Soft Computing approaches on the Bandwidth Problem

Gabriela Czibula[a], Gloria Cerasela Crişan[b], Camelia-M. Pintea[c], Istvan-Gergely Czibula[a]

[a] *Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania*
[b] *Faculty of Science, "Vasile Alecsandri" University of Bacău, Romania*
[c] *Technical University Cluj Napoca, North University Center Baia Mare, Romania*

Abstract

The Matrix Bandwidth Minimization Problem (*MBMP*) seeks for a simultaneous reordering of the rows and the columns of a square matrix such that the nonzero entries are collected within a band of small width close to the main diagonal. The *MBMP* is a NP-complete problem, with applications in many scientific domains, linear systems, artificial intelligence, and real-life situations in industry, logistics, information recovery. The complex problems are hard to solve, that is why any attempt to improve their solutions is beneficent. *Genetic algorithms* and *ant-based systems* are Soft Computing methods used in this paper in order to solve some *MBMP* instances. Our approach is based on a learning agent-based model involving a local search procedure. The algorithm is compared with the classical *Cuthill-McKee* algorithm, and with a hybrid genetic algorithm, using several instances from *Matrix Market* collection. Computational experiments confirm a good performance of the proposed algorithms for the considered set of *MBMP* instances. On *Soft Computing* basis, we also propose a new theoretical *Reinforcement Learning* model for solving the *MBMP* problem.

Keywords: Combinatorial optimization, Matrix Bandwidth Minimization Problem, Soft Computing, Reinforcement Learning

1. Introduction

Combinatorial optimization is the seeking for one or more optimal solutions in a well defined discrete problem space. In real life approaches, this means that people are interested in finding efficient allocations of limited resources for achieving desired goals, when all the variables have integer values. As workers, planes or boats are indivisible (like many other resources), the Combinatorial Optimization Problems (COPs) receive today an intense attention from the scientific community.

The current real-life COPs are difficult in many ways: the solution space is huge, the parameters are linked, the decomposability is not obvious, the restrictions are hard to test, the local optimal solutions are many and hard to locate, and the uncertainty and the dynamism of the environment must be taken into account. All these characteristics, and other more, constantly make the algorithm design and implementations challenging tasks. The quest for more and more efficient solving methods is permanently driven by the growing complexity of our world.

The *Matrix Bandwidth Minimization Problem (MBMP)* is a fundamental mathematical problem, searching for a simultaneous permutation of the rows and the columns of a square matrix that keeps its nonzero entries as much as possible close to the main diagonal. This problem is NP-complete in general [18], and it remains so even in restricted solutions spaces [8], that is why any attempt to improve its solutions is beneficent.

The main contribution of this paper is to emphasize the effectiveness of using *soft computing* methods in order to solve the *Matrix Bandwidth Minimization Problem*. *Genetic algorithms* and *ant-based systems* are natural computing methods used in this paper in order to solve the *MBMP* instances. Computational experiments confirm that these methods provide robust and low-cost solutions for the *MBMP*. We also introduce a new theoretical *reinforcement learning* model for solving the *MBMP*. So far, such a learning model has not been reported in the *MBMP* literature.

The rest of the paper is organized as follows. Section 2 briefly presents the matrix bandwidth minimization problem, emphasizing its relevance and also presenting existing approaches for solving it. The fundamentals of the soft computing approaches considered in this paper, i.e genetic algorithms, ant colony systems and reinforcement learning, are given in Section 3. In Section 4 we propose two *natural computing* methods for solving the *MBMP* instances, namely *genetic algorithms* and *ant colony systems*. A theoretical reinforcement learning model for solving *MBMP* is introduced in Section 5. Section 6 provides an experimental evaluation of the proposed methods and Section 7 contains some conclusions of the paper and future development of our work.

Email addresses: gabis@cs.ubbcluj.ro (Gabriela Czibula), ceraselacrisan@ub.ro (Gloria Cerasela Crişan), cmpintea@yahoo.com (Camelia-M. Pintea), istvanc@cs.ubbcluj.ro (Istvan-Gergely Czibula)

2. The Matrix Bandwidth Minimization Problem

This section introduces the concept and the literature review related to the *Matrix Bandwidth Minimization Problem*.

2.1. Matrix Bandwidth Minimization Problem description

Given a square positive symmetric matrix $\mathcal{A} = (a_{ij})_{1 \leq i,j \leq n}$ the bandwidth β is the value $\beta(\mathcal{A}) = max_{a_{ij} \neq 0} |i-j|$. The Matrix Bandwidth Minimization Problem searches for a row (and column) permutation π that minimizes the bandwidth for the new matrix.

An equivalent form of *MBMP* uses the graph-theory approach, based on the *layout* notion. Given an undirected, connected graph *G=(V, E)*, a layout σ is a bijection between V and $\{1, 2, \ldots |V|\}$. The bandwidth of G is $\beta(G) = min_{\sigma}\ (max_{(u,v) \in V}(|\sigma(u) - \sigma(v)|))$. Intuitively, computing the bandwidth for a graph is to find a linear ordering of its vertices that minimizes the maximum distance between two adjacent vertices.

Starting from the given matrix $\mathcal{A}$, an equivalent graph $G_{\mathcal{A}} = (V, E)$ can be defined and the *MBMP* can be viewed as the problem of minimizing the bandwidth of $G_{\mathcal{A}}$. In this graph, the set of vertices is $V = \{1, 2, \ldots, n\}$ and two vertices i and j are connected through an edge *iff* $a_{ij} \neq 0$, i.e $E = \{(i, j)\ \textit{iff}\ a_{ij} \neq 0\}$.

The current exact approaches devise algorithms that solve the general *MBMP* in $O(4.83^n)$ running time [3]. Classic results for approximation approaches establish an approximation factor of $O(log^{3.5} n)$ for general *MBMP* [7] and $O(log^{2.5} n)$ for trees [10].

The *MBMP* arose in the solving systems of linear equation; the ordering of the system matrix has great impact on the resources needed when actually solving the system, and may lead to a substantial efficiency increase. Minimizing the bandwidth of a matrix helps in improving the efficiency of certain linear algorithms, like Gaussian elimination.

The *MBMP* current applications in computer science include VLSI design, network survivability, data storage. Other applications are in electromagnetic industry [6], large-scale power transmission systems, chemical kinetics and numerical geophysics [15], information retrieval in hypertext [1].

Some generalizations of *MBMP* are currently investigated by the world researchers. For example, the two-dimensional bandwidth problem is to embed a graph into a planar grid such that the maximum distance between adjacent vertices is as small as possible [14].

2.2. Literature review.

The importance of the bandwidth minimization problem is also reflected by the large number of publications describing algorithms for solving it. Cuthill and McKee propose in 1969 in [2] the first stable heuristic method for *MBMP*: the *CM* algorithm with *Breadth-First Search*.

Marti et al. have used in [15] Tabu Search for solving the *MBMP* problem. They used a candidate list strategy to accelerate the selection of moves in the neighborhood of the current solution.

A *GRASP* with *Path Relinking* method given by Pinana et al. in [19] has been shown to achieve better results than the Tabu Search procedure but with longer running times. Lim et al. propose in [11] a Genetic Algorithm integrated with Hill Climbing to solve the bandwidth minimization problem.

A simulated annealing algorithm is shown in [21] for the matrix bandwidth minimization problem. The algorithm proposed by Tello et al. is based on three distinguished features including an original internal representation of solutions, a highly discriminating evaluation function and an effective neighborhood. More recently, the *Ant Colony Optimization* metaheuristic has been used in [13], [20] in order to solve the the *MBMP*.

3. Background

In this section we will briefly review the fundamentals of the soft computing approaches used in this paper for solving the *MBMP*, i.e *genetic algorithms*, *ant colony optimization* and *reinforcement learning*.

Soft Computing is the collection of computing branches that cope with the imprecision, uncertainty, partial truth, and approximation, manifested in nature and naturally (and gracefully) operated by biologic entities (cells, organisms, or collections of individuals). The goal of soft computing approaches is to achieve tractability, robustness and low-cost solutions, facing the real-life, complex, highly-dimensioned problems.

Genetic algorithms (GAs), invented by John Holland in the 1960s, are the most widely used approaches to computational evolution. Genetic algorithms provide an approach to machine learning [16], method motivated by analogy to biological evolution. Hypotheses are often described by bit strings whose interpretation depends on the application, though hypotheses may also be described by symbolic expressions or even computer programs [9].

Ant Colony Optimization (ACO) studies artificial systems inspired by the behavior of real ant colonies and which are used to solve COPs [5]. The *ACO* methods use a set of cooperative artificial ants, each constructing a solution, based on the expected quality of the available moves and on the good solutions found by the community. *ACO* demonstrated a high flexibility and strength by solving with very good results either academic instances of many COPs or real-life problems. To improve the efficiency, the ant-based algorithms are designed using problem-specific information and involve local search methods.

The goal of building systems that can adapt to their environments and learn from their experiences has attracted researchers from many fields including computer science, mathematics, cognitive sciences [22].

Reinforcement learning (RL) is learning what to do - how to map situations to actions - so as to maximize a numerical *reward* signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them. In RL, the computer is simply given a goal to achieve. The computer then learns how to achieve that goal by trial-and-error interactions with its environment.

4. Natural computing models for the MBMP

In this section we propose two *natural computing* methods for solving the *MBMP* instances: *genetic algorithms* and *ant colony systems*. The computational experiments from Section 6 confirm that these methods provide robust and low-cost solutions for the *MBMP*.

4.1. Genetic Algorithm.

In the following, a hybrid genetic algorithm (*HGA*) is proposed for solving the *MBMP*. The algorithm proposed in this section is a slight modification of the Genetic Algorithm integrated with Hill Climbing proposed by Lim et al. in [11].

Let us consider that $\mathcal{A} = (a_{ij})_{1 \leq i,j \leq n}$ is the square symmetric matrix whose bandwidth β has to be minimized.

In the *HGA* we use, a chromosome is a n dimensional sequence $\pi_1, \pi_2, \ldots \pi_n$ representing a permutation π of $\{1, 2, \ldots, n\}$. Thus, a matrix $\mathcal{A}_\pi$ can be associated to a chromosome π, i.e the matrix obtained starting from the matrix $\mathcal{A}$ by permuting it rows (and columns) in the order given by permutation π. The fitness function associated to a chromosome π is defined as the bandwidth of the corresponding matrix, i.e $fitness(\pi) = \beta(\mathcal{A}_\pi)$. The problem consists of minimizing the fitness function, i.e finding the individual with the minimum associated fitness value.

We have used the traditional structure for a genetic algorithm, adding a Hill Climbing step in order to quickly tune solutions to reach local optimum [11]. *HGA* algorithm operates as follows:

i. At the beginning, an initial group of n chromosomes is constructed, as it will be further detailed.

ii. Then, middle-point crossover and a k-swap mutation are performed on this group of chromosomes to generate new chromosomes [11]. Hill Climbing is now applied to each newly-generated chromosome, as proposed in [11]. As the number of individuals within a population remains n, fittest chromosomes will remain in the next generation. After the new generation is formed, a swap mutation is applied on all chromosomes within the new generation, excepting the best one. Then, Hill Climbing is applied again to each newly-generated chromosome.

iii. Step [ii.] is repeated for a given number of generations; the algorithm stops and the best result is reported as solution.

The initial population for the *HGA* is constructed as follows. Starting from matrix $\mathcal{A}$, we construct the corresponding graph $G_\mathcal{A}$. Then, the initial chromosomes are built by performing BFS on the graph, starting from each node. This way, n initial individuals are constructed. Applying Hill Climbing [11] on the obtained individuals, the initial population for the *HGA* is obtained. The construction of the initial population for the *HGA* is slightly different from the method from [11].

As further work we will investigate the appropriateness of replacing the Hill Climbing step from the *HGA* with other local search mechanisms, such as *PSwap* or *MPSwap* procedures that will be described in Subsection 4.2.

4.2. Ant-based system.

A hybridized ACO approach using a local search procedure is proposed in this section for solving the *MBMP*. This local search method is designed to reduce the bandwidth of the current solution and is executed during the local search stage of the ACO framework.

In [20] *Ant Colony System (ACS)* [5] is hybridized with a local search mechanism. The *ACS* model is based on the level structure used by the Cuthill-McKee algorithm [2]. The local search procedure aims at improving *ACS* solutions, by reducing the maximal bandwidth. The integration of a local search phase within the proposed *ACS* approach to *MBMP* facilitates the refinement of ants' solutions.

The main stages of the proposed hybrid *ACS* are as follows.

i. First, the current matrix bandwidth is computed, the pheromone trails are initialized and the parameters values are established.

ii. The construction stage consists of executing the next steps within a given number of iterations. At first all the ants are placed in the node from the first level, then the local search mechanism is applied.

Each ant builds a feasible solution by repeatedly making pseudo-random choices from the available neighbors. While constructing its solution, an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule [5].

After each partial solution is built, in order to improve each ant's solution, the local search mechanism is applied. Finally, once all ants have finished their tour, the amount of pheromone on edges is modified again by applying the global updating rule [5].

iii. The best current solution is listed.

As illustrated above, the local search procedure is used twice within the proposed hybrid model: at the beginning of each iteration and after each partial solution is built, in order to improve each ant's solution.

In [20] two local search mechanisms are introduced: *PSwap* and *MPSwap*. The local search mechanisms are denoted by *hACS* and respectively *hMACS*.

PSwap firstly founds the maximum and minimum degrees. Then, for all indexes x with the maximum degree, it randomly selects an unvisited node y with a minimum degree and then swaps the nodes x and y.

In order to avoid stagnation was introduced *hMACS*. First are found the maximum and minimum degrees. For all indexes x with the maximum degree, it randomly selects an unvisited node y with a minimum degree such as the matrix bandwidth decreases and then swaps the nodes x and y.

The experimental results reported in [20] show that *MPSwap* procedure performs better on small instances, while *PSwap* is better on larger ones.

5. A theoretical reinforcement learning model for solving MBMP

In this section we investigate a reinforcement learning approach for solving the *MBMP* problem and introduce our RL model.

Let us assume, in the following, that $\mathcal{A}$ is the symmetric matrix of order n whose bandwidth has to be minimized.

5.1. Problem definition.

We define the RL problem associated to *MBMP* as in [4]:

- The environment E consists of the set of states $\{1, 2, \ldots, n\}$ extended with a state s_0 that is connected to all other states, i.e. $E = \{1, 2, \ldots, n\} \bigcup \{s_0\}$.
- The initial state si of the agent in the environment is s_0.
- A state $sf \in E$ reached by the agent at a given moment after it has visited states $si, s_1, s_2, \ldots s_k$ is a *terminal* (final) state if the number of states visited by the agent in the current sequence is $n + 1$, i.e. $k = n$.
- The transition function between the states is defined as $h : E \rightarrow P(E)$, where $h(i) = \{1, 2, \ldots n\}$. This means that, at a given moment, from the state i the agent can move to any state from E, excepting the initial state. We say that a state j that is accessible from state i ($j \in h(i)$) is the *neighbor* (*successor*) state of i.
- The transitions between the states are equiprobable, the transition probability $P(i, j)$ between a state i and each neighbor state j of i is $P(i, j) = \frac{1}{n}$.

The RL problem consists in training the *MBMP* agent to find a path $si, \pi_1, \pi_2, \ldots \pi_n$ from the initial to a final state, i.e a permutation π of $\{1, 2, \cdots, n\}$ that minimizes the corresponding matrix bandwidth [4]. Let us consider that, at a given moment, the agent has visited states $si, \pi_1, \pi_2, \ldots \pi_k$, where $k \leq n$, $\pi_i \in E$, $\pi_i \neq si$; $\forall\ 1 \leq i \leq k$ and $\pi_i \neq \pi_j$, $\forall\ 1 \leq i, j \leq k, i \neq j$. Starting from the path $\pi_1, \pi_2, \ldots \pi_k$, we construct a permutation of $\{1, 2, \ldots, n\}$, denoted by $\sigma^{\pi(1..k)} = (\sigma_1^{\pi(1..k)}, \sigma_2^{\pi(1..k)}, \ldots, \sigma_n^{\pi(1..k)})$. An element $\sigma_j^{\pi(1..k)}$ ($\forall 1 \leq j \leq n$) is computed as follows:

- If $j \leq k$, then $\sigma_j^{\pi(1..k)} = \pi_j$.
- If $k < j \leq n$ and $j \notin \{\pi_1, \pi_2, \ldots\ \pi_k\}$, then $\sigma_j^{\pi(1..k)} = j$.
- If $k < j \leq n$ and $j \in \{\pi_1, \pi_2, \ldots\ \pi_k\}$, then $\sigma_j^{\pi(1..k)} = s$, where $1 \leq s \leq n$, $s \notin \{\pi_1, \pi_2, \ldots \pi_k\}$ and $\exists m$, $1 < m < k$ and $i_1, i_2, \ldots, i_m$ ($1 \leq i_q \leq n$ $\forall\ 1 \leq q \leq m$) such that $j = \pi_{i_1}$, $i_1 = \pi_{i_2}, \ldots,$ $i_{m-1} = \pi_{i_m}$, $i_m = \pi_s$.

Based on the definition of $\sigma^{\pi(1..k)}$ given above, it can be proved that $\sigma^{\pi(1..k)}$ is a permutation of $\{1, 2, \ldots, n\}$. Now, a matrix $\mathcal{A}^{\sigma^{\pi(1..k)}}$ can be obtained from the initial matrix $\mathcal{A}$ by permuting its rows (and columns) in the order given by permutation $\sigma^{\pi(1..k)}$.

Consequently, a path $\pi_1, \pi_2, \ldots \pi_k$ of the agent in the environment corresponds to the matrix $\mathcal{A}^{\sigma^{\pi(1..k)}}$ obtained as we have described above.

5.2. Reinforcement function.

As we aim at obtaining a permutation π of $\{1, 2, \ldots, n\}$ that minimizes the matrix bandwidth, we define the reinforcement function as indicated in Equation (1). We mention that an alternative method to define the reinforcement function was considered in [4].

- the reward received in state π_k after states $si, \pi_1, \pi_2, \ldots$ π_{k-1} were visited, denoted by $r(\pi_k | si, \pi_1, \ldots\ pi_{k-1})$ is computed as the bandwith of matrix $\mathcal{A}^{\sigma^{\pi(1..k-1)}}$ minus the bandwidth of matrix $\mathcal{A}^{\sigma^{\pi(1..k)}}$.

$$r(\pi_k | si, \pi_1, \ldots \pi_{k-1}) = \begin{cases} 0 & \text{if } k = 1 \\ \beta(\mathcal{A}^{\sigma^{\pi(1\ldots k-1)}}) - \beta(\mathcal{A}^{\sigma^{\pi(1\ldots k)}}) & \text{otherwise} \end{cases} \quad (1)$$

Considering the reward defined in Equation (1), as the learning goal is to maximize the total amount of rewards received on a path from the initial to a final state, it can be easily proved that the agent is trained to find a permutation π of $= \{1, 2, \ldots, n\}$ that minimizes the bandwidth of the corresponding matrix $\mathcal{A}^{\sigma^{\pi(1..k)}}$.

5.3. The learning process.

During the training step of the learning process, the agent will determine its *optimal policy* in the environment, i.e the *policy* that maximizes the sum of the received rewards. During the training process, the states' utilities estimations converge to their exact values, thus, at the end of the training process, the estimations will be in the vicinity of the exact values.

It is proved that the RL algorithm (such as SARSA [22]) converges with probability 1 to an utility function as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the Greedy policy.

Consequently, after the training step of the agent has been completed, the solution learned by the agent will be constructed by starting from the initial state and following the *Greedy* policy until a solution is reached. From given state i, using the *Greedy* policy, the agent moves to an unvisited neighbor j of i having the maximum utility value. The solution of the *MBMP* reported by the RL agent is a permutation π of $\{1, 2, \ldots, n\}$ such that $U(\pi_1) \geq U(\pi_2) \geq \cdots \geq U(\pi_n)$, U being the utility function learned by the agent during its training. Considering the general goal of a RL agent, it can be proved that the permutation π of $\{1, 2, \ldots, n\}$ learned by the $MBMP$ agent converges to the permutation that corresponds to the matrix with the minimum bandwidth.

6. Computational experiments

In this section follows the comparative evaluation of the techniques proposed in Section 4 in order to solve the *MBMP*. The results are compared with those reported by *CM* algorithm [2].

Nine benchmark instances from *National Institute of Standards and Technology, Matrix Market, Harwell-Boeing sparse matrix collection* [17] were used in the computational experiments. In Table 1 are illustrated, for each considered instance, the following characteristics: number of lines, number of columns and number of nonzero entries.

Table 1: The benchmark instances.

No.	Instance	Euclidean Characteristics
1	can_24	24 24 92
2	can_61	61 61 309
3	can_62	62 62 140
4	can_73	73 73 225
5	can_96	96 96 432
6	can_187	187 187 839
7	can_229	229 229 1003
8	can_256	256 256 1586
9	can_268	268 268 1675

The hybrid genetic algorithm *HGA* (Subsection 4.1) and the hybrid ant systems *hACS* and *hMACS* (Subsection 4.2) were implemented and applied for the instances described in Table 1. Some details regarding the implementations of *HGA*, *hACS* and *hMACS* are following.

The *Hybrid GA* is based on a *Delphi* implementation [23] and is tested with 10% mutation rate, $k = [n/10]$ and 50, respectively 100 generations. $GA1$ is denoted the hybrid genetic algorithm with 50 generations and $GA2$ the hybrid genetic algorithms with 100 generations.

The hybrid ant algorithms [20] *hACS* and *hMACS* are implemented in Java. For each instance, both algorithms are executed 20 times.

The parameter values for both implementations are: 10 ants, 10 iterations, $q_0 = 0.95$, $\beta = 2$, $\rho = 0.001$, $\tau_0 = 0.1$. The algorithms were compiled on an AMD 2600 computer with 1024 MB memory and 1.9 GHz CPU clock.

In Table 2 are comparatively illustrated the best solution (the bandwidth of the matrix) obtained by *CM*, *hACS*, *hMACS*, *GA1* and *GA2* algorithms for the instances given in Table 1.

Table 2: Comparative results.

Instance no.	CM	hACS	hMACS	GA1	GA2
1	8	14	11	**6**	**6**
2	26	43	42	**19**	**19**
3	9	20	12	**8**	**8**
4	27	28	**22**	**22**	23
5	23	**17**	**17**	25	25
6	**23**	63	33	53	51
7	**49**	120	120	63	63
8	116	148	189	**91**	**91**
9	134	165	210	**90**	**90**

A graphical representation of the results is given in Figure 1. Based on Figure 1 some conclusions follows.

Excepting two instances (6 and 7) the hybrid natural-based algorithms provide better result than *CM* algorithm. *hMACS* algorithm performs better than *hACS* algorithm on small instances, while *hACS* algorithm is better than *hMACS* on larger ones. For six instances the hybrid genetic algorithm performed better than ant-based algorithms. The number of generations considered for *GA1* and *GA2* has no significant influence on the results.

In order to assure a better convergence to the solution, the ant-based hybrid models should offer an "ideal" set of parameters and also a good strategy of placing the agents in the environment.

The *Matrix Bandwidth Minimization Problem*'s results could be improved using reinforcement learning in new hybrid natural based-computing techniques.

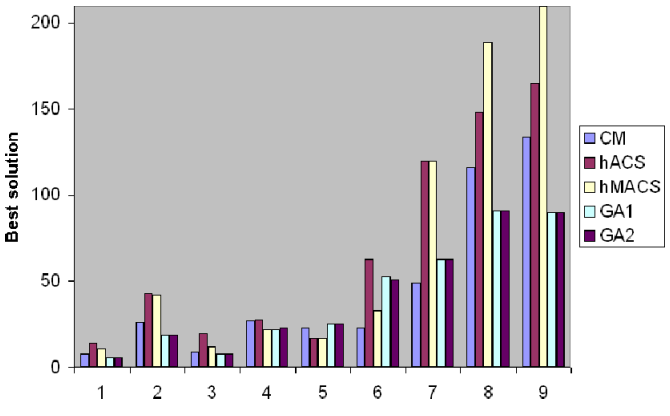

Figure 1: Comparative results.

7. Conclusions and further work

The *Matrix Bandwidth Minimization Problem (MBMP)* is a classic mathematical problem, relevant to a wide range of complex real life applications. The problem is NP-complete and a lot of research was conducted in order to find appropriate solutions.

Nowadays, bio-inspired heuristics are successfully used to solve difficult problems. On this basis, the paper describes several soft computing approaches for solving the *MBMP*. The proposed heuristics are hybrid algorithms: genetic algorithms and ant colony algorithms. Some standard *MBMP* instances are tested using the hybrid bio-inspired algorithms and compared with existing literature. The results are encouraging.

A new theoretical *reinforcement learning* model for solving the considered problem is also introduced. Computational experiments confirmed a good performance of the proposed algorithms, emphasizing the effectiveness of *soft computing* methods in order to solve the *MBMP*.

Further work will be made in order to detail the proposed reinforcement learning model. More exactly, we proposed to develop a *RL* algorithm for training the *MBMP* agent and to experimentally validate the *RL* model. We will also investigate new local search procedures in order to improve the performance of the ant system and of the genetic algorithm proposed for solving the *Matrix Bandwidth Minimization Problem*.

References

[1] M. Berry, B. Hendrickson and P. Raghavan. *Sparse matrix reordering schemes for browsing hypertext.* Lectures in Appl. Math. 32: The Mathematics of Numberical Analysis, 99–123, 1996.

[2] E. Cuthill, E., and McKee, J. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th National Conference ACM*, 157–172, 1969.

[3] M. Cygan and M. Pilipczuk. *Even faster exact bandwidth.* ACM Trans. Algorithms (in press). Also Technical Report abs/0902.1661, arXiv, CoRR, 2009.

[4] Czibula G., Czibula I-G., and Pintea, C-M. A Reinforcement learning approach for solving the matrix bandwidth minimization problem. *Studia Informatica*, LV(4):9–17, 2010.

[5] M. Dorigo and T. Stüzle. *Ant Colony Optimization*, 2004. MIT Press, Cambridge.

[6] A. Esposito, M.S. Catalano, F. Malucell and L. Tarricone. Sparse Matrix Bandwidth Reduction: Algorithms, applications and real industrial cases in electromagnetics, High Performance Algorithms for Structured Matrix Problems. *Advances in the theory of Computation and Computational Mathematics*, 2:27–45, 1998.

[7] U. Feige Approximating the bandwidth via volume respecting embeddings. *Journal of Computer and System Sciences.* 60(3):510–539, 2000.

[8] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for the bandwidth minimization. *SIAM Journal of Applied mathematics.* 34(3):477–495, 1978.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1989, 1st. Addison-Wesley Longman Publishing Co., Inc.

[10] A. Gupta. *Improved bandwidth approximation for trees.* In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 788–793, 2000.

[11] A. Lim, B. Rodrigues, and F. Xiao. Integrated genetic algorithm with hill climbing for bandwidth minimization problem. *Proceedings of the 2003 international Conference on Genetic and Evolutionary Computation, Lecture Notes In Computer Science.* Springer-Verlag, Berlin, Heidelberg, 1594–1595, 2003.

[12] A. Lim, B. Rodrigues and F. A. Xiao. Centroid-based approach to solve the Bandwidth Minimization Problem. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, Hawaii, USA, 2004.

[13] A. Lim, J. Lin, B. Rodrigues and F. Xiao. Ant Colony Optimization with hill climbing for the bandwidth minimization problem. *Appl. Soft Comput.* 6(2):180–188, 2006.

[14] L. Lin and Y. Lin. Two models of two-dimensional bandwidth problems. *Information Processing Letters.* 110(11):469–473, 2010.

[15] R. Marti, M. Laguna, F. Glover and V. Campos. Reducing the Bandwidth of a Sparse Matrix with Tabu Search. *European Journal of Operational Research.* 135(2):211–220, 2001.

[16] M. Mitchell. *An Introduction to Genetic Algorithms.* 1998, MIT Press.

[17] National Institute of Standards and Technology, Matrix Market, Harwell-Boeing sparse matrix collection

[18] C. H. Papadimitriou. The NP-Completeness of the bandwidth minimization problem. *Computing.* 16(3): 263–270, 1976.

[19] E. Pinana, I. Plana, V. Campos and R. Marti. GRASP and Path Relinking for the Matrix Bandwidth Minimization. *European Journal of Operational Research*, 153:200–210, 2004.

[20] C-M. Pintea, G-C. Crisan and C. Chira. A Hybrid ACO Approach to the Matrix Bandwidth Minimization Problem. HAIS 2010, LNCS 6076, 405–412, 2010.

[21] E. Rodriguez-Tello, H. Jin-Kao, and J. Torres-Jimenez. An improved Simulated Annealing Algorithm for the matrix bandwidth minimization. *European J. of Oper. Res.* 185(3):1319–1335, 2008.

[22] R. Sutton and A. Barto. *Reinforcement Learning.* 1998, The MIT Press, Cambridge, London.

[23] C. Zavoianu. *Tutorial: Bandwidth reduction - The CutHill-McKee Algorithm*, 2009.