SCIENTIFIC PUBLICATIONS OF THE STATE UNIVERSITY OF NOVI PAZAR
SER. A: APPL. MATH. INFORM. AND MECH. vol. 6, 2 (2014), 107-117.

A Genetic Algorithm for the Minimum Hitting Set

Bojana Lazović Ćendic

Abstract: In this paper the Minimum Hitting Set (MHS) problem is considered. The problem is solved by a genetic algorithms (GA) that uses binary encoding and standard genetics operators adapted to the problem. In proposed implementation all individuals are feasibility by default, so search is directed into the promising regions. The overall performance of the GA implementation is improved by caching technique. Local search optimization is used to refine the solutions explored by GAs. The algorithm is tested on standard instances from the literature. The obtained results are also compared with the results of existing methods for solving MHS in order to assess their merits.

Keywords: Hitting set, Evolutionary approach, Genetic algorithms.

1 Introduction

This section gives a description of the minimum hitting set (MHS) problem, and the concepts used throughout this paper.

A finite set P and a collection C of n non-empty subsets of P ($C = \{P_1,..,P_n\}$ such that $P_i \subseteq P$ for $i = 1,..,n$) are given. Each subset $P_i \in C$ is a finite set of elements (components from now on), where each of the m elements is represented by a number x_j , $j \in \{1,..,m\}$. The minimum hitting set of C is a set $H \subseteq P$ such that :

$\forall P_i \in C, P_i \cap H \neq \emptyset \wedge \neg\exists H' \subset H : P_i \cap H' \neq \emptyset$ for each $i \in 1,..,n$.

i.e. there is at least a component of H that is a member of all subsets in C, and no proper subset of H is a hitting set. Accordingly, the set H intersects (”hits”) every subset in collection C. In other words, every subset in C must contain at least one element of H.

There may be several minimal hitting sets for C, which constitutes a collection of minimal hitting sets $\Omega = \{H_1,..,H_k,..,H_{|\Omega|}\}$. The computation of this collection is known to be a NP-hard problem (see [14]). The size of the hitting set is given by the number of elements in H, i.e. $|H|$. Therefore in the hitting set problem, the task is to find a hitting set of a minimal size.

Manuscript received March 11, 2014; accepted July 23, 2014.

Bojana Lazović Ćendic is with the Faculty of Mathematics, University of Belgrade, Serbia

Let demonstrate properties of MHS on two little illustrative examples.

Example 1. Let the first set consist of ten elements ($m = 10$) and four subsets ($n = 4$). The subsets are: $P_1 = \{1,2,4,7,9\}$; $P_2 = \{3,4,5,7,9,10\}$; $P_3 = \{2,8,9,10\}$; $P_4 = \{2,3,4,6,7,8,9,10\}$. One of the optimal solutions is set $H = \{9\}$. The optimal objective value is 1 because $P_1 \bigcap P_2 \bigcap P_3 \bigcap P_4 = \{9\}$, thus only one element is enough to represent ("hit") all subsets in the collection.

Example 2. Let the second set consist of three elements ($m = 3$) and three subsets ($n = 3$). The subsets are: $P_1 = \{1,2\}$; $P_2 = \{2,3\}$; $P_3 = \{1,3\}$. One of the optimal solutions is set $H = \{1,2\}$. The optimal objective value is 2 because $P_1 \bigcap P_2 = \{2\}$; $P_1 \bigcap P_3 = \{1\}$; $P_2 \bigcap P_3 = \{3\}$, so there are two elements which may represent ("hit") all subsets in the collection.

MHS problem is one of the most notorious NP-hard problems and it is known to be equivalent to the classical Minimum Set Cover (MSC) : positive and negative approximability results for the MHS can be directly derived from the classical MSC problem [2,4,16]. Recall that in the MSC, given a universe set U, and nonnegative costs for every element of U, a collection T of subsets of U, and it is looking for a subcollection $T' \subseteq T$, such that the union of the sets in T' is equal to U, and T' is of minimal cost.

There have been numerous optimal and heuristic solution approaches for the MHS problem presented in the literature. Identifying minimal hitting sets of a collection of sets is an important problem in many domains, such as in model-based diagnosis (MBD) where the MHS are the solutions for the diagnostic problem. MHS is also one of key problems in the combinatorics of finite sets (see [3]). Known to be a NP-hard problem [14], one (1) desires focusing heuristics to increase the search efficiency and/or (2) limits the size of the return set. Such strategies have the potential to reduce the MHS problem to a polynomial time complexity at the cost of completeness. In [1] a low-cost, approximate MHS algorithm, coined Staccato, uses a heuristic function, borrowed from a lightweight, statistics-based software fault localization approach, to guide the MHS search.

Several others exhaustive algorithms have been presented to solve the MHS problem. Since Reiter [30] showed that diagnoses are MHSs of conflict sets, many approaches to solve this problem in the model-based diagnosis context have been presented. In [15], [7], [30] and [35] the hitting set problem is solved using so-called hit-set tree and all of them need exponential size memory to be implemented. In [9] and [10] the MHS problem is mapped onto an 0/1 integer programming problem, and there is showed that it is possible to find a MHS with minimal cardinality with an algorithm that requires a linear size memory (while it still may needs an exponential time to complete the computation). In [9] is given a first integer linear programming (ILP) formulation of MHS and it is used in this paper. In [36] a method using set-enumeration trees to derive all minimal conflict sets in the context of model-based diagnosis is presented. The authors conclude that this method has an exponential time complexity in the number of elements in the sets. The Quine-McCluskey algorithm [29], [25] originating from logic optimization, is a method for deriving the prime implicants of a monotone boolean function (a dual problem of the MHS problem). This algorithm is, however, of limited use due to its exponential complexity. Algorithms for implicit hitting set problems is given in [5], and in [31], there is a branch-and-reduce algorithm for solving the Minimum Hitting Set Problem.

Many heuristic approaches have been proposed to render MHS computation amenable to large systems. In [24] an approximate method to compute MHSs using genetic algorithms is described. The fitness function used aims at finding solutions of minimal cardinality. Their paper does not present a time complexity analysis, but it is suspected that the cost/completeness trade-off is worse than for GA presenting in this paper. Stochastic algorithms, as discussed in the framework of constraint satisfaction [13] and propositional satisfiability [28], are examples of domain independent approaches to compute MHS. Stochastic algorithms are more efficient than exhaustive methods.

In this paper an evolutionary metaheuristic for solving the minimum hitting set problem is presented. The problem is solved by a genetic algorithm that uses binary encoding and standard genetics operators adapted to the problem.

The paper is organized as follows. In Section 2 Mathematical model is presented. Section 3 includes the main features of a genetic algorithm (GA) implementation designed for the MHS problem, while Section 4 presents computational results on standard instances from the literature.

2 Mathematical model

It is useful to formulate, when it is possible, discrete optimization problems as integer linear programming models, in order to use different well-known optimization techniques for their solving [22, 23, 34]. Following that idea a GAs is used on the mixed integer linear programming formulation for the MHS described below.

Let define parameters and variables as follows:

$$p_{ij} = \begin{cases} 1\,, & j \in P_i \\ 0\,, & j \notin P_i \end{cases} \tag{1}$$

$$x_j = \begin{cases} 1\,, & j\ is\ selected \\ 0\,, & j\ \ is\ not\ selected \end{cases} \tag{2}$$

Parameter (1) indicates whether the j-th element belongs to the P_i-th subset or not. If it belongs to a subset, its value is 1 and otherwise the value is 0.

Variable (2) is a binary variable and it takes the value 1 if the j-th element is selected i.e. belongs to the minimal hitting set , otherwise it gets value 0.

Now, a integer linear programming model can be formulated as follows:

$$\min \sum_{j=1}^{m} x_j \tag{3}$$

subject to :

$$\sum_{j=1}^{m} p_{ij} x_j \geq 1 \quad , \quad for\, every\;\; i = 1,...,n \tag{4}$$

$$x_j \in \{0,1\}, \quad for\, every\;\; j = 1,...,m \tag{5}$$

The objective function (3) minimizes the sum of elements which represent each of the subset P_i in collection C. Since at least one member should belong to every P_i, for every i = 1,.., n, formula (4) is given. Binary nature of variables x_j is specified by constraint (5).

By the definition of the minimum hitting set, the above equation (4)-(5) should be simultaneously satisfied for all i = 1,..,n, then the formulation of the problem is given as an integer linear programming problem. With this setting, identification of the minimal hitting set is then equivalent to solution from equation (3).

3 Proposed GA method

As the minimum hitting set problem represents an NP-hard combinatorial optimization problem, where the size of its solution space can grew exponentially with the problem dimension, the usual way to solve such kind of problem is to apply to a heuristic approach. Although heuristic methods do not offer a guarantee of reaching the optimum, they give satisfactory results for a large range of various problems in a reasonable amount of time.

Genetic algorithms (GAs) have been seen as search procedures that can quickly locate high performance regions of vast and complex search spaces, but they are not well suited for fine-tuning solutions, which are very close to optimal ones. However, genetic algorithms may be specifically designed to provide an effective local search as well. Therefore in this work a local search is used to refine the solutions explored by GA.

GAs are stochastic search techniques which imitate some spontaneous optimization processes in the natural selection and reproduction. At each iteration (generation) GA manipulates a set (population) of encoded solutions (individuals), starting from either randomly or heuristically generated one. Individuals from the current population are evaluated using a fitness function to determine their qualities. Good individuals are selected to produce the new ones (offspring), applying operators inspired from those of genetics (crossover and mutation), and they replace some of the individuals from the current population. Detailed description of GA is out of this paper's scope and it can be found in [26]. Extensive computational experience on various optimization problems shows that GA often produces high quality solutions in a reasonable time, as can be seen from the following recent applications [8, 19, 20, 21, 27, 32].

This section gives a description of a GA implementation for determining the minimum hitting set.

The algorithm uses a binary encoding of the individuals, where each solution H (i.e. a candidate for a minimum hitting set) is naturally represented in the population by a binary string of length m. Digit 1 at the j-th place of the string denotes that element j belongs to H, while 0 shows the opposite. Determined on that way, it is easy to check for every subset if there is an element that represents it i.e. belongs to a minimum hitting set H. The main goal is that at least one element is taken from each subset and that the total number of those elements (i.e. the cardinality of hitting set H) is as little as possible. Objective value is the number of representatives i.e. the cardinality of the obtained minimum hitting set.

Example 3. Let genetic code be 10001001. This means that $x_1 = x_5 = x_8 = 1, x_2 = x_3 = x_4 = x_6 = x_7 = 0$, implying that the elements $\{x_1, x_5, x_8\}$ are potential representatives of the given subset and candidates for the MHS.

If some of the subsets do not have any element that will represent it in the MHS, then through the local search (LS) algorithm, on the accidental way, an element is inserted in that subset. LS algorithm is also used for deleting an element from the subsets which have at least two representatives. That search is done only inside the best subsets or inside the subsets that have the largest number of representatives, and only there the ejection occurs. If it goes through all the subsets, it would slow down the performance. Therefore LS purposely makes a change to one element, in order to reach better results. Once the replacement is made, LS continues searching for the next element that change would improve the result.

Population in each generation contains $N_{pop} = 150$ individuals. The fitness f_{ind} of individual $ind = 1, 2, ..., N_{pop}$ is computed by scaling objective values obj_{ind} of all individuals into the interval [0,1]. Therefore, the best individual ind_{min} has fitness 1 and the worst one ind_{max} has fitness 0. More precisely, $f_{ind} = (obj_{ind_{max}} - obj_{ind})/(obj_{ind_{max}} - obj_{ind_{min}})$. Next, individuals are arranged in non-increasing order of their fitness: $f_1 \geq f_2 \geq ... \geq f_{N_{pop}}$.

The population of the first generation is randomly generated, providing the maximal diversity of the genetic material. In order to prevent an undeserved domination of some individuals in the current population an elitist strategy is used. The fitness of N_{elite} = 100 elite individuals over the population are decreased by the next formula:

$$f_{ind} = \begin{cases} f_{ind} - \bar{f}, & f_{ind} > \bar{f} \\ 0, & f_{ind} \leq \bar{f} \end{cases} ; \quad 1 \leq ind \leq N_{elite}; \ \bar{f} = \frac{1}{N_{pop}} \sum_{ind=1}^{N_{pop}} f_{ind} \tag{6}$$

In this way, even non-elite individuals preserve their chance to survive to the next generation. This approach gives a possibility to allow high elitism without too high selection pressure and thus too much exploitation in the algorithm.

The first N_{elite} individuals are directly passing to the next generation. Genetic operators are applied to the rest of the population ($N_{nnel} = N_{pop} - N_{elite}$ non-elite individuals), so that only one-third is replaced in every generation. The objective value of elite individuals are the same as in the previous generation, that is why they are calculated only once and this provides significant time savings.

Duplicated individuals, i.e. individuals with the same genetic code are redundant and they are removed from the current population by setting their fitness to zero, except for the first occurrence. This is a very effective method of saving diversity of the genetic material and keeping the algorithm away from a premature convergence. Individuals with the same objective value and different genetic codes, in some cases may dominate in the population and lead the algorithm to a local optimum. For that reason, number of such individuals in one generation is limited to some constant $N_{rv} = 40$. Therefore every individual in the population passes through two steps. In the first step the genetic code of the current individual *ind* is checked if it is identical with the genetic code of any other individuals from 1 to $ind - 1$. If the answer is positive, the fitness of *ind* is set to 0. Otherwise the step two occurs. In the second step the number of individuals from 1 to $ind - 1$, which did not get fitness 0 in the first step and which have the same objective value as *ind*, is counted. If its number is greater than or equal to N_{rv}, the fitness of *ind* is set to 0.

The selection operator, which chooses the parent individuals that will produce offspring in the next generation, is an improved tournament selection operator known as the fine-grained tournament selection-FGTS (see [11,12]). This operator uses a real (rational) parameter F_{tour} which denotes the desired average tournament size. Two types of tournaments are performed: the first type is held k_1 times on $\lfloor F_{tour} \rfloor$ individuals, while the second type is applied k_2 times with $\lceil F_{tour} \rceil$ individuals participated, so $F_{tour} \approx \frac{k_1 \cdot \lfloor F_{tour} \rfloor + k_2 \cdot \lceil F_{tour} \rceil}{N_{nnel}}$.

In this implementation F_{tour} = 5.4 and, therefore, corresponding values k_1 and k_2 for N_{nnel}=50 non-elitist individuals are 20 and 30, respectively.

In crossover operator all non-elitist individuals chosen to produce offspring for the next generation are randomly paired for crossover in $\lfloor N_{nnel}/2 \rfloor$ pairs. When a pair of parents is selected, an one-point crossover operator is applied to them producing two offspring. This operator is performed by exchanging segments of two parents' genetic codes starting with a randomly chosen crossover point. The crossover operator is realized with probability p_{cross} = 0.85, which means that approximately 85% pairs of individuals exchange their genetic material.

The simple mutation operator is performed by changing a randomly chosen gene in the code of the individual, with a certain mutation rate (probability) p_{mut}. In order to prevent a premature convergence the mutation rate is increased on each so-called frozen gene, i.e. a gene on a certain position with the same value which appears in all individuals in the current population. In our implementation the mutation rate depends on dimension n and it is 2.5 times higher on frozen genes (p_{mut} = 1.0/n) then on non-frozen ones (p_{mut} = 0.4/n).

In each generation, we determine positions where all individuals have a given gene fixed and define them as frozen genes. Obviously the set of frozen genes is not fixed, i.e. it may change during the generations.

The run-time performance of GA is optimized by a caching technique, which has shown a very good performance in practice on quite different problems (see for example [8,18,33]). The main idea is to avoid computing the same objective value every time when genetic operators produce individuals with the same genetic code. Evaluated objective values are stored in a hash-queue data structure using the least recently used (LRU) caching technique. When the same code is obtained again, its objective value is taken from the cache memory, that provides time-savings. In this implementation the number of individuals stored in the cache memory is limited to 5000. For detailed information about caching GA see [17].

4 Experimental results

This section presents the results of the GA approach outlined in Section 3 for computing minimum hitting sets on hitting set instances taken from [6]. Used hitting set instances include different numbers of elements (m = 50, 100, 250, 500) and different numbers of subsets (n = 1000, 10000, 50000). Obtained solutions are compared with already best known solutions. All computations were executed on 2.5 GHz single processor PC computer with 1 Gb RAM under Windows operating system. Genetic algorithm was coded in C programming language.

In the Table 1 the characteristics of MHS instances are presented. In the first and second column there are the number of elements *m* and the number of subsets *n*, one after another. The best known solutions for these instances are placed in the third column. In one case, for instance of *m*=50 and *n*=1000, GA reaches the new solution (number 6) that is better than the previous best known solution obtained by other methods.

In the Table 2 the results obtained by GA are compared to the best results ever known. The first and the second column are the same as in the Table 1. The third column is reserved for the best known solution. The results on GA are presented in the fourth, fifth and the sixth columns. The best value of GA is given in the fourth column. The mark "bk" is written if GA finishes its work and produces the best known solution. The running time of GA is given in the fifth column and the last column shows the number of generations through which GA passes to obtain the appropriate solution.

Comparisons with the best known results from literature [4] show the appropriateness of applying proposed algorithm components. In order to enhance and assess the reliability of the GA performance, each test instance mentioned in Tables 2 (except the largest) is replicated 20 times. The large-scale instances were run 10 times because of relatively time consuming objective function computation for these instances. The stopping criterion of GA was the maximum number of generations equal to 5000. The algorithm also stops if the

Table 1: Characteristics of MHS instances

m	*n*	*Best known solution*	
50	1000	6	*new*
50	10000	10	
100	1000	6	
100	10000	9	
100	50000	39	
250	1000	10	
250	10000	12	
500	1000	10	
500	10000	16	
500	50000	21	

Table 2: GA results

m	*n*	*Best known*	*GA sol*	*t* (*sec*)	*num gener*
50	1000	6	b.k.	3.239	2304
50	10000	10	b.k.	77.146	2608
100	1000	6	b.k.	5.778	2767
100	10000	9	b.k.	210.767	2830
100	50000	39	40	694.656	3619
250	1000	10	11	10.571	3679
250	10000	12	13	426.919	3492
500	1000	10	11	17.31	3832
500	10000	16	17	550.356	3914
500	50000	21	b.k.	2657.596	3796

best individual or the best value remains unchanged through 2000 successive generations.

New best solution is found for instance of m=50 and n=1000 in a very small running time (3.239 seconds). For instances up to m =100, n =10000 and for the largest instance (m =500,n =50000) GA obtains solutions that match already known best results from literature in a short period of time. Other solutions are very close to the best known solutions (the difference is 1) and they are also reached in a very small running time.

5 Conclusions

In this paper an evolutionary metaheuristic for solving the minimum hitting set problem is presented. The problem is solved by a genetic algorithm that includes the binary representation of individuals, a fine-grained tournament selection, one-point crossover, the mutation with frozen genes, a limited number of different individuals with the same objective value and the caching GA technique. Solution quality is improved by using the local search heuristic that is efficiently implemented in GA. It is shown that GAs may be specifically designed with the aim of performing an effective local search.

As can be seen from experimental results, this approach seems to be good candidate for solving MHS. The proposed GA reaches the new best solution ($|H|$=6) for instance of m=50 and n=1000, in a very small running time (3.239 seconds). For the other instances, in half of them GA obtains the best known solutions, including the largest one, and for the rest, solutions are very close to the best known. Accordingly to the size of instances, all solutions are achieved in a very small running time.

This research can be extended in several ways. It would be desirable to investigate the combination of GA approach with exact methods using the integer linear programming formulation. Also, it should be directed to parallelization of presented genetic algorithm and testing on more powerful multiprocessor computer systems.

References

[1] R. Abreu, A. J. C. van Gemund, *A Low-Cost Approximate Minimal Hitting Set Algorithm and its Application to Model-Based Diagnosis*, In: Proceedings of the 8th Symposium on Abstraction, Reformulation and Approximation (SARA09), Lake Arrowhead, CA, USA, 2009.

[2] G. Ausiello, A. DAtri, M. Protasi, *Structure preserving reductions among convex optimization problems*, Journal of Computer and System Sciences, Vol. 21, 1 (1980), 136-153.

[3] C. Berge, *Hypergraphs: Combinatorics of Finite Sets*, Elsevier-North Holland, Amsterdam, The Netherlands, 1989.

[4] T. M. Chan, E. Grant, *Exact algorithms and APX-hardness results for geometric packing and covering problems*, Computational Geometry, Vol 47, 2 (2014), 112124.

[5] K. Chandrasekaran, R. Karp, E. Moreno-Centeno, S. Vempala, *Algorithms for implicit hitting set problems*, In: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms, 2011, pp. 614-629.

[6] V. Cutello, G. Nicosia, *A clonal selection algorithm for coloring, hitting set and satisfiability problems*, Lecture Notes in Computer Science, Vol. 3931 (2006), 324-337.

[7] J. de Kleer, *Diagnosing intermittent faults*, In: Proceedings of 18th International Worskhop on Principles of Diagnosis (DX-07), 2007.

[8] B. Djurić, J. Kratica, D. Tošić, V. Filipović, *Solving the maximally balanced connected partition problem in graphs by using genetic algorithm*, Computing and Informatics, Vol. 27, 3 (2008), 341-354.

[9] A. Fijany, F. Vatan, *New approaches for efficient solution of hitting set problem*, In: Proceedings of the winter international synposium on Information and communication technologies (WISICT04), Cancun, Mexico: Trinity College Dublin, 2004.

[10] A. Fijany, F. Vatan, *New high performance algorithmic solution for diagnosis problem*, In: Proceedings of the IEEE Aerospace Conference (IEEEAC05), 2005.

[11] V. Filipović, J. Kratica, D. Tošić, I. Ljubić, *Fine grained tournament selection for the simple plant location problem*, In: Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications - WSC5, 2000, pp. 152-158.

[12] V. Filipović, *Fine-grained Tournament Selection Operator in Genetic Algorithms*, Computing and Informatics, Vol.22, (2003), 143-161.

[13] E. C. Freuder, R. Dechter, M. L. Ginsberg, B. Selman, E. P. K. Tsang, *Systematic versus stochastic constraint satisfaction*, In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI95), 1995, pp. 2027-2032.

[14] M. Garey, D. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*,Freeman, San Francisco, 1979.

[15] R. Greiner, B. A. Smith, R. W. Wilkerson, *A correction to the algorithm in Reiters theory of diagnosis*, Artificial Intelligence, Vol.41, 1, (1989), 79-88.

[16] M. Hasan, S. M. S. Hossain, Md. M. Rahman, M. S. Rahman, *Solving Minimum Hitting Set Problem and Generalized Exact Cover Problem with Light Based Devices*, International Journal of Unconventional Computing, Vol 7, 1-2 (2011), 125-140.

[17] J. Kratica, *Improving Performances of the Genetic Algorithm by Caching*, Computers and Artificial Intelligence, Vol.18, (1999), 271-283.

[18] J. Kratica, Z. Stanimirović, D. Tošić, V. Filipović, *Two genetic algorithms for solving the uncapacitated single allocation p-hub median problem*, European Journal of Operational Research, Vol.182, 1, (2007), 15-28.

[19] J. Kratica, V. Kovačević-Vujčić, M. Čangalović, *Computing strong metric dimension of some special classes of graphs by genetic algorithms*, Yugoslav Journal of Operations Research, Vol.18, (2008), 143-151.

[20] J.Kratica, M. Čangalović, V. Kovačević-Vujčić, *Computing minimal doubly resolving sets of graphs*, Computers and Operations Research, Vol.36, 7, (2009), 2149-2159.

[21] J. Kratica, V. Kovačević-Vujčić, M. Čangalović, *Computing the metric dimension of graphs by genetic algorithms*, Computational Optimization and Applications, Vo. 44, 2, (2009), 343-361.

[22] R. H. Kwon, G. V. Dalakouras, C. Wang, *On a posterior evaluation of a simple greedy method for set packing*, Optimization Letters, (2008), 587-597.

[23] L. Liberti, N. Maculan, Y. Zhang, *Optimal configuration of gamma ray machine radiosurgery units: the sphere covering subproblem*, Optimization Letters, (2009), 109-121.

[24] L. LIN, Y. JIANG, *Computing minimal hitting sets with genetic algorithms*, In: Proceedings of the Internation Workshop on Principles of Diagnosis (DX02), Semmering, 2002.

[25] E. J. MCCLUSKEY, *Minimization of boolean functions*, The Bell System Technical Journal, Vol. 35, 5, (1956), 1417-1444.

[26] M. MITCHELL, *Introduction to genetic algorithms*, MIT Press, Cambridge, Massachusetts, 1999.

[27] A. PASZYNSKA, M. PASZYNSKI, *Application of a hierarchical chromosome based genetic algorithm to the problem of finding optimal initial meshes for the self-adaptive hp-FEM*, Computing and Informatics, Vol. 28, 2, (2009), 209-223.

[28] M. QASEM, A.PRUGEL-BENNETT, *Complexity of max-sat using stochastic algorithms*, In Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO08), 2008, pp. 615-616.

[29] W. QUINE, *A way to simplify truth functions*, American Math Monthly, Vol.62, (1955), 627-631.

[30] R. REITER, *A theory of diagnosis from first principles*, Artificial Intelligence, Vol. 32, 1, (1987), 57-95.

[31] L. SHI, X. CAI, *An Exact Fast Algorithm for Minimum Hitting Set* , Computational Science and Optimization (CSO), Vol.1, (2010), 64-67.

[32] Z. STANIMIROVIĆ, J. KRATICA, DJ. DUGOŠIJA, *Genetic algorithms for solving the discrete ordered median problem*, European Journal of Operational Research, Vol.182, 3, (2007), 983-1001.

[33] Z. STANIMIROVIĆ, A *Genetic Algorithm Approach for the Capacitated Single Allocation P-Hub Median Problem*, Computing and Informatics, Vol. 29, 1 , (2010), 117-132.

[34] C. WANG, M. T. THAI, Y. LI, F. WANG, W. WU, *Optimization scheme for sensor coverage scheduling with bandwidth constraints*, Optimization Letters, (2009), 63-75.

[35] F. WOTAWA, *A variant of Reiters hitting-set algorithm*, Information Processing Letters , Vol. 79, (2001), 45-51.

[36] X. ZHAO, D. OUYANG, *Improved algorithms for deriving all minimal conflict sets in model-based diagnosis*, In: Proceedings of the 3rd International Conference on Intelligent Computing (ICIC07), Lecture Notes in Computer Science, Vol. 4681, (2007), 157-166.