

BABEȘ-BOLYAI UNIVERSITY OF CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE



Doina Logofătu

Evolutionary Algorithms in VLSI-CAD

PhD Thesis Summary

Advisor: D. Dumitrescu

Scientific committee:

Prof. Henri Luchian, Al. I. Cuza University of Iași

Prof. Gabriela Czibula, Babeș-Bolyai University of Cluj-Napoca

Conf. Marcel Cremene, Technical University of Cluj-Napoca

2010

Index of contents

INDEX OF CONTENTS	3
STRUCTURE, INDEX TERMS, ORIGINAL CONTRIBUTIONS.....	5
1.1 THESIS'S STRUCTURE.....	5
1.2 INDEX TERMS	7
1.3 ORIGINAL CONTRIBUTIONS AND PUBLICATIONS.....	7
MAPREDUCE.....	12
2.1 MAPREDUCE PARADIGM.....	12
2.2 ADVANTAGES OF USING APACHE HADOOP.....	12

DATA ORDERING PROBLEM	14
3.1 DESCRIPTIONS OF DOP AND DOPI	14
3.2 PREVIOUS RESEARCH	16
3.3 THE ALGORITHMS RAN, EX AND LB FOR DOP AND DOPI	17
3.4 THE ALGORITHM GREEDY MIN SIMPLIFIED (GMS).....	17
3.5 THE ALGORITHMS MUT AND GA FOR DOP AND DOPI	17
3.6 THE PARALLEL DISTRIBUTED ALGORITHM MRPGA FOR DOP AND DOPI	18
3.7 NUMERICAL EXPERIMENTS AND STATISTICAL TESTS	20
3.8 CONCLUSIONS AND FUTURE WORK	21
DATA COMPACTION PROBLEM	22
4.1 PROBLEM DESCRIPTION TCP.....	22
4.2 OPTIMAL ALGORITHM FOR TCP	23
4.3 GREEDY ALGORITHMS GRNV AND GRBT FOR TCP	23
4.4 EVOLUTIONARY ALGORITHM GATC FOR TCP	24
4.5 DISTRIBUTED EVOLUTIONARY ALGORITHM MRPEA FOR TCP	24
4.6 EXPERIMENTAL RESULTS AND STATISTICAL TESTS	26
4.7 CONCLUSIONS AND FUTURE WORK	28
BIBLIOGRAPHY	29

Structure, index terms, original contributions

1.1 Thesis's structure

The dissertation comprises theoretical and experimental aspects of two actual optimization problems from *Very Large Scale Integration* (VLSI) domain, specifically regarding the data ordering and compaction. *Data Ordering Problem* (DOP), *Data Ordering Problem with Inversion* (DOPI) and *Data Compaction Problem* (DCP) are NP-hard.

After the first introductory chapter, the next two chapters contain basis aspects regarding the optimization problems and the evolutionary computation.

The fourth chapter comprises aspects, methods and numerical experiments for Data Ordering Problem with and without Inversion. An important aim of circuit design is the reduction of the power dissipation. Power consumption of digital circuits is closely related to switching activity. Due to the

increase in the usage of battery driven devices (e.g. PDAs, laptops), the low power aspect became one of the main issues in circuit design in recent years. In this context, the Data Ordering Problem with and without Inversion is very important. Data words have to be ordered and (eventually) negated in order to minimize the total number of bit transitions. These problems have several applications, like instruction scheduling, compiler optimization, sequencing of test patterns, or cache write-back. Useful applications of the problems are found also in computational biology. The author proposes some new efficient approaches, among them also three evolutionary ones and new genetic operators. There are proposed: an efficient greedy algorithms *Greedy Min Simplified* – GMS_DOPI [Log06c, Log08d, Log09a], two evolutionary algorithms MUT_DOPI (evolutionary algorithm with mutation) and GA_DOPI (hybrid genetic algorithm) [Log06a], a parallel genetic algorithm using the new technology MapReduce – *Map Reduce Parallel Genetic Algorithm* – MRPGA_DOPI [Log10c]. The new proposed genetic operators – *Simple Cycle Inversion Operator* (SCIM), *Cycle PMX* (CPMX) and *Cycle OX* (COX) [Log06a] – are used in the algorithms MUT_DOPI and GA_DOPI. The third approach is distributed using the MapReduce paradigm and it is the most efficient one regarding the quality of the results. Additionally, it is capable to work with large input data. There are offered also the results from series of various experiments, as tables, graphics and statistical tests, which compare the proposed methods with the present ones and certify their efficiency. The author offers also a research Open Source project, available on: <http://dopisolver.sourceforge.net/>.

The fifth chapter presents algorithms and results regarding the Data Compaction Problem. In this chapter are proposed some original algorithms for solving this problem, specifically the *Test Compaction Problem* (TCP). This is also NP-hard. Beside issues like the low power dissipation and the increase of defect coverage, test compaction is an important requirement regarding large scale integration (LSI) testing. The overall cost of a VLSI circuit's testing depends on the length of its test sequence; therefore the reduction of this sequence, keeping the coverage, will lead to a reduction of used resources in the testing process. In this paper we study test vectors over a five-valued logic. The problem of finding minimal test sets is NP-complete. Consequently, an optimal algorithm has limited practical use and is only applicable to small problem instances. The proposed algorithms are two greedy ones, *Greedy Naive Algorithm* (GRNV_TCP) and *Greedy Binary Tree Algorithm* (GRBT_TCP) [Log08a, Log08e], an evolutionary one *Genetic Algorithm Test Compaction* (GA_TCP) [Log08b, Log09c] and a distributed evolutionary algorithm using the new technology MapReduce *MapReduce Parallel Evolutionary Algorithm* (MRPEA_TCP) [Log10b]. There is also offered a research Open Source project, available on: <http://dcpsolver.sourceforge.net/>.

The sixth and the last chapter of this dissertation contain conclusions and directions for future work, followed by a bibliographical list. This list comprises over 130 sources used in the thesis elaboration.

The thesis is ending with four appendixes which contain instructions referring to the use of the Open Source projects, as well as some larger sets of experiments for the both problems DOPI and TCP.

1.2 Index terms

Index terms used in thesis: NP-hard problems, digital circuit design, *Very Large Scale Integration* (VLSI), *Computer Aided Design* (CAD), *Backtracking*, *Greedy*, graph theory, complexity, evolutionary computation, *Traveling Salesman Problem* (TSP), *Set Covering Problem* (SCP), low power, transition, total number of transitions, *Greedy Min* (GM), *Greedy Min Simplified* (GMS), genetic algorithms, evolutionary algorithms, distributed algorithms, distributed genetic algorithms, genetic operators, mutation operators, crossover operators, search operators, evaluation function, fitness, selection, MapReduce technology, Map and Reduce functions, Apache Hadoop, Open Source projects, *Data Ordering Problem* (DOP), *Data Ordering Problem with Inversion* (DOPI), *Test Compaction Problem* (TCP), computational biology, deoxyribonucleic acid (DNA), random algorithms, exact algorithm, optimal algorithms, optimization, statistical tests, student-t, H_0 hypothesis, numerical experiments, lower bound, *Partially Mapped Crossover* (PMX), *Cycle PMX* (CPMX), *Ordered Crossover* (OX), *Cycle OX* (COX), Don't Care symbol, bus-invert paradigm, compaction factor, compaction rate, *dopiSolver*, *dcpSolver*.

1.3 Original contributions and publications

DOP and *DOPI* refer to an important aspect regarding the optimization of the energy dissipation in the integrated circuits, *DCP* the optimization of testing process. For the both problems are offered various algorithms, also some ideas for future work. Among the proposed approaches there are also efficient distributed algorithms using the MapReduce technology. The analyzed problems are interdisciplinary extended to the computational biology domain, for example regarding the ordering and compaction of DNA molecules. Suggestive results regarding this domain sustained graphically and by specific notes are presented in two papers for the international conference BICS 2008. Extended versions for them are published in the *American Institute of Physics* journal.

These results were presented among international conferences like EvoHOT 2006, ROSYCS 2006, ISMVL 2006, DAS 2008 and BICS 2008. Original contributions exposed in thesis:

- Formalization of specific problems from VLSI-CAD: DOP, DOPI and DCP (4.2, 4.3, 5.2, 5.3);
- GMS_DOPI: a new greedy algorithm [Log06c] – (local search) for DOPI, with better results than the used one in the industry – Greedy Min (4.5.4);
- MUT_DOPI: a simple evolutionary algorithm with mutation for DOP/DOPI, which leads to enhancements towards all greedy methods, using a new mutation operator SCIM [Log06a] (4.6.1);
- GA_DOPI: an efficient genetic algorithm for DOP/DOPI, using two new crossover operators Cycle PMX and Cycle OX [Log06a] (4.6.2);
- MRPGA_DOPI: a distributed genetic algorithm for DOP/DOPI using the MapReduce technology [Log10c] (4.6.3);
- experimental results for validate the enhancements brought by the grade of freedom of DOPI regarding DOP. The DOPI greedy methods vs. DOPI ones, the greedy methods vs. GA_DOPI and MRPGA_DOPI algorithms [Log06a, Log06c, Log08d, Log09a, Log10c] (4.6.3);
- statistical tests for comparing the greedy methods against evolutionary ones for DOP/DOPI [Log06a] (4.7);
- GRNV_DCP and GRBT_DCP: two greedy algorithms for DCP [Log08a, Log08e, Log09b] (5.7);
- GA_DCP: an efficient genetic algorithm for DCP [Log08b, Log09c] (5.8);
- MRPGA_DCP: a parallel genetic algorithm for DCP using MapReduce [Log10b] (5.9);
- experiments and charts for comparing the proposed approaches in pairs [Log08a, Log08b, Log09b, Log09c, Log10b] (5.10);
- statistical tests for comparing greedy methods against the evolutionary ones for [Log10b, Log10c] (5.10);
- *dopiSolver* and *dcpSolver*: research frameworks for DOP/DOPI and DCP, for executing/extending the various algorithms: <http://dopisolver.sourceforge.net> and <http://dcpsolver.sourceforge.net>;
- future work and new research directions for both problems (4.9, 5.11).

A list with publications explicitly referring the two optimization problems is give below.

Books:

Logofătu, D.: *Algorithmen und Problemlösungen mit C++*, pp. 402-411, Vieweg+Teubner-Verlag, 2010 (Germany).

Logofătu, D.: *Eine praktische Einführung in C*, pp. 207-208, entwickler.press, München, Germany, 2008 (Germany).

Logofătu, D.: *Grundlegende Algorithmen mit Java*, pp. 65-98:205-214, Vieweg-Verlag, Germany, 2008 (Germany).

Logofătu, D.: *Algoritmi fundamentali in C++. Aplicații*, pp. 127-154:265-273, Editura Polirom, Iași, 2007.

Logofătu, D.: *Algoritmi fundamentali in Java. Aplicații*, pp. 125-158:269-277, Editura Polirom, Iași, 2007.

Papers for international conferences:

Logofătu, D., Dumitrescu, D.: Distributed Genetic Algorithm for Data Ordering Problem with Inversion Using MapReduce, *11th International Conference on Parallel Problem Solving from Nature*, PPSN, submitted, April 2010.

Logofătu, D., Dumitrescu, D.: Parallel Evolutionary Approach of Compaction Problem using MapReduce, *11th International Conference on Parallel Problem Solving from Nature*, PPSN, submitted, April 2010.

Logofătu, D.: On the Compaction of DNA Sequence Vectors, *Proceedings Bio-Inspired Computational Methods Used for Difficult Problems Solving. Development of Intelligent and Complex Systems* (BICS 2008), pp. 25-36, Târgu Mureș, Romania, 2008.

Logofătu, D., Gruber, M.: Efficient Approaches for DNA Sequences Ordering, *Proceedings Bio-Inspired Computational Methods Used for Difficult Problems Solving. Development of Intelligent and Complex Systems* (BICS 2008), pp. 59-69, Târgu Mureș, Romania, 2008.

Logofătu, D.: Efficient Evolutionary Approach for the Test Compaction Problem, *Proceedings 9th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS*, pp. 144-148, Suceava, Romania, May 22-24, 2008.

Logofătu, D., Drechsler, R.: Comparative Study by Solving the Test Compaction Problem, *Proceedings 38th International Symposium on Multiple-Valued Logic (ISMVL '08)*, Dallas, USA, pp. 44-49, 2008.

Logofătu, D.: Greedy Approaches for the Data Ordering Problem with Inversion, *Proceedings of ROSYCS - Romanian Symposium on Computer Science*, pp. 65-80, Iași, 2006.

Logofătu, D., Drechsler, R.: Efficient Evolutionary Approaches for the Data Ordering Problem with Inversion, *3rd European Workshop on Hardware Optimization Techniques (EvoHOT)*, LNCS 3907, pp. 320-331, Springer, Berlin/Heidelberg, 2006.

Drechsler, R., Drechsler, N.: Minimization of Transitions by Complementation and Resequencing using Evolutionary Algorithms, In *Proceedings of 21st IASTED International Multi-Conference Applied Informatics (AI 2003)*, Innsbruck, 2003.

Drechsler, N., Drechsler, R.: Exploiting don't cares during data sequencing using genetic algorithms, *ASP Design Automation Conf.*, pp. 303-306, 1999.

Murgai, R., Fujita, M., Oliveira, A.: Using complementation and resequencing to minimize transitions, *Design Automation Conf.*, pp. 694-697, 1998.

Murgai, R., Fujita, M., Krishnan, S. C.: Data sequencing for minimum-transition transmission, *IFIP Int'l Conf. on VLSI*, 1997.

Stan, M., Burleson, W.: Limited-weight codes for low-power I/O, *Int'l Workshop on Low Power Design*, 1994.

Journals, magazines:

Logofătu, D.: Static Test Compaction for VLSI Tests: an Evolutionary Approach, *Advances in Electrical and Computer Engineering*, Vol. 8, Nr. 2, pp. 49-53, Romanian Academy of Technical Sciences, 2008.

Logofătu, D.: DNA Sequences and their Compaction, BICS 2008: Proceedings of the 1st International Conference on Bio-Inspired Computational Methods Used for Difficult Problems Solving: Development of Intelligent and Complex Systems. AIP Conference Proceedings, Vol. 1117, Nr. 1, pp. 29-39, *American Institute of Physics*, 2009.

Logofătu, D., Gruber, M.: DNA Sequences and their Ordering, BICS 2008: Proceedings of the 1st International Conference on Bio-Inspired Computational Methods Used for Difficult Problems

Solving: Development of Intelligent and Complex Systems. AIP Conference Proceedings, Vol. 1117, Nr. 1, pp. 3-11, *American Institute of Physics*, 2009.

For solving DOP/DOPI and DCP were realized Open Source projects, available on <http://dopiSolver.sourceforge.net> and <http://dcpSolver.sourceforge.net>.

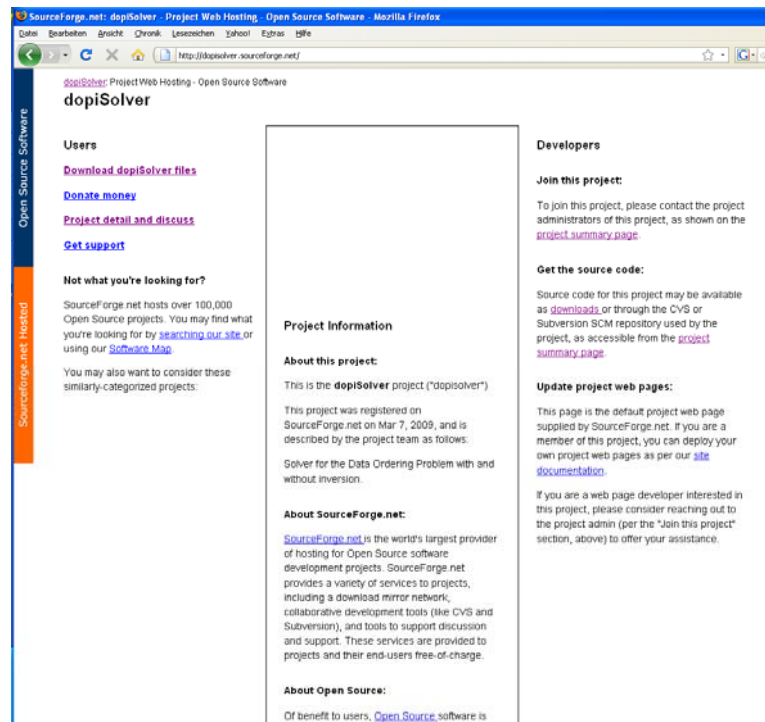


Fig. 1.1. <http://dopisolver.sourceforge.net/>


These frameworks are flexible and offer extension facilities with new approaches for the problems. The appendixes contain the usage instructions for the projects *dopiSolver* and *dcpSolver* from the server *sourceforge.net*, as well as detailed sets of experiments.

SourceForge.net > Find Software > dcpSolver

 **dcpSolver** by dlogofatu

Summary | Files | Support | Develop

Solver for the Data Compaction Problem (dcp) for static test compaction and some specific bionformatic issues. Contains code, charts, tables and test data.

Download Now!
dcpSolver 0.1 (814.3 KB)  OR [View all files >](#)

WWW <http://dcpsolver.sourceforge.net>

TAGS [add](#)
Separate each tag with a space.

Release Date: 2010-04-01

Registered: 2010-03-17

Fig. 1.2. <http://sourceforge.net/projects/dcpsolver/>

MapReduce

2.1 MapReduce Paradigm

The MapReduce technology is a distributed programming model introduced by Google. With this model the user specifies the calculation with help of two functions, *Map* and *Reduce* [Dea08, Eka08, Jin08, Llo10].

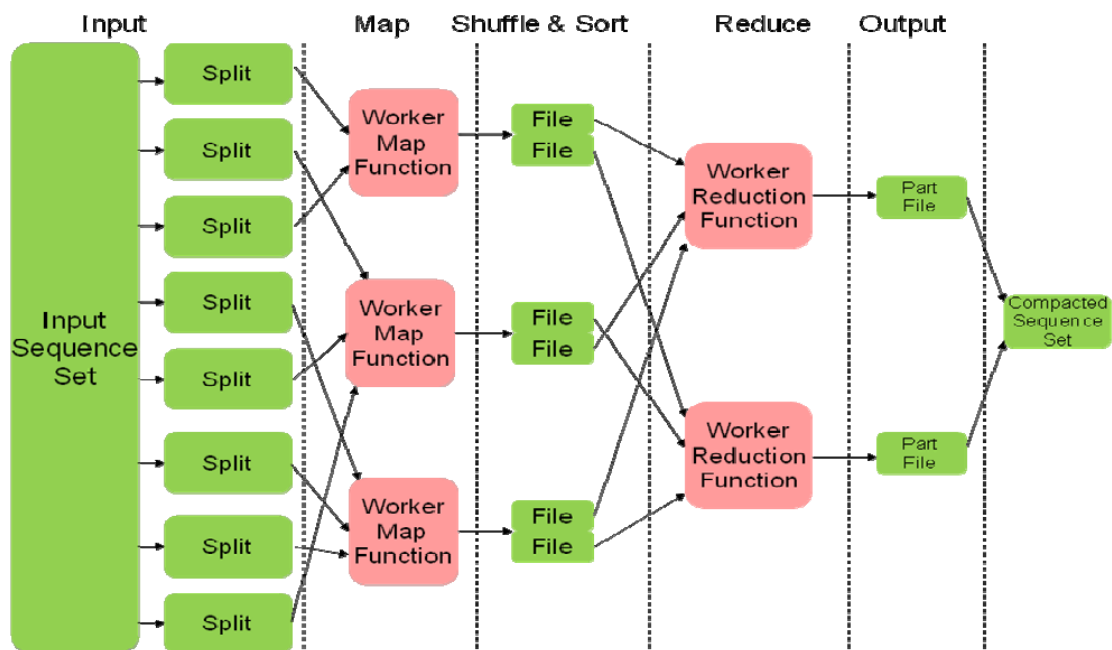


Fig. 2.1. MapReduce dataflow

The MapReduce technology uses these two abstractions, Map and Reduce, for allowing the development of large scale distributed applications, if these applications permit it. Conceptual, the functions Map and Reduce offered by user must have the form:

$$\text{Map}(k_1, v_1) \rightarrow \text{List}(k_2, v_2)$$

$$\text{Reduce}(k_2, \text{List}(v_2)) \rightarrow \text{List}(v_3)$$

The Map calls are distributed among more machines by automated partitioning of the input data. Subsets of input data are parallel processed on different machines. The Reduce calls are also distributed by using a partitioning function.

2.2 Advantages of using Apache Hadoop

Hadoop MapReduce is a batch data processing system for running applications which process vast amounts of data in parallel, in a reliable and fault-tolerant manner on large clusters of compute nodes, eventually running on commodity hardware. It comes with status and monitoring tools and offers a clean abstraction model for programming supporting automatic parallelization and distribution.

A MapReduce *job* splits the input data into independent chunks (splits) which are then processed by the *map tasks* in a completely parallel manner. The framework sorts the maps' outputs and forward them as input to the *reduce tasks*.

Hadoop comes with a distributed file system (*HDFS*) that creates multiple replicas of data blocks and distributes them on compute nodes throughout the cluster to enable reliable, extremely rapid computations. The compute nodes and the storage nodes are typically the same. This allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate rate across the cluster.

The MapReduce framework consists of a single master `JobTracker` and one slave `TaskTracker` per compute node. The master is responsible for scheduling the tasks for the map- and reduce-operations on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks, as directed by the master.

The applications specify the input/output locations, supply *map* and *reduce* functions and eventually invariant (contextual) data. These comprise the *job configuration*. The Hadoop *job client* then submits the job (java byte code packed in jar-archive) and configuration to the `JobTracker` which then distributes them to the slaves; schedules the map-/reduce- tasks and monitors them, providing status and diagnostic information to the *job client*.

Data Ordering Problem

Besides some already existing approaches, the author proposed new algorithms for the both problems DOP and DOPI, among them three evolutionary ones, as well as new genetic operators. There are proposed a new efficient greedy algorithms – *Greedy Min Simplified* – GMS_DOPI [Log06c, Log08d, Log09a], two undistributed evolutionary algorithms MUT_DOPI (evolutionary algorithm with mutation) and GA_DOPI (hybrid genetic algorithm) [Log06a], a parallel genetic algorithms using the MapReduce paradigm – *Map Reduce Parallel Genetic Algorithm* – MRPGA_DOPI [Log10c]. The new proposed genetic operators – *Simple Cycle Inversion Operator* (SCIM), *Cycle PMX* (CPMX) and *Cycle OX* (COX) [Log06a] – are used in the algorithms MUT_DOPI and GA_DOPI. The distributed genetic approach is the best one regarding the quality of the results. In addition, it is capable to deal with large data inputs. There are also offered many-sided experiments results, in form of tables, charts and statistical tests, which compare and certify their efficiency. The author offers as well a research Open Source project, available at <http://dopisolver.sourceforge.net/>.

3.1 The Description of the problems DOP and DOPI

Definition 3.1. A *transition* is the position of a bit which change, i.e. two binar sequences are different on the same position.

Definition 3.2. We denote as the *inversion (complementation)* of a binary word the operation which changes every bit with its negation (complementation - $0 \rightarrow 1, 1 \rightarrow 0$). We denote with \bar{w} the inversed (complemented) word for w .

Definition 3.3. *Number of transitions.* If a word w_r is transmitted immediately followed by w_s , the *number of transitions* is given by the number of bits that change. This is:

$$d(w_r, w_s) = \sum_{j=1}^k w_{rj} \oplus w_{sj} \quad (3.1)$$

Here, the w_{rj} denotes the j^{th} bit of w_r , and \oplus the XOR operation.

The number of transitions is, in fact, the *Hamming distance* between w_r and w_s . Word reordering can change the number of transitions significantly.

Definition 3.4. *Total number of transitions.* The *total number of transitions* is the sum of number of transitions needed for the transmission of all the words. It is denoted with N_T . If σ is a permutation of $\{1, 2, \dots, n\}$, than the total number of transitions will be:

$$N_T = \sum_{j=1}^{n-1} d(w_{\sigma(j)}, w_{\sigma(j+1)}) \cdot \quad (3.2)$$

Examples: $w = 10110001$, $\bar{w} = 01001110$ and $\overline{10110} = 01001$.

For the problem DOPI (*DOP with Inversion*) we allow that some words are inverted by receiver.

Definition 3.5. *Phase-assignment.* The *phase-assignment* (polarity) δ is a function defined on $\{1, \dots, n\}$ with values in $\{0, 1\}$, which specify if a word is sent normal or complemented (negated, inverted).

Definition 3.6. *DOP.* Find a permutation σ of the bit strings w_1, w_2, \dots, w_n such that the total number of transitions:

$$N_T = \sum_{j=1}^{n-1} d(w_{\sigma(j)}, w_{\sigma(j+1)}) \quad (3.3)$$

is minimized.

Definition 3.7. *DOPI.* Find a permutation σ and a phase-assignment δ of the bit strings w_1, w_2, \dots, w_n such that the total number of transitions:

$$N_T = \sum_{j=1}^{n-1} d(w_{\sigma(j)}^{\delta(j)}, w_{\sigma(j+1)}^{\delta(j+1)}) \cdot \quad (3.4)$$

is minimized.

The problems DOP and DOPI are NP-hard. Generally, if there are considered sequences with the same length, the Hamming distance for pairs of them gives the total number of dissimilarities. We denote the generalizations of these problems also with DOP and DOPI. There is given a set of n sequences with the same length k over an alphabet \mathcal{A} . In the case of DOPI is given also a permutation of length k , which denotes the inversion (complementation) function. It is asked to find a permutation of the words (in the case of DOPI also an assignment of them) which minimizes the total number of dissimilarities (transitions).

3.2 Previous research

In the past few years some heuristics were developed for both DOP and DOPI (most of them in relation with the TSP problem):

1. *Double Spanning Tree* (DST) [Gar79]
2. *Spanning Tree/Minimum Matching* (ST-MM) [Gar79]
3. *Greedy Min* (GM) [Mur97]
4. *Greedy Simple* (GS) [Dre97]
5. *Evolutionary Heuristics* [Dre03]

The most powerful polynomial heuristic known so far is *Greedy Min* and it can be applied to both DOP/DOPI:

- 1) Computes the Hamming distance for all (distinct) pairs of given words and selects the pair with a minimum cost.
- 2) Chooses the most convenient pair of words. The beginning sequence will contain these two words.
- 3) Builds progressively the sequence, adding in every step of the most convenient word (that was not yet added). This word can be added either at the beginning or at the end of the sequence, depending where the Hamming distance is minimal.

The EAs are the best algorithms regarding the quality of results. Such evolutionary approaches provide better results than the above-presented *Greedy Min*, but with significantly more time resources. EAs which perform high-quality optimizations are presented in [Dre97], [Dre03]. In [Dre03] are presented evolutionary algorithms for both DOP and DOPI. For DOPI the mutation and crossover operators are applied in parallel for creating new individuals. This is also a hybrid EA, since the initial individuals are preprocessed using greedy methods. The results provided by the EA are better than the *Greedy Min* results, but the maximal number of words is 100.

In [Mur98] a graph theory related model for DOPI is introduced, together with a relevant graph theoretic background. For a DOPI instance with n words, each of length k , a multigraph can be created accordingly. The vertices are the words, if they are in the same phase-assignment (either both 0 or both 1), then the distance between them is the same. Also, if they are in different phase-assignment, the distance remains the same. There are two edges between two vertices (one if the words are in the same phase, one for the case if they are transmitted in different phases). In this manner DOPI is transformed in the equivalent NP-complete problem of finding the Hamiltonian path with the minimum length.

3.3 The algorithms RAN, EX and LB for DOP and DOPI

RAN is the simplest heuristic for generating start solution candidates for such problems. The algorithm *random* (RAN) for DOP generates a random permutation and return it, together with the related cost. For DOPI it will be generated a permutation and a bit string representing the assignation [Log06a, Log07c].

The simplest optimal algorithm for DOP/DOPI is based on the exhaustive search. It runs through all the permutations (assignations) and retains the one with minimum cost. This algorithm can be improved using dynamic programming variations, *branch-and-bound* or *linear programming*. For testing some basis cases there are implemented variants using *brute-force*.

The exact algorithm for DOP, EX, generates all permutations and keeps the first with a minimum cost. The one for DOPI generates all possible permutation-assignation pairs and retains the first with minimum cost [Log06a, Log07c].

The lower-bound algorithm uses the minimal spanning tree und proves to be very useful for evaluating the efficiency of the proposed greedy and evolutionary algorithms.

3.4 The algorithm Greedy Min Simplified (GMS)

The algorithm *Greedy Min Simplified* (GMS) [Log06c, Log08d, Log09a] works, as well as *Greedy Min* [Mur97], for the both problems, *DOP* and *DOPI*. The distinction is that, at every step, adding a new word is allowed only at an end of the sequence (with the possibility to complement some words). The explanation is that one spares time by testing only one ending and the performances are very near *Greedy Min*, even better for large data sets. For the transmission operation (by dynamic testing or among Internet), the time is a very important factor.

3.5 The algorithms MUT and GA for DOP and DOPI

MUT_DOPI [Log06a] is a simple approach which operates only on one single individual (initialized with *Greedy Min*). There is proposed a new mutation operator, *Simple Cycle Inversion Mutation* (SCIM) [Log06a], which is applied together with *Simple Inversion Mutation* (SIM) [Hol75]. This algorithm accomplishes enhancements in a short execution time.

GA_DOPI [Log06a] provides enhancements using the classical scheme of the hybrid genetic algorithms. The genetic operators are synchronously applied for permutation and bit string, with the scope to keep the good trait of the individuals. On the individuals from the current population are applied the mutation and crossover operators with a given probability, and the best *populationSize* individuals are kept in the selection phase. The total number of transitions from the formula (3.2) is used to measure the objective function for an individual:

$$eval(\sigma, \delta) = \sum_{j=1}^{n-1} d(w_{\sigma(j)}^{\delta(j)}, w_{\sigma(j+1)}^{\delta(j+1)}) \quad (3.5)$$

The crossover operators used by the evolutionary algorithm and lead to the quality results are the classical *Partially-Mapped Crossover* – PMX [Gol85], *Ordered Crossover* – OX [Dav85] and two derived ones proposed by the author: *Cycle PMX* (CPMX) [Log06a] and *Cycle OX* (COX) [Log06a]. They are alike with the 2-opt move, with the difference that the sequence is inversed between the cut points. As operators, there are used the operators SIM and SCIM. The parameter settings are chosen by experiments. Because the genetic algorithm is applied for different data, from very small to very large, it becomes necessary to adapt these parameters.

3.6 The distributed parallel algorithm MRPGA for DOP and DOPI

It is proposed a parallel distributed algorithm MRPGA_DOPI (*MapReduce Parallel Genetic Algorithm*) [Log10c] which use the MapReduce technology, the distributed programming model introduced by Google and presented in the previous chapter. We use MapReduce for applying the roulette-wheel method in parallel for more different sets of individuals. Every individual is a pair (permutation, assignation) related to the input data set. The number of the sets of individuals (populations) is given by the total number of individuals divided by the given factor of reduction (reductionFactor). This number represents also the number of *Reduce*-tasks ran by Hadoop in parallel. In the *Map* function, for every sequence will be randomly chosen the set of individuals where the selection is done. In the *Reduce* function, the input key represents the number of the selection set and the input values are the pairs permutation/assignation which constitute the set. There is applied a selection with the roulette-wheel selection and the individuals from the next generation are written as output. After the job termination, the individuals from the new generation are read from their files wrote by the reduction nodes and follow a new ordering step. The implementation MRPGA_DOPI is a Hadoop application wrote in Java. The binary version and the dependent libraries are archived in the file *dopiSolver.jar*. The driver attached with the archive records the application Hadoop with the name *dopi*.

ALGORITHM_MRPGA_DOPI

```

Initialize(populationSize)
Initialize(crossoverRate)
Initialize(mutationRate)
Initialize(reductionFactor)
Initialise_GreedyMin_individuals()
Initialise_GreedyI_individuals()
Initialise_Random_individuals()
def MRPGA_MAP( initialIndex,pair<permutation, assignment>) = {
    reductionSetIndex = random(0.. reductonFactor-1)
    context.write(reductionSetIndex, pair<permutation, assignment>)
}
def MRPGA_REDUCE
    (reductionSetIndex,Iterable< pair<permutation, assignment>>)= {
    Apply_Crossover_operators(numCrossovers);
    Apply_Mutation_operators(numMutatios);
    Calculate_fitness(allNewIndividuals);
    Remove_WorstInviduals (numCrossovers+numMutations);
    for (pair<new_perm, new_assign> : allNewIndividuals)
        context.write
            (reductionSetIndex,pair<new_perm, new_assign>)
    }
for ( i ← 1; i ≤ numGenerations; step 1) execute
    job← NewJob(MRPGA_MAP, MRPGA_REDUCE)
    job.configuration.set(populationHDSFPath)
    job.configuration.set_crossoverRate)
    job.configuration.set_mutationRate)
    job.configuration.setNumReduceTasks(reductonFactor)
    job.submit_and_wait
    collect_new_population
end_for

```

END_ALGORITHM_MRPGA_DOPI**Fig. 3.1.** Pseudo code for MRPGA_DOPI.

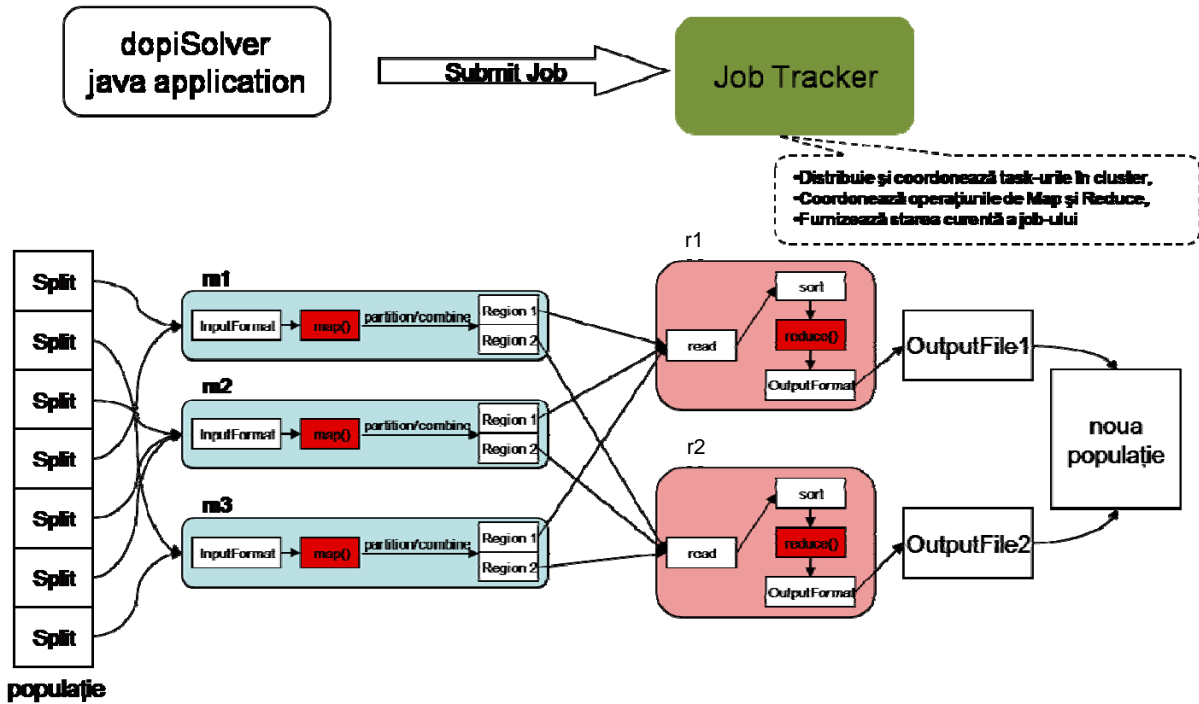


Fig. 3.2. Implementation MRPGA_DOPI with Java and Hadoop.

3.7 Numerical experiments and statistical tests

The numerical experiments are concentrated on DOPI variant, which leads to better results than the DOP one (because the freedom grade in choosing the assignment). This freedom paradigm increases considerably the complexity of the problem. The test program initializes random instances with a uniform distribution for every given pair (n, k) : n sequences, every of length k . For these instances are applied the presented algorithms.

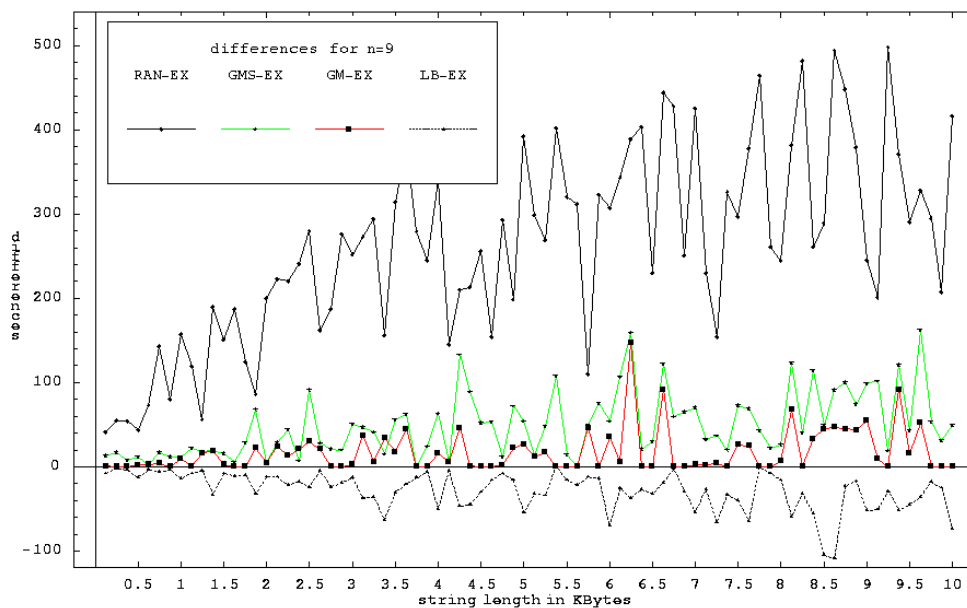
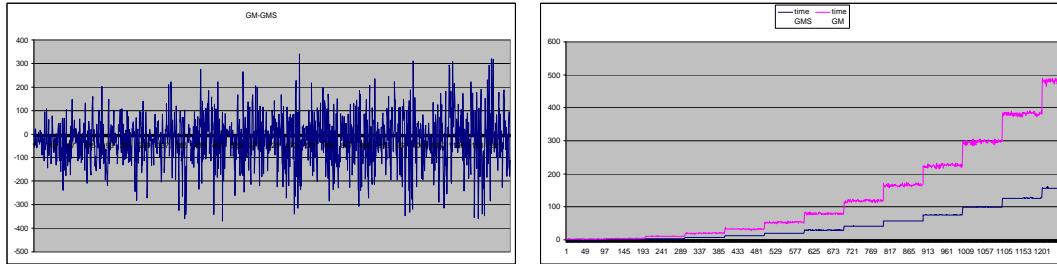


Fig. 3.3. Values RAN-EX, GMS-EX, GM-EX and LB-EX for the parameters $n=9, k=500, 1000, \dots, 10000$.



3.4. Values GM–GMS (left) and execution times (sec., right) for 1000 experiments, $n = 1000, 1500..7000, k = 50, 100..1000$. The positive values left represent more efficient results for GMS. The execution time for GMS is considerably smaller than the one for GM.

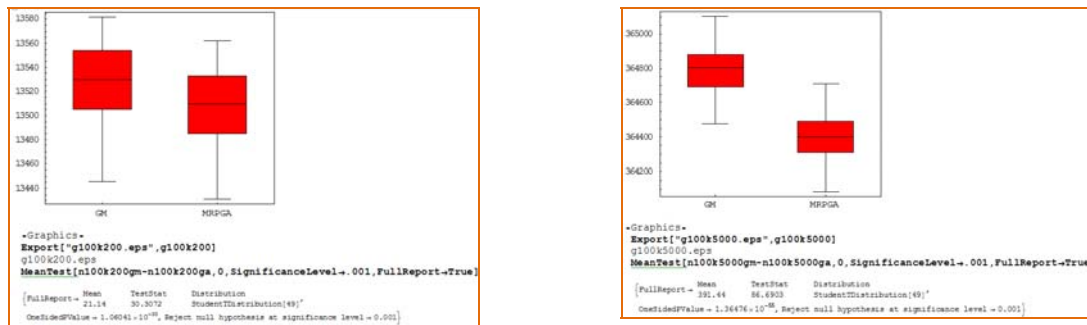


Fig. 3.5. Medians of all values GM_DOPI and MRPGA_DOPI and statistical student-t tests which certify the superiority of the method MRPGA_DOPI against GM_DOPI with a probability 0.999. Used parameters: 50 runs, $n=100, k=200$ (left), 5000 (right).

The thesis contains numerous experiments and statistical tests for comparing the proposed algorithms in pairs, the quality of their results, as well as the execution time.

3.8 Conclusions and future work

Beside other algorithms, we presented here two greedy approaches and three evolutionary ones. The algorithms random, optimal and lower bound helped on framing the quality of the results. One of the greedy approaches (GMS) is new and it proves to be very efficient, especially for large data inputs. The evolutionary approaches are recommended when the focus is the quality of the results and the time/space resources are irrelevant. One of the future work proposals could be the profitable combination of these techniques and testing them on extended experiments scenarios, many-sided also from the distribution of the input data point of view. The evolutionary algorithm MRPGA_DOPI is the best with respect of the quality of the results. Due the use of the MapReduce technology, it can be used also with large data sets, which is a real advantage for the applications in the industrial field.

Data Compaction Problem

In this chapter the author proposes a set of original algorithms for solving the *Test Compaction Problem* (TCP). This important problem is NP-hard and refers the activity of testing digital circuits, playing an important role in their design. There are proposed two greedy algorithms *Greedy Naive Algorithm* (GRNV_TCP) and *Greedy Binary Tree Algorithm* (GRBT_TCP) [Log08a, Log08e], an efficient evolutionary algorithm *Genetic Algorithm Test Compaction* (GA_TCP) [Log08b, Log09c] and a distributed evolutionary algorithm based on the new technology *MapReduce – MapReduce Parallel Evolutionary Algorithm* (MRPEA_TCP) [Log10b]. Additionally, there is also offered a scientific project Open Source available on: <http://dcpsolver.sourceforge.net/>.

4.1 Problem Description TCP

Definition 4.1. *Compaction Set–CS.* Every test is seen as a string which contains characters from the set $CS = \{ '0', '1', 'U', 'Z', 'X' \}$. 'X' is the “Don't Care” characters.

Definition 4.2. *Compatible characters and merge-operation.* Two characters c_1 and c_2 are compatible if they are the same or if one of them is 'X'. We will denote this relation with ' \equiv ' and its negation with ' $\not\equiv$ '.

Definition 4.3. *Compatible tests and merge operation.* Two given tests are compatible if they have the same length and all corresponding characters (the same position) in pairs are compatible. The merged test is obtained by substituting every character sequentially with the merged character from the corresponding position in the two given strings. We will denote this relation also with ' \equiv ' and its negation with ' $\not\equiv$ '.

Definition 4.4. Coverage Set. For two given sets of tests $S_1 = \{t_{11}, t_{12}, \dots, t_{1i}\}$ and $S_2 = \{t_{21}, t_{22}, \dots, t_{2j}\}$, S_2 is a coverage set of S_1 if for every test t in S_1 there is a compatible one in S_2 .

Definition 4.5. Test Compaction Problem (TCP): Given a set of tests $S_1 = \{t_{11}, t_{12}, \dots, t_{1i}\}$, every test t_{1k} , for $k=1, \dots, i$, has the same length. Find a coverage set $S_2 = \{t_{21}, t_{22}, \dots, t_{2j}\}$ of S_1 , such that the cardinality of S_2 (number of elements) is minimal over all coverage sets of S_1 .

The DNA sequence compaction problem is the same as the one of VLSI circuit's testing; only the characters' alphabet is different. We will name the general problem as the *Data Compaction Problem* (DCP): Given n sequences of length k , each of them containing symbols from a given alphabet \mathcal{A} and a symbol 'X' meaning „Don't Care”, you need to find a set of compacted sequences that covers the initial one.

4.2 Optimal algorithm for TCP

The optimal algorithm uses backtracking. The merge operation is applied on pairs of compatible tests and the algorithm is called recursively over all obtained vectors. The boolean variable *compact* verifies if there are compatible pairs, otherwise remains set on *true*. Due to its exponential complexity, this algorithm can be applied in practice only for small dimensions of the problem.

4.3 Greedy algorithms GRNV and GRBT for TCP

The author proposed GRNV and GRBT in [Log08a, Log08e, Log09b]. The transformation is done over the set of tests and operates only with the processed set. On each step, the first two compatible tests are substituted with the result of their fusion. If there are no more tests found, then current set of tests constitutes the result. The complexity is polynomial $O(n^3)$. An important factor in the execution of the greedy algorithm is the choice of the two compatible tests for fusion. This can influence significantly the quality of the results. There are two possibilities. For the first one the tests are stored in a one-dimensional vector structure and the algorithm fuses the first two compatible tests found. We name this algorithm GRNV – **G**reedy **N**aive. For the second case it is used a binary search tree where the tests are descending ordered after their number of 'X' symbols. We name this algorithm GRBT - **G**reedy **B**inary **S**earch **T**ree.

4.4 Evolutionary algorithm GATC for TCP

This algorithm is based on the classical scheme of the genetic algorithms, with the difference that the initial population contains clones of the start sequence. The mutations are applied on the current population and the best individuals are kept for the next iteration. After a certain number of iterations, we apply for every individual obtained the greedy algorithm GRBT previous described for obtaining the covering set of compacted tests. The particularity of the algorithm is the initialization phase using clones of the sequences and the final part that uses the greedy method. A candidate solution is a covering for the start sequence. This implies a set of tests which at least a compatible test for each test from the starting set of tests. A population is a set of such candidate solutions. The algorithm applies successively the mutation operators on the population's individuals, replacing two random compatible tests with the result of their fusion. For the selection function it is used the total number of X symbols:

$$N_X(t_1, t_2, \dots, t_n) = \sum_{i=1}^n \#X(test_i), \quad (4.1)$$

where $\#X()$ is a function that returns the number of 'X' characters from the input test.

4.5 Distributed evolutionary algorithm MRPEA for TCP

The proposed parallel evolutionary algorithm is based on GATC [7] and uses Hadoop. This is the OpenSource MapReduce [Dea08] framework implementation from Apache. A MapReduce job splits the input data into independent chunks (splits) which are then processed by the map tasks in a completely parallel manner. The framework sorts the maps' outputs and forward them as input to the reduce tasks.

The parallel evolutionary algorithm runs for a given number of generations. For each generation, it generates a random number of permutations of the "current" set of sequences, splits the set of permutations in several subsets (map operation) and executes GATC [Log08b, Log09c] on each of them during the reduce operation. Afterwards, it collects the best individuals (permutations) resulted on each Reducer node in the cluster and concatenate them. The result constitutes the new set of sequences for the next generation. A MapReduce *job* splits the input data into independent chunks (splits), which are then processed by the *map tasks* in a completely parallel manner. The framework sorts the *maps'* outputs and forwards them as input to the *reduce tasks*. The *map* operation associates for each generated permutation a subset index between 0 and $reductionFactor-1$ randomly, as output key. This index represents the partition for reduction. In the *Shuffle & Sort* phase, the sequences get sorted after their indices, and each *reducer* node receives and executes GATC in one call over its subset of sequences having the same index. The Hadoop application for DCP is available as the *Open Source* project "dcpSolver" on *sourceforge.net*. The Java binary application and its dependent libraries

are archived in “dcpSolver.jar”. The archive's entry point (*dcpSolver.DcpSolverDriver*) is a Hadoop program driver that registers the DcpSolver's application (*dcpSolver.DcpSolverJob*). The start command is "hadoop jar dcpSolver.jar", and it gets the name of the DcpSolver's application: *dcp*.

ALGORITHM_MRPEA_DCP

```

initialize(currentSequenceSet)
initialize(mutationRate)
initialize(reductionFactor)
for ( $i \leftarrow 1; i \leq \text{numGenerations}; \text{step } 1$ )
    population  $\leftarrow$  GenerateRandomPermutations(currentSequenceSet)
    def MRPGA_TCP_MAP(seqIndex, sequence) =
        context.write(random(0.. reductionFactor-1), sequence)
    def MRPGA_TCP_REDUCE(subsetIndex, Iterable<sequence>)= {
        // run GATC on subset
        numMutations  $\leftarrow$  populationSize*mutationRate
        apply_Mutation_Operators(numMutations);
        calculate_Fitness(allNewIndividuals);
        remove_Worst_Individuals (populationSize/2);
        complete_With_Copy_Individuals(populationSize/2)
        context.write(subsetIndex, best_element(individuals))
    }
    job $\leftarrow$  NewJob(MRPGA_TCP_MAP, MRPGA_TCP_REDUCE)
    job.configuration.setNumReduceTasks (reductionFactor)
    job.submit_and_wait
    currentSequenceSet = concatenate_reducers_parts(job)
endfor
return currentSequenceSet

```

END_ ALGORITHM_MRPEA_DCP

4.6 Experimental results and statistical tests

Regarding the proposed algorithms for TCP there are offered many-sided experiments, which compare their efficiency in pairs. For example, in over 600 tested cases, with n between 5 and 30 and k between 20 and 40, only in 10 cases had OPT better results as GRNV.

Other experiment compares the algorithms GRNV and GRBT, $100 \leq n \leq 1100$, $25 \leq k \leq 1025$ and different compaction factors (13, 25, 38, 50, 62, 75, 88).

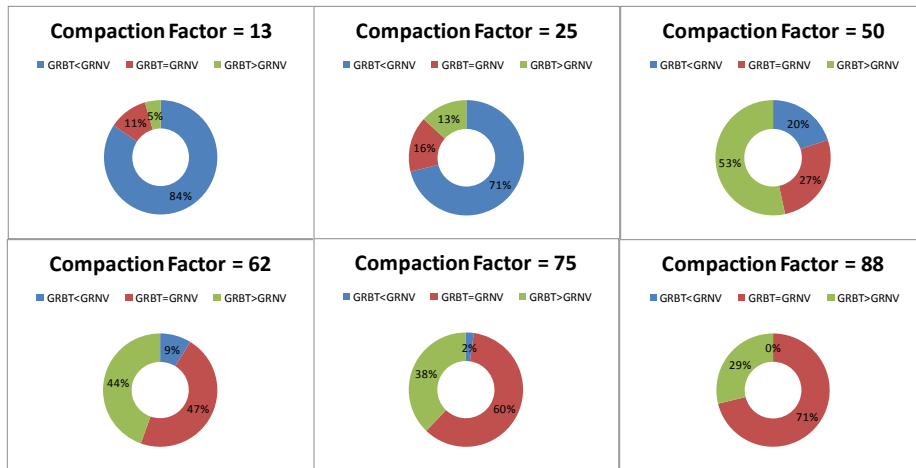


Fig. 4.1. Case numbers GRBT<GRNV, GRBT=GRNV and GRBT>GRNV with $100 \leq n \leq 1100$, $25 \leq k \leq 1025$, compaction factor = 13, 25, 50, 62, 75 and 88 – 1000 runs for every compaction factor.

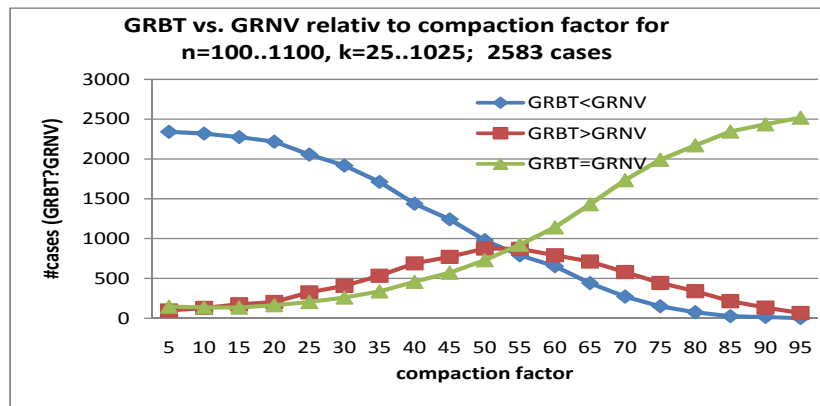


Fig. 4.2. Evolution of the difference GRBT–GRNV among 2583×19 experiments with $100 \leq n \leq 1100$, $25 \leq k \leq 1025$, compaction factor = 5, 10, 15... 95.

The previous 49077 experiments were globally statistic evaluated: difference (GRBT–GRNV) mean is -0.79 , execution time (time_GRBT–time_GRNV) mean is -0.52 . These results confirm the efficiency of the GRBT from both points of view, quality of results and execution time.

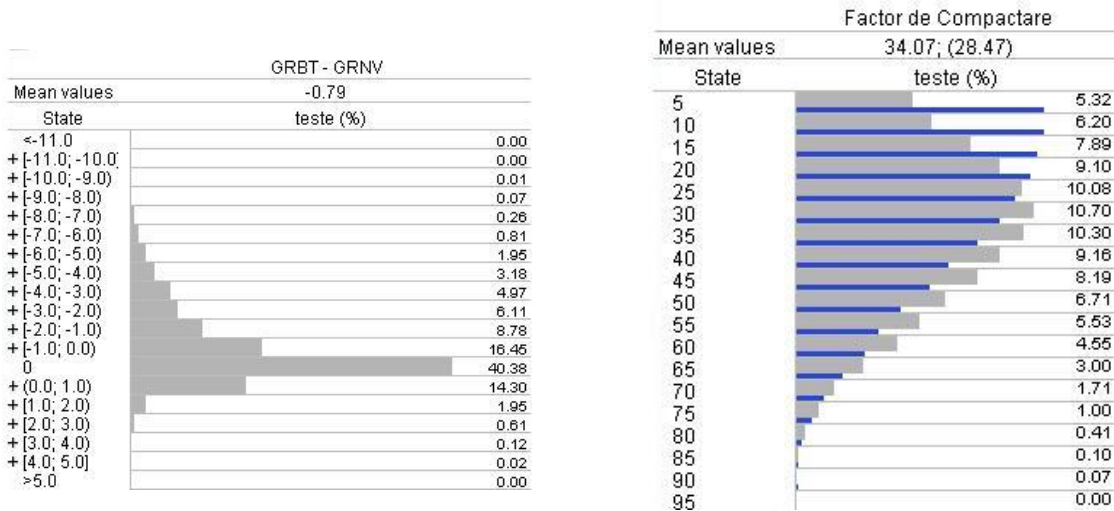


Fig. 4.3. Left: the distribution on intervals of difference GRBT–GRNV among 49077 experiments with $100 \leq n \leq 1100$, $25 \leq k \leq 1025$, compaction factor = 5, 10, 15... 95. Right: Distribution of the time difference among the 9915 tests for that GRBT<GRNV and the execution time GRBT is smaller as the GRNV one, the reference are all tests where GRBT is better as GRNV.

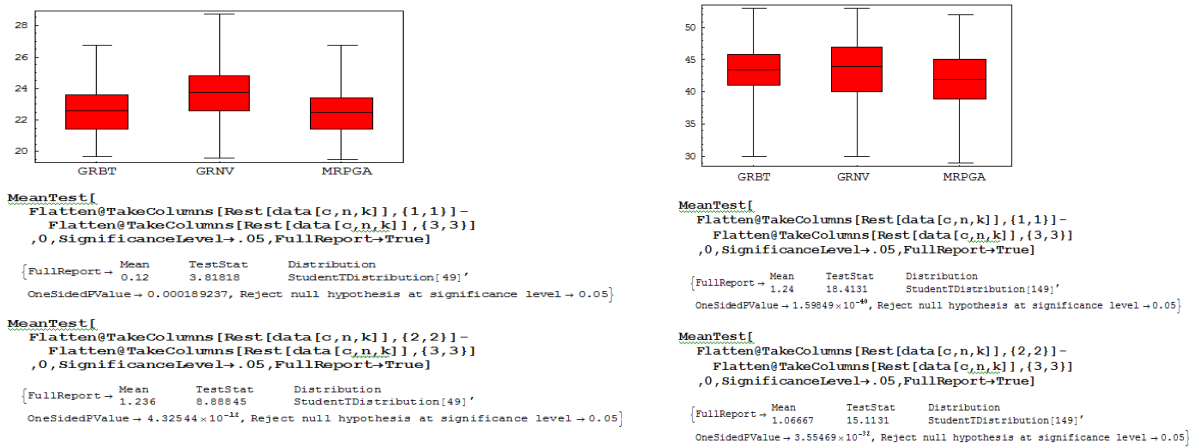


Fig. 4.4. Medians and student-t statistical tests confirming the superiority of the method MRPEA for TCP against GRBT and GRNV with a probability of 0.95. Left: 50 runs, $cf=20$, $n=1000$, $k=2000$. Right: 150 runs, $cf=40$, $n=100$, $k=200$.

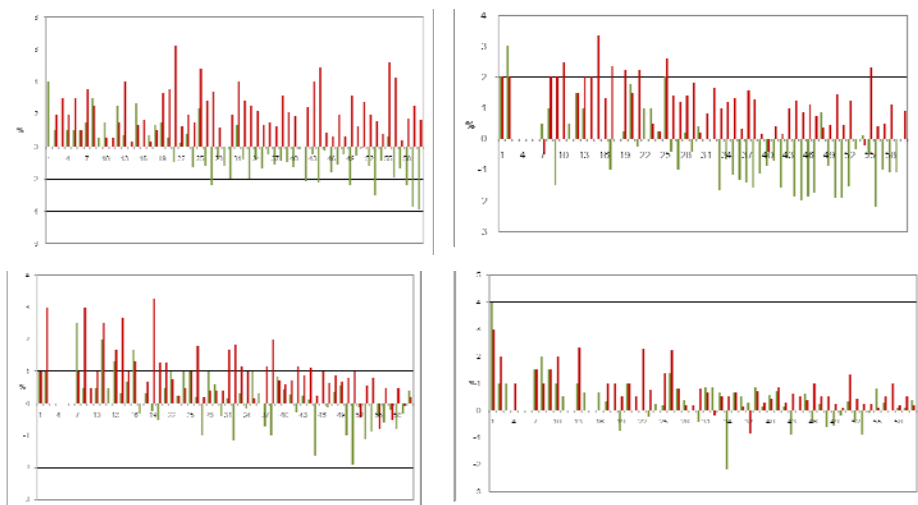


Fig. 4.5. Experimental results for sequence sets with $n = 100$ to 1000 , step 100 , $k = 50$ to 550 , step 100 , $pf=3$, (a) $cr=30$, (b) $cr=40$, (c) $cr=50$, (d) $cr=60$. Green (GRBT–MRPEA), red (GRNV–MRPEA).

4.7 Conclusions and future work

After a formal description of TCP, it is presented an optimal solution based on the backtracking method. Its complexity is exponential. Therefore it can be used only on very small data sets. Additionally, there are presented two greedy methods and an evolutionary one, applied on representative data sets. The experiments show the very good quality of results provided by the proposed methods, but also the differences (quality, execution time) between them related to different parameters for the input data and the compaction factor. The term of compaction factor is introduced here and proves to be a primal element in designing and choosing the algorithm for a specific data set. In this chapter there are proposed also original algorithms for solving the test compaction problem (TCP). This important problem is NP-hard. It refers to the testing activity for the integrated circuits as plays a base role in designing them. The proposed algorithms are two greedy ones, *Greedy Naive Algorithm* (GRNV_TCP) and *Greedy Binary Tree Algorithm* (GRBT_TCP) [Log08a, Log08e], an efficient evolutionary one *Genetic Algorithm Test Compaction* (GA_TCP) [Log08b, Log09c] and a distributed evolutionary algorithm using the MapReduce technology, *MapReduce Parallel Evolutionary Algorithm* (MRPEA_TCP) [Log10b]. As well, it is offered a research open source project available at:

<http://dcpsolver.sourceforge.net/>.

It comprises the implementations of the presented algorithms and data sets for testing; it can be used for research purposes. The experiments show the data structure of binary tree is more efficient for the greedy algorithm and a smaller than 50% compaction rate. The behavior of the proposed approaches varies right with different parameters: number of tests, test length, compaction rate. The quality of the provided results by the greedy algorithms could be improved by adapting the data structure to the different parameters. Improvements could be done also by the implementation methods. Other direction could be the classification of the sequences with more *Don't Cares* on the same positions (or other criteria) and the division of the input data in independent sets, which are solved with different algorithms, then the combination step between these results. The presented algorithms could be refined by adding some sophisticated techniques, like using Tabu-lists or improving the search tree with an A* algorithm. Finding an efficient lower bound algorithm can enriched the future research in the field. The transformation of a problem instance in another known problem instance, for example SAT, and solving it with a specific solver could bring another useful insights.

Bibliography

- [Adl94] Adleman, L. M.: Molecular Computation of Solutions to Combinatorial Problems, *Science*, vol. 266, pp. 1021-1024, 1994.
- [Adl83] Adleman, L.M., Promerance, C., Rumely, R.S.: On distinguishing prime numbers from composite numbers, *Annals of Mathematics* 117 (1983), pp. 173-206, 1983.
- [Agr04] Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P , *Annals of Mathematics* 160 (2004), pp. 781-793, 2004.
- [Alo06] Alon, N., Moshkovitz, D., Safra, M.: Algorithmic construction of sets for k -restrictions, *ACM Transactions on Algorithms (TALG)*, v. 2, n. 2, pp. 153-177, 2006.
- [App03] Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for large traveling salesman problem, *INFORMS Journal of Computing* 15 (2003), pp. 82-92, 2003.
- [App07] Applegate, D.L., Bixby, R., Chvátal, V., Cook, W.J.: *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2007.
- [Aro98] Arora, S.: Polynomial Time Approximation Schemes for Euclidian Travelling Salesman and Other Geometric Problems, *Journal of the ACM*, Vol. 45, No. 5, pp. 753-782, 1998.
- [Bak87] Baker, J.E.: Reducing bias and inefficiency in the selection algorithm, *Proceedings of the International Conference on Genetic Algorithms*, vol. 2, pp. 14-21, 1987.
- [Boo87] Booker, L.: Improving Search in Genetic Algorithms, *Genetic algorithms and simulated annealing*, Davis, L. (ed.), Morgan Kaufmann Publishers, pp. 61-73, 1987.
- [Bli95] Blicke, T., Thiele, L.: A Comparison of Selection Schemes used in Genetic Algorithms, 2. Ed., *TIK Report* No. 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zürich, 1995.
- [Car06] Cardei, M., Wu, J.: Energy-efficient coverage problems in wireless ad-hoc sensor networks, *Computer Communications*, v. 29, n. 4, pp. 415-420, 2006.
- [Cha92] Chandrakasan, A. P., Potkonjak, M., Rabaey, J., Brodersen, R. W.: HYPER-LP: a system for power minimization using architectural transformations, *Int'l Conf on CAD*, pp. 300-303, 1992.
- [Chr76] Christofides, N.: Worst-case analysis of a new heuristic for the traveling salesman problem, *Technical Report* 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, 1976.
- [Coo71] Cook, S. A., The complexity of theorem-proving procedures, *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151-158, Shaker Heights, Ohio, USA, 1971.

- [Cor01] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.: *Introduction to Algorithms*, 3rd Edition, MIT Press, Boston, 2003.
- [Dav85] Davis, L.: Applying adaptive algorithms to epistatic domains, *Proceedings of IJCAI*, pp. 162-164, 1985.
- [Dav91] Davis, L.: *Handbook of Genetic Algorithms*, van Nostrand Reinhold, New York, 1991.
- [Dav60] Davis, M., Putnam, H., A Computing Procedure for Quantification Theory, *Journal of the ACM* 7 (1), pp. 201-215, 1960.
- [Dav62] Davis, M., Logemann, G., Loveland, D.: A Machine Program for Theorem Proving, *Communications of ACM* 5(7), pp. 394-397, 1962.
- [Dea08] Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters, *Commun. ACM*, 51(1), pp. 107-113, 2008.
- [Dev95] Devadas, S., Malik, S.: A survey of optimization techniques targeting low power VLSI circuits, *Design Automation Conf.*, pp. 242-247, 1995.
- [Dor96] Dorigo, M., Gambardella, L.M.: *Ant Colonies for the Traveling Salesman Problem*, Universitet Libre de Bruxelles, Belgium, 1996.
- [Dre02] Drechsler, R., Drechsler, N.: *Evolutionary Algorithms for Embedded System Design*, Kluwer Academic Publisher, 2002.
- [Dre03] Drechsler, R., Drechsler, N.: Minimization of Transitions by Complementation and Resequencing using Evolutionary Algorithms, In *Proceedings of 21st IASTED International Multi-Conference Applied Informatics (AI 2003)*, IASTED International Conference on Artificial Intelligence and Applications (AIA 2003), Innsbruck, 2003.
- [Dre97] Drechsler, R., Göckel, N.: A genetic algorithm for data sequencing. *Electronic Letters*, 33(10), pp. 843-845, 1997.
- [Dre98] Drechsler, R.: *Evolutionary Algorithms for VLSI CAD*, Kluwer Academic Publisher, 1998.
- [Dre99] Drechsler, N., Drechsler, R.: Exploiting don't cares during data sequencing using genetic algorithms. In *ASP Design Automation Conf.*, pp. 303-306, 1999.
- [Dum00] Dumitrescu, D., Lazzerrini, B., Jain, L.C.: *Evolutionary Computation*, CRC Press, 2000.
- [Dum06] Dumitrescu, D.: *Algoritmi genetici și strategii evolutive - aplicații în Inteligența Artificială și în domenii conexe*, reeditare, Editura Albastră, 2006.
- [Edm65] Edmonds, J.: Minimum partition of a matroid into independent subsets, *Journal of Research of the National Bureau of Standards* B 69 (1965), pp. 67-72, 1965.
- [Eka08] Ekanayake, J., Pallickara, S., Fox, G.: Mapreduce for data intensive scientific analyses, *ESCIENCE '08: Proceedings of the 2008 Fourth IEEE International Conference on eScience*, IEEE Computer Society, pp. 277-284, Washington, DC, USA, 2008.

- [Fei98] Feige, U.: A Threshold of $\ln N$ for Approximating Set Cover, *Journal of the ACM (JACM)*, v. 45, n. 4, pp. 634-652, 1998.
- [Fed95] Fredman, M. L., Johnson, D. S., McGeoch, L. A., Ostheimer, G.: Data Structures for Travelling Salesmen, *Journal of Algorithms* 18, pp. 432-479, 1995.
- [Fog66] Fogel, L.J., Owens, A.J., Walsh, M.J., *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.
- [Gar79] Garey, M. R., Johnson, D. S.: *Computers and Intractability – A Guide to NP-Completeness*, Freeman, San Francisco, 1979.
- [Gar97] Garzon, M., Deaton, R., Neathery, P., Murphy, R. C., Stevens Jr., S. E., Franceschetti, D. R.: A new metric for DNA computing, *Genetic Programming 1997, Proceedings of the Second Annual Conference*, Stanford University, pp. 479-490, AAAI, 1997.
- [Gen00] Gen, M., Cheng, R.: *Genetic Algorithms and Engineering Optimization*, John Wiley Sons Inc., USA, 2000.
- [Ghe03] Ghemawat, S., Gobioff, H., Leung, S., -T: The google file system, *SIGOPS Oper. Syst. Rev.*, 37(5), pp. 29-43, 2003.
- [Gol85] Goldberg, D. E., Lingle, R.: Alleles, loci, and the travelling salesman problem, *Int'l Conference on Genetic Algorithms*, pp. 154-159, 1985.
- [Guo01] Guo, R., Pomeranz, I., Reddy, S. M., On improving static test compaction for sequential circuits, *VLSI Design*, Fourteenth International Conference, pp. 111-116, 2001.
- [Hel70] Held, M., Karp, R. M.: The traveling-salesman problem and minimum spanning trees, *Operations Research* 18, pp. 1138-1162, 1970.
- [Hel71] Held, M., Karp, R.M.: The travelling-salesman problem and minimum spanning trees, part II. *Mathematical Programming* 1, pp. 6-25, 1971.
- [Hel98] Helsgaun, K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic/Roskilde University, *Writings on Computer Science*, 1998.
- [Hel00] Helsgaun, K.: An effective implementation of the Lin-Kernighan Traveling Salesman Heuristic, *European Journal of Operation Research* 126, pp. 106-130, 2000.
- [Hel06] Helsgaun, K.: An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic / Roskilde University, *Writings on Computer Science*, 2006.
- [Hig06] Higami, Y., Kajihara, S., Pomeranz, I., Kobayashi, S., Takamatsu, Y.: On Finding Don't Cares in Test Sequences for Sequential Circuits, *IEICE Transactions on Information and Systems*, v. E89, n. 11, 2006.
- [Hoc98] Hochbaum, D., S., Pathria, A.: Analysis of the greedy approach in problems of maximum k -coverage, *Naval Research Logistics*, v. 45, n. 6, pp. 615-627, 1998.
- [Han95] Hansen, N., Ostermeier, A., Gawelczyk, A.: On the Adaptation of Arbitrary Mutation Distributions in Evolution Strategies: The Generating Set Adaptation, *Proceedings of*

- the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 57-64, 1995.
- [Hol75] Holland, J. H.: *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.
- [Hur04] Hurkens, C.A.J., Woeginger, G.J.: On the nearest neighbour rule for the traveling salesman problem, *Operations Research Letters* 32 (2004), pp. 1-4, 2004.
- [Ima94] Iman, S., Pedram, M.: Multilevel network optimization for low power, *Int'l Conf. On CAD*, pp. 372-377, 1994.
- [Jin08] Jin, C., Vecchiola, C., Buyya, R.: MRPGA: An extension of mapreduce for parallelizing genetic algorithms, eScience '08, Proceedings of the 2008 Fourth IEEE International Conference on eScience, pp. 214-221, 2008.
- [Joh95] Johnson, D. S., McGeoch, L. A.: *The Traveling Salesman Problem: A Case Study in Local Optimization*, 1995.
- [Joh96] Johnson, D. S., McGeoch, L. A., Rothberg, E. E.: Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound, *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms* (1996), pp. 341-350, 1996.
- [Kar72] Karp, M. R.: Reductibility Among Combinatorial Problems, *Complexity of Computer Computations* (Symposium Proceedings), Plenum Press, 1972.
- [Knu97] Donald E. Knuth, *The Art of Computer Programming, Volume I: Fundamental Algorithms*, Addison-Wesley Longman, Amsterdam; Ed. 3, 1997.
- [Kre99] Donald L. Kreher, Douglas R. Stinson, *Combinatorial algorithms. Generation, Enumeration, and Search*, CRC Press, 1999.
- [Lad75] Ladner, R.E.: On the structure of polynomial time reducibility, *Journal of the ACM* 22 (1975), pp. 155-171, 1975.
- [Lan60] Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems, *Econometrica* 28 (1960), pp. 497-520, 1960.
- [Lin93] Lin, S., Kernighan, B. W.: An Effective Heuristic Algorithm for the Travelling-Salesman Problem, In *Operations Research* 21, pp. 498-516, 1973.
- [Lit63] Little, J.D.C., Murty, K.G., Sweeny, D.W., Karel, C.: An algorithm for the traveling salesman problem, *Operations Research* 11 (1963), pp. 972-989, 1963.
- [Llo10] Llorca, X, Verma, A., Campbell, R. H., Goldberg, D., E.: When Huge is Routine: Scaling Genetic Algorithms and Estimation on Distribution Algorithms via Data-Intensive Computing, *Parallel and Distributed Computational Intelligence*, pp. 11-41, Springer, 2010.
- [Log05] **Logofătu, D.**: Programare dinamică: sume de puteri , *GInfo* 15/2, pp. 40-43, 2005.

- [Log06a] **Logofătu, D.**, Drechsler, R.: Efficient Evolutionary Approaches for the Data Ordering Problem with Inversion, *3rd European Workshop on Hardware Optimization Techniques (EvoHOT)*, LNCS 3907, pp. 320-331, Springer, Berlin/Heidelberg, 2006.
- [Log06b] **Logofătu, D.**: *Algorithmen und Problemlösungen mit C++*, pp. 388-397, Vieweg-Verlag, 2006.
- [Log06c] **Logofătu, D.**: Greedy Approaches for the Data Ordering Problem with Inversion, *Proceedings of ROSYCS - Romanian Symposium on Computer Science*, pp. 65-80, Iași, 2006.
- [Log06d] **Logofătu, D.**: Problema ordonării datelor cu și fără inversiune, *GInfo* 16/2, pp. 8-14, 2006.
- [Log07a] **Logofătu, D.**: *Algoritmi fundamentali in C++. Aplicații*, pp. 127-154:265-273, Editura Polirom, Iași, 2007.
- [Log07b] **Logofătu, D.**: *Algoritmi fundamentali in Java. Aplicații*, pp. 125-158:269-277, Editura Polirom, Iași, 2007.
- [Log07c] **Logofătu, D.**: *Grundlegende Algorithmen mit Java*, pp. 65-98:205-214, Vieweg-Verlag, Germany, 2008.
- [Log08a] **Logofătu, D.**, Drechsler, R.: Comparative Study by Solving the Test Compaction Problem, *Proceedings 38th International Symposium on Multiple-Valued Logic (ISMVL '08)*, Dallas, USA, pp. 44-49, 2008.
- [Log08b] **Logofătu, D.**: Efficient Evolutionary Approach for the Test Compaction Problem, *Proceedings 9th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS*, pp. 144-148, Suceava, Romania, May 22-24, 2008.
- [Log08c] **Logofătu, D.**: *Eine praktische Einführung in C*, pp. 207-208, entwickler.press, München, Germania, 2008.
- [Log08d] **Logofătu, D.**, Gruber, M.: Efficient Approaches for DNA Sequences Ordering, *Proceedings Bio-Inspired Computational Methods Used for Difficult Problems Solving. Development of Intelligent and Complex Systems (BICS 2008)*, pp. 59-69, Târgu Mureș, România, 2008.
- [Log08e] **Logofătu, D.**: On the compaction of DNA Sequence Vectors, *Proceedings Bio-Inspired Computational Methods Used for Difficult Problems Solving. Development of Intelligent and Complex Systems (BICS 2008)*, pp. 25-36, Târgu Mureș, România, 2008.
- [Log09a] **Logofătu, D.**, Gruber, M.: DNA Sequences And Their Ordering, *American Institute of Physics*, Vol. 1117, pp. 3-11, 2009.
- [Log09b] **Logofătu, D.**: DNA Sequences And Their Compaction, *American Institute of Physics*,

- vol. 1117, pp. 29-39, 2009.
- [Log09c] **Logofătu, D.:** Static Test Compaction for VLSI Tests: an Evolutionary Approach, *Advances in Electrical and Computer Engineering*, pp. 49-53, 2009.
- [Log10a] **Logofătu, D.:** *Algorithmen und Problemlösungen mit C++*, pp. 402-411, Vieweg+Teubner-Verlag, 2010.
- [Log10b] **Logofătu, D., Dumitrescu, D.:** Parallel Evolutionary Approach of Compaction Problem Using MapReduce, 11th International Conference on Parallel Problem Solving from Nature, Cracovia, Polonia, 2010. (submitted)
- [Log10c] **Logofătu, D., Dumitrescu, D.:** Distributed Genetic Algorithm for Data Ordering Problem with Inversion Using MapReduce, 11th International Conference on Parallel Problem Solving from Nature, Cracovia, Polonia, 2010. (submitted)
- [Lou03] Lourenço, H. R., Martin, O. C., Stützle, T.: Iterated Local Search, In Glover, F. (Ed.), Kochenberger, G. (Ed.): *Handbook of Metaheuristics*, Kluwer Academic Publishers, pp. 321-353, 2003.
- [Lun94] Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems, *Journal of the ACM (JACM)*, v. 41 n. 5, pp. 960-981, 1994.
- [Mat10] Matthews, S. J., Williams, T. L.: MrsRF: an efficient MapReduce algorithm for analyzing large collections of evolutionary trees, *BMC Bioinformatics*, 11(1), doi: 10.1186/1471-2105-11-S1-S15, 2010.
- [Mal03] El-Maleh, A., Osais, Y.: Test vector decomposition based static compaction algorithms for combinatorial circuits, *ACM Trans. Des. Autom. Electron. Syst.*, vol. 8, pp. 430-459, 2003.
- [Maz98] Mazumder, P., Rudnick, E.: *Genetic Algorithms for VLSI Design, Layout & Test Automation*, Prentice Hall, 1998.
- [Mic94] De Micheli, G.: *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, Inc., 1994.
- [Mic96] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Ed. 3rd, Springer-Verlag, Berlin Heidelberg New York, 1996.
- [Mic07] Michiels, W., Aarts, E., Korst, J.: *Theoretical Aspects of Local Search*, Springer, Berlin, 2007.
- [Mil05] Milenkovic, O., Kashyap, N.: DNA Codes that Avoid Secondary Structures, *Proceedings of IEEE International Symposium on Information Theory*, Adelaide, Australia, 2005.
- [Mil06] Milenkovic, O., Kashyap, N.: On the Design of Codes for DNA Computing, *Coding and Cryptography (Lecture Notes in Computer Science 3969)*, pp. 100-119, Springer Verlag, Berlin-Heidelberg, Germany, 2006.

- [Mil04] Miltersen, P. B., MILP, ILP and TSP, Course Notes for Search and Optimization, <http://www.daimi.au.dk/dSoegOpt/ilp.pdf>, Spring, 2004.
- [Mur07] Murray, A., T., Kim, K., K., Davis, J., W., Machiraju, R., Parent, R.: Coverage optimization to support security monitoring, *Computers, Environment and Urban Systems*, vol. 31, n. 2, pp 133-147, 2007.
- [Mur97] Murgai, R., Fujita, M., Krishnan, S. C.: Data sequencing for minimum-transition transmission, *IFIP Int'l Conf. on VLSI*, 1997.
- [Mur98] Murgai, R., Fujita, M., Oliveira, A.: Using complementation and resequencing to minimize transitions, *Design Automation Conf.*, pp. 694-697, 1998.
- [Mur94] T. Murata, H. Ishibuchi, Performance evaluation of genetic algorithms for flow shop scheduling problems, *International Conference on Evolutionary Computation*, pp. 812–817, 1994.
- [Müh94] Mühlenbein, H.: The Breeder Genetic Algorithm - a provable optimal search algorithm and its application, *Colloquium on Applications of Genetic Algorithms*, IEEE 94/067, London, 1994.
- [Müh93] Mühlenbein, H., Schilerkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm, *Continuous Parameter Optimization. Evolutionary Computation*, 1 (1), pp. 25-49, 1993.
- [Oli87] Oliver, I. M., Smith, D. J., Holland, J. R. C.: A study of permutation crossover operators on the traveling salesman problem, *Int'l Conference on Genetic Algorithms*, pp. 224-230, 1987.
- [Ost93] Ostermeier, A., Gawelczyk, A., Hansen, N.: A Derandomized Approach to Self Adaptation of Evolution Strategies, *Technical Report TR-93-003*, TU Berlin, 1993.
- [Pap82] Papadimitriou, C.H., Steiglitz, K.: *Combinatorial Optimization; Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, cap. 17-19, 1998.
- [Pap06] Papadimitriou, C.H., Vempala, S.: On the approximability of the traveling salesman problem, *Combinatorica* 26 (2006), pp. 101-120, 2006.
- [Pra75] Pratt, V.: Every prime has a succinct certificate, *SIAM Journal on Computing* 4 (1975), pp. 214-220, 1975.
- [Rag07] Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating mapreduce for multi-core and multiprocessor systems, *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, 2007.
- [Rec73] Rechenberg, I.: *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Fromman-Holzboog, Stuttgart, 1973.
- [Rec73] Schwefel, H.-P.: *Numerical optimization of computer models*, Wiley&Sons, Chichester, 1981.

- [Ros77] Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M.: An analysis of several heuristics for the traveling salesman problem, *SIAM Journal on Computing* 6 (1977), pp. 563-581, 1977.
- [Rud96] Rudolph, G.: Convergence of Evolutionary Algorithms in General Search Spaces, *International Conference on Evolutionary Computation*, pp. 50-54, 1996.
- [Sas07] Sastry, K., Goldberg, D., E., Llorca, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm, *Proceedings of 9th annual conference on Genetic and evolutionary computation GECCO '07*, pp. 577-584, ACM, New York, 2007.
- [Sel96] Selman, B., Kautz, H., Cohen, B.: Local Search Strategies for Satisfiability Testing, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26, AMS, 1996.
- [Sel93] Selman, B., Kautz, A.: Domain-Independent Extension to GSAT: Solving Large Structured Satisfiability Problems, In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 290-295, 1993.
- [She95] Shen, W.-Z., Lin, J.-Y., Wang, F.-W.: Transistor reordering rules for power reduction in CMOS gates, *ASP Design Automation Conf.*, pp. 1-6, 1995.
- [Shm90] Shmoys, D.B., Williamson, D.P.: Analyzing the Held-Karp TSP bound: a monotonicity property with application, *Information Processing Letters* 35 (1990), pp. 281-285, 1990.
- [Sle95] Sleator, D. D., Tarjan, R. E.: Self-adjusting Binary Search Trees, *Journal of the Association for Computing Machinery*, Vol. 32, pp. 652-686, 1985.
- [Sta94] Stan, M., Burleson, W.: Limited-weight codes for low-power I/O, *Int'l Workshop on Low Power Design*, 1994.
- [Sys90] G. Syswerda, Schedule Optimization Using Genetic Algorithms, L. Davis, Editor, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1990.
- [Tiw94] Tiwari, V., Malik, S., Wolfe, A., Lee, M.: Power analysis of embedded software: A first step towards software power minimization, *Int'l Conf. on CAD*, pp. 384-390, 1994.
- [Tiw96] Tiwari, V., Malik, S., Wolfe, A., Lee, M.: Instruction level power analysis and optimization software, *VLSI Design Conf.*, 1996.
- [Tom81] Ioan Tomescu, *Probleme de combinatorică și teoria grafurilor*, Editura Didactică și Pedagogică, București, 1981.
- [Tsu93] Tsui, C., Pedram, M., Despain, A. M.: Technology decomposition and mapping targeting low power dissipation, *Design Automation Conf.*, pp. 68-73, 1993.
- [Tur36] Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society* (2) 42 (1936), pp. 230-265 și 43 (1937), pp. 544-546, 1936-1937.
- [Vai93] Vaishnav, H., Pedram, M.: PCUBE: A performance driven placement algorithm for low power design, *European Design Automation Conf.*, pp. 72-77, 1993.

- [Whi89] Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesman: The genetic edge recombination operator, *Int'l Conference on Genetic Algorithms*, pp. 133-140, 1989.
- [Wol80] Wolsey, L.A.: Heuristic analysis, linear programming and branch and bound, *Mathematical Programming Study* 13, pp. 121-134, 1980.

Resurse Web:

<http://scipy.org/scipy/scikits/wiki/MILP>

[Mixed-Integer Linear Problems - MILP]

<http://javailp.sourceforge.net/>

[Java ILP – Java Interface to ILP solvers]

<http://dopisolver.sourceforge.net/>

[dopiSolver – *framework Open Source* pentru rezolvarea DOPI]

<http://dcpsolver.sourceforge.net/>

[dcpSolver – *framework Open Source* pentru rezolvarea DCP]

<http://hadoop.apache.org/>

[Apache Hadoop]

<http://lsiwww.epfl.ch/LSI2001/teaching/webcourse/ch02/ch02.html#2.5>

<ftp://ftp.cs.cmu.edu/user/sleator/splaying/>

[implementări C și Java pentru *splay trees*]

