Genetic Algorithm and Variable neighbourhood search for BDD Variable Ordering Problem

ABDESSELEM LAYEB [1]
NACER TABIB [2]
BILEL BRIREM [3]

University of Constantine
Department of Computer Science
MISC Lab, Algeria
[1] layeb.univ@gmail.com
[2] tabib@misc-umc.org
[3] Brirem.b.univ@gmail.fr

Abstract. The Binary Decision Diagram (BDD) is used to represent in symbolic manner a set of states. It is largely used in the field of formal checking. The variable ordering is a very important step in the BDD optimization process. A good order of variables will reduce considerably the size of a BDD. Unfortunately, the search for the best variables ordering has been showed NP-difficult. In this article, we propose a new iterative approach based on a hybrid Genetic Algorithm and Variable Neighborhood Search Algorithm. The obtained results are very encouraging and show the feasibility and effectiveness of the proposed hybrid approach.

Keywords: Binary decision diagram, Genetic algorithm,Variable neighborhood search,Hybrid methods

(Received December 11, 2010 / Accepted May 02, 2011)

1 Introduction

The objective of the checking application and electric circuits is to detect the errors which they contain or to show that they function well. One of the methods used in system checking is the model-checking [17]

One of the difficulties encountered in the domain of formal verification is the combinatorial explosion problem. For example in the model checking, the number of states in the transition graphs can reach prohibitive level, which makes their manipulation difficult or impossible. Consequently, compression methods are used in order to reduce the size of the state graph. The compression is done by using data structures in order to represent in a concise manner the set of states. In this case, the operations are done so on set of states rather than on explicit states.

The representation by the Binary Decision Diagrams BDD [6] is among the most known symbolic notations. The BDD is a data structure used to represent Boolean functions. The BDD is largely used in several fields since they offer a canonical representation and an easy manipulation. However, the BDD size depends on the selected variable order. Therefore it is important to find variable order which minimizes the number of nodes in a BDD. Unfortunately, this task is not easy considering the fact that there is an exponential number of possible variable ordering. Indeed, the problem of variable ordering was shown NP-hard [2]. For that, several methods were proposed to find the best BDD variable order and which can be classified in two categories. The first class tries to extract the good order by inspecting the logical circuits [10], whereas, the second class is based on the dynamic optimization of a given order[11] .

Evolutionary computation has been proven to be an effective way to solve complex engineering problems.

It presents many interesting features such as adaptation, emergence and learning. Artificial neural networks, genetic algorithms and artificial immune systems are examples of bio-inspired systems used to this end. One of the iterative methods that have been developed recently to solve this type of problem is Genetic Algorithms GA [16, 8]. It is a stochastic iterative algorithm which maintains a population of individuals. GA adapts nature optimizing principles like mechanics of natural selection and natural genetics. Each individual represents a feasible solution in the problem search space. Basically, a genetic algorithm consists of three essential selection, crossover, and mutation. The selection evaluates the fitness of each individual and keeps the best ones among them. The others are removed from the current population. The crossover merges two individuals to provide new ones. The operator of mutation allows moving each solution to one of its neighbours in order to maintain a good diversity during the process of optimization. GA allows guided search that samples the search space. Although GAs have been showed to be appropriate for solving BDD ordering problem [9], their computational cost seems to be a dissuasive factor for their use on large instances. To overcome this drawback and in order to get better speed and quality convergence, their implicit parallelism is exploited.

Metaheuristics are a family of optimization techniques inspired by nature used to solve difficult optimization problems for which we do not know the most effective method. The heuristic search procedures are recognized as the neighboring structures (NS), which transform a solution to one of its neighbourhood by applying some perturbations. Variable neighbourhood search (VNS) [14] is a recent metaheuristic for solving combinatorial optimization problems. The main idea of VNS is to change systematically the neighbourhood by simply passing one NS to another while they execute a search Local. The VNS changes the neighbourhood structure where the search is trapped in a local minimum. VNS offers a multiple neighbourhood structure with which one recovers the solutions trapped via the others. The main idea here is to choose heuristics (neighbourhood structures) complementary to each other [3].

In this context, we propose in this article, a new iterative approach called GAVNSBDD based on a hybrid GA and variable neighbourhood search. For that, a problem formulation in terms of genetic representation and evolutionary dynamic borrowing evolutionary operators were defined. To foster the convergence to optimality, the VNS has been embedded within the optimization process. VNS helps the search process to avoid local optima and explores the solution space economically and effectively without getting trapped into cycles. The experiences carried out on GAVNSBDD showed the feasibility and the effectiveness of our approach.

Consequently, the remainder of the paper is organized as follows: *section 2* presents some basis concepts of BDD. A brief introduction to variable neighbourhood search is presented in *section 3*. The proposed approach is described in *section 4*. *Section 5* illustrates some experimental results. Then, we finish by giving conclusion and some perspective.

2 Binary Decision Diagram

A Binary Decision Diagram or BDD is data structure used for representation of Boolean functions in the form of rooted directed acyclic graph. A BDD is a rooted directed acyclic graph $G = (V, E)$ with node set V containing two kinds of nodes, *non-terminal* and *terminal* nodes (Figure1). A non-terminal node v has as tag a variable $index(v) \in \{x_1, x_2, \ldots, x_n\}$ and two children $low(v)$, $high(v) \in V$. The final nodes are called *0-final* and *1-final*. A BDD can be used to compute a Boolean function $f(x_1, x_2, \ldots, x_n)$ in the following way. Each input $a = (a_1, a_2, \ldots, a_n) \in \{0, 1\}^n$ defines a computation path through the BDD that starts at the root. If the path reaches a non-terminal node v that is labelled by x_i, it follows the path $low(v)$ if $a_i = 0$, and it follows the path $high(v)$ if $a_i = 1$. The label of the terminal node determines the return value of the BDD on input a. the BDD is called "ordered" if the different variables appear in the same order on all the ways from the root (figure 1). It is important to note that for a given order of variables, the minimal binary decision graph is single. A BDD can be reduced while using the two following rules [7, 6, 4]:

- Recognize and share identical sub-trees.
- Erase nodes whose left and right child nodes are identical.

It is very important to take into account the order of variables to be used when using the BDD in practice. The size of a BDD is largely affected by the choice of the variable ordering (Figure 2).

Unfortunately, there are an exponential number of possible orders (permutation). It is completely clear that the problem of variables ordering is NP-difficult. The use of heuristics is essential to find acceptable solutions within reasonable times. Within this perspective, we are interested in applying evolutionary computing principles to solve the variable ordering problem.

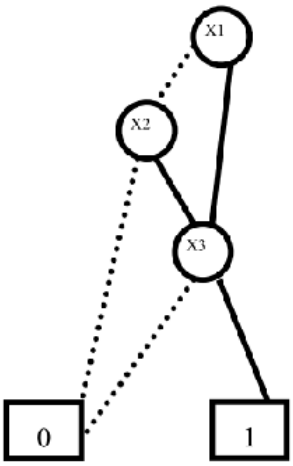

Figure 1: Binary Decision Diagram for the Boolean function $f = x1x3 + x2x3$

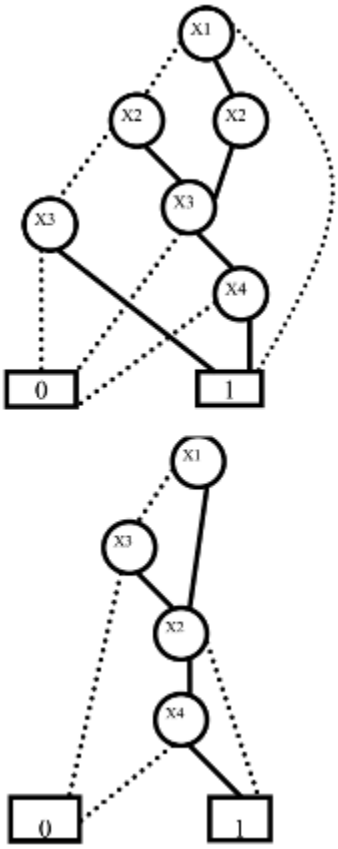

Figure 2: Two BDDs representation of the function $(x1 \vee x3) \wedge (x2 \Rightarrow x4)$, in the top the order of variable is: $x1, x2, x3, x4$; and in the bottom the order is: $x1, x3, x2, x4$

3 Variable neighbourhood search

A Variable Neighbourhood Search (VNS) is a recent metaheuristic based on the idea of a systematic change in the search space. Basically, a local search algorithm performs exploration in a limited region of in the candidate solutions space. The effectiveness of VNS compared to simple local search method is based on the systematic change of neighbourhood. A simple algorithm VNS starts from an initial solution from the search space, than enhances it through a two nested loop in which the core one alters and explores via two main functions so called *shake* and *local search*. Shaking step is performed by randomly selecting a solution from the first neighbourhood. This is followed by applying a local search algorithm. This procedure is repeated as long as a new incumbent solution is found [14, 3].

In order to develop a competent VNS algorithm, the structure of neighbourhood and heuristic functions used must be selected with great accuracy in order to construct a powerful VNS algorithm. We can use more than one neighbourhood structure for shake and local search procedures. The VNS comprises the following steps:

1. **Initialization:** Find an initial solution x.
2. **Repeat:** the following steps until the stopping condition is met:
   (a) **Shake Procedure:** Generate at random a starting solution x'.
   (b) **Local Search:** Apply a local search from the starting solution x' using the base neighbourhood structure until a local minimum x" is found.
   (c) **Improve or not:** If x'' is better than x, do $x \leftarrow x'$.

The integration of the VNS in the genetic algorithm can be done in two ways. This integration can replace completely the mutation operation in a simple genetic algorithm, as we can combine the two operations into a single operation of mutation. Although this method has proved effective in solving several difficult problems, it remains, nonetheless difficult to adapt to the problem of BDD ordering. This is mainly due to the large number of parameters to define: initial solution, local search method, neighbourhood function, number of neighbours to explore.

3.1 Initial solution

It has been shown that the effectiveness of approaches based on the principle of local search depends deeply on

the quality of the initial solution. In our approach, the initial solution is obtained from the genetic algorithm and passed as a parameter to the operator of VNS. So, the solutions passed to the VNS procedure are mainly dependent on the quality of the current individual population in the genetic algorithm.

3.2 Neighbourhood function

The choice of the neighbourhood is very important in this type of local search method; it must be a compromise between efficiency and quality. The complexity of solving approach based on VNS depends mainly on the size of the current solution neighbourhood and the valuation method of each of these neighbours to determine which minimizes the cost function. It has been demonstrated that almost 90% of the execution time is spent in neighbourhood's evaluation. Therefore, it is interesting to use if possible neighbourhoods with constraints to reduce the runtime complexity. The best neighbour generated belonging to the first neighbourhood is selected and the total number of neighbourhoods to generate for each iteration is well defined. The mechanism for the generating of neighbouring solutions is based on choosing a first point to move, then a second point which will be the target of movement. We selected a number of neighbourhoods equal to two, the point of movement is selected from the first neighbourhood defined by the VNS or we undertake a systematic change in the neighbourhood.

4 The Proposed Approach

The development of the suggested approach called GAVNSBDD is based basically on a genetic representation of the research space associated with the problem and an evolutionary dynamic used to explore this space by operating on the genetic representation by using evolutionary operations.

4.1 The fitness function

To select the best individuals of the current population, we must first evaluate these individuals to compute their efficiency or adaptation. The fitness in our case is the number of nodes in the resulting BDD. The best individual is one who gives a minimum number of nodes; the greater the size of the BDD is minimal, the resulting solution is optimal. In order to compute the fitness of variable order, we must first construct the corresponding BDD and then we compute the number of nodes in the resulting BDD. The implementation of the fitness function must be especially optimized because it will be executed many times during the execution of the genetic algorithm. Consequently, the algorithm convergence depends largely on the runtime of the fitness function.

4.2 Genetic representation of variable order

The problem of variable ordering can be mathematically formulated as follow
Given a set of variables $V = \{x_1, x_2 \ldots, x_n\}$, the problem of BDD variables ordering can be defined by specifying implicitly a pair (Ω, SC) where is the set of all possible solutions that is potentials variables order and SC is a mapping $\Omega \longrightarrow \Re$ called score of the variable ordering. This score is the BDD size. Each solution is viewed as permutation of the V variables. Consequently, the problem consists to define the best permutation of V that gives the minimal BDD size. In our approach, the variable order is represented as integer vector (figure 3) satisfying the following criteria:

- For N variable, the size of the vector is N.
- There is no repetitive variable in this vector

1	5	3	4	6	2

Figure 3: Genetic representation of the variable ordering

4.3 Initial Population

The initial population is an important factor in evolutionary algorithms. The generation process of individuals in this population must be carefully selected in order to provide to the genetic algorithm a set of potential and diverse individuals. The goal is to construct a number of different variable orders to be used by the genetic algorithm. In our case, the initial population was created randomly. However, it is important to use heuristics to construct initial solutions of good quality and thus to reduce the convergence time. Another side, the population size has a great impact on the performance of evolutionary algorithms. A large population represents a large space search of solutions, but increases the computational cost. On the other hand, a small population size can lead to local solutions. View we have used a local search method in the core of genetic algorithm, it is preferably to reduce the population size. We have found that a population between 10 and 30 can give good results.

4.4 Genetic operators

Seclection: The selection is to select individuals that have the best fitness to construct the intermediate population in order to apply the evolutionary operators. Each individual is selected with a probability related to its fitness. Thus, individuals with high affinity value have more chance of being selected for the next generation. Selection strategies that we can use in genetic algorithms such as [8]:

- The roulette wheel selection.
- The opposite roulette wheel selection.
- The tournament selection.
- Selection by rank.

In our approach, we have used the roulette wheel selection. The basic motivation for adopting this fitness proportional rule is to preserve diversity of good and bad individuals in the pool, so that it can contain a large space search of potential solutions and to avoid being trapped in local minimum.

Mutation operator: this operator performs permutation between two variables (figure4). It allows moving from the current solution to one of its neighbors. It consists first in selecting pairs of two variables are chosen randomly according to a defined probability, and then swap these two variables as in figure 4 . This operator allows exploring new solutions and thus enhances the diversification capabilities of the search process.

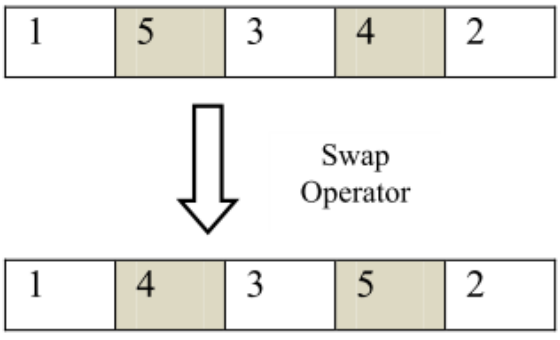

Figure 4: Mutation Operator

Crossover operators: Crossovers are important for promoting the exchange of high quality blocks within the population (figure 5). They exchange subparts of two chromosomes. We have used Partial Mapped Crossover (PMC). The PMC was recommended by Goldberg and Linge[13] . It passes ordering and value information from the parent orders to the offspring orders. A portion of one parent's string is mapped onto a portion of the other parent's string and the remaining information is exchanged.

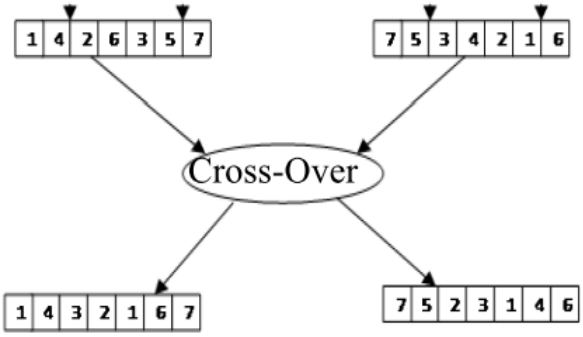

Figure 5: Cross-over Operator

4.5 Local search procedure

To improve the efficiency of the exploration process, we incorporated a local search method in evolutionary dynamics. This form of hybridization has proved advantageous in the context of BDD problems. Indeed, the performances of the local search method allow deeper exploration of certain solution space identified as particularly promising. On the other hand, the diversify ability of the genetic algorithm allows the periodic shift of the solution search to regions rarely visited so far. The proposed hybrid approach is based on genetic algorithm enhanced by the local search method called variable neighbourhood search.

The local search method used in the VNS procedure is the Tabu Search (TS) [1, 12]. TS is among the most popular and robust local search methods. TS is found to be practical in many hard combinatorial optimization problems. The general procedure of TS is given in the figure6 . The main idea underlying is to diversify the search and avoid becoming trapped in local by forbidding or penalizing moves which take the solution, in the next iteration, to point in the solution space previously visited. For this, the algorithm creates a memory list of undesirable moves called "tabu list".

However, Tabu restrictions may be overruled under certain conditions, in which, a Tabu move leads to a better solution, this principle is called aspiration criterion.

Our approach is flexible, so we can used other stochastic local search algorithms.

```
k := 1.
Generate initial solution
WHILE the stopping condition is not met DO
   Identify N(s).      (Neighbourhood set)
   Identify T(s,k).    (Tabu set)
   Identify A(s,k).    (Aspirant set)
   Choose the best s'∈ N(s,k) = {N(s) -
T(s,k)}+A(s,k).
   Memorize s' if it improves the previous best
known solution
   s := s'.
   k := k+1.
END WHILE
```

Figure 6: Tabu Search procedure

4.6 Outline of the proposed framework

Now, we describe how the representation scheme including genetic representation and evolutionary operators has been embedded within a variable neighbourhood search algorithm and resulted in a hybrid stochastic algorithm performing BDD variable ordering.

Our approach is an evolutionary algorithm based on a genetic core. The approach consists of a population of individuals where each individual represents a specific variable orders. To find the best solution, the optimization process of our algorithm consists of a set of steps. The first step is to create the initial population generated randomly. At each iteration of the algorithm, we apply the classical operators: selection, crossover, mutation, fitness evaluation, etc. To increase the performance optimization of our approach, we have incorporated variable neighbourhood search (VNS) in the genetic core. In more details, the proposed approach can be described as follow:

Input:A set of variable *ord*

1. Construct the initial Population of Chromosomes POP
2. Evaluate the population
3. save the best solution
4. **Repeat**
5. Apply a crossover operation on POP according to crossover probability.
6. Apply a mutation operation on POP according to the probability pm.
7. Apply the variable neighbourhood search.
8. Evaluate the new population.
9. Update the best solution
10. Apply selection and reduce and merge operators
11. **Until** a termination-criterion is reached

Output:the best variable order.

5 Implementation and Evaluation

Our approach is implemented with Visual C ++ 2008 and tested on a PC with a 3.0 GHZ processor and 512 MB of memory. We have used the Paradiseo platform [5] to implement our approach. In addition, we have used the package Buddy [15], which contains a set of tools for creating and handling BDDs. For the performance evaluation of our approach, we have used several set of tests created with the gates $NOT, AND, XOR, NAND$. The Building of complex circuits is made by using the ITE operator. An operator, ite, is defined as follows: for logic functions $f, g,$ and h, $ite(f, g, h) = f.g + \sim f.h(\sim not)$. The ite operator can be used to realize all Boolean operations with two variables. For example, $f + g = ite(f, 1, g)$, $f.g = ite(f, g, 0)$, ...etc. We compared the results found by our approach with those of two programs based on a pure genetic algorithm and a pure VNS using Friedman statistic tests. The Friedman test is used test whether the difference between the medians of the methods is not significant. Test results are shown in tables 1,2,3,4. In each table, the second column shows the type of multifunction operator implemented by ITE, the third column contains the size of the BDD obtained from the method GABDD based on pure genetic algorithm, the fourth column shows the size of the BDD obtained from the method VNSBDD based on pure VNS and the last column shows the size of the BDD obtained from the method GAVNSBDD based on hybrid method between GA and VNS. In all experiments, the parameters that we used in our method are: the population size is 10, the mutation rate is 0.1, the taboo list is static array of size 10, the number of the generated neighbours is 2 and the number of iterations in the genetic algorithm core is 100. In GABDD method, it was used the following parameters: the population size is 10, the mutation rate is 0.01, and the number of iterations is 1000.

The results of our method illustrate clearly the effectiveness of merging the genetic algorithm and VNS

algorithm to perform the binary decision diagram ordering problem. The Freidman test (figure7) confirms that our approach ranks high in this experiment. However the performance of the pure genetic algorithm GABDD is a poor compared to GAVNSBDD or VNSBDD. Moreover, Friedman's test shows that the results obtained by the pure VNS approach are near to those of the hybrid method GAVNSBDD, this reflects the good method that uses VNS to change systematically the neighbourhood to find the optimal solution. The effectiveness of our approach is explained by the good combination between diversification and intensification which leads the algorithm to effectively explore the search space and locate a good solution.
The search for the optimal solution is not performed randomly but it runs iteratively by gradually improving the solution until the stop criterion is satisfied. Figures 8 and 9 show how the algorithm converges to the optimal solution. In the first figure (obtained by a test of two functions each of 20 variables), our method starts with an initial solution equal to 766 then it is gradually improved, sure there are mediocre solutions but it's part of the evolution process which leads to better solutions. The algorithm then converges gradually to reach a solution equal to 200 which is the optimal solution found.

Table 1: Results: BDD functions with 20 variables

	$Initialsolution$	GA	VNS	$GAVNS$
$FxorG$	766	491	483	190
$FandG$	882	360	299	206
$ForG$	828	282	242	219

Table 2: Results: BDD functions with 30 variables

	$Initialsolution$	GA	VNS	$GAVNS$
$FxorG$	1846	1443	811	617
$FandG$	1175	750	448	244
$ForG$	1918	1565	695	444

Table 3: Results: BDD functions with 40 variables

	$Initialsolution$	GA	VNS	$GAVNS$
$FxorG$	8437	3183	2713	1208
$FandG$	2629	2097	2000	1704
$ForG$	1394	883	2557	889

Table 4: Results: BDD functions with 60 variables

	$Initialsolution$	GA	VNS	$GAVNS$
$FxorG$	34555	24787	15010	21041
$FandG$	21342	13898	21038	18727
$ForG$	22838	17394	21300	15198

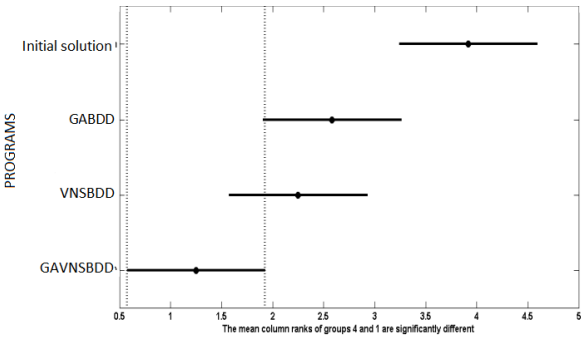

Figure 7: Freidman test The nearest to zero is the best program

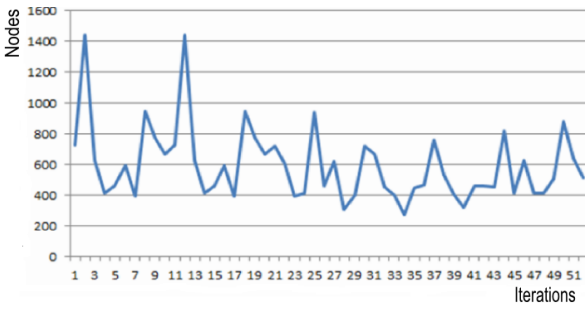

Figure 8: The behaviour of the best fitness (two functions of 20 varibales)

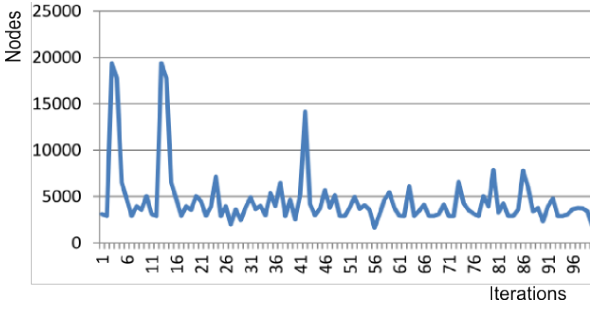

Figure 9: The behaviour of the best fitness (two functions of 40 variables)

6 Conclusion

In this work, we have proposed a new approach called GAVNSBDD to deal with the BDD variable problem. GAVNSBDD is based on a hybridizing of genetic algorithm and variable neighbourhood search method. The experimental studies prove the feasibility and the effectiveness of our approach. We have shown that the uses of variable neighbourhood search can help the genetic algorithm to find better solutions. In the future, we try to use other search local strategies more adapted to the BDD problem like sifting technique. In order to accelerate our approach, we may apply parallelization techniques. Finally, the performance of the algorithm may be improved by using a clever startup solution.

References

[1] Blum, C., Roli, A., and Sampels, M. Hybrid metaheuristics (editorial). *Journal of Mathematical Modelling and Algorithms,Special issue on Hybrid Metaheuristics*, 5(1), April 2006.

[2] Bollig, B. and Wegener, I. Improving the variable ordering of OBDDs is NPcomplete. *IEEE Trans. on Comp*, 45(9):993–1002, 1996.

[3] Brimberg, J., Mladenović, N., Urošević, D., and Ngai, E. Variable neighborhood search for the heaviest k-subgraph. *Computers and Operations Research*, 36(11):2885–2891, 2009.

[4] Bryant, R. E. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM computing surveys*, 24(3):293–318, September 1992.

[5] Cahon, S., Melab, N., and Talbi, E.-G. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, May 2004.

[6] Drechsler, R. and Becker, B. *Binary Decision Diagrams: Theory and Implementation*. Kluwer Academic Publisher, 1998.

[7] Ebendt, R., Fey, G., and Drechlser, R. *Advanced BDD optimization*. Springer, 2005.

[8] Eiben, A. E. and Smith, J. E. *Introduction to Evolutionary Computing*. Springer, 2003.

[9] Forrest, S. Genetic algorithms: Principles of natural selection applied to computation science. 261:872–878, 1993.

[10] Fujii, H., Ootomo, G., and Hori, C. Interleaving based variable ordering methodes for ordered binary decision diagrams. In *Int'l Conf on CAD*, pages 38–41, 1993.

[11] Fujita, M., Fujisawa, H., and Kawato, N. Evaluation and improvements of boolean comparison method based on binary decision diagrams. In *Intl's conf on CAD*, pages 2–5, 1988.

[12] Glover, F. Tabu search part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[13] Goldberg, D. E. and Lingle, R. Alleles, loci, and the tsp. In *Proceedings of the first International Conference on Genetic Algorithms*, pages 154—-159, 1985.

[14] Hansen, P., Mladenović, N., and Moreno-Pérez, J. A. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[15] Jørn, L.-N. `http://sourceforge.net/projects/buddy/`.

[16] Layeb, A. and Saidouni, D. E. A new quantum evolutionary algorithm with sifting strategy for binary decision diagram ordering problem. *International Journal of Cognitive Informatics and Natural Intelligence*, 4(4):47–61, December 2010.

[17] Raimondi, F. and Lomusico, A. Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. *J.Applied Logic (JAPLL)*, 5(2):235–251, 2007.