

[2] L. Idoumghar and R. Schott and J.Y Greff, Application of Markov Decision Processes to the Frequency Assignment Problem, *Journal of Applied Artificial Intelligence*, pp. 761-773, Vol. 18, N 8, September 2004.

[3] S. Hurley and D.H. Smith and S.U. Thiel, Fasoft: A System for Discrete Frequency Assignment, *Radio Science*, 32(5), 1997, 1921-1939.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman and Co., 1979.

[5] J.M. Coupé, "Using a Tabu Search to Solve the Frequency Assignment Problem", *Patent TDF*, N. 9902768, 1999.

[6] F. Glover and E. Taillard, A User's Guide to Tabu Search, *Annals of Operations Research*, 41, 1993, 3-28.

[7] L. Idoumghar and Ph. Debreux, New Modeling Approach for the Frequency Assignment Problem in Broadcasting, *Journal of IEEE Transactions on Broadcasting*, pp. 293-298, Vol. 48, N 4, December 2002.

[8] F. Comellas and J. Ozón, An Ant Algorithm for the Graph Colouring Problem, *Proc. ANTS'98- From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimiszation*, Brussels, Belgium, October 1998.

[9] J. H. Holland, Adaptation in Natural and Artificial Systems, *PhD thesis*, University of Michigan press., 1974.

[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, *Addison Wesley*, 1989.

[11] R. Bellman,Dynamic Programming, Princeton University Press 1957.

[12] R. A. Howard, Dynamic Programming and Markov Processes, Cambridge, Massachussets: MIT Press, 1960.

[13] G. Syswerda, Uniform Crossover in Genetic Algorithms, *Proc. of Int. Conf. on Genetic Algorithms (ICGA'89)*, pp. 2-9, 1989.

[14] W.M. Spears and K.A. De Jong, On the Virtues of Parametrized Uniform Crossover, *Proc. of the* 4^{th} *Int. Conf. on Genetic Algorithms and Their Applications*,

University of California, San Diego, Morgan Kauffmann Publishers, pp. 230-236, 1991.

[15] L.P. Kaelbling, M.L. Littman and A.R Cassandra, Planning and Acting in Partially Observable Stochastic Domains, *Artificial Intelligence*, 101(1-2), 1998, 99-134.

[16] R. Givan, S. Leach and T. Dean, Bounded-parameter Markov Decision Processes, *Artificial Intelligence*, 122(1-2), 2000, 71-109.

[17] Michael L. Littman and Thomas L. Dean and Leslie Pack Kaelbling, On the Complexity for Solving Markov Decision Problems, *Proc. of Int. Conf. on Uncertainty in Artificial Intelligence*, 1995, 394-402.

[18] M.L. Puterman, Markov Decision Processes, New York: John Wiley & Sons, 1994.

[19] Bertsekas, D.P, Dynamic Programming: Deterministic and Stochastic Models. Prentice-Hall, Englewood Cliffs, N.J, 1987.

[20] The "Conseil suprieur de l'audiovisuel" is an independent administrative authority that was created in France to guarantee broadcasting freedom in the conditions laid down by the modified Law of September 30th, 1986. "http://www.csa.fr/multi/introduction/intro.php?l=uk"

[21] Hyeong Soo Chang and Hong-Gi Lee and Michael C. Fu and Steven I. Marcus, Evolutionary Policy Iteration for Solving Markov Decision Processes, *Journal of IEEE Trans. on Automatic Control*, 50, Nov. 2005, pp. 1804-1808.

[22] Hyeong Soo Chang and Walter J. Gutjahr and Jihoon Yang and Sungyong Park, An Ant System Approach to Markov Decision Processes, September 2003.