

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

Some other issues in discrete optimization

Resume of the PhD Thesis

Scientific advisor:
Professor Militon Frențiu, Ph.D.

Ph.D. Student:
Alexandra Băicoianu

Cluj-Napoca
2016

Contents

1	Introduction	20
2	Rectangular two dimensional cutting stock problems	28
2.1	Basic notions concerning cutting and covering problem	28
2.2	The generating cutting-covering solutions using Euclid's algorithm . . .	31
2.2.1	Problem statement and formulation	31
2.2.2	Problem solving steps	31
2.3	The determination of the guillotine restrictions for a rectangular covering model	38
2.3.1	Problem statement and formulation	38
2.3.2	Cuts determination	42
2.3.3	The algorithm for the verification of the guillotine restrictions . .	44
2.4	The determination of the guillotine restrictions for a rectangular cutting-stock pattern	57
2.4.1	Problem statement and formulation	57
2.4.2	Cuts determination	60
2.4.3	The algorithm for the verification of the guillotine restrictions . .	62
2.5	Contributions and results	71
3	Three dimensional bin packing problems	73
3.1	Basic notions concerning three dimensional bin packing problems . . .	74
3.2	A topological order for a rectangular three dimensional bin packing problem	75
3.2.1	Problem statement and formulation	76
3.2.2	The compound graph for the packing problem	82
3.2.3	Topological sorting algorithm	87
3.3	The determination of the guillotine restrictions for a rectangular three dimensional bin packing pattern	90
3.3.1	Problem statement and formulation	90
3.3.2	Cuts determination	93
3.3.3	Verification test for guillotine restrictions	96

3.4 Contributions and results	108
4 Optimization with Logical Analysis of Data	109
4.1 Theoretical aspects	112
4.1.1 Logical Analysis of Data	112
4.1.2 Classification algorithms	123
4.2 Computational experiments	127
4.2.1 Datasets - Examples and comparisons	127
4.2.2 Experiments	130
4.2.3 Results and analysis	136
4.3 Summary and future work	139
5 Conclusions	141
Appendices	146

List of publications

- **A. Băicoianu**: A Comparative Study of Some Classification Algorithms using WEKA and LAD Algorithm, Annals of the Tiberiu Popoviciu Seminar of Functional Equations, Approximation and Convexity, ISSN 1584-4536, Volume 12, 2014, pp. 7381.
- **A. Băicoianu**, A. Vasilescu, R. Pândaru: Upon the performance of a Haskell parallel implementation, Bulletin of the Transilvania University of Braşov, **2013**, Series III: Mathematics, Informatics, Physics, pp. 61-72.
- (textbook) **A. Băicoianu**, I. Plajer: Formal Languages and Automata - practice, Transilvania University Press, Braşov, Romania, 2012.
- D. Marinescu, **A. Băicoianu**, S. Dimitriu: Software for Plagiarism Detection in Computer Source Code, Proc. of the 7th International Conference on Virtual Learning ICVL, 2012, pp. 373-379.
- D. Marinescu, **A. Băicoianu**, D. Simian: The determination of the guillotine restrictions for a Rectangular Three Dimensional Bin Packing Pattern, Recent Researches in Computer Science, Proc. of the 15-th WSEAS International Conference on Computers CSCC, Corfu, 2011, Volume 1, pp. 491-496.
- D. Marinescu, **A. Băicoianu**: A test of the guillotine restrictions determination for a Rectangular Three Dimensional Bin Packing Problem, WSEAS Transactions on Information Science and Applications, Volume 8, Issue 6, 2011, pp. 253-263.
- A. Vasilescu, **A. Băicoianu**: Algebraic Model for the CPU Logic Unit Behaviour, Recent researches in computer science, Proc. of the 15-th WSEAS International Conference on Computers, Corfu, 2011, pp. 521-526.
- D. Marinescu, **A. Băicoianu**: The determination of the guillotine restrictions for a rectangular cutting-stock pattern, Latest Trends on Computers, Proc. of the 14-th WSEAS International Conference on Computers CSCC, ISSN: 1792-4251, Corfu, 2010, Volume 1, pp. 121-126.

- D. Marinescu, **A. Băicoianu**: An algorithm for the guillotine restrictions verification in a rectangular cutting-stock pattern, WSEAS Transactions on Computers, Volume 9, Issue 10, 2010, pp. 1160-1169.
- D. Marinescu, **A. Băicoianu**: The determination of the guillotine restrictions for a rectangular cutting-stock pattern, Latest Trends on Computers, Volume 1, 14th WSEAS International Conference on Computers, 2010, pp. 121-126.
- D. Marinescu, **A. Băicoianu**: An algorithm for determination of the guillotine restrictions for a rectangular cutting-stock pattern, Journal WSEAS Transactions on Computers archive, Volume 9 Issue 10, October 2010, pp. 1160-1169.
- **A. Băicoianu**, S. Dumitrescu: Data mining meets Economic Analysis: Opportunities and Challenges, Bulletin of the Transilvania University of Braşov, **2010**.
- D. Marinescu, **A. Băicoianu**: An Algorithm for the guillotine restrictions verification in a rectangular covering Model, WSEAS Transactions on Computers, ISSN: 1109-2750, Issue 8, Volume 8, 2009, pp. 1306-1316.
- D. Marinescu, **A. Băicoianu**: The determination of the guillotine restrictions for a rectangular covering model, Proceedings of the 13th WSEAS International Conference on Computers, Rodos, 2009, pp. 307-312.
- D. Marinescu, P. Iacob, **A. Băicoianu**: A Topological Order for a Rectangular Three Dimensional Bin Packing Problem, Proceedings of the 12th WSEAS International Conference on COMPUTERS, Heraklion, Greece, July 23-25, 2008, WSEAS Press, ISSN: 1790-5109, pp. 285-290. (ISI Web of Knowledge)
- D. Marinescu, P. Iacob, **A. Băicoianu**: A plan for loading the Boxes for a Three Dimensional Bin Packing Model, WSEAS Transactions on System, ISSN: 1109 – 2777, Issue 10, Volume 7, 2008, pp. 830-839.
- P. Iacob, D. Marinescu, **A. Băicoianu**: The generating of the cutting-covering receipts using Euclid's algorithm, Proceedings of the 12th WSEAS International Conference on COMPUTERS, Heraklion, Greece, 2008, WSEAS Press, ISBN 978-960-6766-85-5, pp. 280-284. (ISI Web of Knowledge)

Keywords

- discrete optimization
- cutting and covering problems
- bin packing problems
- NP-hard
- pattern
- dataset
- data mining
- classification
- LAD
- WEKA
- OCTAVE

Resume

In the context of the current problems of discrete optimization research, our aim is to study various types of discrete optimization problems which also apply in practice. Solving this particular set of problems involves searching for the optimal solution among a finite or countably infinite array of potential solutions.

This thesis includes information obtained from the authors own research work and results thus provided were achieved either individually or in collaboration with other researchers. Such research areas include concrete problems specific to various discrete optimization fields, such as rectangular two dimensional cutting problems area, three dimensional bin packing problems area, data mining area and some other further related areas. Also, the LAD methodology is compared in this thesis with five classification algorithms used in the machine learning literature, and their implementation in WEKA and OCTAVE. The comparison aims at exhibiting the fact that LAD is a classification methodology absolutely comparable with other well-known algorithms in the field of data mining, which have had even more satisfactory results in some particular cases.

The thesis comprises five chapters, a bibliographical reference (which includes 85 titles, out of which 17 were written by the author) and three appendices. These appendices are meant to highlight and sustain the results from chapter four.

Chapter 1. This chapter serves as introduction to this thesis. The aim of this section of the thesis is to inform the reader on the issues to be assessed within the content section of the thesis, as well as to provide the necessary documentation for the results obtained by the author within the covered areas. Furthermore, the introduction will also present the main contributions of the present research in the field of discrete optimization.

Chapter 2. It contains newly found theoretical results, which have led to the generation of a set of algorithms: algorithm for solving two dimensional cutting stock problems by means of the Euclid's algorithm, algorithm for checking guillotine type restrictions by means of the decomposition of a graph in connected components, algorithm for checking guillotine type restrictions when gaps are allowed.

In the first section, we begin our exposure by explaining what the cutting and covering (packing) problems are and we analyzed the two dimensional cutting stock problem. We defined one special case of the general two dimensional problem in which all cuts must go from one edge of the rectangle to be cut to the opposite edge, which is a guillotine type.

In the second section, our objective was to define some receipts for solving the cutting and covering problem, using a polynomial method based on Euclid's algorithm for the greatest common divisor. We improve the complexity by a Greedy variant. The complexity of this method is based on Euclid's algorithm and Lamé's theorem. We define the properties for our algorithm, we construct the set of receipts and we establish that the construction in one direction of the cutting-covering receipt is a geometrical construction of Euclid's algorithm, see Theorems 1 and 2. We were able to conclude the complexity of the new method based on the number of divisions in Euclid's algorithm using Lamé's theorem. The novelty of this solution is given by using Euclid's algorithm for solving this kind of problems. The results within this section can be found in the paper [34].

In the third section, we emphasize the cutting and covering problems with guillotine restrictions. It is possible to use an analytic method to verify if the obtained pattern is with guillotine restrictions or not [45]. This method is not so easy to use because the cutting pattern is represented as an array model, that means a large matrix representation. Using the graph representation of the cutting or covering pattern we give in this section another analytic method for testing of the guillotine restriction based on the decomposition of a graph in connected components. In [51] we used the graph representation of the cutting and covering pattern to prove the connection between guillotine and the connected components of the graph. We started from this connection and we present in this section an algorithm which can be used to verify the guillotine restrictions in a two dimensional covering model. We introduced the notions of rectangular covering model, "guillotine restrictions", "downward adjacency", "rightward adjacency" and we considered the graph of downward adjacency and graph of rightward adjacency for defining the cuts for the rectangular cutting and covering problem. The results from Theorems 13 and 14 suggest an algorithm for the verification of the guillotine restrictions, using the decomposition of graphs G'_d or G'_r defined in [44] in connected components. The novelty for this section is given by the manner of defining the cuts used to pronounce the algorithm. The correctness of the algorithm follows from the Theorems 13 and 14, that make the connection between a guillotine cut and the decomposition of the graph G'_d or G'_r in connected components. For the determination of the Polish notation we preserve only one connected component from this decomposition. The algorithm for determination of the connected components has the complexity $O(m)$, where m is the number of the arches of the graph attached to the model. So the complexity of V-CUT or H-CUT procedures defined is also $O(m)$. It follows that the complexity of PREORDER procedure for a rectangular covering model of k items with guillotine restrictions is $O(km)$. Using the decomposition of graphs G'_d or G'_r in connected components and the algorithm defined in [47] we present an extended example for a practical situation.

In the last section, we consider a two dimensional rectangular cutting stock problem in case of a cutting pattern with gaps. First we present two new graph representations of the cutting pattern, weighted graph of downward adjacency and weighted graph of rightward adjacency. Using this kind of representation we propose a method to verify guillotine restrictions of the pattern which can be applied for cutting-stock pattern with

gaps but also for the covering pattern without gaps and overlapping. The results from the Theorem 22 suggest an algorithm for the verification of the guillotine restrictions, in case of a cutting-stock pattern with gaps. The defined algorithm has as input the weighted graphs G_d or G_r attached to a rectangular cutting pattern, and the output is the s-pictural representation of the cutting pattern like a formula in a Polish prefixed form.

The algorithm constructs the syntax tree for the s-pictural representation of the cutting pattern, starting from the root to the leaves procedure PRORD. For every vertex of the tree it verifies if it is possible to make a vertical cut, procedure VCUT or horizontal cut, HCUT procedure, using an algorithm for the decomposition of a graph in two components. We note that we can apply this algorithm also in the case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. The correctness of the algorithm follows from the Theorem 22, that makes the connection between a guillotine cut and the decomposition of a graph in two subgraphs. The procedure PREORD() represents a preorder traversal of a graph, so the complexity is $O(k)$, where k is the number of the cutting items. Also, in the procedure VCUT, respectively HCUT we traverse a subgraph of the initial graph. So, the complexity of the algorithm is $O(k^2)$.

Chapter 3. The problem addressed in this chapter is that of three dimensional bin packing problem and solutions. We consider the rectangular three dimensional bin packing problem with one finite bin, where the bin is packed with a set of rectangular boxes, without gaps or overlapping. Starting from a solution of the three dimensional bin packing model, our objective was to determine an order for the loading the boxes in the bin so that a box is packed in the bin only if there are no empty spaces down to this box and the origin of the box is in a fixed position, determinate by the boxes situated in the West and North neighborhood.

In the first section, there is an introduction to the three dimensional bin packing problems outlining the presentation of the problem and putting it into context. Before we proceed with particular algorithms for solving this kind of problems, we gave in this section some preliminaries definitions and information about the three dimensional bin packing problem.

The second section is an extension of one of the previous work regarding the two dimensional covering problem to a rectangular three dimensional bin packing problem, where a bin is packed with a set of rectangular boxes, without gaps or overlapping. We present a kind of topological sorting algorithm for this problem, of linear complexity, OVERDIAG-3D Algorithm.

By extending the two dimensional covering model, we define in this section three kinds of adjacency relations, adjacency in the direction of Ox , Oy and Oz , Definitions 23, 25, 27. Starting with these three kinds of adjacency we define three kinds of graphs: the graph of adjacency in direction Ox , Oy and Oz and we gave a concrete example of a packing model.

The novelty of this problem is given, on the one hand, by the mathematical models that we introduce and, on the other hand, by the fact that we use these extended types

of graphs for this kind of three dimensional bin packing problem. Also, we discovered some important properties, see Theorems 32, 33. Due the Theorem 33 it is possible to represent simultaneously these obtained graphs by a single adjacency matrix, a matrix with elements from the set $\{0, 1, 2, 3\}$. For any packing model we define a network, a graph of compound adjacency, Definition 34 and we prove that the graph is acyclic, see Theorem 35.

To determine a topological order we used a new algorithm, OVERDIAG-3D which is an extension of a topological sorting algorithm presented in. This algorithm is based on the particularity of the compound graph defined, respectively on the form of the resulted matrix, attached to the graph. The authors' achievements within this section can be found in paper [56]. It completes the results obtained in [44, 55].

In the third section, we discussed the rectangular three dimensional bin packing problem, where a bin is loaded with a set of rectangular boxes, without overlapping. One of the most popular restriction for the solution to the three dimensional bin packing problem is the guillotine restriction, see Definition 40. Our objective here is to find a method for verifying if a solution of the bin packing problem has the guillotine constraints or not. For this purpose we use a weighed graph representation, Definition 43 of a solution of the problem, the generalization of this kind of representation obtained by us for two dimensional cutting stock problem in [48, 49, 50]. Theorem 45 introduce some properties for the weighed graphs attached to the pattern. The results from the previous theorem suggest an algorithm for the verification of the guillotine restrictions, in case of the bin-packing pattern with gaps but without overlapping [57]. We remark that we can apply this algorithm also in case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. An extended example is discussed here and the algorithm iterations are highlighted. The prefix Polish notation for the resulted syntactic tree is given and the algorithm's complexity was studied in [58].

Chapter 4. Our aim for this chapter was to focus on a combinatorial optimization based data analysis methodology, that is able to perform classification with justification, Logical Analysis of Data (LAD). The objective is to present this methodology, that is based on data mining principles, combinatorics, Boolean functions and operational research. In addition, the LAD methodology is compared with the most important classification algorithms used in the machine learning literature: C4.5, Random Forest, Support Vector Machines, Multilayer Perceptron, Logistic regression, and their implementation in WEKA and OCTAVE, for a greater impact on final results.

In the first section, we briefly presented the LAD methodology, as well as some basic notions on using WEKA and OCTAVE. Within the same section, we provide general information on the functions and guiding principles of chosen algorithms, compared to LAD.

In section "Computational experiments" we organize the experiments we have achieved, the steps we followed in doing so together with conclusions we have drawn following LAD testing and the algorithms: C4.5, Random Forest, Support Vector Machines, Mul-

tilayer Perceptron, Logistic regression represent our original contribution to this thesis. At the end of this section we will analyze results, while also highlighting the advantages of the LAD methodology and commenting on its advantages and disadvantages as for more particular cases. Herein, we investigate the accuracy of LAD and we did some remarks about its execution time and the quality of the results. We outline the manner in which this classifying methodology works and we have concluded that it has shortcomings, but also strengths compared to the other methods. Thus, our conclusion is that, when compared to other classification algorithms, the LAD methodology requires simple notions of discrete optimization, while providing an accurate classification on considered situations. A significant contribution of the author in this chapter consists in providing and evaluating computationally the LAD methodology and making some comparison with some specific machine learning algorithms. We evaluate the coverage cases - which percentage of the outcome is covered through a solution - in the case of LAD and other specialized methodologies, as well. Overall, it is to be noticed that LAD counts as a highly intelligent method of classification.

The tests that we did with LAD, WEKA and OCTAVE prove that LAD methodology is used like a great data mining tool for classification, justified by:

- The relatively simple concepts LAD models use → see set covering problem, Boolean functions, simplification of Boolean functions, partial Boolean functions, simplex method, etc.;
- The efficiency/quality of LAD patterns → LAD provides, as any other algorithms, a set of patterns covering both positive and negative observations, but unlike other considered algorithms, these patterns are justifiable, which means that positive patterns cover only those observations ratified as positive, without covering any of the negative patterns, and the other way round. Since the quality and transparency of patterns are very important characteristics when applying classification algorithms, LAD provides these two features, enabling professionals within different fields of application to easily understand them;
- The accuracy provided by LAD → following experiments achieved by the author, it has been observed that results provided by means of LAD are more specific and have a greater degree of accuracy, close to or even exceeding the level of accuracy of other known classification algorithms;
- Time of execution → when based on the processing of Boolean functions, the generation of valid patterns is a process characterized by a low degree of complexity;

Amazing results obtained in the field of healthcare and not only have encouraged us to choose this problem classification methodology. Our aim was to observe whether LAD is as competitive in other fields of interest as in the field of medicine. In order to better emphasize the results of the LAD methodology, we have achieved a comparative study

with more specific data mining algorithms. In order for the comparison to be relevant, the algorithms were chosen so that they share particular features with LAD. The classical implementation used for LAD was modified by the author, in order to obtain expected results earlier. Appendix two includes all these changes. The main reasons for the author of this paper to have chosen WEKA are its versatility and the authors own wish to test all considered algorithms within a common environment. The implementations in OCTAVE represent an additional contribution, meant to strengthen results obtained by means of WEKA.

In the last section of this chapter we described general conclusions on the whole chapter and some further directions for development. As a perspective on this presented data mining tool, we would like to extend our previous work on Haskell [12] to some pattern mining algorithms. We intend to use specific libraries, like HLearn in some specific problems of data mining, particularly on LAD. An interesting perspective would be to find a Haskell alternative solution for practical problems basically solved with LAD methodology.

Chapter 5. In this chapter we highlight the main conclusions and contributions of this thesis.

Appendix 1 In order to serve as anchorage for our tests, for some of the considered data mining algorithms, Logical Regression and Multilayer Perceptron, we also carried out an OCTAVE implementation, using a purely OCTAVE code, with no external libraries or plugins. The implementation was modular, so that algorithm testing for the datasets selected might be less intricate.

Appendix 2 This Appendix is meant to highlight the changes the author underwent on the existing LAD processing tool. We remind here that the adaptations of the original application were:

- converting the application in order that it could be used in Windows (the original application was developed for Linux)
- converting the application so that it becomes a console application, not a command line application
- the optimal rewriting in $C++$ of some methods (dynamic allocation instead of static allocation, we change the parameters of some functions, in those cases where they had more than four parameters, we changed some structures in classes and we wrote some Object-oriented programming modules)
- we added some new necessary methods, like the one for calculating standard deviation
- additionally, because k -folding method is the most frequently used cross-validation technique, we evaluate the accuracy of LAD using one random 10-fold cross-validation.

Appendix 3 is meant to highlight and sustain the results from chapter four.

The original contributions in this thesis are:

In Chapter 2:

- the algorithm for solving the cutting and covering problem, using a polynomial method based on Euclid's algorithm for the greatest common divisor. We improve the complexity by a Greedy variant. The complexity of this method is based on Euclid's algorithm and Lamé's theorem 1. We define the properties for our algorithm, we construct the set of receipts and we establish that the construction in one direction of the cutting-covering receipt is a geometrical construction of Euclid's algorithm, see Theorems 1 and 2. The novelty of this solution is given by using Euclid's algorithm for solving this kind of problems.
- the results from Theorems 13 and 14 suggest an algorithm for the verification of the guillotine restrictions, using the decomposition of graphs in connected components. The novelty for this algorithm is given by the manner of defining the cuts used to pronounce the algorithm. The correctness of the algorithm follows from the Theorems 13 and 14, that make the connection between a guillotine cut and the decomposition of the graphs in connected components.
- the results from the Theorem 22 suggest an algorithm for the verification of the guillotine restrictions, in case of a cutting-stock pattern with gaps. The defined algorithm has as input the weighted graphs G_d or G_r attached to a rectangular cutting pattern, and the output is the s-pictural representation of the cutting pattern like a formula in a Polish prefixed form. The complexity of the algorithm is $O(k^2)$.

In Chapter 3:

- we define three kinds of adjacency relations, adjacency in the direction of Ox , Oy and Oz , Definitions 23, 25, 27. Starting with these three kinds of adjacency we define three kinds of graphs: the graph of adjacency in direction Ox , Oy and Oz and we gave a concrete example of a packing model.
- we introduce new mathematical models, see Theorems 32, 33 and we discovered some important properties. Due the Theorem 33 it is possible to represent simultaneously these obtained graphs by a single adjacency matrix, a matrix with elements from the set $0, 1, 2, 3$. For any packing model we define a network, a graph of compound adjacency, Definition 34 and we prove that the graph is acyclic, see Theorem 35.
- to determine a topological order we used a new algorithm, OVERDIAG-3D which is an extension of a topological sorting algorithm presented in. This algorithm is based on the particularity of the compound graph defined, respectively on the form of the resulted matrix, attached to the graph.

- we use a weighed graph representation, Definition 43 of a solution of the bin packing problem. Theorem 45 introduce some properties for the weighed graphs attached to the pattern. The results from the previous theorem suggest an algorithm for the verification of the guillotine restrictions, in case of the bin-packing pattern with gaps but without overlapping. We remark that we can apply this algorithm also in case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. An extended example is discussed here and the algorithm iterations are highlighted. The prefix Polish notation for the resulted syntactic tree is given and the algorithm's complexity was studied.

In Chapter 4, after having achieved a set of changes with the classical implementation technique used for LAD, a comparative study between LAD and five classification algorithms, by means of WEKA and OCTAVE. In view of an accurate and complete comparison between these algorithms and the LAD methodology, we have taken into consideration two main comparison directions: default parameters, where we compared results obtained with LAD with those results obtained with WEKA (for C4.5, Multilayer Perceptron, Logistic, SMO and Random Forest algorithms) - Explorer and optimal parameters, WEKA provides a series of means for automating the process of finding optimal parameters for a classifier, Experimenter. From the two comparative perspectives, accuracy and confusion matrices, LAD represents a methodology which allows comparison with the given algorithms, sometimes with even better results. The comparison directions together with all tests and conclusions belong entirely to the author.

Bibliography

- [1] G. Alexe, S. Alexe, P.L. Hammer: Pattern-Based Clustering and Attribute Analysis, *Soft Computing Springer Journal*, 2006, pp. 442-452.
- [2] G. Alexe, S. Alexe, T.O. Bonates, A. Kogan: Logical analysis of data the vision of Peter L. Hammer, *Annals of Mathematics and Artificial Intelligence*, 49(1-4):265-312, 2007.
- [3] G. Alexe, P.L. Hammer: Spanned patterns for the logical analysis of data, *Discrete Appl. Math.*, 154 (2006), pp. 1039 - 1049.
- [4] G. Alexe, P.L. Hammer: Accelerated algorithm for pattern detection in logical analysis of data, *Discrete Appl. Math.*, 154 (2006), pp. 1050 - 1063.
- [5] G. Alexe, S. Alexe, P.L. Hammer, A. Kogan: Comprehensive vs. comprehensible classifiers in logical analysis of data, *Discrete Applied Mathematics*, in press, 2007.
- [6] S. Alexe, et al.: Coronary Risk Prediction by Logical Analysis of Data. *Annals of Operations Research*, 119:15 - 42, 2003.
- [7] R.R. Amossen, D. Pisinger: Multi-dimensional Bin Packing Problems with Guillotine Constraints, *Computers & OR* 37, No. 11, pp. 1999-2006, 2010.
- [8] M.Z. Arslanov, D.U. Ashigaliev, E.E. Ismail: Polynomial algorithms for guillotine cutting of a rectangle into small rectangles of two kinds *European, Journal of Operational Research*, Vol. 185, Issue 1, 2008, pp. 105-121.
- [9] R.E. Banfield: A comparison of decision tree ensemble creation techniques, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 (2007), pp. 173-180.
- [10] A. Băicoianu, I. Plajer: *Formal Languages and Automata - practice*, Transilvania University Press, Braşov, Romania, 2012.
- [11] A. Băicoianu: A Comparative Study of Some Classification Algorithms using WEKA and LAD Algorithm, *Annals of the Tiberiu Popoviciu Seminar of Functional Equations, Approximation and Convexity*, ISSN 1584-4536, vol 12, 2014, pp. 7381.

- [12] A. Băicoianu, A. Vasilescu, R. Pândaru: Upon the performance of a Haskell parallel implementation, *Bulletin of the Transilvania University of Braşov*, **2013**, Series III: Mathematics, Informatics, Physics, 61-72.
- [13] A. Băicoianu, S. Dumitrescu: Data mining meets Economic Analysis: Opportunities and Challenges, *Bulletin of the Transilvania University of Braşov*, 2010.
- [14] T.O. Bonates, P.L. Hammer: Pseudo-Boolean Regression, *RRR 3-2007*, January, 2007.
- [15] E. Boros, P.L. Hammer, T.I Baraki, A. Kogan: Logical analysis of numerical data, *Mathematical Programming*, 79 (1997), pp. 163 - 190.
- [16] E. Boros, P.L. Hammer, et al.: An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering* 12 (2), 2000,292 – 306.
- [17] E. Ciurea, L. Ciupală: Algoritmi - Introducere în algoritmica fluxurilor în reţele, Ed. Matrix ROM Bucureşti, 2006.
- [18] P. Chen, Y. Chen, M. Goel, F. Mang: Approximation of Two-Dimensional Rectangle Packing, CS270 Project Report, 1999, www.cad.eecs.berkeley.edu/~fmang/cs270/report.pdf.
- [19] T.H. Cormen, C.E. Leiserson, R.R Rivest: *Introduction to Algorithms*, MIT Press, 1990.
- [20] I.G. Czibula, G. Şerban: A Hierarchical Clustering Algorithm for Software Design Improvement, *KEPT 2007, Knowledge Engineering: Principles and Techniques*, Babeş-Bolyai University, June 6-8, 2007, Cluj-Napoca, pp. 316-323.
- [21] I.G. Czibula, G.S. Cojocar: A Hierarchical Clustering Based Approach in Aspect Mining, *Computing and Informatics*, Vol. 29, 2010, 881900.
- [22] D. Dumitrescu, H.F. Pop, C. Sârbu: Fuzzy hierarchical cross-classification of Greek muds, *Journal of Chemical Information and Computer Science*, 35, pp. 851-857, 1995.
- [23] D. Dumitrescu: *Principiile matematice ale teoriei clasificării*, Academiei Române, Bucureşti, 1999.
- [24] H. Dyckhoff: A typology of cutting and packing problems, *European Journal of Operational Research* 44 (1990) 145-159.
- [25] C. Filippi, A. Agnetis: An asymptotically exact algorithm for the high-multiplicity bin packing problem, *Mathematical Programming*, 2005.

-
- [26] M. R. Garey, D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.
- [27] S. Godoy-Calderon, J.D. Batiz, M. Lazo-Cortez: *A non-Classical View of Coverings and its Implications in the Formalization of Pattern Recognition Problems*, Proceedings of the WSEAS Conference, 2003, paper 459-151.
- [28] P.L. Hammer: *The Logic of Cause-effect Relationships*, Lecture at the International Conference on Multi-Attribute Decision Making via Operations Research-based Expert Systems, Passau, Germany, 1986.
- [29] P.L. Hammer, T. Bonates: *Logical Analysis of Data: From Combinatorial Optimization to Medical Applications*, Rutgers Center for OR.
- [30] J. Han, M. Kamber: *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, August 2000.
- [31] D.J. Hand, H. Mannila, P. Smyth: *Principles of Data Mining*. MIT Press, Cambridge, MA, 2001.
- [32] P. Iacob, D. Marinescu, C. Luca: *Covering a rectangle with rectangular pieces*, Proceedings - Strategies and Applications, Thessaloniki, Greece, October 18-20, 2001, pp. 506-513.
- [33] P. Iacob, D. Marinescu, K. Kiss-Jakab: *Some algorithms for generating receipts in the rectangular cutting-covering problem*, NAUN, International Journal of Mathematical Models and Methods in Applied Science, Issue 3, Vol. 1., 2007, pp. 182-187.
- [34] P. Iacob, D. Marinescu, A. Băicoianu : *The generating of the cutting-covering receipts using Euclid's algorithm*, International Conference on Computers, Heraklion, Greece, July 23-25, 2008.
- [35] P. Iacob, D. Marinescu, C. Luca: *L-Shape room*, Proceeding of WMSCI , Orlando, Florida, USA, 2005, Vol III, pp. 175-179.
- [36] H.W. Ian, E. Frank: *Data Mining: Practical machine learning tools and techniques*. 2nd edition. Morgan Kaufmann, San Francisco, 2005.
- [37] Z. Jin, T. Ito: *The three Dimensional Bin Packing Problem and its Practical Algorithm*, *JSME*, Series , Vol. 46, No.1, 2003.
- [38] S.B. Kotsiantis: *Supervised Machine Learning: A Review of Classification Techniques*, *Informatica* 31(2007) 249-268, 2007.
- [39] M. Lejeune: *Pattern definition of the p-efficiency concept*, *Annals of Operations Research* 200(1):1-14, November 2012.

- [40] M. Lauer, S. Alexe, et al.: Use of the Logical Analysis of Data Method for Assessing Long-Term Mortality Risk After Exercise Electrocardiography. *Circulation*, 106:685 - 690, 2002.
- [41] P. Lémaire: The ladoscope gang: Tools for the Logical Analysis of Data. <http://www.kamick.org/lemaire/LAD/>
- [42] J.E. Lewis, R.K. Ragade, A. Kumar, W.E. Biles: A distributed chromosome genetic algorithm for bin-packing, 14th Intern. Conf. on Flexible Automation and Intelligent Manufacturing Vol. 21, Issues 4-5, Aug-Oct 2005, pp. 486-495.
- [43] D. Marinescu: An integer programming model for two-dimensional cutting stock-problems, Babeş-Bolyai University, Fac. Math. Phys., Res. Semin. 8, pp. 65-74 (1987).
- [44] D. Marinescu: A representation problem for a rectangular cutting - stock model, *Foundations of Computing and Decision Sciences*, Vol. 32, 2007, No. 3, pp. 239-250.
- [45] D. Marinescu: A bidimensional Turing machine for the cutting stock problem with guillotine restrictions, *Proceeding of the scientific symposium with the contribution of teachers and researchers from the Republic of Moldova, Braşov 1991*, pp. 73-83.
- [46] D. Marinescu: A s-picture language for a cutting-stock model with guillotine restrictions, *Bulletin of the Transilvania University of Braşov - seria C*, Vol XXXIII 1991 pp. 39-45.
- [47] D. Marinescu: Graphs attached to a rectangular cutting-stock model (French), *Itinerant Seminar of Functional Equation, Approximation and Convexity, Cluj-Napoca 1988*, Preprint No. 6, pp. 209 -212.
- [48] D. Marinescu, A. Băicoianu: The determination of the guillotine restrictions for a rectangular cutting-stock pattern, *Latest Trends on Computers, Proc. of the 14-th WSEAS International Conference on Computers CSCC, ISSN: 1792-4251, Corfu, 2010, Vol. 1*, pp. 121-126.
- [49] D. Marinescu, A. Băicoianu: An algorithm for the guillotine restrictions verification in a rectangular cutting-stock pattern, *WSEAS Transactions on Computers, Volume 9 Issue 10, Oct. 2010*, pp. 1160-1169.
- [50] D. Marinescu, A. Băicoianu: An Algorithm for the guillotine restrictions verification in a rectangular covering Model, *WSEAS Transactions on Computers, ISSN: 1109-2750, Issue 8, Vol. 8, Aug. 2009*, pp. 1306-1316.
- [51] D. Marinescu, A. Băicoianu: The determination of the guillotine restrictions for a rectangular covering model, *Proc. of the 13-th WSEAS International Conference on Computers, Crete, 2009*, pp. 307-312.
- [52] D. Marinescu, A. Băicoianu: The determination of the guillotine restrictions for a rectangular cutting-stock pattern, *Proceeding of the 14-th WSEAS CSCC, Corfu, 2010*, pp. 121 - 126.

- [53] D. Marinescu, A. Băicoianu: An algorithm for determination of the guillotine restrictions for a rectangular cutting-stock pattern, *Journal WSEAS Transactions on Computers archive*, Volume 9 Issue 10, October 2010, pp. 1160-1169.
- [54] D. Marinescu, A. Băicoianu, S. Dimitriu: Software for Plagiarism Detection in Computer Source Code, *Proc. of the 7th International Conference on Virtual Learning ICVL 2012*, pp. 373-379.
- [55] D. Marinescu, P. Iacob, K. Kiss-Jakab: A topological order for a rectangular covering model, *Proc. of the 11-th WSEAS CSCC, Crete, Vol. 4, 2007*, pp. 82-85.
- [56] D. Marinescu, P. Iacob, A. Băicoianu: A Topological Order for a Rectangular Three Dimensional Bin Packing Problem, *Proceedings of the 12th WSEAS International Conference on COMPUTERS, Heraklion, Greece, July 23-25, 2008*, WSEAS Press, ISSN: 1790-5109, pp. 285-290.
- [57] D. Marinescu, A. Băicoianu, D. Simian: The determination of the guillotine restrictions for a Rectangular Three Dimensional Bin Packing Pattern, *Recent Researches in Computer Science, Proc. of the 15-th WSEAS International Conference on Computers CSCC, Corfu, 2011, Vol. 1*, pp. 491-496.
- [58] D. Marinescu, A. Băicoianu: A test of the guillotine restrictions determination for a Rectangular Three Dimensional Bin Packing Problem, *WSEAS Transactions on Information Science and Applications, Volume 8 Issue 6, June 2011*, pp. 253-263.
- [59] D. Marinescu, P. Iacob, A. Băicoianu: A plan for loading the Boxes for a Three Dimensional Bin Packing Model, *WSEAS Transactions on System, ISSN: 1109 – 2777, Issue 10, Vol. 7, Oct. 2008*.
- [60] E. Mayoraz: C++ Tools for Logical Analysis of Data, *Technical Report, RTR 1 – 95, Rutgers University, July 1995*.
- [61] T. Papke, A. Bortfeldt, H. Gehring: Software Programs for Solving a Cutting Problem in the Wood-Working Industry. A Case Study, *WSEAS Transactions on Information Science & Applications, Issue 5, Volume 4, May 2007*, pp. 932-938.
- [62] S.A. Paschos, V.Th. Paschos, V. Zissimopoulos: A model to approximately cover an area by antennas, *Proc. of WSEAS Conference, 2003*, pp. 458-187.
- [63] Z. Pawlak: *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers Norwell, USA, 1992.
- [64] H.F. Pop, C. Sârbu, O. Horowitz, D. Dumitrescu: A fuzzy classification of the chemical elements, *Journal of Chemical Information and Computer Sciences* 36, 3 (1996), pp. 465-482.
- [65] H.F. Pop: *Sisteme inteligente în probleme de clasificare*, PhD Thesis, Babeş-Bolyai University, 1995.

- [66] H.F. Pop, M. Frențiu: Applications of Principal Components Methods, (ISI) Proceedings of the International Conference "Complexity and Intelligence of the artificial and Natural Complex Systems. Medical Applications of the Complex Systems, Biomedical Computing", IEEE Computer Society Press, 2009, pp. 103-109, ISBN 78-0-7695-3621-7.
- [67] J.R. Quinlan: C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
- [68] K.H. Rosen: Handbook of discrete and combinatorial mathematics, CRC Press, 2000.
- [69] X. Song, C. Chu, Y. Nie: A Heuristic Dynamic Programming Algorithm for 2D Unconstrained Guillotine Cutting, Proceedings of WSEAS Conference , 2004, pp. 484-108.
- [70] P. Sweeney, E. Paternoster: Cutting and packing problems: a categorized application-oriented research bibliography, Journal of the Operational Research Society , 43/7 (1992) 691-706.
- [71] A. Vasilescu, A. Băicoianu: Algebraic Model for the CPU Logic Unit Behaviour, Recent researches in computer science, Proc. of the 15-th WSEAS International Conference on Computers, Corfu, 2011.
- [72] I.H. Witten, E. Frank: Data Mining: Practical Machine Learning Tools with Java Implementations, Morgan Kaufmann, San Francisco, 1999.
- [73] Three dimensional Packing Problems (2003) The Institute for Algorithms and Scientific Computing (SCAI, Germany).
- [74] ***, <https://hackage.haskell.org/packages/>
- [75] ***, <http://mathworld.wolfram.com/OptimizationTheory.html>
- [76] ***, <http://dictionary.sensagent.com>
- [77] ***, <http://rutcor.rutgers.edu/pub/LAD/>
- [78] ***, <http://www.cs.waikato.ac.nz/~ml/weka>
- [79] ***, <https://weka.wikispaces.com/Optimizing+parameters>
- [80] ***, <http://www.cs.waikato.ac.nz/ml/weka/documentation.html>
- [81] ***, https://en.wikipedia.org/wiki/Confusion_matrix
- [82] ***, <https://izbicki.me/public/papers>
- [83] ***, <http://archive.ics.uci.edu/ml/datasets.html>
- [84] ***, <http://www.gnu.org/software/octave/download.html>
- [85] ***, <http://www-mdp.eng.cam.ac.uk/web/CD/engapps/octave>

Chapter 1

Introduction

Just as life is about making decisions, most optimization problems require several objectives to be achieved, and these are generally conflicting. Thus, in order for the solution to be simpler, these objectives will be regarded as having only one solution to start with, following that the remaining ones are regarded as constraints.

Operations research, often considered to be a sub field of mathematics, is a discipline that deals with the application of advanced methods in helping to make better decisions. It leads to the optimal solutions or near optimal solutions to complex decision-making problems. It is often concerned with determining the maximum (of profit, performance, or yield) or minimum (of loss, risk or cost) of some real world objective.

It is well known that the operations research, as it is nowadays, appeared during the second world war, from the practical necessity to find the more efficient military sources. We recall that the operations research give a generic answer to the following question: if we suppose that if the implications of the choice of the elements of a given set is known, then we should determine the element (better said all the elements) of the set which satisfies some given conditions such that the result of the above implication is optimal. Over time, operations research knew a very quick development. Hence, nowadays it has a lot of branches among which we point out the optimization theory. Optimization theory includes the calculus of variations, control theory, convex optimization theory, decision theory, game theory, linear optimization, etc. The study of optimization problems was closely connected to practical problems.

The scientific results obtained within this thesis were accompanied by a variety of optimization obstacles, mainly caused by concrete data mining and problem cutting and covering. It is therefore important to emphasize the justified high ranking of the optimization theory among other mathematical areas, since it provides such impressive array of applications within more practical fields of science. When confronted with the specific problem to be solved, we select the method or algorithm which can be used for solving it, as well as a set of necessary and sufficient conditions for ensuring optimal solutions for each suggested problem.

As the title of the thesis indicates, our aim is to study various types of discrete optimization problems which also apply in practice. Solving this particular set of problems involves searching for the optimal solution among a finite or countably infinite array of potential solutions. The definition of the theory of optimality considers a set of criterion function meant to be maximized or minimized. All problems studied herein stand on specific applications to be encountered in real life situations. The solutions may be combinatorial structures like arrangements, sequences, combinations, choices of objects, subsets, subgraphs, chains, routes in a network, assignments, schedules of jobs, packing schemes, etc.

When living in a flat in the city, one of the greatest challenges is finding all the necessary room for storing various possessions - too many things, too little space. In the suburbs, this problem is as big of a threat as it is for central city dwellers, since it is easy to notice that the option of using off-site storage facilities is on a steady rise. When attempting to summarize this in more metaphorical terms, it would seem that we are facing an insufficient amount of bins for all storing necessities. That's where mathematics comes in to suggest the bin packing problem and its relatives.

The bin packing problem raises the following question: given a finite collection of n weights $w_1, w_2, w_3, \dots, w_n$, and a collection of identical bins with capacity C (which exceeds the largest of the weights), what is the minimum number k of bins into which the weights can be placed without exceeding the bin capacity C ? Stripped of the mathematical formulation, we want to know how few bins are needed to store a collection of items. This problem, known as the one-dimensional bin packing problem, is one of many mathematical packing problems which are of both theoretical and applied interest in mathematics and computer science.

We consider that having the opportunity to study such an impressive field is a real privilege. This thesis includes information obtained from the authors own research work and results thus provided were achieved either individually or in collaboration with other researchers. Such research areas include concrete problems specific to various discrete optimization fields, such as rectangular two dimensional cutting problems area, three dimensional bin packing problems area, data mining area and some other further related areas.

Next, we will refer to the content of this thesis, and we will also provide concise information about the issues that each of the chapters is to deal with. The thesis consists of five chapters, the bibliography section and two Appendices. The Appendices are meant to ensure a deeper understanding of various notions presented in Chapter four, and to prove the accuracy of different implementations made.

Chapter 1. This chapter serves as introduction to this thesis. The aim of this section of the thesis is to inform the reader on the issues to be assessed within the content section of the thesis, as well as to provide the necessary documentation for the results obtained by the author within the covered areas. Furthermore, the introduction will also present the main contributions of the present research in the field of discrete optimization.

Chapter 2. This chapter encloses the original results of the authors scientific re-

search on rectangular two dimensional cutting problems, which can be further referred to in papers [34, 48, 49, 50, 51, 52, 53].

We begin our exposure by explaining what the cutting and covering (packing) problems are, according to H. Dyckhoff [24] and we analyzed the two dimensional cutting stock problem. We defined one special case of the general two dimensional problem in which all cuts must go from one edge of the rectangle to be cut to the opposite edge, which is a guillotine type.

In Section "The generating cutting-covering solutions using Euclid's algorithm" our objective was to define some receipts for solving the cutting and covering problem, using a polynomial method based on Euclid's algorithm for the greatest common divisor. We improve the complexity by a Greedy variant. The complexity of this method is based on Euclid's algorithm and Lamé's theorem. We define the properties for our algorithm, we construct the set of receipts and we establish that the construction in one direction of the cutting-covering receipt is a geometrical construction of Euclid's algorithm, see Theorem 1. We were able to conclude the complexity of the new method based on the number of divisions in Euclid's algorithm using Lamé's theorem.

The novelty of this solution is given by using Euclid's algorithm for solving this kind of problems. The results within this section can be found in the paper authored by P. Iacob, D. Marinescu and A. Băicoianu [34].

In Section "The determination of the guillotine restrictions for a rectangular covering model" we emphasize the cutting and covering problems with guillotine restrictions. It is possible to use an analytic method to verify if the obtained pattern is with guillotine restrictions or not [45]. This method is not so easy to use because the cutting pattern is represented as an array model, that means a large matrix representation. Using the graph representation of the cutting or covering pattern [47, 44] we give in this section another analytic method for testing of the guillotine restriction based on the decomposition of a graph in connex components.

In [51] we used the graph representation of the cutting and covering pattern to prove the connection between guillotine and the connex components of the graph. We started from this connection and we present in this section an algorithm which can be used to verify the guillotine restrictions in a two dimensional covering model.

We introduced the notions of rectangular covering model, "guillotine restrictions", "downward adjacency", "rightward adjacency" and we considered the graph of downward adjacency and graph of rightward adjacency for defining the cuts for the rectangular cutting and covering problem. The results from Theorems 13 and 14 suggest an algorithm for the verification of the guillotine restrictions, using the decomposition of graphs G'_d or G'_r defined in [44] in connex components. The novelty for this section is given by the manner of defining the cuts used to pronounce the algorithm. The correctness of the algorithm follows from the Theorems 13 and 14, that make the connection between a guillotine cut and the decomposition of the graph G'_d or G'_r in connex components. For the determination of the Polish notation we preserve only one connex component from this decomposition. The algorithm for determination of the connex components has the

complexity $O(m)$, where m is the number of the arches [17, 19]. So the complexity of V-CUT or H-CUT procedures defined is also $O(m)$. It follows that the complexity of PREORDER procedure for a rectangular covering model of k items with guillotine restrictions is $O(km)$.

We note that the results obtained within this section complete the results obtained in [47] and detailed results are found in [51]. Also, the next section completes this section, giving some extra information about the algorithm and a particular example with all iterations.

Using the decomposition of graphs G'_d or G'_r in connex components and the algorithm defined in [47] we present an extended example for a practical situation. The authors achievements within this section can be found in the paper authored by D. Marinescu, A. Băicoianu [49] and we mention that it completes the results obtained in [47].

In section "The determination of the guillotine restrictions for a rectangular cutting-stock pattern" we consider a two dimensional rectangular cutting stock problem in case of a cutting pattern with gaps. First we present two new graph representations of the cutting pattern, weighted graph of downward adjacency and weighted graph of rightward adjacency. Using this kind of representation we propose a method to verify guillotine restrictions of the pattern which can be applied for cutting-stock pattern with gaps but also for the covering pattern without gaps and overlapping. The results from the Theorem 22 suggest an algorithm for the verification of the guillotine restrictions, in case of a cutting-stock pattern with gaps.

The defined algorithm has as input the weighted graphs G_d or G_r attached to a rectangular cutting pattern, and the output is the s-pictural representation of the cutting pattern like a formula in a Polish prefixed form.

The algorithm constructs the syntax tree for the s-pictural representation of the cutting pattern, starting from the root to the leaves (*procedure PRORD*). For every vertex of the tree it verifies if it is possible to make a vertical cut (*procedure VCUT*) or horizontal cut (*HCUT procedure*), using an algorithm for the decomposition of a graph in two components. We note that we can apply this algorithm also in the case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. The correctness of the algorithm follows from the Theorem 22, that makes the connection between a guillotine cut and the decomposition of a graph in two subgraphs. The procedure *PREORD()* represents a preorder traversal of a graph, so the complexity is $O(k)$ [19], where k is the number of the cutting items. Also, in the procedure *VCUT*, respectively *HCUT* we traverse a subgraph of the initial graph. So, the complexity of the algorithm is $O(k^2)$.

All results revealed within this thesis are fully discussed in the paper written by D. Marinescu, A. Băicoianu [52]. Using the algorithm presented in [52] we studied a particular example for which we gave the Polish notation.

Please note that all problems studied within this chapter refer to new approaches in solving cutting and rectangular problems. In this purpose, we defined various algorithms and we solved a range of specific types of two dimensional rectangular cutting

and covering problems.

Chapter 3, contains original results of scientific research belonging to the author of the thesis and can be found in the following papers [56, 59, 57, 58].

The problem addressed in this chapter is that of three dimensional bin packing problem and solutions. We consider the rectangular three dimensional bin packing problem with one finite bin, where the bin is packed with a set of rectangular boxes, without gaps or overlapping. Starting from a solution of the three dimensional bin packing model, our objective was to determine an order for the loading the boxes in the bin so that a box is packed in the bin only if there are no empty spaces down to this box and the origin of the box is in a fixed position, determinate by the boxes situated in the West and North neighborhood.

In Section "Background notions concerning three dimensional bin packing problems" there is an introduction to the three dimensional bin packing problems outlining the presentation of the problem and putting it into context. Before we proceed with particular algorithms for solving this kind of problems, we gave in this section some preliminaries definitions and information about the three dimensional bin packing problem.

In Section "A topological order for a rectangular three dimensional bin packing problem" is an extension of one of the previous work [55] regarding the two dimensional covering problem to a rectangular three dimensional bin packing problem, where a bin is packed with a set of rectangular boxes, without gaps or overlapping. We present a kind of topological sorting algorithm for this problem, of linear complexity, **OVERDIAG-3D** Algorithm.

By extending the two dimensional covering model [44], we define in this section three kinds of adjacency relations, adjacency in the direction of Ox , Oy and Oz , Definitions 23, 25, 27. Starting with these three kinds of adjacency we define three kinds of graphs: the graph of adjacency in direction Ox , Oy and Oz and we gave a concrete example of a packing model.

The novelty of this problem is given, on the one hand, by the mathematical models that we introduce and, on the other hand, by the fact that we use these extended types of graphs for this kind of three dimensional bin packing problem. Also, we discovered some important properties, see Theorems 32, 33. Due the Theorem 33 it is possible to represent simultaneously these obtained graphs by a single adjacency matrix, a matrix with elements from the set $0, 1, 2, 3$. For any packing model we define a network, a graph of compound adjacency, 34 and we prove that the graph is acyclic, see 35.

To determine a topological order we used a new algorithm, **OVERDIAG-3D** which is an extension of a topological sorting algorithm presented in [19]. This algorithm is based on the particularity of the compound graph defined, respectively on the form of the resulted matrix, attached to the graph. The authors' achievements within this section can be found in paper [56]. It completes the results obtained in [44, 55].

In section "The determination of the guillotine restrictions for a rectangular three dimensional bin packing pattern" we discussed the rectangular three dimensional bin

packing problem, where a bin is loaded with a set of rectangular boxes, without overlapping. One of the most popular restriction for the solution to the three dimensional bin packing problem is the guillotine restriction, see 40. Our objective here is to find a method for verifying if a solution of the bin packing problem has the guillotine constraints or not. For this purpose we use a weighed graph representation 43 of a solution of the problem, the generalization of this kind of representation obtained by us for two dimensional cutting stock problem in [48, 49, 50]. Theorem 45 introduce some properties for the weighed graphs attached to the pattern. The results from the previous theorem suggest an algorithm for the verification of the guillotine restrictions, in case of the bin-packing pattern with gaps but without overlapping [57]. We remark that we can apply this algorithm also in case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. An extended example is discussed here and the algorithm iterations are highlighted. The prefix Polish notation for the resulted syntactic tree is given and the algorithm's complexity was studied in [58].

Please note that all results comprised in this chapter are provided in four other academic papers, the result of either individual or co-authorships. Each of these academic works advances a series of original perspectives and optimized algorithms for three dimensional bin packing problems.

Chapter 4 is concerned with the study of specific methodology in what regards theories such as discrete optimization and data mining, **Logical Analysis of Data (LAD)**. Our aim was to present a data mining tool, a combinatorial and optimization based data analysis method, that is able to perform classification. In this chapter we describe and investigate the performance of novel optimization applications in classification and regression models with LAD. Also, the LAD algorithm is compared in this section with the main classification algorithms used in the machine learning literature, and their implementation in WEKA [72]. The comparison aims at exhibiting the fact that LAD is a classification methodology absolutely comparable with other well-known algorithms in the field of data mining, which have had even more satisfactory results in some particular cases.

Please note that all statements and results presented herein belong to the author. Part of them is also to be found in [13] and some in [11].

Finally, in **Chapter 5** we highlight the main conclusions and contributions of this thesis.

The personal contribution of the author in the area of discrete optimization, with applications in two dimensional cutting problem, three dimensional bin packing problems and some data mining problems may be highlighted through the following published papers and a textbook:

- **A. Băicoianu:** A Comparative Study of Some Classification Algorithms using WEKA and LAD Algorithm, Annals of the Tiberiu Popoviciu Seminar of Functional Equations, Approximation and Convexity, ISSN 1584-4536, vol 12, 2014, pp. 7381.

- **A. Băicoianu**, A. Vasilescu, R. Pândaru: Upon the performance of a Haskell parallel implementation, Bulletin of the Transilvania University of Braşov, **2013**, Series III: Mathematics, Informatics, Physics, 61-72.
- (textbook) **A. Băicoianu**, Plajer I.: Formal Languages and Automata - practice, Transilvania University Press, Braşov, Romania, 2012.
- D. Marinescu, **A. Băicoianu**, S. Dimitriu: Software for Plagiarism Detection in Computer Source Code, Proc. of the 7th International Conference on Virtual Learning ICVL 2012, pp. 373-379.
- D. Marinescu, **A. Băicoianu**, Simian D.: The determination of the guillotine restrictions for a Rectangular Three Dimensional Bin Packing Pattern, Recent Researches in Computer Science, Proc. of the 15-th WSEAS International Conference on Computers CSCC, Corfu, 2011, Vol. 1, pp. 491-496.
- D. Marinescu, **A. Băicoianu**: A test of the guillotine restrictions determination for a Rectangular Three Dimensional Bin Packing Problem, WSEAS Transactions on Information Science and Applications, Volume 8 Issue 6, June 2011, pp. 253-263.
- P. Iacob, D. Marinescu, **A. Băicoianu**: The generating of the cutting-covering receipts using Euclid's algorithm, Proceedings of the 12th WSEAS International Conference on COMPUTERS, Heraklion, Greece, July 23-25, 2008, WSEAS Press, ISBN 978-960-6766-85-5, pp. 280-284. (ISI Web of Knowledge)
- D. Marinescu, **A. Băicoianu**: The determination of the guillotine restrictions for a rectangular cutting-stock pattern, Latest Trends on Computers, Proc. of the 14-th WSEAS International Conference on Computers CSCC, ISSN: 1792-4251, Corfu, July 2010, Vol. 1, pp. 121-126.
- D. Marinescu, **A. Băicoianu**: An algorithm for the guillotine restrictions verification in a rectangular cutting-stock pattern, WSEAS Transactions on Computers, Volume 9 Issue 10, Oct. 2010, pp. 1160-1169.
- D. Marinescu, **A. Băicoianu**: An Algorithm for the guillotine restrictions verification in a rectangular covering Model, WSEAS Transactions on Computers, ISSN: 1109-2750, Issue 8, Vol. 8, Aug. 2009, pp. 1306-1316.
- D. Marinescu, **A. Băicoianu**: The determination of the guillotine restrictions for a rectangular covering model, Proceedings of the 13th WSEAS International Conference on Computers, Rodos, 2009, pp. 307-312.
- D. Marinescu, **A. Băicoianu**: The determination of the guillotine restrictions for a rectangular cutting-stock pattern, Latest Trends on Computers, vol.1, 14th WSEAS International Conference on Computers, 2010, pp. 121-126.

- D. Marinescu, P. Iacob, **A. Băicoianu**: A Topological Order for a Rectangular Three Dimensional Bin Packing Problem, Proceedings of the 12th WSEAS International Conference on COMPUTERS, Heraklion, Greece, July 23-25, 2008, WSEAS Press, ISSN: 1790-5109, pp. 285-290. (ISI Web of Knowledge)
- D. Marinescu, P. Iacob, **A. Băicoianu**: A plan for loading the Boxes for a Three Dimensional Bin Packing Model, WSEAS Transactions on System, ISSN: 1109 – 2777, Issue 10, Vol. 7, Oct. 2008.
- D. Marinescu, **A. Băicoianu**: An algorithm for determination of the guillotine restrictions for a rectangular cutting-stock pattern, Journal WSEAS Transactions on Computers archive, Volume 9 Issue 10, October 2010, pp. 1160-1169.
- **A. Băicoianu**, S. Dumitrescu: Data mining meets Economic Analysis: Opportunities and Challenges, Bulletin of the Transilvania University of Braşov, **2010**.

Chapter 2

Rectangular two dimensional cutting stock problems

Mathematics applies to real life, for instance when having to study a series of concrete and practical aspects from a mathematical standpoint. As already stated in the introduction to this thesis, the present study was carried out through the analysis of the cutting stock problem an aspect of utmost importance which bears effects on the profit of several processing industries.

The cutting stock problem deals with a set of rectangular items and in theory it is classified as the two dimensional rectangular cutting stock problem. Though having been widely researched within mathematical programming, the findings of such studies fail to consider the cutting process in the practical manufacturing phase. Thus, in this chapter we are to map a set of solutions for the two dimensional rectangular cutting stock problems, while also considering the constraints of the cutting process.

For the mathematical modeling of the above type of problem and its study we use different mathematical tools and notions. Therefore, for the easier lecturing of the present thesis, we start by presenting basic notions and results with respect to cutting stock problems, guillotine cutting patterns, covering models.

2.1 Basic notions concerning cutting and covering problem

Problems of cutting and covering (packing) of concrete or abstract objects appear under various specifications [24]: cutting-stock problems, knapsack problems, container and vehicle loading problems, pallet loading, bin-packing, assembly line balancing, etc.

The problem is NP-hard and it generally emerges within disparate production processes which spread from home textile to glass, steel, wood or even paper manufacturing industries. In such cases, rectangular bodies are cut from larger surfaces of bulk material. What is more, an increasingly difficult problem can be hence derived - Cutting and Covering - which involves cutting a larger piece of fabric into several smaller elements

in order to cover a surface without any overlaps or gaps in between.

So far, the cutting stock problem has been a great focus for the fields of computer science and mathematics, since it enables a wide range of applications in various industries, as for instance in the case of a managing position within a paper cutting division in a factory which has to produce rolls of paper of fixed width. How can the manager tackle the cutting process so that in the end, the last amount of paper goes to waste? Most definitely, this represents a discrete optimization problem, or more specifically, an integer linear programming problem, where linear programming is a branch of applied mathematics which focuses on solving optimization problems.

One such solving method is the simplex method which identifies a basic non optimal point and gradually advances towards determining increasingly good solutions. When no further solutions are to be found, the algorithm achieves with the optimal solution. On the other hand, with the paper roll problem, the issue is to minimize the loss. The constraints of the problem in this case require that we cut enough rolls of certain dimensions, so as to satisfy the need. First, we should create all possible patterns that might be cut from a roll. When the orders involve different widths, it is possible that the problem might include an exponential number of patterns. Consequently, such problem might prove impossible to formulate, let alone to solve.

Then, a reasonable question to ask is, do we really have to generate all the possible patterns even before solving the problem? The answer is no. The delayed column generation method solves the cutting stock problem by starting with just a few patterns. It generates additional patterns when they are needed. The new patterns are introduced by solving another optimization problem called the "knapsack problem". For solving this problem, there are numerous methods, among them are branch-and-bound and dynamic programming. These two problems, the main linear program and the "knapsack problem", are solved in turn until no more patterns can be generated which will reduce the number of rolls cut.

The two dimensional cutting stock problem requires cutting a plane rectangle into smaller rectangular pieces of given sizes and values to maximize the sum of the values of the pieces cut. This version of the problem appears in the problem of cutting steel or glass plates into required stock sizes to minimize waste. By taking the value of a piece to be proportional to its area, we can formulate the waste minimization problem as one of maximizing the value of the pieces cut. The problem also appears in cutting wood plates to make furniture and paper board to make boxes.

A special case of the general two dimensional cutting problem is one in which all cuts must go from one edge of the rectangle to be cut to the opposite edge, the cut has to be of a guillotine type. Some cutting patterns can not be produced by this type of cut. However, the restriction of guillotine cuts appears very often in practice especially for cutting of paper and glass.

In practice cutting problems appear in a constrained form, the most usual constraint being the one that restricts the maximum number of pieces of each type to be cut. In the one dimensional problem upper bound constraints on the variables can easily be included

in a dynamic programming algorithm. However, the two dimensional constrained cutting problem is not amenable to efficient solution by these means.

In what follows, we will study the rectangular two dimensional cutting stock problem and various solutions for this. We highlight in these next sections the importance of this practical problem and we present some solutions for it.

The standard formulation for the cutting-stock problem starts with a list of m orders, each requiring q_j , $j = 1, \dots, m$ pieces. We then construct a list of all possible combinations of cuts, often called *patterns*, associating with each pattern a positive integer variable x_i representing how many times each pattern is to be used. The linear integer program is then:

$$\begin{aligned} & \min \sum_{i=1}^n c_i x_i \\ & \text{s.t. } \sum_{i=1}^n a_{ij} x_i \geq q_j, \quad \forall j = 1, \dots, m \\ & \quad x_i \geq 0, \text{ integer} \end{aligned}$$

where a_{ij} is the number of times order j appears in pattern i and c_i is the cost, often the waste of pattern i . The particular nature of the quantity constraints can lead to different mathematical characteristics. The above formulation's quantity constraints are minimum constraints (at least the given amount of each order must be produced, but possibly more). When $c_i = 1$ the objective minimizes the number of utilized master items and, if the constraint for the quantity to be produced is replaced by equality, it is called the bin packing problem. The most general formulation has two-sided constraints (and in this case a minimum-waste solution may consume more than the minimum number of master items):

$$q_j \leq \sum_{i=1}^n a_{ij} x_i \leq Q_j, \quad \forall j = 1, \dots, m$$

This formulation applies not just to one dimensional problems. Many variations are possible, including one where the objective is not to minimize the waste, but to maximize the total value of the produced items, allowing each order to have a different value.

In general, the number of possible patterns grows exponentially as a function of m , the number of orders. As the number of orders increases, it may therefore become impractical to enumerate the possible cutting patterns. An alternative approach uses delayed column-generation. This method solves the cutting-stock problem by starting with just a few patterns. It generates additional patterns when they are needed. For the one-dimensional case, the new patterns are introduced by solving an auxiliary optimization problem called the knapsack problem, using dual variable information from the linear programming.

2.2 The generating cutting-covering solutions using Euclid's algorithm

The number of publications in the area of cutting-stock problem has increased considerably over the last two decades. The typology of cutting-stock problems introduced by H. Dyckhoff [24], 1990 and by P. Sweeney [70], 1992 initially provided an excellent instrument for the organization and categorization of existing and new literature. These problems are in fact NP-complete problems. However, over the years some deficiencies of this typology have also become evident, which created problems in dealing with recent developments and prevented it from being accepted more generally.

If we know the dimensions of the pieces then we are dealing with a classical cutting-stock problem, which can be modeled as a mixed 0 – 1 programming problem [43]. There are also heuristic models, but it is not our study, for details see the papers [18, 69]. Now, if the dimensions of pieces used for covering are unknown then the problem is more complicated.

Our objective for this section is to present a polynomial method for solving cutting problems based on Euclid's algorithm to compute the greatest common divisor. We improve the complexity by a Greedy variant. The proof of the method's complexity is based on Euclid's algorithm and Lamé's theorem, see [68], page 228. Using the well-known Euclid's algorithm, we gave an original method for extracting the solutions for a two dimensional problem, namely Euclid's algorithm.

2.2.1 Problem statement and formulation

Like we mentioned before, the everyday problem that we are taking on here is to cover a rectangular room with a cover, linoleum or carpet. This material is in a roll of fixed width and by cutting it with a guillotine we get another rectangle. We want to cover the room with a minimum number of pieces and to waste a minimum amount of material (amount of left-overs).

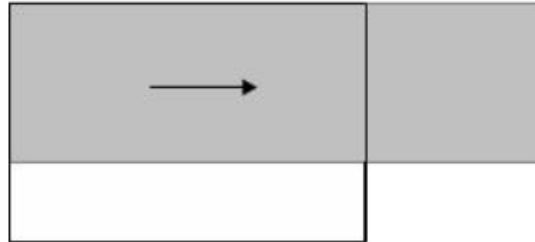
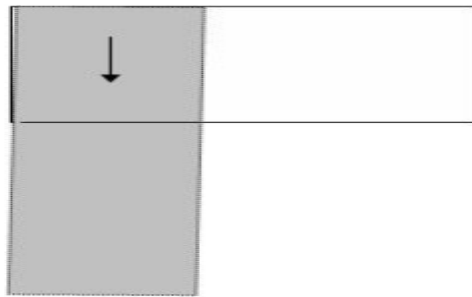
We consider two rectangular surfaces, one of them with the dimensions a , b , the other one with unknown dimensions, x and y . The optimization problem that we have here is to obtain the minimum value of y so that we could cover the first rectangle with smaller pieces from the second one.

2.2.2 Problem solving steps

We want to know how the material with dimensions x and y can be cut to cover a room of dimensions a and b with the condition (c_1):

$$a * b = x * y.$$

It is clear that if $x = a$ (or $x = b$) we have equal rectangles and there is no cutting

Figure 2.1: Direction of Ox Figure 2.2: Direction of Oy

problem. Therefore we will assume that we are in the situation:

$$x < a < b < y.$$

Practically we may consider only the case where x, a, b, y are integers.

Let $m = \gcd(a, b, x, y)$, gcd - greatest common divisor. We may consider the cover divided in squares of dimension m and any coverage of the room will be a permutation of these squares. This is a finite number but unacceptably great. We mention that all these statements can be found in [35].

The method we propose here is based on the following two properties:

1. If we put the cover on the floor overlapping two adjacent sides of the cover over two adjacent sides of the floor and we cut the remainder of material (that does not cover the floor), we obtain a new piece of cover and a new piece of floor that have the same properties described by condition c_1 .
2. The cover may be put on the floor in two directions (see the example from Figure 2.1 and Figure 2.2).

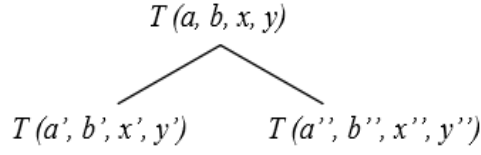


Figure 2.3: The obtained binary tree

- **A first approach**

Keeping the orientation of the pieces in the covering process we have to make a convention: we shall choose fitting units of lengths on a, b, x, y so:

- the same unit of length on a and x and the same unit of length on b and y ;
- the number of units of length on a coincide with the number of units of length on y and the same condition for b and x ;

After we laid out the material in one of the directions, we face the same issue regarding dimensions smaller than a or b or x or y .

The algorithm (proposed by P. Iacob, D. Marinescu, and C. Luca in [32]) is recursively generating a binary tree: if the initial problem is to cover the rectangle of dimensions a and b with a cover of dimensions x and y then it is the root $T(a, b, x, y)$; putting x on a we obtain the right sub-tree with the root $T(a - x, b, x, y - b)$ and putting x on b we obtain the left sub-tree with the root $T(a, b - x, x, y - a)$, where $a' = a, b' = b - x, x' = x, y' = y - a$, and $a'' = a - x, b'' = b, x'' = x, y'' = y - b$.

But, if $a < x$ then $a'' = a, b'' = b - y, x'' = x - a, y'' = y$ and if $b < x$ then $a' = a - x, b' = b, x' = x - b$ and $y' = y$.

It is obvious that if $a = x$ or $b = x$ we have already a solution of cutting covering. As it was shown in [32], the algorithm ends after a finite number of steps. The cutting design with the smallest number of pieces will be the shortest way from the root to a leaf. We can detect some situations when growing the y we can cover the initial rectangle; between them some are Pareto optimum points.

We assume that: $R_0 = T(a - x, b, x, y - b)$ and $R_1 = T(a, b - x, x, y - a)$. The zero index means that the material was laid out on the direction of Oy while the one index 1 value means that material was laid out in the other direction.

Theorem 1. [34] *If $a > x$ and $b > x$ then $R_{01} = R_{10}$.*

Proof. $R_{01} = T(a - x, b - x, x, y - b - a) = T(a - x, b - x, x, y - a - b) = R_{10}$. \square

Nr of receipt	Receipt	Nr of pieces	Waste
1	$R_{00 \dots 0}$ q_a+1 times	q_a+1	$b*(x-r_a)$
2	$R_{00 \dots 010}$ q_a times	q_a+2	$(b-x)*(x-r_a)$
3	$R_{00 \dots 0110}$ q_a times	q_a+3	$(b-2x)*(x-r_a)$
...			
q_b+1	$R_{00 \dots 011 \dots 10}$ q_a times q_b times	q_a+q_b+1	$r_b*(x-r_a)$
q_b+2	$R_{11 \dots 11}$ q_b+1 times	q_b+1	$a*(x-r_b)$
q_b+3	$R_{00 \dots 010}$ q_b times	q_b+2	q_b+2
q_b+4	$R_{00 \dots 0110}$ q_b times	q_b+3	$(a-2x)*(x-r_b)$
...			
q_b+q_a+1	$R_{00 \dots 011 \dots 10}$ q_b times q_a times	q_a+q_b+1	$r_a*(x-r_b)$

Figure 2.4: The solution

So, if $a > x$ and $b > x$ then it doesn't matter in which of the directions we laid out the material. Now we are able to give a few cutting and covering solutions on a $O(q_a + q_b)$, where $a = x * q_a + r_a$ and $b = x * q_b + r_b$, q_a and q_b are quotients, and r_a and r_b are remainders in these equations.

The construction of the set of solutions:

It is obvious that previous solutions have the same number of pieces, meaning that we will choose the one corresponding with $\min(r_a, r_b)$. We will proceed in the same manner with the other solutions having the same number of pieces, choosing the one with minimal waste (minimal remainder).

We consider the following example, where value of x is given and where we used the formulas from Figure 2.4

$$a = 27, b = 16, x = 10.$$

Taking only the solutions having the same number of pieces and minimal losses we obtain the solution from Figure 2.6.

In the cases of $a < x$ and $b < x$ we use the algorithm described in [33]. But this algorithm can also be furthermore optimized. Let us take the following situation when $\frac{a}{x} = \frac{2}{3}$, then the solution is to divide the cover and the surface into $2 * 3 = 6$

Nr of receipt	Receipt	Nr of pieces	Waste
1	R ₀₀₀	3	16*3=48
2	R ₀₀₁₀	4	6*3=18
3	R ₁₁	2	6*27=162
4	R ₁₀₁	3	6*17=102
5	R ₁₀₀₁	4	6*7=42

Figure 2.5: Example



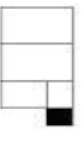
Nr of receipt	Receipt	Nr of pieces	Waste	Figure
1	R ₁₁	2	6*27=162	
2	R ₀₀₀	3	16*3=48	
3	R ₀₀₁₀	4	6*3=18	

Figure 2.6: Some numerical example of solution

pieces. The solution is obvious. By applying the first method a better solution can be obtained. We will take the algorithm from [33] and modify it in the following way:

We choose from table the optimal decomposition of $\frac{a}{x}$ and $\frac{b}{x}$. For each fraction of the decomposition the initial step is applied, but the tree is built in one direction, on the left for the decomposition of $\frac{a}{x}$, respectively on the right for $\frac{b}{x}$.

By applying it for our example, we get:

$$\frac{a}{x} = \frac{27}{10} = \frac{5}{2} + \frac{1}{5} \text{ and } \frac{b}{x} = \frac{16}{10} = \frac{8}{5}.$$

Theorem 2. [34] *The construction in one direction of the cutting-covering solution is a geometrical construction of Euclid's algorithm for the numbers a and x where the fraction $\frac{a}{x}$ is irreducible and $a < x$.*

Proof. Let us assume the situation of $T(a, b, x)$. Computing $y = \frac{a*b}{x}$, the length of the cover material, and $b' = \frac{y}{a} = \frac{b}{x}$. The surface to cover the material is divided into rectangles with dimension 1 on the direction of a and b' on the direction of b (the material will be laid out with x on a).

The obtained rectangles are aligned in the same direction on the surface to be covered and on the covering material. We can cut out these rectangles from the covering material and lay them onto the surface to be covered without any rotation which would lead us to a solution of $a * x$ pieces and 0 waste according with [33]. Another solution would be the construction of a binary tree only on the right side: \square

The successive dimensions of the covering material and the surface to be covered are: $x, a, r_1, r_2, , \dots, r_n, 0$, the remainders of the divisions of Euclid's algorithm. The number of pieces $S = \sum_{i=1}^n q_i$ will be the sum of the successive quotients from the same Euclid's algorithm.

Lemma 3. *If $\frac{a}{x}$ is irreducible and $a < x$ then $S = \sum_{i=1}^n q_i \leq a * x + 1 - a^2$.*

Proof. As we showed in the beginning of the previous proof we can have a cutting-covering solution of $a * x$ pieces. If we apply the same method of cutting as in the previous example - rectangles of dimension 1 respectively b' - but we lay out the first piece of material without cutting it, remaining of dimension a^2 , then we will have $a * x + 1 - a^2$ pieces, and with construction of the binary tree in only one direction we get the same number of pieces. \square

- **A second approach** Uses parts from the first one, taking into account the optimal decomposition of the fraction $\frac{a}{x}$. Details in the paper written by P. Iacob, D. Marinescu, A. Băicoianu [34].

$$\begin{array}{l}
T(a, b, x) \\
T(a, b-y, x-a) \\
T(a, b-2y, x-2a) \\
\cdots \\
T(a, b-q_1*y, r_1) \quad \text{where } x=a*q_1+r_1 \\
T(a-r_1, b', r_1) \\
T(a-2r_1, b', r_1) \\
\cdots \\
T(r_2, b', r_1) \quad \text{where } a=q_2*r_1+r_2 \\
T(r_2, b'-r_2, r_1-r_2) \\
\cdots \\
T(r_{n-1}, 0', r_n)
\end{array}$$

Figure 2.7: The binary tree, right side

We are now able to conclude the complexity of the new method based on the number of divisions in Euclid's algorithm. Lamé's [68] had found that the complexity is less than 5^* (the number of a 's digits). We conclude that the complexity of the new method is:

$$O(5 * MAX(numberOfDigits(r_a), numberOfDigits(r_b))).$$

Cutting stock problems have many applications in production processes in paper, glass, metal and timber cutting industries. There are also unconventional applications, where covering model is used in the formalization of pattern recognition problems [27].

Finally, our cutting-covering problem is also important for the situations in which we don't know the dimensions of the pieces for cutting and covering. For this kind of problems it is possible to use the second approach which is polynomial and it uses a small amount of memory for intermediate data storage. It follows that it is faster even for big integers as input data.

The results within this chapter belong to the author and can be found in the paper authored by P. Iacob, D. Marinescu and A. Băicoianu, [34].

2.3 The determination of the guillotine restrictions for a rectangular covering model

Many kinds of cutting and covering problems or cutting/covering problems were considered in the literature with many kinds of constraints depending on technological restrictions. In the two dimensional or three dimensional applications it is frequently required that the resulting patterns be guillotine-cuttable, i.e. that the item can be obtained through a sequence of edge-to-edge cuts parallel to the edges of the support. These kinds of restrictions are named guillotine restrictions and they are obtained by the technological conditions of the cutting or covering process.

For cutting, covering and cutting-covering problems many models and algorithms are developed such as: formulation as mixed integer programming [37], genetic algorithms [42] or approximation algorithms.

In all of these methods we obtained a pattern or a set of patterns. The guillotine restrictions are difficult to respect in generating this pattern. So, it is possible to use an analytic method to verify if the obtained pattern is with guillotine restrictions or not [45]. This method is not so easy to use because the cutting pattern is represented as an array model, that means a large matrix representation.

Using the graph representation of the cutting or covering pattern [47, 44] we present here another analytic method for the verification of the guillotine restriction based on the decomposition of a graph in connex components. Also, starting from this connection, we propose an algorithm which can be used to verify guillotine restrictions in a two dimensional covering model.

2.3.1 Problem statement and formulation

Let \mathcal{P} , a rectangular plate, characterized by length l and width w . The plate \mathcal{P} is covered with k rectangular items, C_i , $i = 1, 2, \dots, k$, from \mathcal{C} , the set of the rectangular items, without gaps or overlapping. An item is characterized by length l_i and width w_i .

Definition 4. *A rectangular covering model of \mathcal{P} is an arrangement of the k rectangular items C_i , $i = 1, 2, \dots, k$ on the supporting plate \mathcal{P} , so that \mathcal{P} is completely covered by the components C_i , $i = 1, 2, \dots, k$, without gaps or overlapping.*

Let the covering model from Figure 2.8 where $\mathcal{P}(340 \times 172)$, $C_1(96 \times 47)$, $C_2(44 \times 123)$, $C_3(51 \times 59)$, $C_4(51 \times 62)$, $C_5(110 \times 39)$, $C_6(110 \times 59)$, $C_7(135 \times 94)$, $C_8(72 \times 77)$, $C_9(83 \times 77)$, $C_{10}(89 \times 77)$.

Definition 5. *A rectangular covering model has guillotine restrictions if at every moment of the cutting process the remaining supporting rectangle is separated in two new rectangles by a cut from an edge to the opposite edge of the rectangle and the cutting line is parallel with the two remaining edges.*

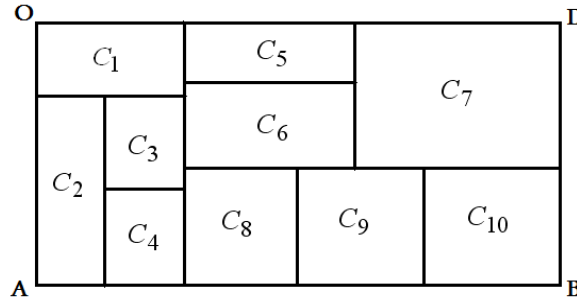


Figure 2.8: The rectangular covering model

In the set of the rectangles $\{C_1, C_2, \dots, C_k\}$ from the covering model we define a downwards adjacency relation and a rightwards adjacency relation.

Definition 6. *The rectangle C_i is downward adjacent with rectangle C_j if in the covering model C_j is to be found downward C_i and their borders have at least two common points.*

Definition 7. *The rectangle C_i is rightward adjacent with rectangle C_j if in the covering model, C_j is to be found rightward C_i and their borders have at least two common points.*

Let $C = \{C_1, C_2, \dots, C_k\}$ and consider $R_d, R_r \notin C$, the northern borderline and the western of the plate P . For any covering model, we can define a graph of downwards adjacency, G_d , and another one of rightwards adjacency, G_r .

Definition 8. [44] *The graph of downward adjacency $G_d = (C \cup \{R_d\}, \Gamma_d)$ has as vertices the rectangles C_1, C_2, \dots, C_k and a new vertex R_d symbolizing the northern borderline of the supporting plate P .*

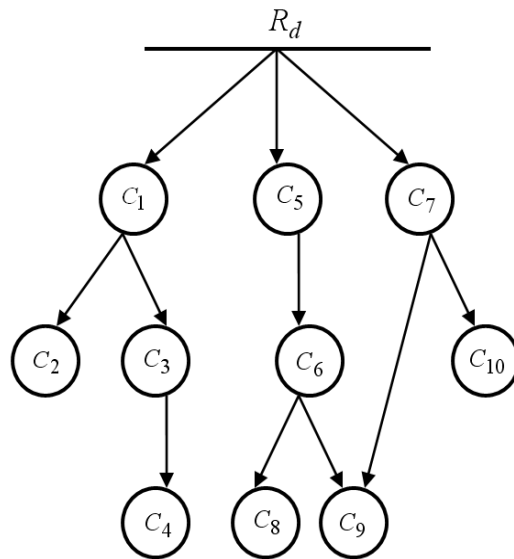
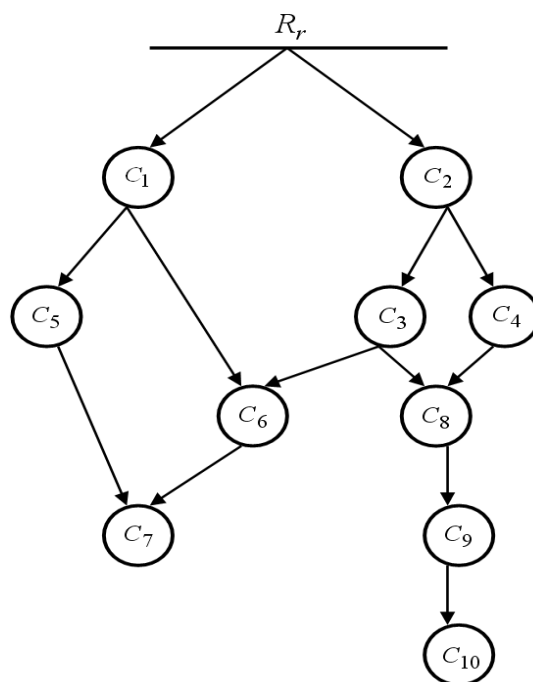
The Γ_d is defined as follows:

$$\left\{ \begin{array}{l} \Gamma_d(C_i) \ni C_j \text{ if } C_i \text{ is downward adjacent} \\ \quad \text{with } C_j \\ \Gamma_d(R_d) \ni C_i \text{ if } C_i \text{ touches the North border} \\ \quad \text{OD of support plate } P \end{array} \right.$$

Definition 9. [44] *The graph of rightward adjacency $G_r = (C \cup \{R_r\}, \Gamma_r)$, where R_r symbolizes the western border:*

The Γ_r is defined as follows:

$$\left\{ \begin{array}{l} \Gamma_r(C_i) \ni C_j \text{ if } C_i \text{ is rightward adjacent} \\ \quad \text{with } C_j \\ \Gamma_r(R_r) \ni C_i \text{ if } C_i \text{ touches the West border} \\ \quad \text{OA of support plate } P \end{array} \right.$$

Figure 2.9: The graph G_d corresponding Figure 2.8Figure 2.10: The graph G_r corresponding Figure 2.8

Let the covering model from Figure 2.8. The graphs G_d and G_r , are represented in Figure 2.9 and Figure 2.10.

We remark that in the graphs G_d and G_r the vertex R_d (respectively R_r) is connected by an arch to the vertex C_i if and only if C_i touches the northern (respectively the western) border of the support \mathcal{P} .

Definition 10. [44] *Let a covering model \mathcal{M} and the graph of downward adjacency G_d . We say that the rectangle C_i is situated above the rectangle C_j in the covering model \mathcal{M} if in the graph of downward adjacency G_d there is a path from C_i to C_j . Similarly, we say that the rectangle C_i is situated to the left of the rectangle C_j in the covering model \mathcal{M} if in the graph of rightward adjacency G_r there is a path from C_i to C_j .*

Remark 11. *From [46] it follows that it is possible to represent a rectangular covering model with guillotine restrictions using an expression with two operations:*

1. \ominus - the line concatenation, an operation for horizontal cuts;
2. \otimes - the column concatenation, an operation for vertical cuts.

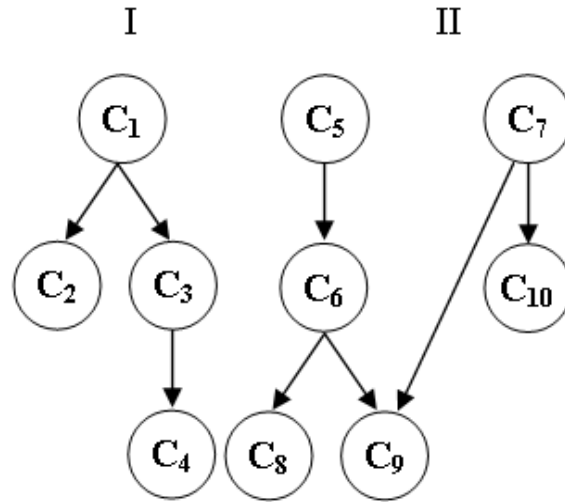
For the rectangular covering model from Example 1 we obtain the following representation:

$$\begin{aligned} & (C_1 \ominus (C_2 \otimes (C_3 \ominus C_4))) \\ & \quad \otimes \\ & (((C_5 \ominus C_6) \otimes C_7) \ominus ((C_8 \otimes C_9) \otimes C_{10})) \end{aligned}$$

where C_1, \dots, C_{10} are the items from our model.

Remark 12. *From the Property 4 from [44] it follows that for two items C_i and C_j from a covering model there is only one of the following situations:*

1. C_i is situated above C_j (there is a path from C_i to C_j in the graph G_d);
2. C_j is situated above C_i (there is a path from C_j to C_i in the graph G_d);
3. C_i is situated to the left of C_j (there is a path from C_i to C_j in the graph G_r);
4. C_j is situated to the left of C_i (there is a path from C_j to C_i in the graph G_r);
5. There is no path between C_i and C_j nor in the graph G_d nor in G_r .

Figure 2.11: The subgraphs G'_d extracted from G_d

2.3.2 Cuts determination

Starting from a rectangular covering model we intend to find a connection between guillotine restrictions and the two graphs of adjacency, G_d and G_r , attached to the covering model.

For this purpose we construct two new subgraphs G'_d and G'_r where:

1. $G'_d = (C, \Gamma'_d)$ is obtained from G_d by elimination of the vertex R_d together with the arches starting from R_d ;
2. $G'_r = (C, \Gamma'_r)$ is obtained from G_r by elimination of the vertex R_r together with the arches starting from R_r .

In Figure 2.11 the subgraph G'_d for the covering model from 2.8 is presented.

In [47] it is proved that the graph G_d , respectively G_r , is a connex graph. By the elimination of a vertex together with the arches starting from this vertex it is not sure that the subgraph G'_d , respectively G'_r , remain a connex graph.

We will prove that for a model with guillotine restrictions at least one of these subgraphs is not a connex graph.

Theorem 13. *Let \mathcal{M} be a rectangular covering model without gaps and overlapping and the graph G'_d attached to \mathcal{M} . In the covering model \mathcal{M} there is a vertical guillotine cut if and only if in the graph G'_d there are at least two connex components.*

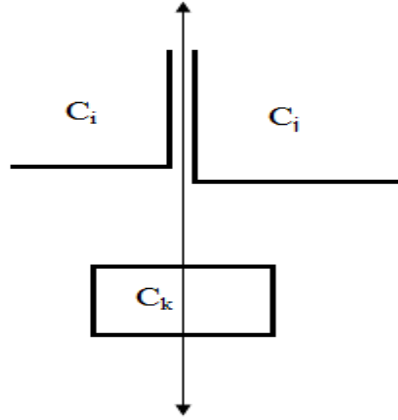


Figure 2.12: Particular case

Proof. Necessity. Suppose that the covering model \mathcal{M} has a vertical guillotine cut. That means the sets of items \mathcal{C} can be separated in two subsets, \mathcal{C}_l , the set of the vertices situated to the left of the cut, and \mathcal{C}_r , the set of the vertices situated to the right of the cut.

Let $C_i \in \mathcal{C}_l$ and $C_j \in \mathcal{C}_r$ two items situated to the left, respectively to the right of the cut. Suppose that the graph G'_d is connex. It follows that there is a chain between C_i and C_j in the graph G'_d . On this chain there are at least two vertices, C_n, C_m so that $C_n \in \mathcal{C}_l$, $C_m \in \mathcal{C}_r$ and C_m is downward adjacent with C_n (or C_n is downward adjacent with C_m), from Definition 7. That means C_m is situated above (downward) C_n and the items C_m and C_n have at least two common points.

But, in this case, it is impossible to separate C_m from C_n by a vertical cut. So our supposition that G'_d is a connex graph is false.

Sufficiency. Suppose there are two connex components of the graph G'_d , G_{d1} and G_{d2} . Let \mathcal{C}_1 be the set of the vertices from G_{d1} and \mathcal{C}_2 be the set of the vertices from G_{d2} . Let $C_i \in \mathcal{C}_1$ and $C_j \in \mathcal{C}_2$ so that C_i is rightward adjacent with C_j . It follows that there is no chain between C_i and C_j in G'_d and of course, no paths.

Suppose that a cut between C_i and C_j intersects an item C_k . This means we have a situation like in Figure 2.12.

In this case, C_i and C_j are situated above the item C_k in the model \mathcal{M} and that means there are two paths, one path from C_i to C_k and another path from C_j to C_k in G'_d . That means there is a chain between C_i and C_j . But it is impossible because C_i and C_j belong to two different connex components. It follows that our assumption that a cut between C_i and C_j intersects an item C_k is false.

So it is possible to separate the model \mathcal{M} in two submodels by a vertical cut. \square

Theorem 14. Let \mathcal{M} be a rectangular covering model and the graph G'_r attached to \mathcal{M} .

In the covering model \mathcal{M} there is a horizontal guillotine cut if and only if in the graph G'_r there are at least two connected components.

Proof. The proof is similar with the proof of previous theorem, for vertical cuts. \square

2.3.3 The algorithm for the verification of the guillotine restrictions

The results from the previous theorems suggest an algorithm for the verification of the guillotine restrictions, using the decomposition of graphs G'_d or G'_r in connex components.

Input data: The graph G'_d or G'_r attached to a rectangular covering model.

Output data: The s-pictural representation of the covering model like a formula in a Polish prefixed form.

Method: The algorithm constructs the syntactic tree for the s-pictural representation of the covering model, defined in Remark 8, starting from the root to the leaves (procedure PREORDER). For every vertex of the tree verifies if it is possible to make a vertical (procedure V-CUT) or horizontal cut (H-CUT), using an algorithm for the decomposition of a graph in two components (procedure CONEXCOMP [17], page 36): one is a connex component and the other is the rest of the graph after extraction of the connex component. The method ADD() is used for addition of the next member in the Polish prefixed form.

```

PROCEDURE PREORDER ( $G, ADD()$ )
begin
  V-CUT( $G, err, G_l, G_r$ );
  if  $err = 0$  then
    if  $|G| = 1$  then ADD( $G$ );
    else ADD( $\emptyset$ );
      PREORDER( $G_l, ADD()$ );
      PREORDER( $G_r, ADD()$ );
    end
  else
    H-CUT( $G, err, G_u, G_d$ );
    if  $err = 0$  then
      if  $|G| = 1$  then ADD( $G$ );
      else ADD( $\ominus$ );
        PREORDER( $G_u, ADD()$ );
        PREORDER( $G_d, ADD()$ );
      end
    else No guillotine restrictions
  end
end

```

The procedures for vertical cut and horizontal cut are presented bellow:

PROCEDURE V-CUT(G, err, G_l, G_r)

```

begin
   $err = 0$ ;
  CONEXCOMP( $G, p, N_l, N_r$ );
  if  $p = 1$  then  $err = 1$ ; ;
  else
     $G_l = N_l$ ;
     $G_r = N_r$ ;
  end
end

```

PROCEDURE H-CUT(G, err, G_u, G_d) \circ

```

begin
   $err = 0$ ;
  CONEXCOMP( $G, p, N_u, N_d$ );
  if  $p = 1$  then  $err = 1$ ; ;
  else
     $G_u = N_u$ ;
     $G_d = N_d$ ;
  end
end

```

Further, we lay out an example with all iterations for our algorithm. Let us take the covering model from Figure 2.8 with the extracted subgraphs, G'_d and G'_r . In Figure 2.13 the first vertical cut of the covering model and the decomposition in two components are presented: a conex component and the remainder component. In the syntax tree from Figure 2.14 the conex component is marked by 1, the remainder component is marked by 2 and they are connected using the operation of column concatenation \circ for the vertical cut.

Let $A = \{C_1, C_2, C_3, C_4\}$ and $B = \{C_5, C_6, C_7, C_8, C_9, C_{10}\}$ be the sets of the vertices from these two components.

The partial prefix Polish notation for this syntax tree from Figure 2.14 is:

\circ .

We continue to make vertical or horizontal cuts for the left and right components from the syntax tree until every components contains only one item from the covering model.

Using the first component we are doing a horizontal cut in Figure 2.15, so we have the decomposition of the set A in two sets which are corresponding to the two new components, one of them contain only the item C_1 and another set $D = \{C_2, C_3, C_4\}$. In Figure 2.16 we are adding to the syntax tree the nodes C_1 and the component 3 which are connected using the operation of line concatenation \ominus for the horizontal cut.

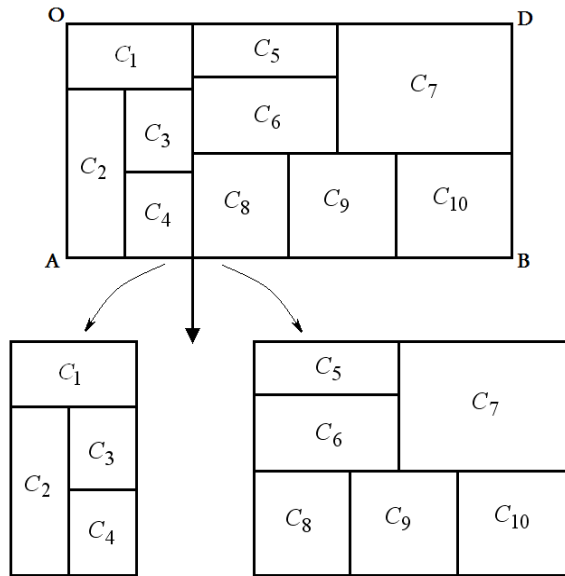


Figure 2.13: The first vertical cut

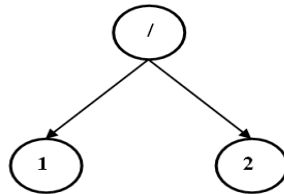


Figure 2.14: The first syntax tree

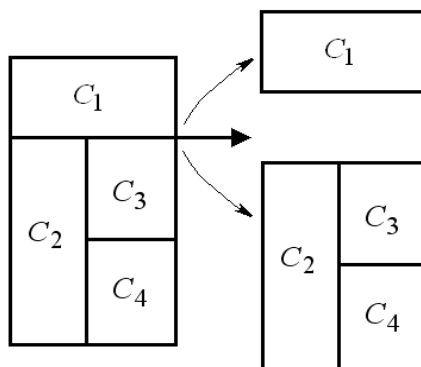


Figure 2.15: Step2. The first horizontal cut

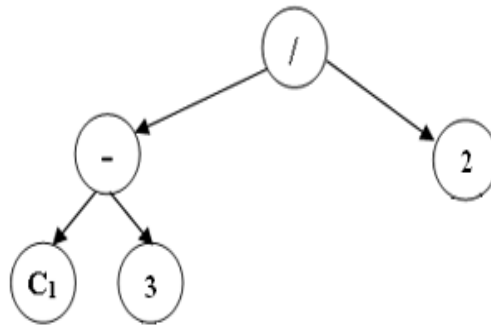


Figure 2.16: Step2. The syntax tree

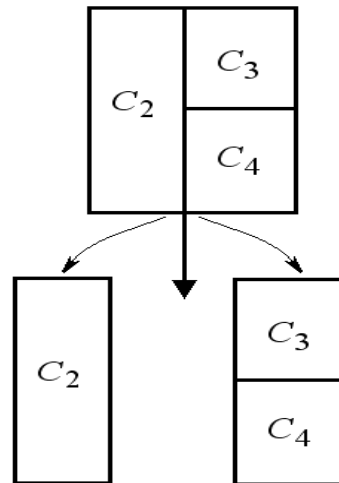


Figure 2.17: Step3. A vertical cut

The partial prefix Polish form for this tree from Figure 2.16 is:

$$\ominus \ominus C_1.$$

On the third component we are trying to do a vertical cut, a decomposition of the set D in 2 sets like in Figure 2.17, so we have other two components. One of these is a leaf of the syntax tree, C_2 and another is the fourth component, let it be the set $E = \{C_3, C_4\}$, Figure 2.18.

The partial prefix notation associated with the syntax tree from Figure 2.18 is:

$$\ominus \ominus C_1 \ominus C_2.$$

The last step is the decomposition of the fourth component, the set E in two components using a horizontal cut Figure 2.19, C_3 and C_4 , both leaves of the syntax tree from Figure 2.20.

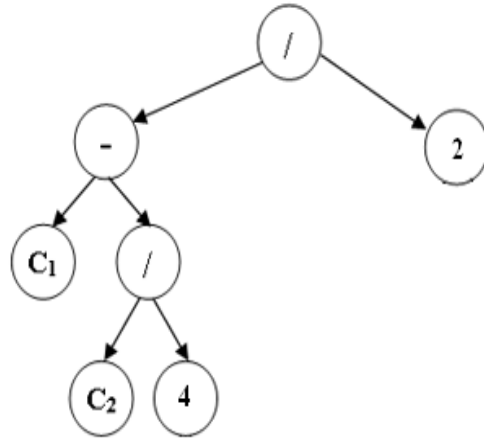


Figure 2.18: Step3. The syntax tree

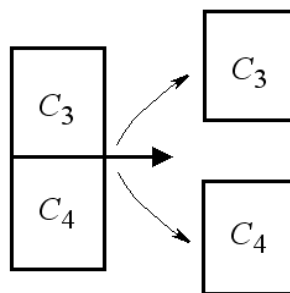


Figure 2.19: Step4. A horizontal cut

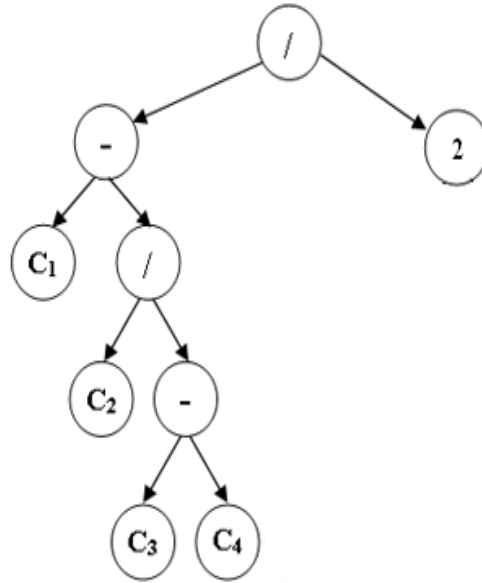


Figure 2.20: Step4. The syntax tree

We determine the all left side of the syntax tree corresponding to our covering model using the two kinds of cuts. The partial Polish form obtained from the left side of the syntax tree is:

$$\emptyset \ominus C_1 \ominus C_2 \ominus C_3 C_4.$$

Let's consider now the right side of the syntax tree. Using the second component obtained from the first cut that we did at the beginning, we are doing a horizontal cut in Figure 2.21, so we have the decomposition of the set B in two sets which are corresponding to the two new components, one of them F contains the items C_5, C_6, C_7 and the other one G which contains the items C_8, C_9, C_{10} . For Step 5 we have associated the syntax tree from Figure 2.22 and the partial Polish notation obtained for this step is:

$$\emptyset \ominus C_1 \ominus C_2 \ominus C_3 C_4 \ominus .$$

On the fifth component we are doing a vertical cut, a decomposition of the set F in 2 sets like in Figure 2.23, so we have other two components. One of these is the seventh component, let it be the set $H = \{C_5, C_6\}$ and the other one is a leaf of the syntax tree, C_7 , see Figure 2.24. The partial prefix Polish form for the tree from Figure 2.24 is:

$$\emptyset \ominus C_1 \ominus C_2 \ominus C_3 C_4 \ominus \emptyset.$$

Using the seventh component we are trying one horizontal cut, a decomposition of the set H in two items C_5 and C_6 , see Figure 2.25, both leaves of the syntax tree from Figure 2.26. The partial prefix Polish notation till this step is:

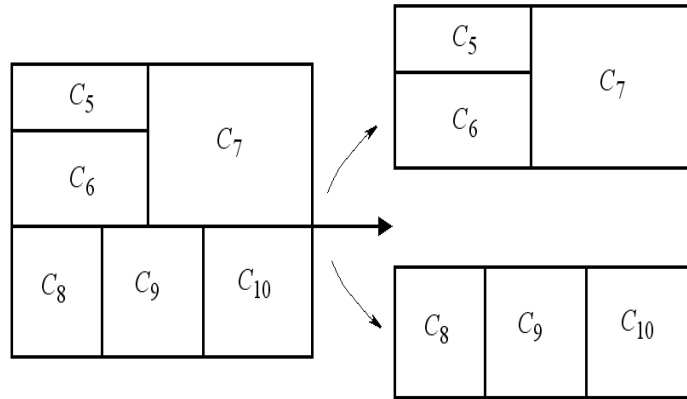


Figure 2.21: Step 5. A horizontal cut

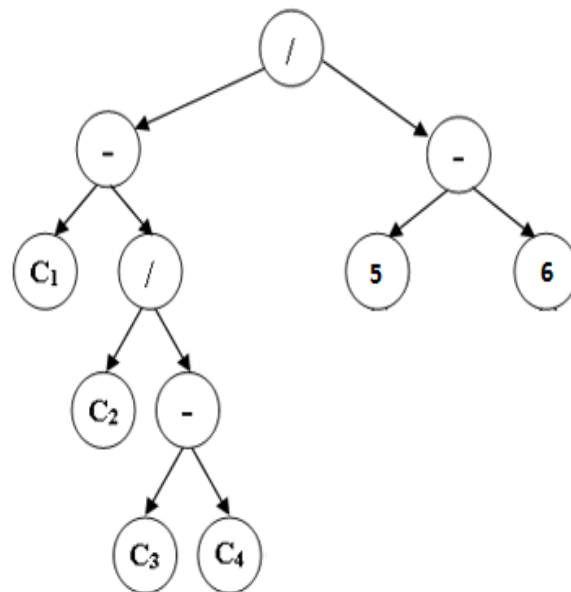


Figure 2.22: Step 5. The syntax tree

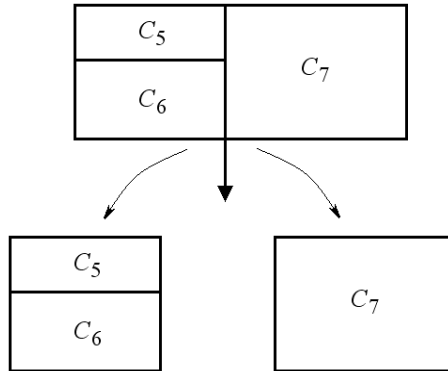


Figure 2.23: Step6. A vertical cut

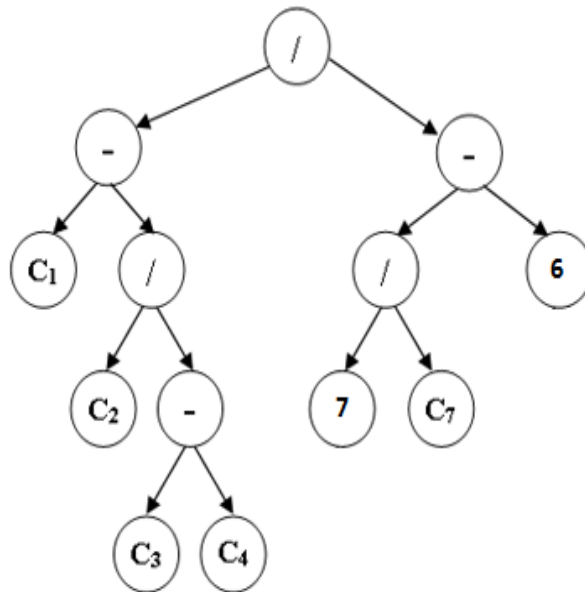


Figure 2.24: Step6. The syntax tree

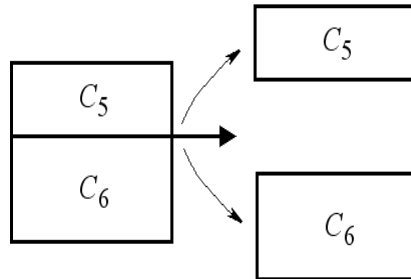


Figure 2.25: Step7. A horizontal cut

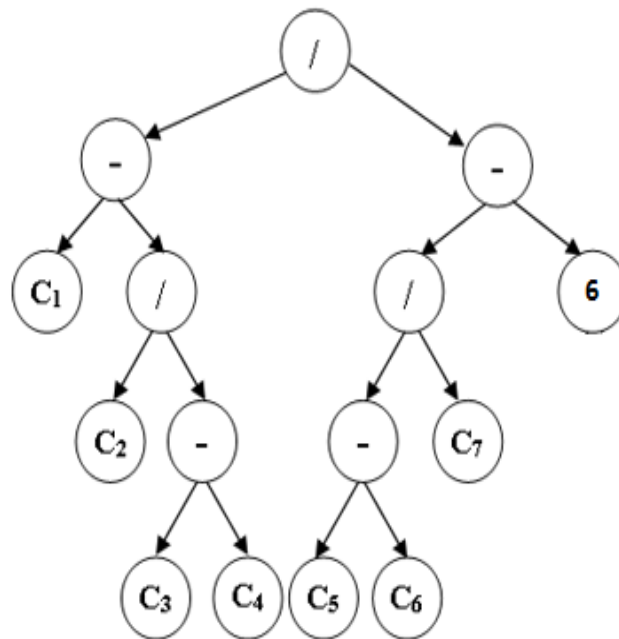


Figure 2.26: Step7. The syntax tree

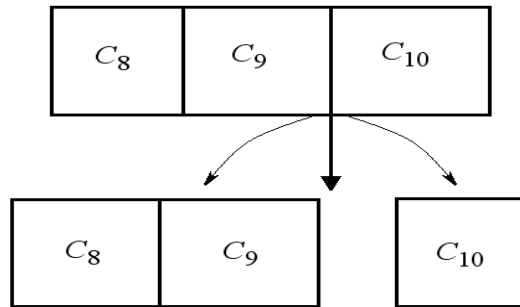


Figure 2.27: Step8. A vertical cut

$$\ominus \ominus C_1 \ominus C_2 \ominus C_3 C_4 \ominus \ominus \ominus C_5 C_6 C_7.$$

Returning to component number 6, we are trying a vertical cut, a decomposition of the set G in two sets, Figure 2.27 one is $I = \{C_8, C_9\}$ and the other one is the item C_{10} , which is a leaf of the syntax tree from Figure 2.28. Our partial Polish form up to this step is:

$$\ominus \ominus C_1 \ominus C_2 \ominus C_3 C_4 \ominus \ominus \ominus C_5 C_6 C_7 \ominus .$$

Using component 8, we are trying a horizontal cut first, but this is impossible so we are doing a vertical one, and this means the decomposition of set I in two items, C_8 and C_9 , Figure 2.29, both leaves of the syntax tree. Finally, we obtain the syntax tree from Figure 2.30, corresponding to the covering model.

The final prefix Polish form, for all the syntax tree is:

$$\ominus \ominus C_1 \ominus C_2 \ominus C_3 C_4 \ominus \ominus \ominus C_5 C_6 C_7 \ominus \ominus C_8 C_9 C_{10}.$$

Guillotine partitions play an important role in many research areas and application domains, e.g., computational geometry, computer graphics, integrated circuit layout, and solid modeling.

We mention here that all the results obtained are in [49] and it completes the obtained results from [48].

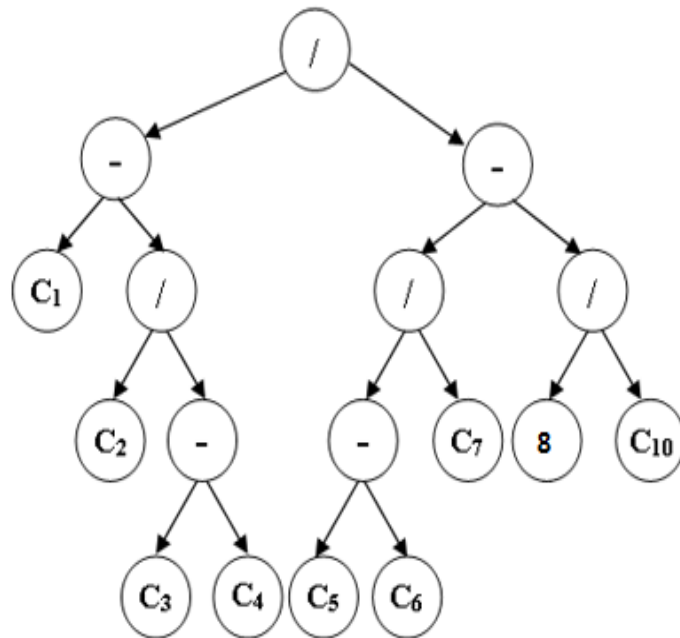


Figure 2.28: Step8. The syntax tree

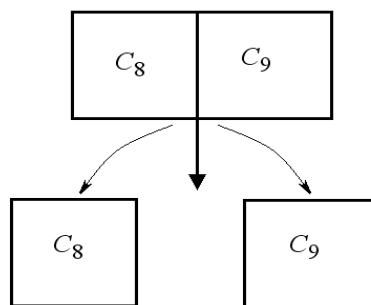


Figure 2.29: Step9. A vertical cut

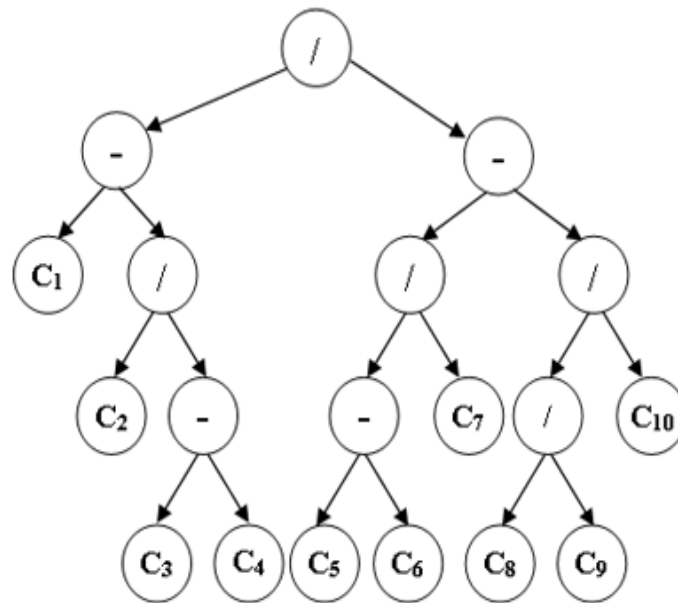


Figure 2.30: The syntax tree

Correctness and complexity

The correctness of the algorithms follows from the Theorems 13 and 14, that make the connection between a guillotine cut and the decomposition of the graph G'_d or G'_r in connex components. For the determining of the Polish notation we preserve only one connex component from this decomposition.

An algorithm for the determination of the connex components has the complexity $O(m)$, where m is the number of the arches. So the complexity of V-CUT or H-CUT is also $O(m)$. It follows that the complexity of PREORDER for a rectangular covering model of k items with guillotine restrictions is $O(km)$.

The problem, the so-called 2-dimensional guillotine problem, is a constraint on a complete partition of 2-dimensional space. Guillotine partitions were introduced in 1980s, and they have numerous applications in computational geometry, computer graphics, etc.

The partitioning of 2-dimensional space is a ubiquitous problem in industry. It appears in many forms from pallet loading to floor tile tessellation. A subset of the problem, the 2-dimensional guillotine problem, is almost as pervasive. Various aspects of the problem are found in industries that produce two dimensional sheets of glass, textiles, paper or other material.

Like the complete partition, the guillotine problem remains NP-hard. For this reason

it is better to use an algorithm for generating an unconstrained covering model and after that to use our algorithm for testing the guillotine restrictions of the pattern.

We note that the results obtained within this section complete the results obtained in [47] and detailed results are found in [51]. Also, all the experimental results are in [48, 49].

The discrete optimization branch of optimization mathematics deals with problems which require the operator to choose an optimal solution from a range of finite possible solutions. The cutting stock problem is a specific example of discrete optimization and it is formulated as an integer programming problem. In this section our aim was to optimally solve a variety of cutting stock problems and determine a set of optimal algorithms for this kind of problems. As these are NP-hard problems, it is obvious that solutions cover a certain class of specific problems.

2.4 The determination of the guillotine restrictions for a rectangular cutting-stock pattern

A frequent constraint in various types of cutting problems, imposed by industrial applications of the two or three dimensional problem, is the so-called guillotine restriction which states that the resulting patterns need to be guillotine cuttable. In literature several techniques that solve the Cutting and Covering problems have been proposed. All these methods result in a pattern or a set of patterns. They are not adequate for constructing cutting patterns when the approach is used to solve the guillotine cutting-stock problem.

In [8] a polynomial algorithm was presented for two special cases of guillotine cutting a rectangle into small rectangles. However, the guillotine restrictions are difficult to respect in the general pattern-generation process. So instead of generating the cutting-stock pattern with guillotine restrictions it is possible to use an analytic method to verify if the pattern, obtained by some methods used in case of non-guillotine cutting, is with or without guillotine restrictions. Nevertheless, the method given in [45] is rather unpractical since the cutting pattern is represented as an array pattern, which implies a large matrix representation.

Another analytical method, presented in [49], used the graph representation of a covering pattern without gaps or overlapping. This method developed an algorithm for guillotine restrictions verification, based on connections between guillotine cut and the connex components of the graphs. But this algorithm is useless in case of a cutting-stock pattern with gaps. For this kind of pattern we propose another method based on two new kinds of graph representations, weighed graph representations, which is more general comparing with the method presented in [49].

We consider a two dimensional rectangular cutting stock problem in case of a cutting pattern with gaps. First we present two new graph representations of the cutting pattern, weighted graph of downward adjacency and weighted graph of rightward adjacency. Using this kind of representation we propose a method to verify guillotine restrictions of the pattern which can be applied for cutting-stock pattern with gaps but also for the covering pattern without gaps and overlapping.

2.4.1 Problem statement and formulation

We consider the rectangular plate \mathcal{P} , characterized by length L and width W . From this we cut k rectangular items, each one is characterized by length l_i and width w_i .

Definition 15. [52] *A rectangular cutting-stock pattern is an arrangement of the k rectangular items C_i , $i = 1, 2, \dots, k$ on the supporting plate \mathcal{P} , so that the borders of the items C_i to be parallel with the borders of the plate \mathcal{P} .*

For this kind of patterns we have presented in [47, 44] two graph representations. Starting from these representations we complete the graphs by adding a value for each arc from the two graphs.

Definition 16. [52] *A rectangular cutting pattern has guillotine restrictions if at every moment of the cutting process the remaining supporting rectangle is separated in two new rectangles by a cut from an edge to the opposite edge of the rectangle and the cutting line is parallel with the two remaining edges.*

In the set of the rectangles $\{C_1, C_2, \dots, C_k\}$ from the covering pattern we define a downwards adjacency relation and a rightwards adjacency relation.

Definition 17. [52] *The rectangle C_i is downward adjacent (rightward adjacent) with rectangle C_j if in the cutting pattern, C_j is to be found downward (respectively rightward) C_i and their borders have at least two common points.*

Let $C = \{C_1, C_2, \dots, C_k\}$ and $R_d, R_r \notin C$. For any covering pattern, we defined in [47, 44], a graph of *downwards adjacency* and another one of *rightwards adjacency*. We define now two new graphs a weighted graph of *downwards adjacency*, G_d , and another one of *rightwards adjacency*, G_r . We will use in these definitions the notation $V(X, Y)$ for the value of the arc (X, Y)

Definition 18. [52] *The weighted graph of downward adjacency $G_d = (C \cup \{R_d\}, \Gamma_d)$ has as vertices the rectangles C_1, C_2, \dots, C_k and a new vertex R_d symbolizing the northern borderline of the supporting plate P . The Γ_d is defined as follows:*

$$\left\{ \begin{array}{l} \Gamma_d(C_i) \ni C_j \text{ if } C_i \text{ is downward adjacent} \\ \quad \quad \quad \text{with } C_j \\ \Gamma_d(R_d) \ni C_i \text{ if } C_i \text{ touches the North border} \\ \quad \quad \quad \text{of the support plate } P \\ V(X, C_j) = w_j, \forall X \in C \cup R_d \text{ and } C_j \in C \end{array} \right.$$

Definition 19. *The weighted graph of rightward adjacency $G_r = (C \cup \{R_r\}, \Gamma_r)$, where R_r symbolizes the western border. The Γ_r is defined as follows:*

$$\left\{ \begin{array}{l} \Gamma_r(C_i) \ni C_j \text{ if } C_i \text{ is rightward adjacent} \\ \quad \quad \quad \text{with } C_j \\ \Gamma_r(R_r) \ni C_i \text{ if } C_i \text{ touches the West border} \\ \quad \quad \quad \text{of the support plate } P \\ V(X, C_j) = l_j, \forall X \in C \cup R_r \text{ and } C_j \in C \end{array} \right.$$

Let the cutting-stock pattern from Figure 2.31. The weighted graphs G_d and G_r , are represented in Figure 2.32 and Figure 2.33.

We remark that in the graphs G_d and G_r the vertex R_d (respectively R_r) is connected by an arc to the vertex C_i if and only if C_i touches the northern (respectively the western) border of the support \mathcal{P} .

Remark 20. *In the following we consider only the rectangular cutting-stock pattern where the rectangles are not situated under or to the right of an empty spaces. When it is not true (see Figure 2.34), we can define another pattern (equivalent) by moving*

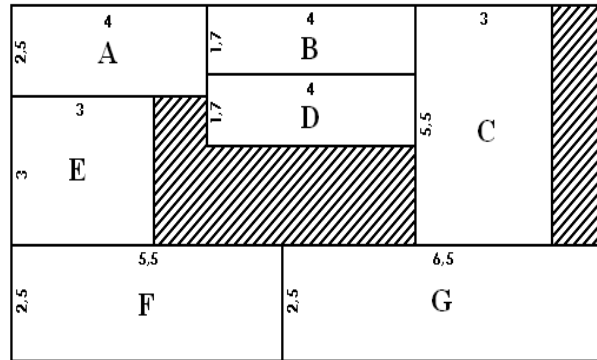


Figure 2.31: The rectangular cutting-stock pattern

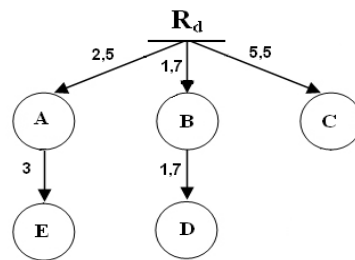


Figure 2.32: The graph G_d

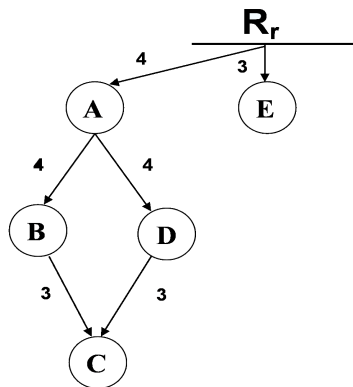


Figure 2.33: The graph G_r

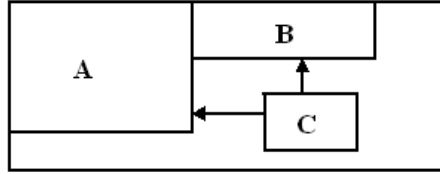


Figure 2.34: Moving directions

the rectangles (in Figure 2.34 this rectangle is C) down or to the left till they touch the border of another rectangle. In the sense of the cutting-stock problem with a minimum rest, for every cutting-stock pattern there is always an equivalent pattern of this form.

From the Remark it results that the weighted graphs G_d and G_r attached to a cutting-stock pattern have the following properties [44]: the graphs are quasi strongly connected and have no circuit.

Let us take a cutting-stock pattern with guillotine restrictions. From [46] it follows that it is possible to represent a rectangular cutting-stock pattern with guillotine restrictions using an expression with two operations:

1. \ominus - the s-line concatenation, an operation for horizontal cuts;
2. \otimes - the s-column concatenation, an operation for vertical cuts.

2.4.2 Cuts determination

In previous section in this thesis, we presented an algorithm for cuts determination in case of a cutting pattern without gaps. But it is not possible to apply this algorithm in the case of a cutting-stock pattern with gaps.

Starting from a rectangular cutting-stock pattern with gaps we intend to find a connection between guillotine restrictions and the two weighted graphs of adjacency, G_d and G_r .

For this purpose we will use the notation $Lpd(R_d, C_i)$, respectively $Lpr(R_r, C_i)$ for the length of the path from R_d to C_i in the graphs G_d , respectively G_r . We remark that $Lpd(R_d, C_i)$ is the distance from the northern border of the plate P to the southern border of piece C_i and similarly $Lpr(R_r, C_i)$ represents the distance from the western border of the plate P to the eastern border of piece C_i .

Remark 21. *If a cutting-stock pattern has a horizontal guillotine cut situated at a distance M from the North border of the supporting plate P then the set of the items, C , can be separated in two subsets S_1 , the set of the items situated above this cut, and S_2 the set of the items situated below this cut. Of course in the weighted graph G_d we have:*

1. $Lpd(R_d, C_i) \leq M$ for every $C_i \in S_1$;

2. $Lpd(R_d, C_i) > M$ for every $C_i \in S_2$.

We obtain a similar result if the cutting-stock pattern has a vertical cut.

The two conditions from the above remarks are necessary but are not sufficient because it is possible for the cut to intersect some items from the set S_2 . We present in the following the necessary and sufficient conditions for a guillotine cut.

Theorem 22. [52] *Take a rectangular cutting-stock pattern with possible gaps and the weighted graph G_d attached to the pattern. The cutting-stock pattern has a horizontal guillotine cut on the distance M from the northern border of the supporting plate if and only if it is possible to separate the sets of the items, C , in two subsets, S_1 and S_2 so that:*

1. $C = S_1 \cup S_2, S_1 \cap S_2 = \emptyset$;
2. For every $C_j \in C$ so that $(R_d, C_j) \in \Gamma_d$ it follows that $C_j \in S_1$;
3. $Lpd(R_d, C_i) \leq M$ for every $C_i \in S_1$;
4. If there is $C_j \in S_1$ so that $Lpd(R_d, C_j) < M$ then all direct descender of C_j will be in S_1 .

Proof. i. Suppose that the cutting-stock pattern has a vertical guillotine cut. That means that the sets of items C can be separated in two subsets, S_1 , the set of the vertices situated above the cut, and S_2 , the set of the vertices situated below the cut. From the Remark 21 it follows that conditions 1, 2 and 3 are fulfilled.

Suppose that condition 4 is not fulfilled. That means there are two items $C_j \in S_1$ and $C_i \in S_2$ so that $Lpd(R_d, C_j) < M$ and the item C_i is a successor of C_j . Because $C_i \in S_2$ it follows that $Lpd(R_d, C_i) > M$ and a horizontal cut situated on the distance M from the northern border of the supporting plate will intersect the item C_i . It means that without the condition 4 it is impossible to separate the set of the items by a horizontal cut. So our supposition that the condition 4 is not fulfilled is false.

ii. Suppose all the conditions 1-4 are fulfilled but it is not possible to make a horizontal cut on the distance M in the cutting-stock pattern. It follows that there is at least item $C_i \in S_2$ which is intersected by such a cut. It means that the distance from the northern border of the supporting plate to the northern border of the item C_i is less than M and the distance from the northern border of the supporting plate to the southern border of the item C_i is greater than M .

But from the Remark 21 it follows that the northern border of the item C_i is identical with the southern border of some item C_j , situated above C_i . That means $(C_j, C_i) \in \Gamma_d$ and $Lpd(R_d, C_j) < M$ and so $C_j \in S_1$. From condition 4, because C_i is a successor of C_j , it follows that C_i must be in S_1 in contradiction with our hypothesis. That means that if the conditions 1-4 are fulfilled then there is a horizontal guillotine cut in the cutting-stock pattern.

We obtain a similar result if we consider the weighted graph of rightwards adjacency. □

2.4.3 The algorithm for the verification of the guillotine restrictions

The results from the previous theorem suggest an algorithm for the verification of the guillotine restrictions, in case of a cutting-stock pattern with gaps.

Input data: The weighted graphs G_d or G_r attached to a rectangular cutting pattern.

Output data: The s-pictural representation of the cutting pattern like a formula in a Polish prefixed form.

Method: The algorithm constructs the syntax tree for the s-pictural representation of the cutting pattern, starting from the root to the leaves (*procedure PRORD*). For every vertex of the tree it verifies if it is possible to make a vertical (*procedure VCUT*) or horizontal cut (*HCUT procedure*), using the algorithm for the decomposition of a graph in two components, S_1 and S_2 .

We will use the following notations:

- G'_r, G'_d are the subgraphs of $G_r|_X$, respectively $G_d|_X$, where we can add, if it is necessary, the root $R_r(R_d)$ and the arcs starting from $R_r(R_d)$, like in Definition 18.

- $L_1(L_2)$ is the weight of the first (second) cutting support which contains all the items from $S_1(S_2)$.

- $succ(C_i|_G)$ is the set of successors of the item C_i in the graph G .

The *ADD()* procedure is used for addition of the next member in the Polish prefixed form.

We remark that we can apply this algorithm also in the case of a cutting-stock pattern without gaps and, of course, in the case of a covering pattern with or without gaps.

Let us take the cutting-stock pattern from Figure 2.31 with the weighted graphs, G_d and G_r [53].

For this example $L = 8$, $W = 5, 5$, and first we are trying a horizontal cut. We obtain two sets, one composed from nodes $\{A, E, B, D, C\}$ and $\{F, G\}$, see Figure 2.35, Figure 2.36 .

In the syntactic tree from Figure 2.37 we have 2 components connected using the operation column concatenation \ominus for the horizontal cut. The prefix Polish notation for this syntactic tree from Figure 2.37 is: \ominus .

Using the first component we are trying a vertical cut, we have $maxM = 2.5$. In Figures 2.38 and 2.39, we have the decomposition in two other sets, one of them contains the items $\{A, E\}$, and the other one $\{B, D, C\}$.

In Figure 2.40 we are adding to the syntactic tree the components 3 and 4 which are connected using the operation of column concatenation \oslash for the vertical cut.

The prefix notation for this tree from Figure 2.40 is:

$$\ominus \oslash .$$

We continue to make horizontal or vertical cut for the left and right components from the syntactic tree until every component contains only one item from the covering pattern.

[52] **PROCEDURE PRORD**($G, C, L, W, ADD()$)

begin

VCUT($G_r, C, L, W, err, S_1, S_2, L_1, L_2$);

if $err = 0$ **then**

if $|C| = 1$ **then** **ADD**(C);;

else **ADD**(\emptyset); **PRORD**($G_d, S_1, L_1, W, ADD()$);

PRORD($G_d, S_2, L_2, W, ADD()$);;

end

else

HCUT($G_d, C, L, W, err, S_1, S_2, W_1, W_2$);

if $err = 0$ **then**

if $|C| = 1$ **then** **ADD**(C);;

else **ADD**(\ominus); **PRORD**($G_r, S_1, L, W_1, ADD()$);

PRORD($G_r, S_2, L, W_2, ADD()$);;

end

else No guillotine restrictions;

end

end

PROCEDURE VCUT($G_r, X, L, W, err, S_1, S_2, L_1, L_2$)

begin

$err = 0$; **CONSTRUCT-SUBGRAPH**(G_r, G'_r, X, R_r);

$V := \bigcup \{C_i | C_i \in X, (R_r, C_i) \in \Gamma_r\}$, where all the elements are unmarked

$maxM := \max\{l_i | C_i \in V\}$

$P_i := \{l_i | C_i \in V\}$

while $\exists C_i \in V$ unmarked element **do**

 mark C_i ;

if $P_i < maxM$ **then**

for $C_j \in succ(C_i | G'_r)$ **do**

$V := V \cup \{C_j | C_j \text{ is an unmarked element}\}$;

$P_j := P_i + l_j$;

if $P_j > maxM$ **then**

$MaxM := P_j$;

end

end

end

end

$maxM := \max\{Lpd(R_r, C_i) | C_i \in V\}$

if $maxM = L$ **then**

$err = 1$;

end

$L_1 := maxM$;

$L_2 := L - maxM$;

$S_1 := V$;

$S_2 := X - V$;

end

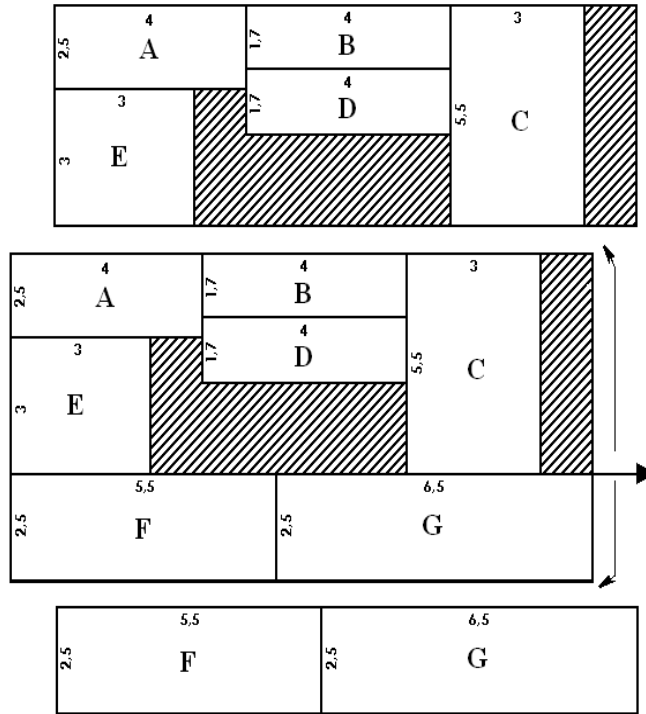


Figure 2.35: The first horizontal cut

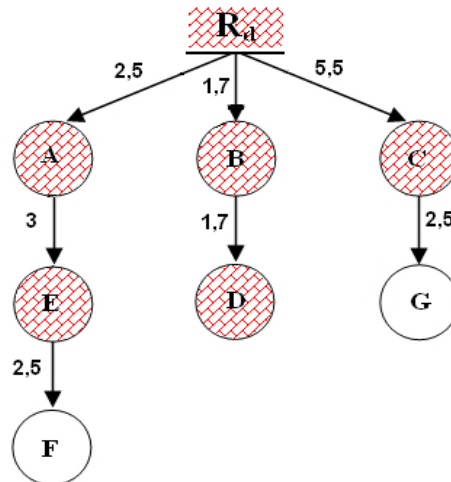


Figure 2.36: The two sets from the horizontal cut

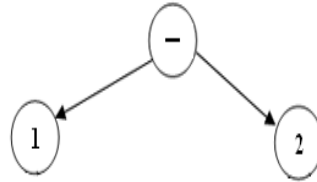


Figure 2.37: The first syntactic tree

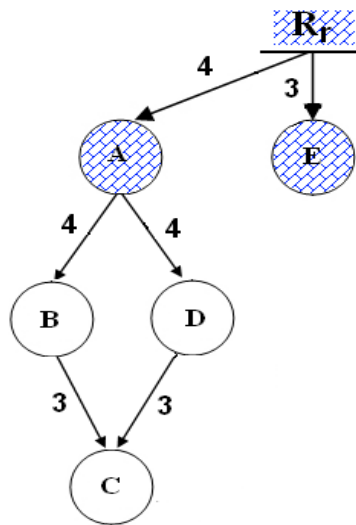


Figure 2.38: The two sets from the vertical cut

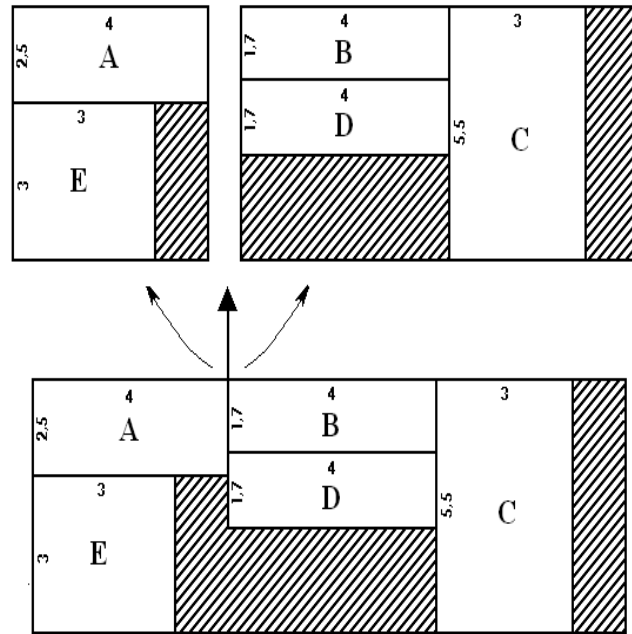


Figure 2.39: A vertical cut

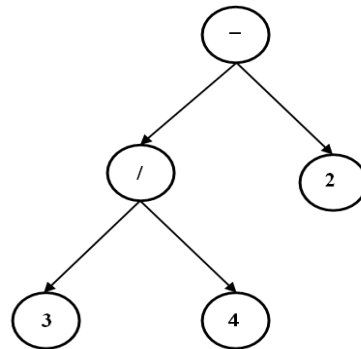


Figure 2.40: The syntactic tree

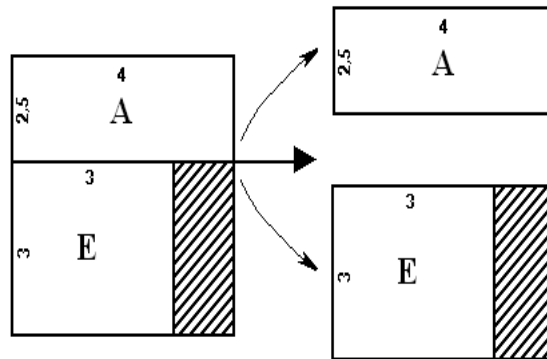


Figure 2.41: One horizontal cut

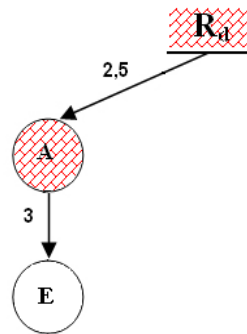


Figure 2.42: The sets obtained from the previous cut

On the third component we are trying to do a horizontal cut, a decomposition of the component 3 in two nodes, A and E . Both of them are leaves of the syntactic tree. We did the horizontal cut in Figure 2.41, and we can see the sets obtained in Figure 2.42.

The syntactic tree associated is presented in Figure 2.43.

The prefixed notation associated to the syntax tree from Figure 2.43 is:

$$\ominus \otimes \ominus AE.$$

On the fourth component, we are looking for a horizontal cut, but this is not possible so we are doing a vertical one, see Figure 2.44 which divides it in a simple node, C and a new component, component number 5. In Figure 2.45 we have made the cuts and in Figure 2.46 we have the syntactic tree, with the leaf of the tree, C .

On the fifth component we are trying a horizontal cut.

We determine the all left side of the syntactic tree corresponding to our covering pattern using the two kinds of cuts.

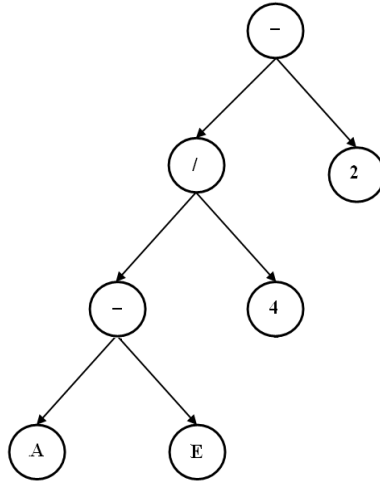


Figure 2.43: The syntactic tree

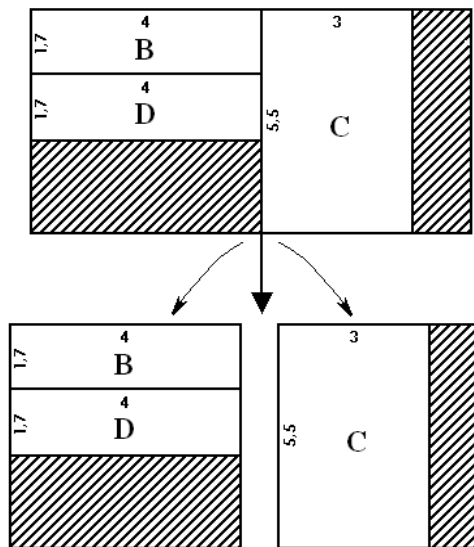


Figure 2.44: A vertical cut on the fourth component

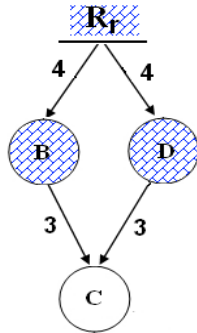


Figure 2.45: The cuts obtained

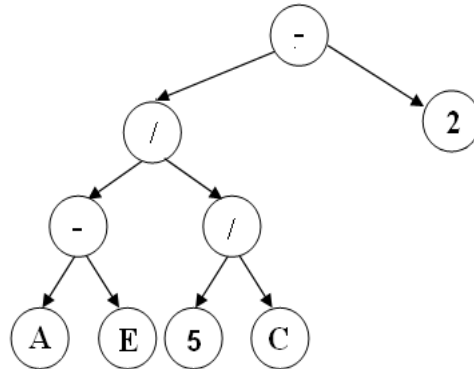


Figure 2.46: The syntactic tree

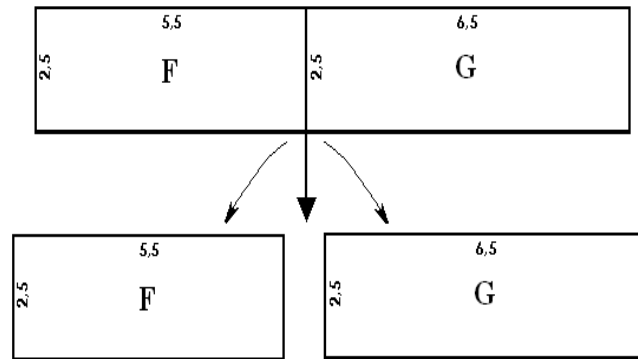


Figure 2.47: The horizontal cut on the second component

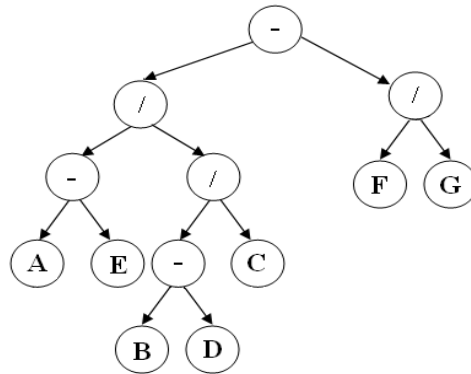


Figure 2.48: The syntactic tree

The Polish notation obtained from the left side of the syntax tree is:

$$\ominus \ominus \ominus AE \ominus \ominus BDC.$$

Let's consider now the right side of the syntactic tree. Using the second component obtained from the first cut that we did at the beginning, we are trying a horizontal cut, but it is not possible, so in Figure 2.47 we have the vertical cut, which means the decomposition of the component 2 in two nodes, F and G .

The syntactic tree derived is in Figure 2.48.

The Polish notation for the tree from Figure 2.48 is [53]:

$$\ominus \ominus \ominus AE \ominus \ominus BDC \ominus FG.$$

All the results exposed here are in [52, 53].

Correctness and complexity

The correctness of the algorithm follows from the Theorem 22, that makes the connection between a guillotine cut and the decomposition of a graph in two subgraphs.

The procedure *PREORD()* represents a preorder traversal of a graph, so the complexity is $O(k)$ [19], where k is the number of the cutting items. Also, into the procedure *VCUT*, respectively *HCUT* we traverse a subgraph of the initial graph. So, the complexity of the algorithm is $O(k^2)$.

No matter if it is a guillotine covering [49] or cutting-stock, the problem is a constraint on a complete partition of two dimensional space. Guillotine partitions were introduced in 1980ies have numerous applications [27] in computational geometry, computer graphics, pattern recognition etc.

Various aspects of the problem are found in industries that produce two dimensional sheets of glass, textiles, paper or other material. A similar problem arises in the design of layouts for integrated circuits or in the design of an optimal placement of a set of solar panels. Like the complete partition, the guillotine problem remains NP hard. For this reason it is better to use an algorithm for generating an unconstrained covering or cutting-stock pattern and, after that or in each step, to use our algorithms for verifying the guillotine restrictions of the generated pattern.

Whether it is a guillotine covering [49] or cutting-stock, with gaps or without gaps, the problem, the so-called two dimensional guillotine problem, is a constraint on a complete partition of two dimensional space. The partitioning of two dimensional space is a ubiquitous problem in industry. It appears in many forms from pallet loading to floor tile tessellation.

A subset of the problem, the two dimensional guillotine problem, is almost as pervasive. Various aspects of the problem are found in industries that produce two dimensional sheets of glass, textiles, paper or other material. A similar problem arises in the design of layouts for integrated circuits, how should the subcircuits the total chip area required should be arranged to minimize or in the design of an optimal placement of a set of solar panels. Like the complete partition, the guillotine problem remains NP-hard. For this reason it is better to use an algorithm for generating an unconstrained covering or cutting-stock pattern and, after that or in each step, to use our algorithms for verifying the guillotine restrictions of the generated pattern.

2.5 Contributions and results

We note that all the studied problems within the present chapter point out some new types of discrete optimization problems which have been studied less so far. The results from this chapter are included in five papers, individual or in joint works. Each of these papers presents some original points of view and optimized algorithms on rectangular

two dimensional cutting problem. The scientific results within this chapter belongs to the author and can be found in the papers [51, 52, 34, 48, 49, 53, 52].

This chapter dealt with the classical two dimensional covering problem. We came up with solutions for particular cases within this problem, by closely studying the covering problem with guillotine cuts limitations. All these problems represent particular cases of discrete optimization.

Chapter 3

Three dimensional bin packing problems

Since computer science started developing over forty years ago, bin packing still continues to be one of the most difficult problems to deal with today. Scientists in the fields of computing and discrete mathematics have been relentlessly trying to analyze and understand this computational puzzle and yet, no algorithm capable of deriving an optimal solution within a reasonable time limit could be found. But fortunately enough, no scientist has ever rejected the possibility of such solution to be available.

bin packing problems alongside other seemingly inexplicable problems are generally classified by theorists as NP-hard and NP-complete problems. So far, there is no optimal solution to the bin packing problem to be derived by a computer without having to derive almost all possible solutions. To be more precise, finding the best match solution to one complex instance of the bin packing problem by means of the highest performance computer ever would take months, or even years.

Nevertheless, as logicians study, evaluate and experiment, industries depend on solutions for such fine problems, even if they are imperfect solutions. Production industries make use of bin packing in wide areas from television programming to designing automobiles and aircrafts. In many such cases, if a computer is capable of revealing a near to perfect solution within reasonable deadlines, this specific solution will temporarily meet the needs of the industries concerned. Therefore, theorists and scientists developed approximation algorithms to bin packing problems, see [62, 18].

This section will specifically show new, original discrete optimization solutions to the three dimensional version of bin packing. bin packing in three dimensions, or box packing received significant attention in computer science during the past decade despite its great importance to industry and computer science.

We consider the rectangular three dimensional bin packing problem with one finite bin, where the bin is packed with a set of rectangular boxes, without gaps or overlapping. Starting from a solution of the three dimensional bin packing model, our objective is to

determine an order of the loading the boxes in the bin so that a box is packed in the bin only if there are no empty spaces down to this box and the origin of the box is in a fixed position, determinate by the boxes situated in the West and North neighborhood.

3.1 Basic notions concerning three dimensional bin packing problems

The three dimensional bin packing optimization problem deals with fitting an arbitrary number of box-shaped items of various sizes into a box-shaped three dimensional area, with utmost efficiency. Approximation algorithms are the tool which enables the user to accommodate as many objects in the least amount of time. Since this is a NP-hard problem, three dimensional bin packing is an aspect of great academic relevance for computer scientists worldwide. In the more practical fields of our contemporary lives, this problem holds a great amount of interest in industrial contexts such as shipping cargo and designing machinery with replaceable parts, such as medical devices or automobiles. The modern industry continuously relies on algorithms to provide optimal solutions to such common problems.

bin packing problem has been shown to be NP complete in the strong sense, see [26]. Various polynomial time approximation algorithms and numerous heuristics have been designed during the last decades. For example, the First Fit Decreasing algorithm is a simple greedy strategy that places items one after each other in decreasing order of their size into the first bin they fit in. FFD has been shown to use not more than $71/60OPT + 1$, where OPT is the optimal solution value [26]. There are Asymptotic Polynomial Time Approximation Schemes which are able to solve bin packing problem to any fixed percentage of OPT if the input is sufficiently large. As bin packing is strongly NP-complete, there is no fully polynomial time approximation scheme as long as $P \neq NP$.

However, there are some special cases in which bin packing can be solved to optimality in polynomial time. For example, if all item sizes are divisible by each other First Fit Decreasing algorithm was shown to produce optimal packing. If there are only two item sizes with high multiplicities, there is an $O(\log^2 C)$ algorithm, where C is the maximum bin capacity. Furthermore, there are polynomial approximation algorithms for the high-multiplicity case with a small number of distinct item sizes [25].

Countless are the applications for bin packing and manifold the techniques that have been applied to the problem over the years. Our individual motivation in writing a chapter of this thesis in the area of bin packing has multifaceted character: firstly, we are particularly interested in efficiently solvable special cases of the problem as solutions of provable quality can be obtained within polynomial time and secondly, we break out in the direction of a popular programming paradigm named constraint programming. Within this scope, we are interested in the applicability and algorithmic evaluation of bin packing as a constraint. Thirdly, we are driven by the challenge to efficiently solve

large highlyconstrained bin packing problems arising from practical applications.

This thesis highlights four individual areas of research which are interconnected to each other by the central issue of Three Dimensional bin packing. It presents novel theoretical considerations, innovative algorithmic techniques and approaches, a variety of applications, and different practical examples in the setting of the Three Dimensional bin packing problem.

3.2 A topological order for a rectangular three dimensional bin packing problem

This section extends a previous work [55] regarding the two dimensional covering problem to a rectangular three dimensional bin packing problem, where a bin is packed with a set of rectangular boxes, without gaps or overlapping.

Our objective here is to determine an order of the loading the boxes in the bin so that a box is packed in the bin only if there are no empty spaces down to this box and the origin of the box is in a fixed position. We define a graph representation, of the packing model in order to obtain an acyclic graph. Starting from this graph it is possible to obtain a topological order of the boxes. At the end we present a kind of topological sorting algorithm for this problem, of linear complexity.

The bin packing optimization problem concerns efficiently placing box-shaped objects of arbitrary size and number into a box-shaped container. Such problems are also commonly referred to as Cutting and Packing problems in [24].

Many one dimensional, two dimensional and three dimensional bin packing problems were considered within this approach, altogether with the specific restraints for the industry concerned. The three dimensional bin packing problems maintain the difficulties of one dimensional and two dimensional bin packing problems, while also priding on unique and important applications. As already known, each object exists in three dimensions, representative for three coordinates: width, length and height. Thus, each of the boxes should efficiently fit into one or several bins.

Like two dimensional bin packing, each box must stay orthogonal, or maintain its orientation in the packing. If two dimensional bin packing equates to rectangle-to-floor plan packing [62], three dimensional bin packing equates to box-to-room packing. Three dimensional bin packing may involve a single bin or multiple bins. The singular bin packing problem involves only one bin with either definite or infinite volume. Bins with infinite volume are defined with finite length and width, but with height extending to infinity. This allows packing solutions to pack until the set of boxes is exhausted. Solutions dealing with infinitely sized bins generally carry most compressing objects effectively.

Another way to approach this problem is by considering multiple bins. Each bin has a definite volume. In this way, if the volume of the objects exceeds the volume of the room, an algorithm must make choices of which boxes to include in the packing and

which to throw away. This approach is good for deterministic approaches to the box packing problem.

All these problems focus around one question: "Are the boxes large enough to accommodate these items?" As in all bin packing problems, there are further constraints to be added to this problem, so as to mimic a real-life situation. For instance, gravity is the constraint which causes boxes to rest on the floor or on pallets in warehouses. Weight distribution represents a constraint, as well. As real objects possess mass and weights, a packing solution may need equal weight distribution within a specific storage area. An equal distribution of masses instead of bulking all items in a single location would prevent a boat from tipping or a truck from becoming unstable, for instance. At the same time, it would be unadvisable to place large weight items on top of lighter objects, such as a one kilo shoe box. Moreover, time might be another constraint when, for instance, the quicker the packing process, the better. What it is known as the classic bin packing problem fails to take into consideration all of the above mentioned constraints, since all these more complex problems allow simplification.

The three dimensional bin packing problem holds importance to many fields. Shipping and moving industries, architecture, engineering, and design are all areas where three dimensional bin packing could apply. Industry uses bin packing for everything from scheduling television programming to stacking cargo in a semi-truck to designing automobiles and airplanes.

Many models and algorithms are developed for bin packing problem such as: formulation as a mixed integer program, which can solve the small sized instance to optimum value [37], genetic algorithms [42], or approximation algorithms.

While an approximation algorithm become a guide that attempts to place objects in the least amount of space and time, a mixed integer program gives solution as the position of the boxes in the bin. For this situation we need a plan for packing the boxes in the bin. This is our objective, to determine the order of packing the boxes in the bin if we know the covering solution of the bin.

3.2.1 Problem statement and formulation

Let \mathcal{P} be a rectangular bin, characterized by length l , width w , height h . The bin \mathcal{P} is filled with k rectangular boxes C_1, C_2, \dots, C_k without gaps or overlapping. A box C_i is characterized by length l_i , width w_i , height h_i . We consider a coordinate system $xOyz$ so that the corner O of the bin is the origin of the coordinate system like in Figure 3.1. We will use the following notations:

- $ABCO$ is the bottom face of the bin
- $GDEF$ is the top face of the bin
- $OADG$ is the West face of the bin
- $OCFG$ is the North face of the bin

- $EBCF$ is the East face of the bin
- $ABDE$ is the South face of the bin
- $O - C_i$ is the O-corner of the box C_i of coordinates x_i, y_i, z_i

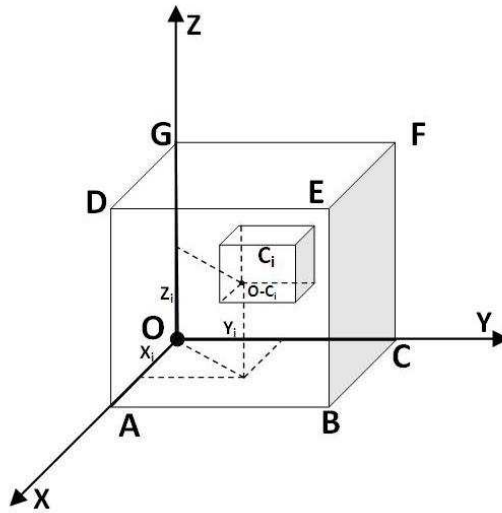


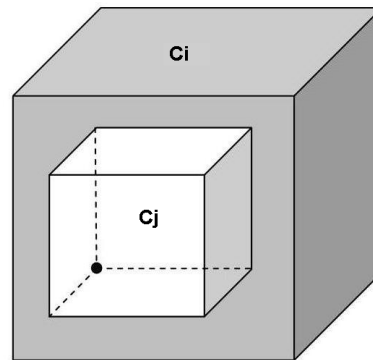
Figure 3.1: The position of box C_i in the bin

By extending the two dimensional covering model [44], we define three kinds of adjacency relations between the boxes C_i and C_j from the bin, adjacency in the direction of Ox , Oy and Oz , Figure 3.1.

Definition 23. [56] *The box C_i is adjacent in Ox direction with the box C_j in the bin \mathcal{P} (Figure 3.2) if the South face of C_i and the North face of C_j have three non-collinear common points, one of which is $O - C_j$.*

Remark 24. *From Definition 23 it follows that if C_i is adjacent with C_j in the direction of Ox we will have:*

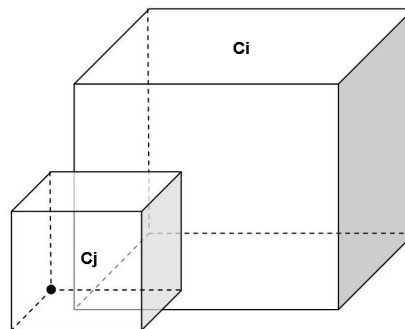
$$x_i < x_j$$

Figure 3.2: Adjacency in Ox direction

$$y_i \leq y_j$$

$$z_i \leq z_j$$

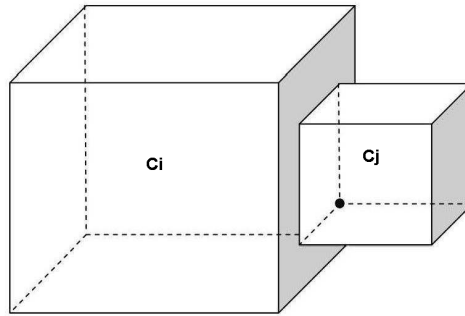
In Figure 3.3 is one situation when the two boxes are not adjacent in the direction of Ox .

Figure 3.3: Not adjacency in Ox direction

Definition 25. [56] The box C_i is adjacent in Oy direction with box C_j in the bin \mathcal{P} (Figure 3.4) if the East face of C_i and the West face of C_j have three non-collinear common points, one of which is $O - C_j$.

Remark 26. From Definition 25 it follows that if C_i is adjacent with C_j in Oy direction we will have:

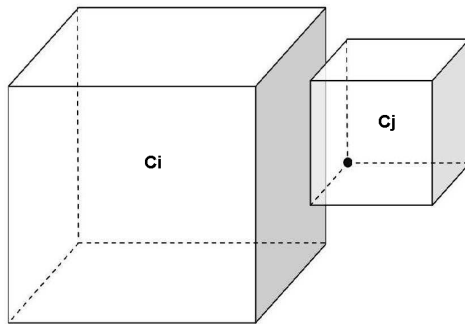
$$y_i < y_j$$

Figure 3.4: Adjacency in Oy direction

$$x_i \leq x_j$$

$$z_i \leq z_j$$

In Figure 3.5 is one situation when the two boxes are not adjacent in the direction of Oy .

Figure 3.5: Not adjacency in Oy direction

Definition 27. [56] The box C_i is adjacent in Oz direction with box C_j in the bin \mathcal{P} (Figure 3.6) if the North face of C_i and the South face of C_j have three non-collinear common points.

Remark 28. In Definition 27 it is not necessary for C_i and C_j to have $O - C_j$ as a common point because our purpose is to give an order of packing and from this point of view we will pack the box C_j only if all the boxes situated downward C_j were already packed. For this reason if C_i is adjacent in Oz direction with C_j conclude that $z_i < z_j$ and there are no more conditions.

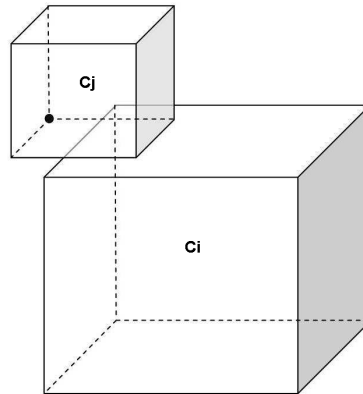
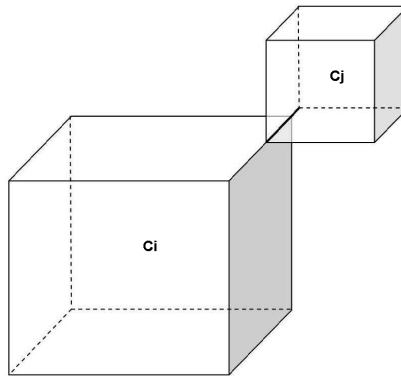
Figure 3.6: Adjacency in Oz directionFigure 3.7: Not adjacency in Oz and Oy directions

Figure 3.7 is one situation when the two boxes are not adjacent in the direction of Oz and Oy .

Starting with these three kinds of adjacency we define three kinds of graphs:

- G_{Ox} - the graph of adjacency in direction Ox ;
- G_{Oy} - the graph of adjacency in direction Oy ;
- G_{Oz} - the graph of adjacency in direction Oz .

Definition 29. [56] The graph of adjacency in Ox direction is $G_{Ox} = (C \cup x, \Gamma_{Ox})$, where the vertices are the boxes from $C = (C_1, C_2, \dots, C_k)$ and X represents the face $GOCF$ situated on the yOz plane, and

$$\left\{ \begin{array}{l} \Gamma_{Ox}(C_i) \ni C_j \text{ if } C_i \text{ is adjacent in} \\ \text{direction } Ox \text{ with } C_j \\ \Gamma_{Ox}(X) \ni C_i \text{ if the North face of } C_i \\ \text{touches the } yOz \text{ plan} \end{array} \right.$$

Definition 30. [56] The graph of adjacency in Oy direction is $G_{Oy} = (C \cup y, \Gamma_{Oy})$. where the vertices are the boxes from $C = (C_1, C_2, \dots, C_k)$ and Y represents the face $DAOG$ situated on the xOz plane, and

$$\left\{ \begin{array}{l} \Gamma_{Oy}(C_i) \ni C_j \text{ if } C_i \text{ is adjacent in} \\ \text{direction } Oy \text{ with } C_j \\ \Gamma_{Oy}(Y) \ni C_i \text{ if the West face of } C_i \\ \text{touches the } xOz \text{ plan} \end{array} \right.$$

Definition 31. [56] The graph of adjacency in Oz direction is $G_{Oz} = (C \cup z, \Gamma_{Oz})$. where the vertices are the boxes from $C = (C_1, C_2, \dots, C_k)$ and Z represents the face $ABCO$ situated on the xOy plane, and

$$\left\{ \begin{array}{l} \Gamma_{Oz}(C_i) \ni C_j \text{ if } C_i \text{ is adjacent in} \\ \text{direction } Oz \text{ with } C_j \\ \Gamma_{Oz}(Z) \ni C_i \text{ if the bottom face} \\ \text{of } C_i \text{ touches the } xOy \text{ plan} \end{array} \right.$$

Example 1.

Let us consider the packing model from Figure 3.8 and 3.9.

Then the G_{Ox} , G_{Oy} and G_{Oz} are the graphs from Figures 3.10, 3.11, 3.12.

The graphs G_{Ox} , G_{Oy} and G_{Oz} have important proprieties:

Theorem 32. The graphs G_{Ox} , G_{Oy} and G_{Oz} for a packing model are acyclic.

Proof:

We will prove the theorem for G_{Ox} graph because for G_{Oy} and G_{Oz} the proof is similar.

Let the graph G_{Ox} for a packing model of a bin \mathcal{P} with the boxes C_1, C_2, \dots, C_k . Assume that the graph G_{Ox} is cyclic. That means there is a simple path in G_{Ox} which leaves from an element C_{i_1} and returns to C_{i_1} .

Let this path be $\mu = [C_{i_1}, C_{i_2}, \dots, C_{i_1}]$.

From Remark 24 it follows that:

$$x_{i_1} < x_{i_2} < \dots < x_{i_\mu} < x_{i_1} \Rightarrow x_{i_1} < x_{i_1}$$

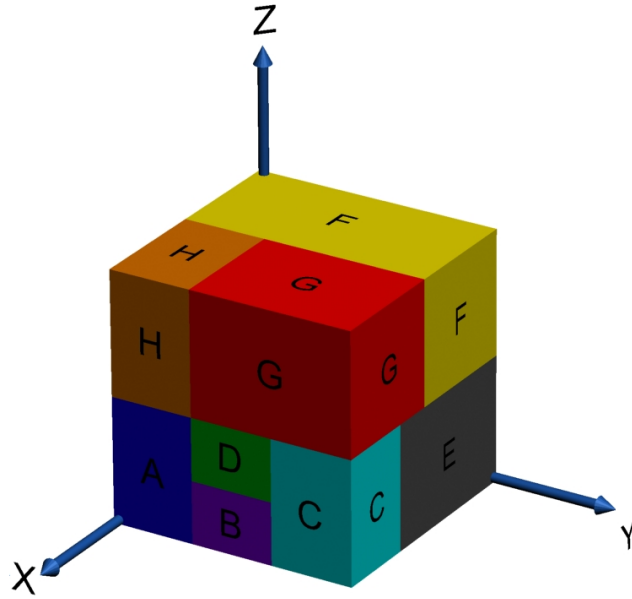


Figure 3.8: The Packing model from the Example 1.

This is impossible, so the presumption G_{Ox} is cyclic, is false.

Theorem 33. [56] Assume we have the graphs G_{Ox} , G_{Oy} and G_{Oz} for a packing model of the bin \mathcal{P} with boxes C_1, C_2, \dots, C_k . We have the following properties:

- if $C_j \in \Gamma_{Ox}(C_i)$ then $C_j \notin \Gamma_{Oy}(C_i) \cup \Gamma_{Oz}(C_i)$
- if $C_j \in \Gamma_{Oy}(C_i)$ then $C_j \notin \Gamma_{Ox}(C_i) \cup \Gamma_{Oz}(C_i)$
- if $C_j \in \Gamma_{Oz}(C_i)$ then $C_j \notin \Gamma_{Ox}(C_i) \cup \Gamma_{Oy}(C_i)$

Proof:

Assume that $C_j \in \Gamma_{Ox}(C_i)$. It follows that C_i is adjacent with C_j in the Ox direction. From Definition 23 it follows that the South face of C_i and the North face of C_j have at least three non-collinear points. Now we can use Remark 28 and it follows is impossible for the East face of C_i and the West face of C_j to have also three non-collinear points. It is impossible also for the top face of C_i and bottom face of C_j to have three non-collinear points. The results for the other two situations are similar.

3.2.2 The compound graph for the packing problem

Due to Theorem 33 it is possible to represent simultaneously these graphs by single adjacency matrix T , a matrix with elements from the set $0,1,2,3$. So we will use 1 for G_{Ox} , 2 for G_{Oy} , 3 for G_{Oz} and 0 if there is no adjacency.

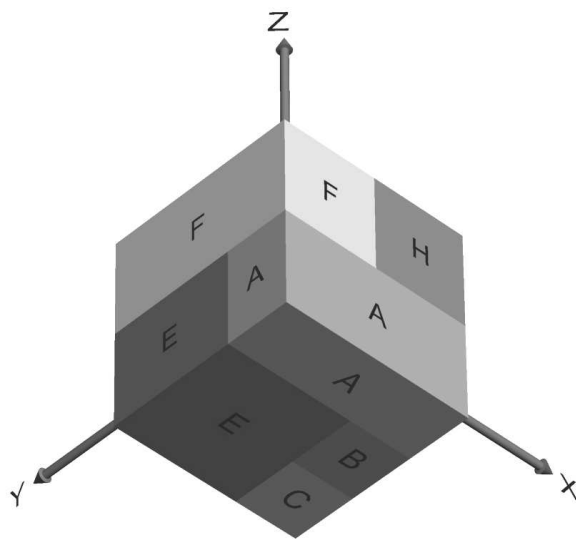


Figure 3.9: The Packing model from the Example 1.

The matrix T is defined like:

$$T_{ij} = \begin{cases} 1, & \text{if } \Gamma_{Ox} \ni C_j \\ 2, & \text{if } \Gamma_{Oy} \ni C_j \\ 3, & \text{if } \Gamma_{Oz} \ni C_j \\ 0, & \text{the other cases} \end{cases}$$

Definition 34. For any packing model we define a network, a graph of compound adjacency $G_c = (C, \Gamma_c)$ where $\Gamma_c(C_i) \ni C_j$ if $T_{ij} \neq 0$. Additionally, the value of the arch (C_i, C_j) is T_{ij} , if $T_{ij} \neq 0$.

Exemple 2.

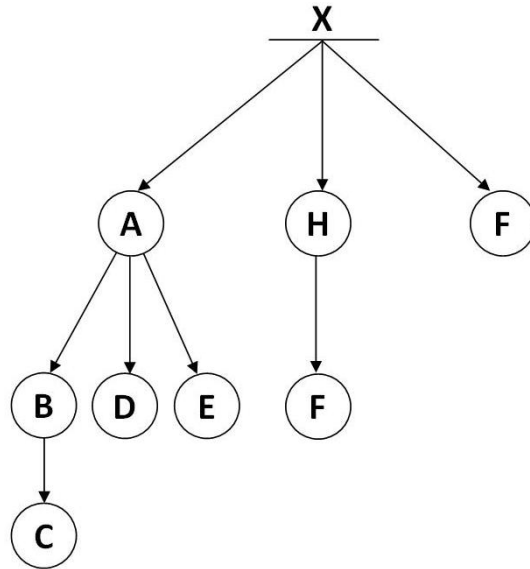


Figure 3.10: Graph G_{Ox}

For the packing model from the Example 1. the matrix T is:

$$T = \begin{matrix} & \begin{matrix} A & B & C & D & E & F & G & H \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 3 & 0 & 3 \\ 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix} \text{ and the graph is represented like in}$$

Figure 3.13.

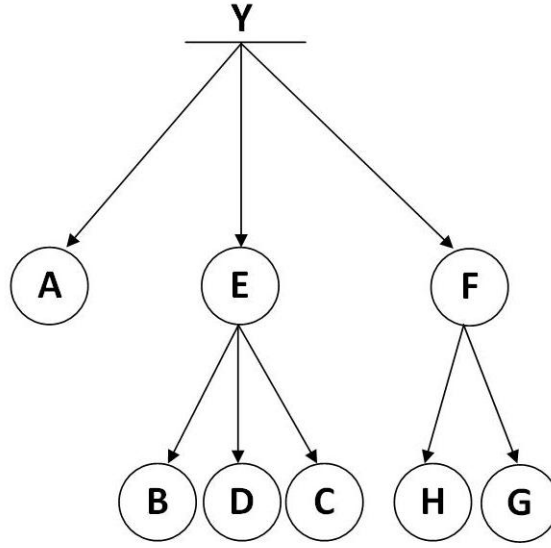
In [55] it is shown that the adjacency relation in one plan which is parallel with xOy define an acyclic compound graph.

Let's see the situation when these relations are three dimensional.

Theorem 35. *The graph G_c for a packing model of the boxes C_1, C_2, \dots, C_k in the bin \mathcal{P} is acyclic.*

Proof:

(i) Assume that we have a cycle $C_{i_1}, C_{i_2}, \dots, C_{i_n}$ in the compound graph G_c is composed only from arches with value 1 and 2. It means that we consider only the arches from G_{Ox} and G_{Oy} . From the Remark 24 and the Remark 26 it follows that:

Figure 3.11: Graph G_{Oy}

$$x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_n}$$

$$y_{i_1} \leq y_{i_2} \leq \dots \leq y_{i_n}$$

and there is at least one index i_k so that $x_{i_k} < x_{i_{k+1}}$ or $y_{i_k} < y_{i_{k+1}}$. It follows that $x_{i_1} < x_{i_n}$ or $y_{i_1} < y_{i_n}$ that means $C_{i_1} \neq C_{i_n}$ and is impossible to have cycle in the graph G_c .

(ii) Suppose that we have a cycle $C_{i_1}, C_{i_2}, \dots, C_{i_n}$ with arches from G_{Ox} and from G_{Oy} and there is at least one arch $(C_{i_k}, C_{i_{k+1}})$ from G_{Oz} . From the Remarks 24, 26 and 28 we have that:

$$z_{i_1} \leq z_{i_2} \leq \dots \leq z_{i_k} < z_{i_{k+1}} \leq \dots \leq z_{i_n}$$

It follows that:

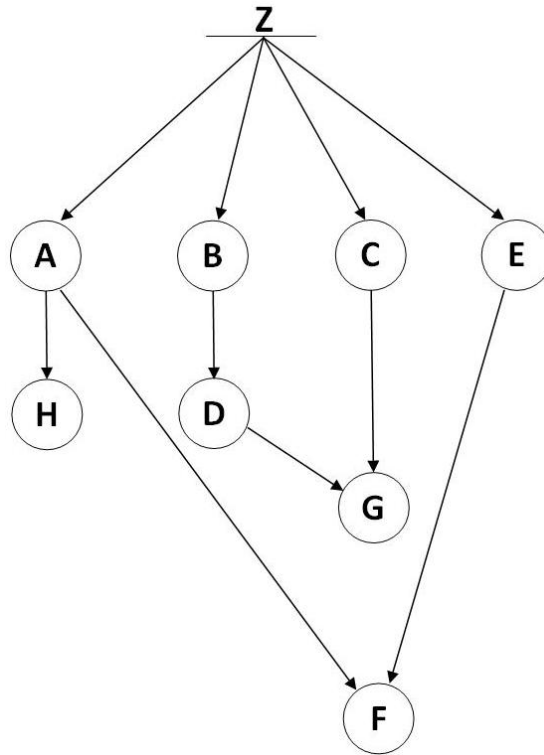
$$z_{i_1} < z_{i_n}$$

and so $C_{i_1} \neq C_{i_n}$ and the path $C_{i_1}, C_{i_2}, \dots, C_{i_n}$ is not a cycle.

Definition 36. [17] A topological sorting of a directed acyclic graph $G = (C, \Gamma)$ is a linear ordering of all its vertices so that, if G contains an arch (C_i, C_j) then C_i appears in the order before C_j .

Theorem 37. There is a topological order of the vertices from the set C in the compound G_c .

Proof:

Figure 3.12: Graph G_{Oz}

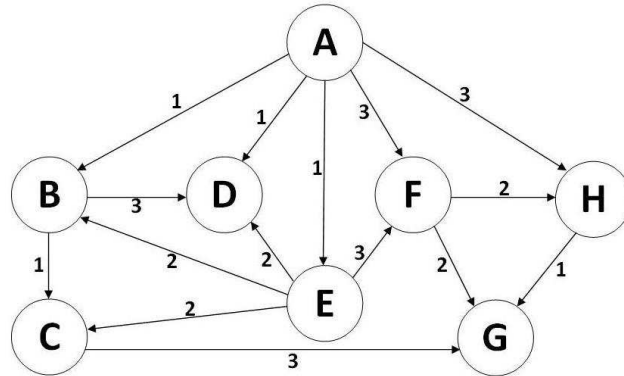
It results directly from Theorem 35 and from [17] because the compound graph G_c is acyclic.

From the definition of the compound graph G_c , a topological order of the vertices from the set C means that if there is an arch from C_i to C_j in G_c (i.e. if C_i is adjacent with C_j in direction of Ox , Oy or Oz) then C_i must appear before C_j in the topological order. Overview the significance of the compound graph G_c for the packing model, it follows that a box C_j is packed only if the corner $O - C_j$ is in a fixed position (with one neighbor box on the West and one neighbor box on the North side) and all the boxes situated bellow C_j , which are adjacent with C_j in Oz direction were already packed. This is a reasonable condition from a practical point of view.

Theorem 38. *The compound graph for a rectangular covering model G_c is a particular network, where there is a single vertex without ascendants.*

Proof:

Let S_1, S_2, \dots, S_k be the topological order of the vertices from the set C . Let C_i so that $O - C_i$ is O , that means C_i is situated in the corner O of box P . From the definition of graphs G_{Ox} , G_{Oy} and G_{Oz} it follows that $C_i \in \Gamma_{Ox}(X) \cap \Gamma_{Oy}(Y) \cap \Gamma_{Oz}(Z)$. If we

Figure 3.13: The compound graph G_c

eliminate the vertices X, Y, Z from G_c it follows that C_i is a vertex without ascendants in the graph G_c . It is evident that $S_1 = C_i$, the first in the topological order. More, for every $S_i \in C$ there is a path from S_1 to S_i .

3.2.3 Topological sorting algorithm

To determine a topological order of the boxes C_1, C_2, \dots, C_k we can use a topological sorting algorithm from [19] or a new algorithm, OVERDIAG-3D which is an extension of previous algorithm from [19]. This algorithm is based on the particularity of the compound graph G_c , respectively on the form of the matrix T , attached to the graph G_c .

OVERDIAG-3D Algorithm

From Theorem 37 it follows that there is a topological order in G_c . Then the matrix T , attached to the compound graph G_c , is an over diagonal matrix with the main diagonal equal to 0, when the vertices are in topological order.

We will base our algorithm on two observations:

1. By changing lines and columns in the adjacency matrix, the number of elements equal to zero remains unchanged ;
2. We can always find a column with the necessary number of zeroes.

For finding the topological order of the set C we extend our matrix $T(k \times k)$ to the matrix $A(k \times (k + 1))$ by attaching a new column, $k + 1$, to the columns of T , which preserve the original number of rows.

Finally this new column, $k + 1$, of the matrix A represents the topological order of the set C .

OVERDIAG-3D(A, n)

```

for  $i = 1, n - 1$ 
   $jc = 0;$ 
   $jc3 = 0;$ 
   $jl = 0;$ 
   $jl3 = 0;$ 
   $j = 1;$ 
  repeat if  $A(i, j) = 0$  then  $j = j + 1$ 
  until  $(A(i, j) \neq 0$  or  $j = n + 1)$ 
   $k = i + 1$ 
  repeat if  $A(k, j) = 0$  then  $k = k + 1$ 
  (test if the column is 0)
  until  $(k = n + 1$  or  $A(k, j) \neq 0)$     if  $num = num + 1$ 
  endif
  endfor
  if  $(num = k - i + 1)$  then
     $ifix = j$ 
    break
  endfor
  { changing line[ $ifix$ ] with line[ $i$ ]
    and column[ $ifix$ ] with column[ $i$ ] }
  for  $j = i$  to  $k$ 
     $T'[ifix, j] \leftrightarrow T'[i, j]$ 
  endfor
  for  $r = i$  to  $k$ 
     $T'[r, ifix] \leftrightarrow T'[r, i]$ 
  endfor
  { now we have to fix two corners of the rectangle }
   $T'[ifix, ifix] \leftrightarrow T'[i, i]$ 
endfor
return

```

Correctness and complexity

Applying the OVERDIAG-3D algorithm we change the order of the vertices C_i so that the matrix T for the compound graph G_c became an over diagonal matrix. It follows that $T_{ij} = 0$ for all $i \geq j$ and it is possible to have $T_{ij} \neq 0$ only if $i < j$.

For the compound graph G_c that means there is an arch from C_i to C_j only if $i < j$ so C_i appears before C_j in the ordered set C . It follows that C is topologically sorted.

Remark that the OVERDIAG-3D algorithm is linear in k^2 , the maximal number of edges in the compound graph G_c .

The three dimensional bin packing problem holds importance to many fields. Shipping and moving industries, architecture, engineering, and design are all areas where three dimensional bin packing could apply.

Another application of the three dimensional bin packing problem is for cutting in the wood industry [61]. Industry uses bin packing for everything from scheduling television programming to stacking cargo in a semi-truck to designing automobiles and airplanes. Recently, the Institute for Algorithms and Scientific Computing in Germany used three dimensional packing for research in molecular biology and chemistry and also with automobile design for manufacturer Mercedes-Benz [73].

A problem here, after the the determination of a packing model, is to determine the order in which the boxes will be packed in the bin, the plan of packing.

This kind of order can be the topological order given by us, where the placement begins with the northwestern-bottom corner of the bin and it ends with a box situated on the top of the bin.

We intend to apply these results and previous results [55] in the field of the Renewable Energy for determine the optimal placement of the set of photovoltaic cells in a field of cells.

All results obtained within this section of the research belong to the author and they can be accessed in [59, 56]. The bin packing problem is a very practical problem, while its general solution is still a subject of debate. We have focused on a class of problems which deal with packing smaller containers into one larger container, so that in the end there is virtually no space left and the setting of the small containers is done according to dimensions. We will have to take into account the size of the containers because in practice it is of utmost importance not to place large containers on top of smaller containers (such as the case when a similar setting would cause damage to containers). The results thus obtained are satisfactory because while imposing a defined set of conditions for the containers, we can assign a topological order which satisfies all restrictions imposed. We also mention the fact that the algorithm was initially implemented for a three dimensional puzzle game, where scores are obtained according to the puzzle pieces loading efficiency. Our intention is to implement such an algorithm for real-life situations, as well.

3.3 The determination of the guillotine restrictions for a rectangular three dimensional bin packing pattern

This section deals with to the rectangular three dimensional bin packing problem, where a bin is loaded with a set of rectangular boxes, without overlapping. One of the most popular restriction for the solution of the three dimensional bin packing problem is the guillotine restriction. That means that the packing patterns are so that the boxes can be obtained by sequential face-to-face cutting plane parallel to a face of the bin.

Our objective is to find a method for verifying if the solution of the three dimensional bin packing problem has the guillotine constraints or not. For this purpose we use a weighed graph representation of a solution of the problem, the generalization of this kind of representation obtained by us for two dimensional cutting stock problem in [48, 49, 50].

From the computational complexity theory point of view, the bin packing problem is a combinatorial NP-hard problem. In it, box-shaped objects of different sizes must be placed into a finite number of bins in a way that minimizes the number of bins used. In [24] many kinds of bin packing problems were presented, one dimensional, two dimensional and three dimensional with many kinds of constraints depending on technological restrictions. Of course the difficulty of the problem increases in three-dimensional bin packing problem as compared to the difficulty of fewer dimensional bin packing problems, but keeps special and important applications.

In the three-dimensional bin packing problem each bin represents triplets containing three values: width, length, and height. Each box should fit into a bin or bins most efficiently. Like two dimensional bin packing, each box must stay orthogonal, or maintain its orientation in the packing. Like all bin packing problems, extra constraints may be added to the problem to create a more real-world-like problem. Such constraints are: gravity, weight distribution or delivery time and so-called guillotine constraint. The guillotine constraint requires that the patterns should be such that the boxes can be obtained recursively by cutting the bin in two smaller bins, until each bin will contain only one box and no box has been intersected by a cut.

In [7] the authors solve a general packing problem, where in each step they test for satisfaction of guillotine constraints. Using some graph representations, defined by us in [56], we present now another guillotine test for the three dimensional bin packing patterns, by generalizing the results obtained in [48, 49] for the two dimensional cutting-stock patterns.

3.3.1 Problem statement and formulation

We consider a three dimensional rectangular bin \mathcal{B} , a container with length L , width W , height H . The bin \mathcal{B} is filled with k rectangular boxes, C_1, C_2, \dots, C_k without overlapping. Every box C_i has length l_i , width w_i , height h_i .

Definition 39. A rectangular 3D-bin packing pattern is an arrangement of the k rectangular boxes C_i in the container \mathcal{B} , so that the faces of the boxes C_i be parallel with the faces of the container \mathcal{B} .

Definition 40. A rectangular bin packing pattern has guillotine restrictions if the bins can be recursively separated in two new bins by a cutting plane which is parallel with a face of the original bin, until each bin contains only one box.

We presented in [44, 56] a representation for a bin packing pattern by means of some graphs of adjacency. Now we complete the graphs of adjacency by adding a value for each arc of these graphs like in [48, 49].

We consider a bin $OABCDEFGH$ and a coordinate system $xOyz$ so that the corner O is the origin of the coordinate system like in Figure 3.1.

We mention that in Figure 3.1, $[AB]$ is the length L , $[BC]$ is the width W and $[AD]$ is the height H . We will use the adjacency relations [56] to express the connections between two boxes C_i and C_j from the bin packing pattern.

Definition 41. The box C_i is adjacent in Ox direction with the box C_j in the bin packing pattern of \mathcal{B} (Figure 3.2), if the South face of C_i and the North face of C_j have at least three non-collinear common points.

Similarly we can define the adjacency relations in the direction Oy and Oz .

Remark 42. In the following we consider only the bin packing pattern where the boxes are not situated above, to the East directions and to the North directions of an empty space. That means every box C_j is adjacent with at least three boxes: one situated down, one to the West and one to the South, or C_j is situated on the down face, respectively West face, or on the South face of the bin. Otherwise we will push the box S downwards, either towards the South or the West directions, like in Figure 3.15 until S will satisfy these conditions.

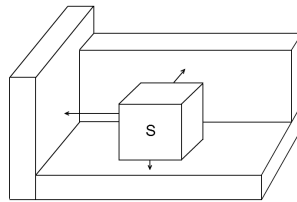


Figure 3.14: The moving directions

Starting from the three kinds of adjacency we have defined in [56] three kind of graphs:

1. G_{Ox} - the graph of adjacency in direction Ox

2. G_{Oy} - the graph of adjacency in direction Oy
3. G_{Oz} - the graph of adjacency in direction Oz .

Now we complete these graphs by adding the values for every arc from the graphs which represent a bin packing pattern with respect to the restrictions from Remark 42.

Definition 43. *The weighed graph of adjacency in Ox direction for the bin packing pattern is $G_{Ox} = (C \cup R_X, \Gamma_{Ox})$, where the vertices are the boxes from $C = C_1, C_2, \dots, C_k$, R_X represents the face $GOCF$ situated on the yOz plane, and*

$$\left\{ \begin{array}{l} \Gamma_{Ox}(C_i) \ni C_j \text{ only if } C_i \text{ is adjacent in} \\ \quad \text{direction } Ox \text{ with } C_j \\ \Gamma_{Ox}(X) \ni C_i \text{ only if the North face of } C_i \\ \quad \text{touches the } yOz \text{ plan} \\ Value(U, C_j) = w_j, \forall U \in C \cup R_X \text{ and } C_j \in C \end{array} \right.$$

Similarly we can define a graph of adjacency in Oy direction and another of adjacency in Oz direction, using w_j respectively h_j for the every value of an incoming arc of C_j .

From the Remark 42 and from [44] it follows that all of the three weighed graphs of adjacency are strongly quasi connected.

Example 1.

We consider a bin packing pattern described in the Figures 3.15 and 3.16 where the bin has the dimensions $L = 3, W = 3, H = 4$ and the boxes are of the dimensions (l_i, w_i, h_i) like in the following:

- the box A of dimension $(1, 3, 2)$
- the box B of dimension $(1, 1, 1)$
- the box C of dimension $(1, 1, 2)$
- the box D of dimension $(1, 1, 1)$
- the box E of dimension $(2, 2, 2)$
- the box F of dimension $(3, 1.5, 2)$
- the box G of dimension $(2, 1.5, 2)$
- the box H of dimension $(1, 1.5, 2)$

Then the G_{Ox}, G_{Oy} and G_{Oz} are the weighed graphs from Figures 3.17 and 3.18.

We observe that the bin packing pattern from Figures 3.15 and 3.16 has guillotine restrictions. Similar with [46] it follows that it is possible to represent a bin packing pattern with guillotine restrictions using a Polish expression with three operations:

1. \oplus - the vertical concatenation, an operation for a horizontal cutting plane;

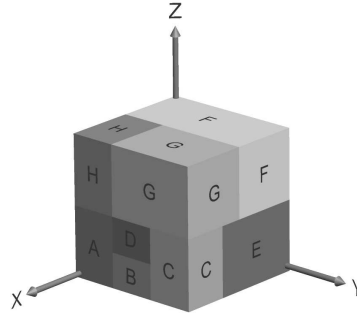


Figure 3.15: A pattern view from the top-right-front corner for Example 1.

2. \ominus - the W-E concatenation, an operation for a vertical cutting plane perpendicular on Ox ;
3. \otimes - the N-S concatenation, an operation for a vertical cutting plane perpendicular on Oy .

For example, the cutting pattern from Figure 3.15 will be described by the following Polish expression:

$$\oplus \otimes A \ominus E \otimes \oplus BDC \ominus F \otimes HG$$

.

3.3.2 Cuts determination

In the previous sections we presented two methods for cuts determination in case of a 2D-cutting pattern without overlapping: one for pattern without gaps [50] and one for the pattern with gaps [48, 49].

Now we consider the 3D-bin packing pattern without overlapping but with possible gaps, with respect to the conditions from Remark 42.

Following the way described in [48, 49] we intend to find a connection between guillotine restrictions and the three weighed graphs of adjacency, G_{Ox} , G_{Oy} and G_{Oz} .

First we will use the notation $Lpd(R_X, C_i)$ for the length of the path from R_X to C_i in the graph G_{Ox} . Similarly we will use the notations $Lpr(R_Y, C_i)$ for the length of the path from R_Y to C_i in the graph G_{Oy} , respectively $Lpr(R_Z, C_i)$ for the length of

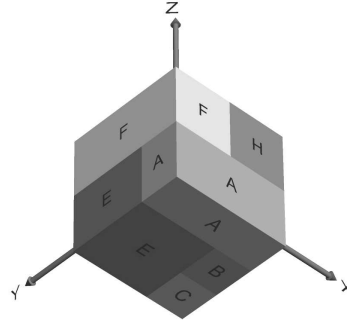


Figure 3.16: A pattern view from the bottom-left-back corner for Example 1.

the path from R_Z to C_i in the graph G_{Oz} . We remark that $Lpd(R_X, C_i)$ represents the distance from the northern face of the bin B to the southern face of box C_i , $Lpr(R_Y, C_i)$ represents the distance from the western face of the bin B to the eastern face of box C_i and $Lpr(R_Z, C_i)$ represents the distance from the bottom face of the bin to the top face of bin C_i .

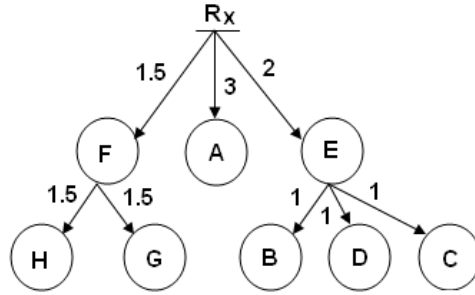
Remark 44. *If a cutting-stock pattern has a horizontal guillotine cutting plane (perpendicular on Oz) situated at a distance M from the down face of the bin B then the set of items, C , can be separated in two subsets B_1 , the set of items situated below this cutting plane, and B_2 the set of items situated above this plane. Of course in the weighed graph G_{Oz} we have:*

1. $Lpd(R_Z, C_i) \leq M$ for every $C_i \in B_1$;
2. $Lpd(R_Z, C_i) > M$ for every $C_i \in B_2$.

We obtain a similar result if the cutting-stock pattern has a vertical cutting plane perpendicular on Ox or a vertical cutting plane perpendicular on Oy .

The two conditions from the above remark are necessary but are not sufficient, because it is possible for the cutting plane to intersect some items from the set B_2 . In the following we present necessary and sufficient conditions for a guillotine cut.

Theorem 45. *Let us consider a 3D-bin packing pattern with possible gaps and the weighed graph G_{Oz} attached to this pattern. The bin packing pattern has a horizontal guillotine cutting plane situated at the distance M from the downwards face of the*

Figure 3.17: Graph G_{O_x} for Example 1.

bin if and only if it is possible to separate the sets of the items, C , in two subsets, B_1 and B_2 so that:

1. $C = B_1 \cup B_2, B_1 \cap B_2 = \emptyset$;
2. For every $C_j \in C$ so that $(R_Z, C_j) \in \Gamma_z$ it follows that $C_j \in B_1$;
3. $Lpd(R_Z, C_i) \leq M$ for every $C_i \in B_1$;
4. If there is $C_j \in B_1$ so that $Lpd(R_Z, C_j) < M$ then all direct descendants of C_j will be in B_1 .

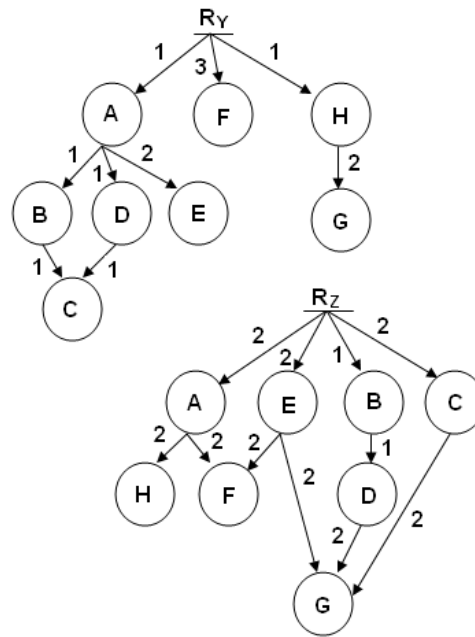
Proof:

i. Suppose that the bin packing pattern has a horizontal guillotine cutting plane and let the weighed graph G_{O_z} attached to the pattern. That means the sets of items C can be separated in two subsets, B_1 , the set of the vertices situated above the cutting plane, and B_2 , the set of the vertices situated below the cutting plane. From the Remark 44 it follows that conditions 1, 2 and 3 are fulfilled.

Suppose that the condition 4 is not fulfilled. That means there are two items $C_j \in B_1$ and $C_i \in B_2$ so that $Lpd(R_Z, C_j) < M$ the item C_i is a direct successor of C_j and suppose that $C_i \in B_2$. It follows that $Lpd(R_Z, C_i) > M$ and a horizontal cutting plane situated on the distance M from the downwards face of the bin will intersect the box C_i . It means that without the condition 4 it is impossible to separate the set of items by a horizontal cutting plane. So our supposition that the condition 4 is not fulfilled is false.

ii. Suppose all the conditions 1-4 are fulfilled but it is not possible to have a horizontal cutting plane at the distance M in the cutting-stock pattern. It follows that there is at least an item $C_i \in B_2$ which is intersected by such a cut. It means that the distance from the bottom face of the bin to the bottom face of the box C_i is less than M and the distance from the downwards face of the bin to the top face of box C_i is greater than M .

But from the Remark 44 it follows that the bottom face of box C_i is identical with the top face of some box C_j , situated downwards C_i . That means $(C_j, C_i) \in \Gamma_z$ and $Lpd(R_Z, C_j) < M$ and so $C_j \in B_1$. From condition 4, because C_i is a direct successor

Figure 3.18: Graphs G_{Oy} , G_{Oz} for Example 1.

of C_j , it follows that C_i must be in B_1 in contradiction with our hypothesis. That means that if conditions 1-4 are fulfilled then there is a horizontal guillotine cutting plane in the bin packing pattern.

We obtain a similar result if we consider the weighed graphs of adjacency in the directions Ox or Oy .

3.3.3 Verification test for guillotine restrictions

From Theorem 45 [57], we obtained an algorithm for the verification of the guillotine restrictions, in case of a bin packing pattern with gaps but without overlapping.

Input data: The weighed graphs G_{Ox} or G_{Oy} or G_{Oz} attached to a bin packing pattern.

Output data: The s-pictural representation of the cutting pattern [46] like a formula in a Polish prefixed form.

Method: Using a depth-first search method, the algorithm constructs the syntactic tree for the Polish expression representation of the cutting pattern, starting from the root to the leaves (procedure PRORD). For every vertex of the tree it verifies if it is possible to make a guillotine cut by a cutting plane perpendicular on Oz (procedure ZCUT) or perpendicular on Ox (procedure XCUT) or perpendicular on Oy (procedure YCUT), using an algorithm for decomposition of a set C of boxes in two subsets, B_1 and B_2 .

We will use for the algorithm the following notations:

- $G'_{Ox}, G'_{Oy}, G'_{Oz}$ are the subgraphs of $G_{Ox}|_U$, respectively $G_{Oy}|_U$ and $G_{Oz}|_U$ where we can add, if it is necessary, the root $R_X(R_Y, R_Z)$ and the arcs starting from R_X (R_Y, R_Z).

- $succ(C_i|_G)$ is the set of successors of the box C_i in the graph G .

The method $ADD()$ is used for addition of the next member in the Polish prefixed form.

```
[58] PROCEDURE PRORD( $G, C, L, W, H, ADD()$ ) begin
  ZCUT( $G_{Oz}, C, L, W, H, err, B_1, B_2, H_1, H_2$ );
  if  $err = 0$  then
    if  $|C| = 1$  then ;
      ADD( $C$ )
    else ADD( $\oplus$ );    PRORD( $G_{Ox}, B_1, L, W, H_1, ADD()$ );
      PRORD( $G_{Ox}, B_2, L, W, H_2, ADD()$ ); ;
    end
  else
    XCUT( $G_{Ox}, C, L, W, H, err, B_1, B_2, W_1, W_2$ );
    if  $err = 0$  then
      if  $|C| = 1$  then ;
        ADD( $C$ )
      else ADD( $\ominus$ );    PRORD( $G_{Oy}, B_1, L, W_1, H, ADD()$ );
        PRORD( $G_{Oy}, B_2, L, W_2, H, ADD()$ ); ;
      end
    else
      YCUT( $G_{Oy}, C, L, W, H, err, B_1, B_2, L_1, L_2$ );
      if  $err = 0$  then
        if  $|C| = 1$  then ;
          ADD( $C$ )
        else ADD( $\odot$ );    PRORD( $G_{Oz}, B_1, L_1, W, H, ADD()$ );
          PRORD( $G_{Oz}, B_2, L_2, W, H, ADD()$ ); ;
        end
      else N;
        o guillotine restrictions
      end
    end
  end
end
```

The procedures ZCUT, YCUT and XCUT are presented below:

We will consider the covering pattern from Figures 3.15 and 3.16, with the weighed graphs G_{Ox}, G_{Oy}, G_{Oz} from Figure 3.17 and Figure 3.18. By examining the weighed graphs we observe that it is possible to make the first horizontal cut by a cutting plane perpendicular on Oz , procedure ZCUT. In Figure 3.19 this first horizontal cut of the

```

PROCEDURE ZCUT( $G_{Oz}, U, L, W, H, err, B_1, B_2, H_1, H_2$ ) begin
   $err = 0$ ; SUBGRAPH( $G_{Oz}, G'_{Oz}, U, R_{Oz}$ );
   $V := \bigcup \{C_i | C_i \in U, (R_{Oz}, C_i) \in \Gamma_{Oz}\}$ , where all the elements are unmarked
   $maxM := \max\{h_i | C_i \in V\}$ 
   $P_i := \{h_i | C_i \in V\}$  while  $\exists C_i \in V$  unmarked do
    mark  $C_i$ ;
    if  $P_i < maxM$  then
      for  $C_j \in succ(C_i \text{ in the graph } G'_{Oz})$  do
         $V := V \cup \{C_j | \text{where } C_j \text{ is unmarked}\}$ ;
         $P_j := P_i + h_j$ ;
        if  $P_j > maxM$  then
           $maxM := P_j$ ;
        end
      end
    end
  end
   $maxM := \max\{Lpd(R_{Oz}, C_i) | C_i \in V\}$ 
  if  $maxM = H$  then
     $err = 1$ ;
  end
  else
     $H_1 := maxM; H_2 := H - maxM; B_1 := V; B_2 := U - V$ ;
  end
end

```

```

PROCEDURE YCUT( $G_{Oy}, U, L, W, H, err, B_1, B_2, L_1, L_2$ ) begin
   $err = 0$ ; SUBGRAPH( $G_{Oy}, G'_{Oy}, U, R_{Oy}$ );
   $V := \bigcup \{C_i | C_i \in U, (R_{Oy}, C_i) \in \Gamma_{Oy}\}$ , where all the elements are unmarked
   $maxM := \max\{l_i | C_i \in V\}$ 
   $P_i := \{l_i | C_i \in V\}$  while  $\exists C_i \in V$  unmarked do
    mark  $C_i$ ;
    if  $P_i < maxM$  then
      for  $C_j \in succ(C_i \text{ in the graph } G'_{Oy})$  do
         $V := V \cup \{C_j | \text{where } C_j \text{ is unmarked}\}$ ;
         $P_j := P_i + l_j$ ;
        if  $P_j > maxM$  then
           $maxM := P_j$ ;
        end
      end
    end
  end
   $maxM := \max\{Lpd(R_{Oy}, C_i) | C_i \in V\}$ 
  if  $maxM = L$  then
     $err = 1$ ;
  end
  else
     $L_1 := maxM; L_2 := L - maxM; B_1 := V; B_2 := U - V$ ;
  end
end

```

```

PROCEDURE XCUT( $G_{Ox}, U, L, W, H, err, B_1, B_2, W_1, W_2$ ) begin
   $err = 0$ ; SUBGRAPH( $G_{Ox}, G'_{Ox}, U, R_{Ox}$ );
   $V := \bigcup \{C_i | C_i \in U, (R_{Ox}, C_i) \in \Gamma_{Ox}\}$ , where all the elements are unmarked
   $maxM := \max\{w_i | C_i \in V\}$ 
   $P_i := \{w_i | C_i \in V\}$  while  $\exists C_i \in V$  unmarked do
    mark  $C_i$ ;
    if  $P_i < maxM$  then
      for  $C_j \in succ(C_i \text{ in the graph } G'_{Ox})$  do
         $V := V \cup \{C_j | \text{where } C_j \text{ is unmarked}\}$ ;
         $P_j := P_i + w_j$ ;
        if  $P_j > maxM$  then
           $maxM := P_j$ ;
        end
      end
    end
  end
   $maxM := \max\{Lpd(R_{Ox}, C_i) | C_i \in V\}$ 
  if  $maxM = W$  then
     $err = 1$ ;
  end
  else
     $W_1 := maxM; W_2 := W - maxM; B_1 := V; B_2 := U - V$ ;
  end
end

```

packing pattern that separates the set of the boxes in two components it is presented , the set $\{A, E, B, C, D\}$ and the other set $\{H, F, G\}$. It can be seen that on the graph G_{O_z} we did the cut at distance 2. In the syntactic tree, these components are connected using the horizontal concatenation \oplus , an operation for a horizontal cutting plane, Figure 3.20.

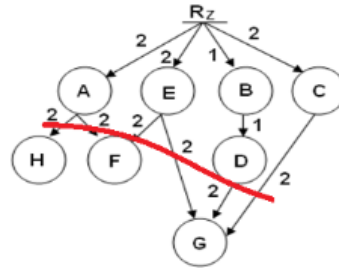


Figure 3.19: The first vertical cut of the bin packing pattern

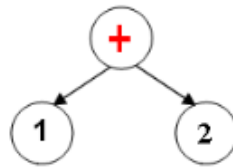


Figure 3.20: The first syntactic tree

The prefix Polish notation for this syntactic tree from Figure 3.20 is: \oplus .

We continue to make horizontal, vertical N-S or W-E cuts for the left and right components from the syntactic tree until every components will contain only one item from the covering model, considering the extracted subgraphs.

Now considering the component 1, we have the three subgraphs $R1_X(A, E, B, D, C)$, $R1_Y(A, E, B, D, C)$ and $R1_Z(A, E, B, D, C)$ from Figure 3.21 and Figure 3.22, obtained by extracting only the nodes from the set, $\{A, E, B, D, C\}$.

Applying the procedure YCUT we observe that we can make a vertical W-E cut on $R1_Y(A, E, B, C, D)$, by a cutting plane perpendicular on O_y , at distance 1 from top of the subgraph, Figure 3.22. We extract two sets again, one containing just $\{A\}$ and another $\{E, B, C, D\}$, named component 3. In the syntactic tree from Figure 3.23 we see that the operation between box A and this new component, 3, is \otimes the notation for a cutting plane perpendicular on O_y .

The prefix Polish notation for this syntactic tree from Figure 3.23 is: $\oplus \otimes A$.

We continue our algorithm using component 3, by extracting the three subgraphs $R3_Y(E, B, C, D)$, $R3_Z(E, B, C, D)$ and $R3_X(E, B, C, D)$, see Figure 3.24 and Figure

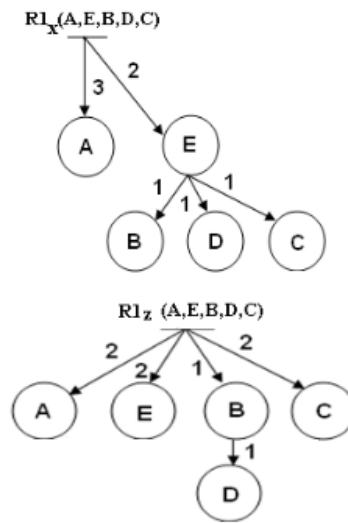


Figure 3.21: The subgraphs $R1_X(A, E, B, D, C)$ and $R1_Z(A, E, B, D, C)$ derived from the first set

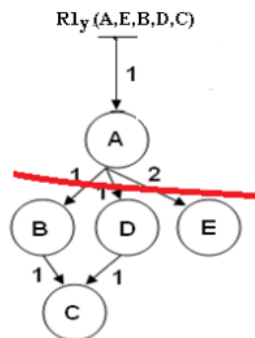


Figure 3.22: The subgraph $R1_Y(A, E, B, D, C)$ derived from the first set

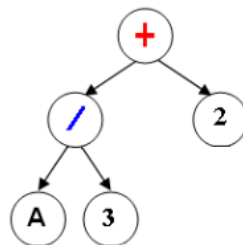


Figure 3.23: The syntactic tree. Step 2

3.25.

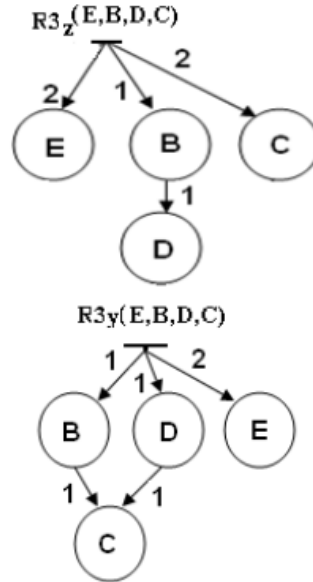


Figure 3.24: The subgraphs $R3_Y(E, B, D, C)$ and $R3_Z(E, B, D, C)$ derived from the third set

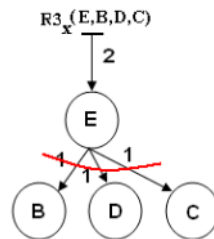


Figure 3.25: The subgraph $R3_X(E, B, D, C)$ derived from the third set

Now we use the procedure XCUT that makes a vertical N-S cut on $R3_X(A, E, B, C, D)$, at distance 2 from top of the subgraph, see Figure 3.25. We extract two sets, one containing just $\{E\}$ and another $\{B, C, D\}$, named component 4. In the syntactic tree from Figure 3.26 we have the operation \ominus for a vertical cutting plane perpendicular on O_x between E and component 4.

The prefix Polish notation for this syntactic tree from Figure 3.26 is: $\oplus \otimes A \ominus E$.

Using component 4, we extract the three subgraphs $R4_X(B, C, D)$, $R4_Z(B, C, D)$ and $R4_Y(B, C, D)$, see Figure 3.27 and Figure 3.28.

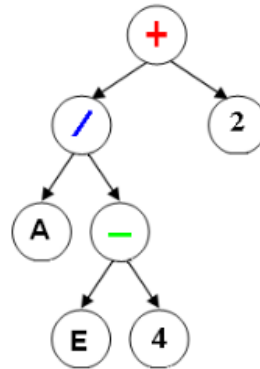


Figure 3.26: The syntactic tree. Step 3

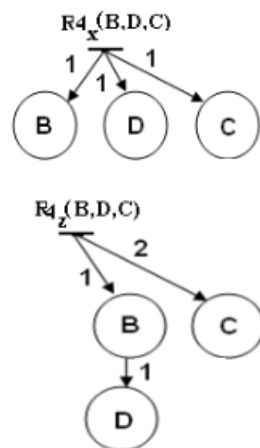


Figure 3.27: The subgraphs $R4_X(B, C, D)$ and $R4_Z(B, C, D)$ derived from the fourth set

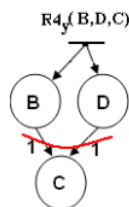


Figure 3.28: The subgraph $R4_Y(B, C, D)$ derived from the fourth set

We can make a cut on $R_4Y(B, C, D)$, at distance 1 from top of the subgraph, by a cutting plane perpendicular on O_y , see Figure 3.28. We extract two sets, one containing just $\{C\}$ and another $\{B, D\}$, named component 5. In the syntactic tree from Figure 3.29 we have the operation for a vertical cutting plane perpendicular on O_y between C and component 5.

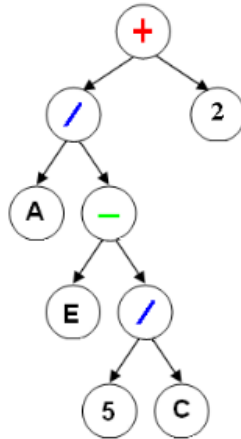


Figure 3.29: The syntactic tree. Step 4

Making the same steps with component 5, we obtained a vertical cutting plane perpendicular on O_z and the syntactic tree from Figure 3.30.

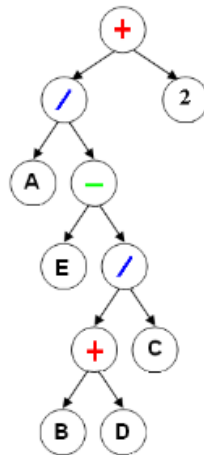


Figure 3.30: The syntactic tree. Step 5

The prefix Polish notation for this syntactic tree from Figure 3.30 is: $\oplus \oslash A \ominus E \oslash$

$\oplus BDC$.

Let's turn to set number 2, that one composed of $\{H, F, G\}$. We have the subgraphs from Figure 3.31 and Figure 3.32 and we've done a cut on $R2_X(F, H, G)$, at the distance 1.5, by a cutting plane perpendicular on Ox . The two sets are: $\{F\}$ and $\{H, G\}$.

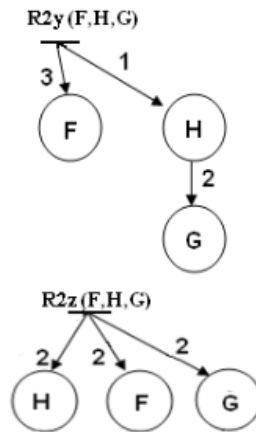


Figure 3.31: The subgraphs $R2_Y(F, H, G)$ and $R2_Z(F, H, G)$ derived from the second set

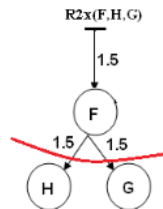


Figure 3.32: The subgraph $R2_X(F, H, G)$ derived from the second set

The syntactic tree from Figure 3.33 has the component 6 connected with F thru a W-E cut.

Using component 6, we have the last three subgraphs $R6_X(H, G)$, $R6_Z(H, G)$ and $R6_Y(H, G)$. The final syntactic tree is in Figure 3.34.

This syntactic tree corresponds to the prefix Polish notation:

$$\oplus \circledast A \ominus E \circledast \oplus BDC \ominus F \circledast HG,$$

exactly the one that we considered in previous section. Like we mention before, the algorithm's complexity is $O(k^2)$ [58].

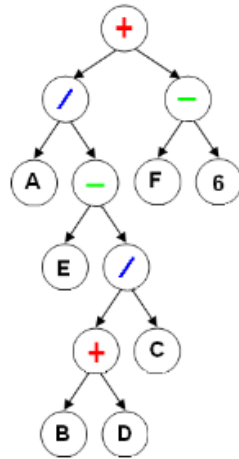


Figure 3.33: The syntactic tree. Step 6

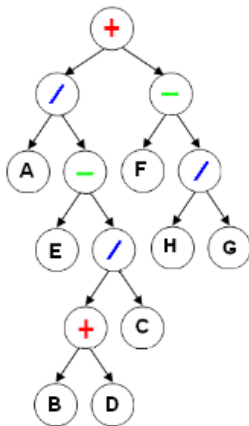


Figure 3.34: The final syntactic tree

Correctness and complexity

The correctness of the algorithm follows from the Theorem 45, that makes the connection between a guillotine cut and the decomposition of a graph in two subgraphs.

The procedure *PREORD()* represents a preorder traversal of a graph, so the complexity is $O(k)$ [17, 19], where k is the number of the cutting boxes. Also, in the procedure *ZCUT*, respectively *XCUT* and *YCUT* we traverse a subgraph of the initial graph. So, the complexity of the algorithm is $O(k^2)$.

The three dimensional bin packing problem holds importance to many fields. Shipping and moving industries, architecture, engineering and design are all areas where three dimensional bin packing could apply. Industry uses bin packing for everything from scheduling television programming to stacking cargo in a semi-truck to designing automobiles and airplanes. Many of the applications of the three dimensional bin packing problem need packing patterns with guillotine restrictions. So a way of solving this is to use some algorithms for packing patterns determination and to use our algorithm for verifying if the patterns have guillotine restrictions or not. This guillotine test can also be used in a constraint programming approach for solving the packing problem. The test for guillotine restrictions presented in this paper is based on a representation of the bin packing pattern by three weighed graphs of adjacency. These graphs, introduced first for the two-dimensional cutting stock problem, were very useful to prove some properties of a cutting or covering pattern [44] or to find out an order of packing for loading a container [56].

We remark that we can apply this algorithm for guillotine determination also in case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps.

3.4 Contributions and results

In this chapter, we have presented four individual areas of research which are interconnected by the central issue of Three Dimensional bin packing. It presents novel theoretical considerations, innovative algorithmic techniques and approaches, a variety of applications, and different practical examples in the setting of the Three Dimensional bin packing problem. The developed algorithms for these different approaches create feasible solutions and all the presented algorithms are quite general and thus might be adapted to a wide variety of three dimensional bin packing problems.

We note that all the results from this chapter are included in four papers, individual or in joint works. The scientific results within this chapter belongs to the author and can be found in the papers [56, 59, 57, 58].

Chapter 4

Optimization with Logical Analysis of Data

Our aim for this chapter is to focus on a combinatorial optimization based data analysis methodology, that is able to perform classification with justification, Logical Analysis of Data (LAD). The justified classification is an ideal aimed by many methodologies within the field of data mining. The justification of a classification algorithm is based on the idea of using information extracted from an amount of data, so that the classification of new data can be easily achieved.

If in the first two chapters of this thesis we looked at covering two dimensional and three dimensional problems, this chapter focuses on variations of the covered topic, which is the study of a methodology based on discrete optimization fundamentals. The LAD methodology involves the knowledge of some basic discrete/combinatorial optimization concepts and principals. Herein, we investigate the accuracy of LAD and we did some remarks about its execution time and the quality of the results. In addition, the LAD methodology is compared with the most important classification algorithms used in the machine learning literature, and their implementation in WEKA [72] and OCTAVE [85], for a greater impact on final results.

We outline the manner in which this classifying methodology works and we have concluded that it has shortcomings, but also strengths compared to the other methods. Thus, our conclusion is that, when compared to other classification algorithms, the LAD methodology requires simple notions of discrete optimization, while providing an accurate classification on considered situations.

A significant contribution of the author in this chapter consists in providing and evaluating computationally the LAD methodology and making some comparison with some specific machine learning algorithms. We evaluate the coverage cases - which percentage of the outcome is covered through a solution - in the case of LAD and other specialized methodologies, as well. Overall, it is to be noticed that LAD counts as a highly intelligent method of classification.

It is nevertheless doubtless that in order to achieve classification, we can use any of the considered algorithms, while the LAD methodology is supported by arguments such as:

- The relatively simple concepts LAD models use - see set covering problem, Boolean functions, simplification of Boolean functions, partial Boolean functions, simplex method, etc.;
- The efficiency/quality of LAD patterns - LAD provides, as any other algorithms, a set of patterns covering both positive and negative observations, but unlike other considered algorithms, these patterns are justifiable, which means that positive patterns cover only those observations ratified as positive, without covering any of the negative patterns, and the other way round. Since the quality and transparency of patterns are very important characteristics when applying classification algorithms, LAD provides these two features, enabling professionals within different fields of application to easily understand them;
- The accuracy provided by LAD - following experiments achieved by the author, it has been observed that results provided by means of LAD are more specific and have a greater degree of accuracy, close to or even exceeding the level of accuracy of other known classification algorithms;
- Time of execution - when based on the processing of Boolean functions, the generation of valid patterns is a process characterized by a low degree of complexity;
- Amazing results obtained in the field of healthcare and not only have encouraged us to choose this problem classification methodology. Our aim was to observe whether LAD is as competitive in other fields of interest as in the field of medicine.

In order to better emphasize the results of the LAD methodology, we have achieved a comparative study with more specific data mining algorithms. In order for the comparison to be relevant, the algorithms were chosen so that they share particular features with LAD. The classical implementation used for LAD was modified by the author, in order to obtain expected results earlier. Appendix two includes all these changes. The main reasons for the author of this paper to have chosen WEKA are its versatility and the authors own wish to test all considered algorithms within a common environment. The implementations in OCTAVE represent an additional contribution, meant to strengthen results obtained by means of WEKA.

Within section "Theoretical aspects" we will briefly present the LAD methodology, as well as some basic notions on using WEKA and OCTAVE. Within the same section, we will provide general information on the functions and guiding principles of chosen algorithms, compared to LAD.

In section "Computational experiments" we organize the experiments we have achieved, the steps we followed in doing so together with conclusions we have drawn following

LAD testing and the algorithms: C4.5, Random Forest, Support Vector Machines, Multilayer Perceptron, Logistic regression represent our original contribution to this thesis. At the end of this section we will analyze results, while also highlighting the advantages of the LAD methodology and commenting on its advantages and disadvantages as for more particular cases.

The section "Summary and future work" includes general conclusions on the whole chapter and some further directions for development.

4.1 Theoretical aspects

This section provides an overview of some aspects which are relevant for understanding the main issues of this chapter: LAD methodology and its main concepts and the "How to use it" principle of C4.5 Algorithm, RF Random Forest Algorithm, SVM Support Vector Machines Algorithm, MLP Multilayer Perceptron Algorithm, Logistic regression and WEKA package. We focus on the methodology of LAD and discuss its main components and functions. Additionally, we shortly outline the ideas of some well known classification and regression algorithms and present the validation procedures used to estimate the accuracy in our experiments with WEKA.

LAD is a methodology which achieves generation and analysis of such aggregates of variables meant to characterize both the positive and negative attributes of specific observations. This is why this methodology has been so far successfully tested on various areas of knowledge like: clinical, sport, computer science, healthcare. A variety of studies have proved that the accuracy of LAD bears close similarity to some of the best methods used in data analysis [2], while at times even exceeding results obtained with methods which are most frequently used.

For the practical part of the LAD we have chosen specifically designed software for experimentation. In the first part of this chapter we will briefly remind what LAD is, and then we describe how various experimental components, described in full detail in [60], are used. Secondly, we will shortly describe some algorithms or methodologies used in matters of classification. The experiments are done in WEKA. In the end, we can conclude how efficient LAD is comparative to other methodologies.

A whole set of specialized literature has been published on various aspects concerning classification methods, as well as solving algorithms [30, 31, 20, 22, 64, 65]. Nevertheless, the methodology chosen for this particular thesis holds a great advantage - that of reasonable, with justification classification, a quite important benefit, especially when we want the end results to be easily readable by individuals outside the field of informatics.

4.1.1 Logical Analysis of Data

LAD is an analysis methodology for great amounts of data which are all based on concepts for optimization criteria, operational research, combinatorics, Boolean functions. It was first introduced in the '80s in [28] and the basic implementation was extended to the case of non-binary data, using the binarization process, in which every numerical is discretized and replaced by one or more binary variables, bringing the problem to the usual binary form. In [16] the implementation of a LAD classification methodology was described and several further developments to the original theories were proposed. A more recent overview of LAD can be found in [29]. This methodology was first applied in medicine in 2002 – 2003 after S. Alexe published reports of a joint research carried out together with clinicians from an academic medical center [29].

LAD differentiates itself from other data mining classical methodologies since it is capable of generating and analyzing a major subset of such sequences of variables described as the negative and positive aspects of all observations. It uses discrete optimization techniques to extract models designed with the support of a defined number of combinatorial patterns - a combination of attributes values appearing together only in some attributes values from dataset - generated in this way [16]. Among known issues of classification, a pattern is a pair of variables, where the first is a collection of aspects and the second is a notion behind the aspect, the name of it. The pattern is as valuable as it is fit enough for separating/discriminating examples from various classes. Specimens from the same class should have similar values, while those belonging to different classes, have different values, at times even opposing.

The goal of a classifier is to partition feature space into two or more different decision regions. Many of the approaches used for classification in traditional data mining are generally based on statistics, such as clustering, decision trees or association rules. Still, the novelty of LAD is that it proposes an innovative manner of data analysis by means of techniques such as combinatorial logic, Boolean functions and discrete optimization. The capability of detecting logical combinatorial information about classified information differentiates LAD from any other traditional data mining technologies. Many of the techniques used for classification in traditional data mining [65, 21, 66] are generally based on statistics, such as clustering, decision trees or association rules. Still, the novelty of LAD is that it proposes an innovative manner of data analysis by means of techniques such as combinatorial logic, Boolean functions and discrete optimization. The capability of detecting logical combinatorial information about classified information differentiates LAD from any other traditional data mining technologies. The LAD methodology detects patterns for which all the satisfying observations have a decidedly higher or reduced show degree than the considered population. From this point of view, we can see some similarity to algorithms such as C4.5 [67], and rough sets classification [63, 65].

The most remarkable characteristic of LAD is its capability to identify any hidden patterns in the information. Since patterns represent sequences of specific features, it is possible to use them for building decision boundaries which can then serve LADs classification. This is possible through the provision of essential information which helps distinguish observations in one class from observations in another. Using patterns could lead to a more stable performance when challenged with classifying not only negative, but also positive classes, since they are not prone to measurement errors. One specificity of LAD is that it favors to select a too larger number of patterns by the provision of solutions for a set covering problem used in building a classifier. The condition is that each observation in the set covering problem of LAD is covered by a pattern. This is regardless of the fact that the specific observation is an outlier.

In this section, the following terminology and notations are extensively used. A "dataset" is a collection of information in which each information can be represented as one event and it is characterized by a vector of values. LAD browses the dataset in order

to determine repetitive collections of conditions on attributes. These are called "patterns". The observation/information which satisfies a specific pattern has a much lower or higher show rate compared to the rest of the population. Consequently, new observations may be classified in two types - positive/show and negative/no show, depending on the patterns they predominantly satisfy. Each type has its particular sequence of patterns. Within those, there are four instances to be noticed: in the first two, the new generated result achieves only a part of the positive or negative patterns and is thus classified as positive or negative, in the third, the new result achieves patterns of the two categories and is consequently classified as either positive or negative, according to the importance assigned to particular patterns of both types covering the observation. Whereas in the fourth instance, the result does not achieve any of the patterns expected, and thus cannot be classified. One of the most worthy of mention practical outcomes of such an approach is the opportunity of offering reasonable description or meaning for LAD results. Another important practical issue is the opportunity to determine brand new categories of information. After discovering them, the possibility to study their purpose and type is a real worthy advantage.

Considering these advantages, viewing LAD in economic parameters as in [13] is not difficult because the performance of this methodology has been so far endorsed through many successful applications of data analysis problems. Perfect classification performance is often impossible, thus it is common to seek minimum error rate classification. Within this paper we have highlighted the importance of using an efficient methodology and its cost-effectiveness. The more exact and more error less the classification algorithm is, the quicker and cheaper the solution. This can count as one of the contributions made in the purpose of developing the array of applicability of the LAD methodology. The paper referenced herein represents the author's own contribution and its main benefit is that of highlighting the economic value of the suggested method.

By means of LAD, it is possible to automatically scan a tremendous amount of highly complex interactions, while retaining only the most relevant for the matter at hand. It may be inferred that such benefits of the LAD methodology could inspire research in view of a deeper understanding of any connected relations of cause and effect. Several studies also revealed that the discussed LAD rating method is objective, transparent, while it can be easily generalized and is sustainable through its pertinent model. In particular, some medical centers are progressively using LAD in the actual process of clinical investigation for a diversity of affections and their ability to adopt improved LAD therapies involved a significant cost reduction.

The main objective in LAD is to implement some decomposition/aggregation methods to some variables of dimension n containing the positive and negative observations [2]. It is to be acknowledged that numerous mathematical problems necessary for approaching such steps depend upon the results of a set covering problem. Definitely, it can be further formulated as an integer linear programming problem [68].

The initial purpose of LAD is to analyze datasets characterized by attributes which take binary values. The need for a binarization process arose as it was noticed that most

daily activities are expressed as real, and not binary values. Binarization consists in the introduction of a set of binarization thresholds [29], based on which each considered attribute is to be classified as below or above the considered value. Choosing an accurate binarization threshold is essential for obtaining results in line with the real attributes considered. This threshold can be interactively chosen or it can be determined by means of various procedures [15]. Relatively small variations of the value of the binarization threshold can greatly influence the result of the binarization process. The automatic choice of a binarization threshold represents a mandatory request in most situations. Generally, this choice is made based on a set on information previously extracted from the considered attributes. Nevertheless, there are particular cases when the choice of the binarization threshold is possible if certain features of the attribute to be elaborated are known or imposed. It will become clear in subsequent sections that the overall computational performance of LAD methodology is related to the number of variables actually tested [2]. In [14] we have the formulation of a set covering problem.

”Several real-life datasets contain variables that are neither binary nor numerical, but instead assume one of a set of nominal, or categorical, values. Let $S = \{s_1, \dots, s_{|S|}\}$ be the set of possible values that a certain nominal variable x_j can assume. In some cases, there is a total order between the possible values, for instance: low, medium, and high. In such a case, the binarization of x_j can be done by associating a sequence of $\lceil \log(|S|-1) \rceil + 1$ binary variables to it in such a way that each value s_i of x_j is represented in binarized form simply as the binary representation of the integer $i - 1$. Alternatively, or when no total order among the elements of S is available, the binarization of x_j can be accomplished by associating to it a set of $|S|$ binary variables, each one of which represents a possible value of x_j . In the resulting binarized representation of x_j exactly one of the associated indicator variables would have the value 1” [14].

In the case of a binary dataset a pattern ”is simply a homogeneous subcube of $\{0, 1\}^n$, i.e., a subcube having the following properties:

- a nonempty intersection with one of the sets Ω^+ or Ω^- ;
- an empty intersection with the other set (Ω^- or Ω^+ , respectively).” [29]

We shall consider the hypothesis that a specific element is positive when covered only by positive patterns, without being covered by any negative patterns. Consequently, a given element is negative if it only achieves negative patterns, without achieving any negative patterns. Note that for a dataset with numerical variables that has been binarized according to the procedure described in the previous section, the definition of a pattern can be seen as a set of constraints that are simultaneously imposed on the values of one or more of the original variables. Indeed, let the j -th binary (indicator) variable be associated to the k -th original (numerical) variable and with the cutpoint value c . Then, a pattern requiring that the j -th binary variable equals 1 corresponds to the requirement that the k -th original variable exceeds the cutpoint c .

The quality of a pattern is related to its ability to discriminate observations from different classes. The "prevalence" of a positive pattern with respect to a given set of observation is the percentage of positive observations that are covered by the pattern, while the coverage of a pattern is simply the number of observations covered by the pattern in the given set. The "homogeneity" of a positive pattern is the percentage of positive observations among the set of observations covered by it. These concepts can be defined for negative patterns in a similar way [14].

Previous implementations of LAD methodology [3, 4, 15, 14] have all relied on some type of pattern generation procedure that implies the generation of considerable collections of positive and negative patterns, each encourage some requirements in terms of prevalence and homogeneity. The notion of pattern is connected with the supposition that the judgments are accurate. Thus, it is assumed that all measurements of attribute values are accurate, as are all recorded observations. When applied to real-life situations, these assumptions do not apply wholly. Such example is the assessment of patterns on a testing set, which shows that positive patterns largely covered (on training set) not only cover an important number of positive statements, on the other hand a relatively small number of negative statements. It is therefore justifiable to permit patterns to cover a reduced number of observations of the opposite class.

The expected results are that the loosening constraints which define both positive and negative patterns will lead to an important rise in the coverage of negative/positive observations. Further on, we shall consider samples not covering the opposite class as pure or homogenous. In the meanwhile, other existing patterns are sometimes referred to as non-homogenous.

Therefore, we can say that "a pattern is a subcube of $\{0, 1\}^n$ with the property that it covers mostly observations from one of the sets Ω^+ or Ω^- , covering only a small number of observations from the other set" [39]. "A pattern is positive if the percentage of observations from Ω^+ covered by the pattern is larger than the percentage of observations from Ω^- covered by it" [39]. A negative sample is described in a symmetric manner.

More generally, we can construct subcubes that do not necessarily satisfy the conditions for being considered patterns. Let us consider each subcube as described by its associated Boolean conjunction, where a literal is used whenever the value of the corresponding binary variable is fixed in the subcube. For instance, let $n = 5$ and $C = x_1\bar{x}_3$. The conjunction C is associated to the subcube of $\{0, 1\}^5$ containing all points in which the first component takes the value 1 and the third component takes the value 0. For the task of classification we are only interested in those conjunctions having certain values of prevalence and homogeneity. Indeed, for extracting the information that can be used to classify unseen observations, we are only interested in finding a conjunction with the property that the set of observations that it covers has a distribution of points from Ω^+ and Ω^- that is significantly different from that originally in Ω .

Since the positive and negative patterns were defined according to similarities they share (symmetric features), the generation procedures obtained will also be symmetric. By reason of being concise, we shall only refer here to the procedures used in generating

positive patterns. Similarly, one can follow the same procedure seeing the generation of negative pattern. The unequivocal approach we have chosen in order to generate patterns is rooted in the use of combinatorial enumeration techniques.

In view of the existence of various possible measures of quality of any given pattern, it is important for the pattern generation procedure not to miss any of the best patterns. These are the patterns that highlighting the most the features of one class. Pattern generation techniques can follow a top-down or a bottom-up approach, details in [10].

In a nutshell, the top-down design commences when joining all positive considerations to their characteristic terms. These specific terms translate as patterns and even if particular literals are removed, what hence results is still a pattern. The purpose of the top-down design is to remove literals until achieving a single remaining pattern. On the other hand, the bottom-up design begins with one-degree terms which cover a set of positive observations. In the case when this specific term does not cover any negative observation whatsoever, it can be called a pattern. Alternatively, literals shall be gradually added until a pattern has been generated. This method of pattern generation is not entirely unfamiliar since it has so far been implemented in other machine learning techniques [10]. This type of generation is not entirely new, we can see it in other machine learning techniques.

This pattern generation process is guided by two natural objectives: "simplicity principle" - it gives preference to the generation of short patterns and "comprehensiveness principle" - we attempt to cover every positive observation by at least one pattern. This is accomplish by a combination bottom-up/top-down technique which favors the bottom-up strategy. We start by using the bottom-up approach to generate all the patterns of very small degrees (usually up to four or five) and then use a top-down direction to cover those positive observations (if any) that remained uncovered after the bottom-up step.

"The bottom-up pattern generation is based on term enumeration. The number of terms of degree d over n Boolean variables is $2^d C_n^d$. Even for n fixed at a moderate value (say, for $n = 20$), this is a very rapidly growing function of d . Therefore, the term enumeration method used for pattern generation must be extremely selective. One of the methods used is a breadth-first enumerative technique which produces, at each stage d , all the positive prime patterns of degree d , as well as the list of the so-called candidate terms to be examined at stage $d + 1$ of the algorithm. A candidate term is any term that covers at least one negative and at least one positive observation. The terms of degree d examined by the algorithm at stage d are all those from which one gets a candidate term of degree $d - 1$ (generated at stage $d - 1$) by eliminating any of its literals" [16]. "The terms of degree d examined by the algorithm are then partitioned into the following three sets:

- P_d , consisting of those terms which cover at least one positive and no negative observations;
- C_d , consisting of those terms which cover at least one positive and at least one negative observation;

- G_d , consisting of all the remaining terms.

It is easy to see that the set P_d consists of all positive prime patterns of degree d and the set C_d consists of all candidate terms of degree d . We note that the set G_d is eliminated from any further consideration. Additional reductions in the volume of computations are obtained by examining the terms of C_d in the lexicographic order induced by the linear order

$$x_1 \prec \bar{x}_1 \prec x_2 \prec \bar{x}_2 \prec \dots$$

of the literals. Since it is sufficient to generate each term only once, the terms of degree $d + 1$ are generated from C_d by adding, in all possible ways, to a term $T \in C_d$ a literal which is larger (in this order) than any literal in T . Indeed, let the indices of the literals in T be $i_1 < i_2 < \dots < i_d$. Suppose that a term T' is obtained by adding to T a literal of index $i < i_d$. Let T'' be the term whose literals have the indices $i_1, i_2, \dots, i, \dots, i_{d-1}$. Clearly, T' can also be obtained by adding to T'' the literal of index i_d . If $T'' \notin C_d$, T' does not have to be examined because the term T' is then neither a prime pattern, nor a candidate term. If $T'' \in C_d$, then T' was already considered, because T'' is lexicographically smaller than T' [16]. The data structure used in the implementation of this algorithm is described in [16].

The generation of patterns starts with the bottom-up stage and proceeds until all patterns of degree up to a certain (problem dependent) D are generated. At that time, those observations in the archive which are not covered by any patterns generated so far are extracted and used in a top-down procedure to generate some additional patterns that cover them. The resulting set of patterns can be further reduced through the exclusion of the unnecessary/redundant patterns. A pattern is defined as unnecessary or redundant if all items it covers had already been covered by an alternative pattern, which, moreover, completes the former by covering an extra array of features. Such computational experiments prove that this simple procedure leads to radical cuts in the number of valid patterns.

In many cases, it is important to restrict attention to prime patterns with special properties. A typical example occurs when reliability considerations require the usage of only those prime positive/negative patterns that cover at least a certain number of positive/negative observation points. Additionally, only those prime positive/negative patterns are to be considered whose appropriately defined distance to the set of negative/positive observations is large. Let the Hamming distance from [1] between a term and a point be the number of variables in which they conflict. Then, the distance from a term to a set of points can be defined as the average of Hamming distances between the term and individual points.

A particularly significant property of certain datasets that has to bear in mind in pattern generation consists of the existence of some, commonly named, "monotone variables". Monotone variables appear frequently in real-life problems, being associated with attributes having a well defined impact on the outcome. It is well-known that the

prime implicants of a Boolean function do not contain the negations of positive variables, or the non negated forms of negative variables. The pattern generation procedure described in [16] can be easily modified to prevent the generation of unwanted patterns. This modification may reduce substantially the number of patterns generated by this procedure.

Logical Analysis of Data models

We shall consider the hypothesis that a specific element is positive when covered only by positive patterns, without being covered by any negative patterns. Consequently, a given element is negative if it only achieves negative patterns, without achieving any negative patterns. The definitions of a LAD model and discriminant can be found in [5].

Designing a LAD model for a given dataset requires, as a general rule, the creation of a larger set of patterns, of which a subset, fully compliant with the definition of the LAD model presented above, shall be selected. Moreover, each pattern in the model should fulfill specific conditions in what regards the prevalence and homogeneity within the group.

”Given a LAD model $M = M^+ \cup M^-$ and a new observation $w \notin \Omega$, the classification of w is determined by the sign of a so-called discriminant function $\Delta : \{0, 1\}^n \rightarrow R$ associated to the given model. Let $M^+ = \{P_1, \dots, P_{M^+}\}$ and $M^- = \{N_1, \dots, N_{M^-}\}$, and let us associate to each positive pattern $P_i \in M^+$ a real weight α_i and to each negative pattern $N_j \in M^-$ a real weight β_j . The associated discriminant function on w is given by:

$$\Delta(w) = \sum_{P_i \in M^+} \alpha_i P_i(w) - \sum_{N_i \in M^-} \beta_i N_i(w)$$

where $P_i(w)$ equals 1 if w is covered by P_i , and 0 otherwise (similarly for $N_i(w)$). The sign of $\Delta(w)$ determines the classification of w . If $\Delta(w) = 0$ the observation is either left unclassified, or is classified according to the more frequent class” [5].

In most applications of LAD, equal weights are used among the positive patterns of a LAD model, and also among its negative patterns. A more sophisticated way of selecting weights is used in [15] and consists of the solution of a linear programming model that chooses the weights in such a way so as to maximize the separation of the training set in the so-called pattern-space. The pattern-space representation of an observation w is simply the characteristic vector of the patterns covering w . Attention that in this approach the discriminant function is optimized over a given set of patterns that was previously selected during the pattern generation procedure. Since the number of samples can be considerable, the use of this approach assumes that a relatively small set of patterns has been previously identified.

Accuracy and validation

When designing a classification model it is of utmost importance to always evaluate the accuracy of the resulting model, expressed as the mean value of correct predictions from the total amount of predictions made. This process bears the name of classification accuracy. One of the characteristic component within the study of medical data is the validation, otherwise called the cross-validation of results obtained. If the initial dataset is big enough to permit the separation of all subsequent information into a training set - 70% - and a test set - 30% -, the former is usually used for designing an analytical pattern for the specific task and draw necessary conclusions, while the latter serves at validating conclusions hence derived. One of the main difficulties encountered in the medical field is that when working with large numbers of patients, each affected by certain conditions, medical datasets consist in observations of poorer quantity. Thus, within this particular field of science, the most appropriate tool for assessing the quality of derived conclusions upon the study of clinical data provided is the cross-validation/or rotation estimation technique.

By ordinary, k -fold cross validation process is used [30]. Unlike the method that tests a model on one set of testing and reports the performance, k -fold cross validation method produces k testing steps and so k values of model accuracy. The mean value of these rates represents the performance estimation pattern. Moreover, in the practical usage, this represents the preferred methodology for quantifying the quality of a given pattern and providing relevant comparison with other patterns. The method consists in the initial division of a bulk of information into a k amount of pieces. The division is random or it can follow a preset order. One of the k pieces is associated to data testing, while the other $k - 1$ pieces are kept for training data. The result obtained by the training of the $k - 1$ pieces is to be tested on the k -piece parameter. The process shall be further on repeated k times, and for each of such instances the test bulk together with training data are to be changed, so that each of the k pieces is a test bulk in turns. The accuracy of the process is provided by the accuracy average for the amount of k tests.

LAD is making a division, particularly a new applicant can either be positive, negative or indeterminate. The accuracy is defined like: "simply the proportion of correctly classified patients in the test set" [29].

The field of medicine also uses, on quite a frequent basis, two other concepts in the analysis of samples - the sensitivity and the specificity of the test. Sensitivity is defined as the fraction of samples classified as being positive within a given set of positive tests (e.g. From patients known to be ill), while the specificity is the fraction of samples classified as being negative within a given set of negative tests (e.g. From patients known to be healthy).

Next, we also mention the importance of the notion "hazard ratio of a set of tests" [29] - another testing measure used in medicine - which is the mean fraction of two other fractions, that is, the fraction of the positive tests and the fraction of positive tests within a complementary set. Usually, this set is the one foreseen to be positive by LAD or other

data study method.

The classic Logical Analysis of Data implementation

In order for LAD to be implemented, a set of important parameters controlling the type of patterns generated and designed needs to be calibrated. Among these usual parameters relating to general pattern creation procedures, we mention: pattern degree, pattern family (prime, spanned, strong patterns, or mixed combinations), the minimum degree of homogeneity of a pattern and its minimum prevalence.

Except for the pattern generation process, the design of models involves other important decisions, as well. The elaboration of the model and its power reach consensus when each statement from the training set is covered by the patterns in the model. Another essential decision is the discriminant function to be used upon the design stage of the model. According to general standards, the implementation of LAD is made through the use of a simple linear function, where all positive and negative patterns hold equal weights. It follows that the discriminant function generally used in LAD classifies an observation depending on the percentage of one and the other positive and negative patterns covering it.

The LAD discriminant data can be regarded as a hyper-plane separation in the pattern space. It is possible to design various discriminant functions on the same pattern-space and obtain quite varied classifications veracities. There is also a great chance that a discriminant function which was cautiously designed might trigger differences in the resulting accuracy of a model.

While undergoing real-life research, it has been noted that a cautious calibration of the LAD parameters would grant the design of highly accurate models, which most frequently end up by outperforming the models designed using more common methods, such as SVM, decision trees, neural networks, etc. What is more, such LAD models allow the user to access a deeper understanding of the data analyzed, which would otherwise be very hard to obtain with the use of a single machine learning/ data mining algorithm. Among such examples, we mention the relative influence of variables, the automatic detection of outliers and possibly misclassified observations, together with the detection of representative subgroups of observations within one of the classes analyzed. Nevertheless, one of the major drawbacks is that in order to indulge on all the benefits that the classification power and data mining capabilities of LAD have to offer, one must waste precious time in order to calibrate all the above mentioned parameters.

Tools for Logical Analysis of Data

As already mentioned, the purpose of LAD is to extract from a set of items sharing a common feature one or more logical patterns to be satisfied by numerous of the specific items, so that such logical relations are satisfied by a minimum of the items not sharing

the particular common feature. These logical relations are thus representative for the items which share the common feature.

There are several tools designed for working with LAD, most of which have been developed for research purposes. One of them is the application available for download at [77]. The wide array of functions which compose this application follow and achieve all the steps LAD requires, with modularity as a must, so that each stage of LAD can be easily tested, modified or optimized. In order to develop the application, the authors have used the C++ language, mainly for its popularity and its reasonably high level of abstraction. Most methods used therein are easy to read and understand. All aspects discussed in relation to the description of LAD are necessary for a full understanding of the code. The application is implemented in such a way that it flawlessly works on Linux. Since we were looking for an easier operation of such modules, the translation of the Windows operating system represents a personal contribution of the author. Another modification to the initial version is that due to the fact that the application developed by E. Mayoraz worked only for the order line, we considered that using the tool with input text files would be more helpful. This implementation can be divided into three phases: binarization, generation of patterns and models. The binarization phase is designed to handle multiple classes. The other two stages are restricted to problems with two classes. The author has attempted to generalize the two stages, which should thereafter enable classification for more classes, but this was achieved only by repeating the steps for two by two classes. A generalized development of the concept is on the priority shortlist in the future.

Next, we have chosen three datasets from two different fields, medicine and economy, and we have proved how optimum the LAD methodology is, when compared with other methodologies based on the same kind of algorithms. In order to achieve the comparative study, we have undergone the following steps:

- we have tested the results obtained by means of LAD in various specialized papers [40, 29, 28] and we emphasized the fact that LAD really is a top method for classification, providing as support examples for the datasets: **breast-cancer-wisconsin.arff**, **diabetes.arff** and **credit-g.arff**;
- we have applied other methods for the correct classification of instances, by choosing one of the most well-known methodologies and algorithms, such as C4.5 Algorithm, RF Random Forest Algorithm, SVM Support Vector Machines Algorithm or MLP Multilayer Perceptron Algorithm. In order to apply such algorithms we have used the WEKA package, for its great suitability with the task. Further details on WEKA are to be found within the following sections.

We have assessed the accuracy of these classification models using the one random 10-fold cross validation method, see section "Accuracy and Validation".

There are many methods for conventional data mining classification and most of them stem from statistics. Such methods include: clustering, decision trees, association

rules, etc. The difference between these methods and LAD is that not only does LAD suggest a new mechanism for data analysis by means of combinatorial logic, Boolean functions and optimization techniques, but it also has the capacity to detect further logical combinatory information based on the remarks made. Moreover, the LAD method of classification identifies patterns characterized by satisfying observations of a distinctly higher or smaller degree than the investigated population. From this perspective, we can notice some degree of similarity to other algorithms such as C4.5 from [67] and classifications of rough sets from [63].

Further details on the working principles of the application developed for LAD can be found in [29]. We also mention that there are other tools for LAD processing, such as Datascope - package written by S. Alexe for Windows, Ladoscope Gang [41] - a set of programs written in the language Objective Caml by P. Lemaire, PLAD - written in Perl, available by request from E. Boros, etc, but the one we have chosen to analyze within this paper has proved to be of the greatest performance and the easiest to extend. The results obtained for LAD are remarkable, and that is why we felt the need of comparing it with some more classical classification tools. All these methodologies/algorithms are integrated in WEKA. At the same time, the modifications made on the original code developed by E. Mayoraz are attached in Appendix 2.

4.1.2 Classification algorithms

Within this section we will shortly focus on the analysis and the particular details of the algorithms we are to use in the next section, in order to assess the quality of the computational results obtained on LAD. Whenever possible, we will also emphasize existing connections between specific features of such algorithms and basic concepts of LAD.

We will discuss the main parameters for each algorithm and we present them based on their implementation in WEKA package. We remark the fact the "WEKA's classify panel enables the user to apply classification and regression algorithms (called classifiers in WEKA) to the resulting dataset, to estimate the accuracy of the resulting predictive model, and to visualize erroneous predictions, ROC curves, etc., or the model itself (if the model is amenable to visualization like, e.g., a decision tree). [78] The methodology used in order to compare obtained remarks along those of methods executed in WEKA package was the LAD accuracy definition, which has already been presented in the previous section.

Further on, we will shortly revise some classical algorithms and methods used in classification. Each of the examples provided are similar to the proposed methodology of LAD. Detailed mathematical principles about how these algorithms works can be found in [23].

C4.5 algorithm

C4.5 is a classification algorithm, which generates a decision tree using the training data. The approach of information gain is used to build a decision classification tree as regards a target classification previously chosen. Evidently, the separation of the input space into subspaces in the decision tree context is closely related to that of patterns in LAD methodology. We state the fact that in the case of decisions trees the set of patterns is significantly smaller than that of LAD. Because usually the decision tree models are simpler classifiers, they are less expensive computationally generated and, if pruned, are less prone to overfitting the data. What differentiates LAD classifiers from decision tree models is a higher degree of generalization for the former. The next section will also allow us to conclude that the results obtained both for LAD and decision trees have similar values; even if they still remain fundamentally different from other perspectives.

C4.5 works after following principle: for each node it is selected the specific feature of the data which most adequately divides the set of instances into further subsets enriched either in one of the classes. The division criterion in such instance is the normalized information gain (difference in entropy). The feature holding the greatest amount of normalized information gain will be selected for further decision making. The typical algorithm used for creating decision trees is to be found in [38].

In the case of all decision tree models, each node generally represents a specific test which relates to the values of one or more features, while each of its subsequent branches correlates to one of the possible results of the test. If the specific feature used in the test predicates nominal or discrete values, it is then possible to achieve tests with a larger array of results. Thus, in the case of a training dataset, each individual test divides the observations into one or several subsets.

When applying successively such tests, the user can design a tree like structure which partitions data into progressively lesser parts of the input space. Normally, this process is achieved in a greedy manner. By means of the best test that divides training data, the entire amount of training data is split into one or several subsets, following that each of the obtained subset is further split in the same manner, until each emerging subset of observations is homogenous enough in what concerns the set observation classes. The criterion used for assessing the suitability of a particular test is a quantitative measure, such as the entropy.

It is obvious that the partition of the input space into subspaces in the decision tree context is closely related to that of patterns in LAD. The sequence of tests down a path in a decision tree corresponds to a pattern in LAD, where it defines a subset of the training observations that satisfies certain conditions of prevalence and homogeneity. But, we should remark the fact that in the case of decisions trees the set of patterns is significantly smaller than that of LAD, since test section is made based on the particular test's capacity to provide locally enhanced separation of the set of observations. In the classic LAD enumeration scheme, all patterns are generated and evaluated, and from this, a complex way to manipulate them. Since decision tree models are considered to be

simpler classifiers, they are less expensive to generate. Moreover, by pruning them, the risk of overfitting data diminishes. On the other hand, LAD classifiers are more general than decision tree models. We will see in our next section that the obtained results for LAD and decision tree are close enough, even if they are quite different in some aspects [11].

Random Forest algorithm

The Online Chemical Modeling Environment has integrated an implementation of the random forest algorithm by WEKA package. Random forests refers to the combination of learning methods which operates through the design of a great number of decision trees at training time and the output of the class which is the mechanism of the classes output by individual trees. As mentioned above while it is quite cheap to build trees, among the most important benefits of using such simple structures we mention the high accuracy of results obtained with this particular classification model [9]. Given that a sufficiently large number of trees is generated, we expect that random forests models have a very close performance to that of LAD models from [3]. We proved these theoretical results through tests, using WEKA.

SVM algorithm

In machine learning, SVM is a supervised learning model that analyzes data and recognizes patterns. The typical SVM's classifier consists of a linear discriminant function that separates the training data in a very similar way to the LAD. Besides this, the optimization model for optimizing the weights of the discriminant function in LAD is almost identical with the one utilized in SVM. We will see in the next section, that on the same dataset, these two algorithms have tight results.

Logistic regression

The method of logistic regression serves at evaluating concepts such as conditional probability and classification. This model was initially designed to work with two classes, but it was further enhanced in order to discriminate between several types of classes. This led to it being also called multinomial logistic regression. We will mention two reasons why linear regressions are not suitable for the accurate prediction on a given binary variable:

1. A linear regression will predict values outside the acceptable range;
2. Since the branched experiments can only result in two values, there will be no distribution of the residuals within the predicted line.

Alternately, the outcome of any logistic regression is the logistic curve, generally limited to values between 0 and 1. The only notable difference between the logic regression and the linear regression is that the curve of the former is designed by means of the natural logarithm of the odds of the variable addressed, rather than by means of the probability factor. Furthermore, it is not mandatory for predictors to be ordinarily distributed or for them to be equally variable within each of the groups. Here are some instances of issues generally encountered with two classes of predictors (seen as logistic regression): labeling an email as spam or no spam (posing contents, details about the subject of the email, sender is or not in the list of contacts), classification of tumors as benign or malignant (following the analysis of results), classification of visuals, such as the image of fruit as apple and grapes. Other examples of problems include the case when there are multiple classes of predictors and they are generally encountered in such instances as classification of emails in terms of typology: news, weather, jobs, family, spam, etc.; classification of visuals, such as the image of fruit as apple, grapes, cherries or oranges.

Further on, the experimental section will provide us with evidence that the results obtained for LAD and this specific algorithm are mostly similar.

Multilayer Perceptron algorithm

A multilayer perceptron (MLP) is a "feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs" [38]. "A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. The number of nodes in the first layer is equivalent to the number of input variables in the data, and each of the specific nodes holds a single input, which corresponds to the correlated input variable. The last layer consists of a number of nodes which equals that of predictors in the data, but there is no connection between different nodes. The intermediate layers, where the case, provide the network with extended flexibility" [38].

We will supplement the description of this algorithm in the experimental section, where we will analyze the performance of these heuristics in a series of computational experiments, with different publicly available datasets from WEKA package repository and UC Irvine repository.

4.2 Computational experiments

In previous section, we summarily define the LAD methodology and we described some essential classification algorithms. In this section, we discuss the comparative accuracy of the LAD "models" and the accuracy and results for these algorithms. We state that we use the C++ application developed by E. Mayoraz, and modified by the author for LAD models and WEKA package for all the other algorithms. For all the next examples, we used 3.9.0 WEKA version [78].

One of the fields of knowledge which ascertained the irrefutable value of LAD in real life applications is medicine. That is the reason for choosing a dataset from the economic environment and another two from the medical environment as a means to exemplify the LAD principles. The LAD methodology has several practical applications, be it in the medical or the economic fields. One way of emphasizing the importance of using the LAD methodology within the economic field, as well as highlighting the importance of this method within various practical aspects, is shown in [13].

The subsection "Datasets - Examples and comparisons" includes the description of data sets used in comparison, as well as the two main comparison directions suggested by the author. Within the same section, we remind about the authors original input in developing suggested algorithms, as well as modifications to the original implementation purpose of LAD.

The subsection "Experiments" includes the experiments achieved and it displays them as comparative tables.

The subsection "Results and analysis" includes the analysis of obtained results, together with the advantages LAD provides, suggestions on when it is recommended and when it should be avoided, as well as which is the status of the specific methodology compared to considered algorithms.

4.2.1 Datasets - Examples and comparisons

Within this section we would like to compare the LAD methodology with some of the best-known classification algorithms. The comparison aims at exhibiting the fact that LAD is a classification methodology absolutely comparable with other well-known algorithms in the field of data mining, which have had even more satisfactory results in some particular cases.

Following experiments, it might appear that LAD is regarded as a feasible tool allowing the user to attain superior classification performance at low computational expenses. The veracity of LAD models are equal to or even exceed the level of accuracy obtained with models created by means of other algorithms.

For the LAD methodology, we have used an application which was originally designed by E. Mayoraz (a public tool serving research purposes) and was further improved by the author. In order to obtain results with the other algorithm considered, we used WEKA. In order to serve as anchorage for our tests, for some of the considered data

mining algorithms we also carried out an OCTAVE implementation, Appendix 1, using a purely OCTAVE code, with no external libraries or plugins.

In view of an accurate and complete comparison between these algorithms and the LAD methodology, we have taken into consideration two main comparison directions:

- Comparison 1 - default parameters: where we compared results obtained with LAD with those results obtained with WEKA (for J48, Multilayer Perceptron, Logistic, SMO and Random Forest algorithms), with default parameters for each algorithm. Further on, for all named classifiers, we shall present the values of used parameters in the experiments;
- Comparison 2 - optimal parameters: while such task as finding optimal parameters for a classifier could prove to be a dreary practice, WEKA provides a series of means for automating the process. Meta-classifiers enable the optimization of specific parameters of base classifiers. These "meta-classifiers" [79] are:

```
"weka.classifiers.meta.CVParameterSelection
weka.classifiers.meta.GridSearch
weka.classifiers.meta.MultiSearch (3.7.11+)
Auto-WEKA (3.7.13+) "
```

For both comparisons, a 10-fold experiment was achieved and it combined all possible parameters. The final report consists in the maximum average degree of accuracy obtained.

Datasets description

The first dataset to be considered within the field of medicine:

breast-cancer-wisconsin.arff - 699 instances.

The specialized literature acknowledges that this is a clean dataset - the dataset does not contain incomplete, incorrect, inaccurate, irrelevant, etc. records - for which a wide variety of analysis methods offer diagnostic models characterized by a high degree of accuracy. In the Figure 2 from Appendix 3 we can see how this file looks in WEKA.

It is known from that Breast Cancer Wisconsin is a data file upon which frequent data studies offer rigorous diagnostic results. Using WEKA graphical user interface we can see it is a cytological test described by: number of numerical attributes (10), number of instances (699 total - 458 benign/label "2", 241 malignant/label "4"), name of the attributes, minimum value, maximum value, etc. WEKA compute some basic statistics on each attribute, we can easily see: minimum value, maximum value, mean, standard deviation, etc. The outcome is a binary variable, with values for benign or malignant nature of the tumor, "yes" - tested positive, "no" - tested negative. This dataset is updated frequently. In our experiments, for abbreviation, we will refer to this dataset like "b-c-w".

The second dataset is an economic one: **credit-g.arff** - 1000 instances.

Since the economic relevance of LAD was already mentioned in [13], it should be obvious why the author choose a dataset such as "credit-g.arff". It characterize applicants for credit cards. The output represents the approval or rejection of the request.

As soon as data is loaded from the visualization window, WEKA provides the possibility of determining those attributes that the data set depends on. Alongside visualization, WEKA offers a set of basic static elements for each visualized attribute, see Figure 3 from Appendix 3. We can see that from all those 1000 instances, 700 are labeled "good"/positive and 300 "bad"/negative.

The third dataset is another medical one: **diabetes.arff** - 768 instances.

For this dataset, class value 1 (total number of instances 268) is interpreted as "tested positive for diabetes" and the rest "tested negative for diabetes". We can load data from WEKA and it admits the attributes, Figure 4 and Figure 5 from Appendix 3.

We have chosen three datasets for the experiments which shall be detailed within this section. In the case of all these sets, the output has only two possible values - "yes"/"no". The reason for choosing such sets is the fact that the LAD methodology is hence developed so that it can classify certain instances as being part of two appointed and possible classes. It is most certain that a part of the chosen algorithms offers the possibility that the output is more generous, but for the sake of making an accurate comparison by means of LAD, we have chosen those sets which provide a valid solution of this type.

LAD - C++ application

All results for LAD hence obtained were verified and reconstructed using a modified version of the application designed by E. Mayoraz [60].

We remind here that the adaptations of the original application were:

- converting the application in order that it could be used in Windows (the original application was developed for Linux);
- converting the application so that it becomes a console application, not a command line application;
- the optimal rewriting in $C++$ of some methods (dynamic allocation instead of static allocation, we change the parameters of some functions, in those cases where they had more than four parameters, we changed some structures in classes and we wrote some Object-oriented programming modules);
- we added some new necessary methods, like the one for calculating standard deviation;

- additionally, because k -folding method is the most frequently used cross-validation technique, we evaluate the accuracy of LAD using one random 10-fold cross-validation. For this purpose, we used the methods:

```
void divide(Matrix<T>& partition, int k)
void setFold(multiDS<T>& data,
const Matrix<T>& partition, const int fold)
```

The first method constructs a partition of k equal parts of the dataset. For the purpose of equalize all the parts, the number of elements in two different parts will never differ from more than 1. The parameters of the method are: the dataset received like a matrix with different (integer) values, the number of parts (for our particular case is 10).

The second method take the fold and if this one is greater than 0, we set data to the fold-th partition of the current object, if fold is less that 0, we set the data to everything but the fold-th partition of the current object, fold $\{-k, \dots, -1\} \cup \{1, \dots, k\}$. The code for these methods is in Appendix 2.

By means of the above mentioned application we will display the results obtained by means of LAD for all datasets, and we will also carry out a comparative study of these results with those obtained by means of WEKA algorithms.

4.2.2 Experiments

Comparison 1

Dataset **breast-cancer-wisconsin.arff**.

In Table 4.1 accuracy is provable by tests of the 10-fold cross validation type. Within the previous section, we have explained this type of testing. The table shows results obtained by means of LAD and results obtained by means of considered algorithms. We use Experimenter Environment from WEKA, we choose the algorithms (J48, Multilayer Perceptron, Logistic, SMO, Random Forest) and we run all the algorithms in a single work session, with one repetition and 10-fold cross validation. The results can be seen in Figure 6 from Appendix 3. This option offered by WEKA is very useful, in case one wishes to run more algorithms at once.

Note that the WEKA algorithms need some calibration of some parameters. For this type of comparison we used the default values of the parameters. For instance, in practice, J48 algorithm takes into account a penalty term, in order to allow some observations in the training dataset to be incorrectly classified. The so-called C -parameter (default value - 1.0) imposes the degree of importance the model should allow to the perfect separation of the training data, compared to maximizing the separation margin of most observations. An equally important parameter is the kernel function, which refers to the

feature of the space chosen (in the specific case, the polynomial function). For these values see the official documentation for WEKA [80].

The classification accuracy of LAD with a 10-fold cross validation for this dataset is 96.10%. Because the application for LAD did not originally have the calculation of standard deviation, we add a new method that calculates it. It works out the mean, the simple average of the tests values, then for each value: subtract the mean and square the result, then work out the mean of those squared differences and take the square root of that. In this way, we have a standard deviation of 2.15.

We repeat the experiment for 10 repetition and 10-fold cross validation (10X10 results). In this experiment, the J48, MultilayerPerceptron, Logistic, SMO, RandomForest algorithms are run 10 times with 10-fold cross-validation procedure. The next step, each of the 10 cross-validation folds is averaged. This procedure results in a line for each run for a summarized 50 result lines. All 500 (10x50) results are sent to the classification. It is obvious that this type of experiment affects the execution time, but the mean absolute error is different. The results can be found in Figure 7 from Appendix 3.

The accuracy obtained with LAD methodology are at least comparable to the accuracies of the chosen algorithms. LAD stands out as a highly competitive and alternate method.

Dataset **credit-g.arff**.

In Table 4.1 we have the results with 10-fold cross validation method for LAD and the WEKA algorithms. All the results can be found in Appendix 3, Figure 8.

Dataset **diabetes.arff**.

In Table 4.1 we have the results with 10-fold cross validation method for LAD and the WEKA algorithms. It informs the comparison of the chosen algorithms from WEKA and LAD.

Dataset	LAD	C4.5(J48)	SMO(SVM)	Random Forest	Logistic	MLP
b-c-w (ACC)	96.10%	94.56%	96.99%	96.57%	96.57%	95.28%
b-c-w (StdDev)	(±2.15)	(±3.63)	(±2.07)	(±2.15)	±2.15	(±2.61)
credit-g (ACC)	83.25%	70.50%	75.10%	76.40%	75.20%	71.50%
credit-g (StdDev)	(±4.15)	(±3.60)	(±3.45)	(±3.92)	(±3.43)	(±2.80)
diabetes (ACC)	76.30%	73.83%	77.34%	75.79%	77.22%	75.40%
diabetes (StdDev)	(±3.05)	(±5.66)	(±4.07)	(±3.49)	(±4.57)	(±4.66)

Table 4.1: Results for datasets

From these tables we can clearly observe that the LAD methodology is one competitive with the other classification algorithms from the analysis of the classification accuracy. Of course it is important to understand that differences between results are normal, especially since two different tools were used in order to obtain them. Each of these tools have different particularities and different advantages. For instance, the

LAD application has advantages such as: application developed in a quick programming language (C++), the LAD classifying steps were fully respected, the modules of the application have been modified, see Appendix 2, in order to obtain the most efficient local optimizations, which in turn would lead to the global optimization of the classification.

WEKA combines many classification algorithms into one single application, the implementations for the algorithms being quite efficient. For all the algorithms, we have chosen to use 10-fold cross-validation in the hopes of obtaining a better classification percentage. The execution time is not very important, but, even so, we can observe that from this perspective as well the LAD classification is not less efficient than other algorithms.

We can easily accept that the capability of LAD is the same as that of Random Forests and SMO, at the same time having at least the same results as the other algorithms from WEKA [2]. In correlation with the most frequently used machine learning/data mining algorithms, the proposed methodology of LAD was shown here that bears comparison to several other methods used in classification.

All the tests that we did here were with Experimenter Environment, where we can easily see the values for accuracy and standard deviation. But, if we want to compare other values for these algorithms, we should use Explorer Environment.

We also compare the confusion matrix [81] for these datasets. Table 4.2 contains confusion matrix for the above algorithms and for LAD. The confusion matrix is a helpful mechanism for analyzing how good our classifiers can recognize observations from different classes. We state that for **breast-cancer-wisconsin.arff**, parameter a is for the class labeled by "2", benign and parameter b is for the class labeled by "4", malignant. For **credit-g.arff**, parameter a - "good", parameter b - "bad" and for **diabetes.arff**, parameter a - "tested_negative", parameter b - "tested_positive".

From this table we can observe that LAD has at least the same precision like the other algorithms, sometimes even bigger. In Appendix 3 we have all the results obtained with WEKA and reported in Table 4.2. From these results we can also report other values like: "TP Rate", "FP rate", "Precision - $= TP/(TP + FP)$ ", etc. [80] Obviously, we can still change some parameters of the algorithms and we can still make some tests with other algorithms from WEKA, but this is considered a future work in this topic.

Comparison 2

Finding the optimal parameters for a classifier might prove to be laborious. Thus, WEKA provides a set of solutions for automating the process. There are available four meta-classifiers that allow you to optimize some parameters of your base classifier. From these, we choose `weka.classifiers.meta.CVParameterSelection` for our tests. We can optimize the C parameter of

```
weka.classifiers.functions.SMO, but not the  $C$  of an
weka.classifiers.functions.SMO
within a weka.classifiers.meta.FilteredClassifier.
```

Dataset	Algorithm	a	b
breast-cancer-wisconsin.arff	LAD	442	16
		8	233
	C4.5(J48)	438	20
		18	223
	SMO(SVM)	446	12
		9	232
	Random Forest	444	14
		10	231
	Logistic	446	12
		12	229
	MLP	440	18
		15	226
credit-g.arff	LAD	693	95
		70	142
	C4.5(J48)	588	112
		183	117
	SMO(SVM)	610	90
		159	141
	Random Forest	642	58
		178	122
	Logistic	605	95
		153	147
	MLP	561	139
		146	154
diabetes.arff	LAD	420	78
		98	172
	C4.5(J48)	407	93
		108	160
	SMO(SVM)	449	51
		123	145
	Random Forest	418	82
		104	164
	Logistic	440	60
		115	153
	MLP	416	84
		105	163

Table 4.2: Confusion Matrix for all algorithms

Shortly, `weka.classifiers.meta.CVParameterSelection` selects best value for a parameter, this means that it optimizes performance, using cross-validation and optimizes accuracy.

For exemplification, we consider two of our algorithms J48 and SMO. J48 has two parameters, confidenceFactor C and minNumObj M . We enter in the ArrayEditor for CVParameters the following string: `C 0.1 0.5 5`. This will test the confidence parameter from 0.1 to 0.5 with step size 0.1 (= 5 steps). For SMO and its complexity parameter C we enter the following string: `C 2 8 4`. This will test the complexity parameters 2, 4, 6 and 8 (= 4 steps). In order to report the comparison between these classifiers, we have to know the symbols: *** - significant poor, *v* - significant better, *blank* - we can not tell if significant poor or significant better.

The results for **breast-cancer-wisconsin.arff** are in Figure 4.1.

```

Tester:      weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -V -result-matrix "weka.experiment.Re
Analysing:   Percent_correct
Datasets:    1
Resultsets:  2
Confidence:  0.05 (two tailed)
Sorted by:   -
Date:        7/7/16 5:27 PM

Dataset      (1) meta.CVParamet | (2) meta.CVPara
-----
breast_cancer_92  (10)  96.71(2.24) |  94.56(3.29)
-----
                                 (v/ /*) |      (0/1/0)

Key:
(1) meta.CVParameterSelection '-P \"C 2.0 8.0 4.0\" -X 10 -S 1 -W functions.SMO -- -C 1.0 -L 0.001 -P 1.0E-12 -N 0
(2) meta.CVParameterSelection '-P \"C 0.1 0.5 5.0\" -X 10 -S 1 -W trees.J48 -- -C 0.25 -M 2' -6529603380876641265

```

Figure 4.1: **breast-cancer-wisconsin.arff** with meta-classifier CVParameterSelection

We can see that standard deviation is smaller for the same percent for J48 and SMO is better in correct percent, 96.99% and 96.71%.

The results for **credit-g.arff** are in Figure 4.2. Here we enter in the ArrayEditor for CVParameters the following string: `C 0.1 0.3 3.0` (for J48), because the dataset has more instances, according to WEKA [80]. For SMO we enter the following string: `C 2.0 6.0 3.0`.

The correct percent is slightly better than the percent obtained in previous comparison, this means that CVParameterSelection optimizes the accuracy. The same is for **diabetes.arff**.

The results for **diabetes.arff** are in Figure 4.3.

It is easy to notice that accuracy percentages are higher, but also that the standard deviation has lower values. Finding optimum parameters and their adequate calibration is quite a sensitive topic. WEKA provides these meta-classifiers in order to ensure better results for the data sets chosen. A fairly easy option for finding accurate ranges for the hyper-parameter of each algorithm is the multiple attempts with the WEKA Explorer.


```

Tester:   weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -V --result-matrix "weka.experiment.ResultMatrixPlai
Analysing: Percent_correct
Datasets: 1
Resultsets: 2
Confidence: 0.05 (two tailed)
Sorted by: -
Date:     7/8/16 4:40 PM

Dataset          (1) meta.CVParameter | (2) meta.CVPara
-----
german_credit    (10) 76.00(2.11) | 75.20(3.71)
-----
                (v/ /*) | (0/1/0)

Key:
(1) meta.CVParameterSelection '-P \"C 0.1 0.3 3.0\" -X 10 -S 1 -W functions.SMO -- -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
(2) meta.CVParameterSelection '-P \"C 2.0 6.0 3.0\" -X 10 -S 1 -W functions.SMO -- -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K

```

Figure 4.2: **credit-g.arff** with meta-classifier CVParameterSelection

```

Tester:   weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -V --result-matrix "weka.experiment.ResultMatrixPlai
Analysing: Percent_correct
Datasets: 1
Resultsets: 2
Confidence: 0.05 (two tailed)
Sorted by: -
Date:     7/8/16 4:50 PM

Dataset          (1) meta.CVParameter | (2) meta.CVPara
-----
pima_diabetes    (10) 77.60(4.53) | 73.45(6.51)
-----
                (v/ /*) | (0/1/0)

Key:
(1) meta.CVParameterSelection '-P \"C 2.0 6.0 3.0\" -X 10 -S 1 -W functions.SMO -- -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1
(2) meta.CVParameterSelection '-P \"C 0.1 0.5 5.0\" -X 10 -S 1 -W trees.J48 -- -C 0.25 -M 2' -6529603380876641265

```

Figure 4.3: **diabetes.arff** with meta-classifier CVParameterSelection

Following this, we notice which are the satisfactory values and use the WEKA Experimenter in order to operate a search by means of range. We also mention that these meta-classifiers are only accessible with newer WEKA versions 3.7.11+.

In order to serve as anchorage for our tests, for some of the considered data mining algorithms we also carried out an OCTAVE implementation, Appendix 1, using a purely OCTAVE code, with no external libraries or plugins. We report that the accuracy for **breast-cancer-wisconsin.arff** was: Logistic - 97.138%, MLP - 96.566%, **credit-g.arff** Logistic - 72.05%, MLP - 70.98% and **diabetes.arff** Logistic - 74.25%, MLP - 71.80%.

Within this section, we explored the performance of LAD and to evaluate such characteristics as accuracy. Moreover, we shortly summarized the concept at the basis of various well-known classification algorithms used as reference and which serve at evaluating accuracy within the validation procedures of our computational studies.

Though a decade ago research mainly focused on theoretical enhancements and on the generic computational implementation, more recently the focus has switched to everyday application of such methodologies as LAD in fields of utmost priority, namely medicine. Nevertheless, in order to fully explore the classification power and data mining capabilities of LAD, the researcher will need to sacrifice great amounts of time.

This section was intended to provide an accurate computational evaluation of LAD models and compare them with other data mining algorithms. Further research in the field aims at assessing the time needed for designing a LAD model, together with finding means for reducing it as much as possible, while also ensuring accurate results. Following this stage, we can focus on the development of LAD models for various other types of medical issues.

4.2.3 Results and analysis

There are two main directions we have focused on within previous section of the thesis. The first was to assess the utility of the LAD methodology, as well as the suitability of LAD for the two datasets, attended by a judgment of the results obtained. The main conclusions of this stage are:

- LAD is a methodology which combines combinatorics basic notions, operational research and optimization in order to achieve a classification of a satisfactory percentage. This conclusion can be verified for "breast-cancer-wisconsin.arff", where the accurate classification percentage was of 96.10% or the others datasets, where the accurate classification percentage was more than 75%-80%. The assessment was made by means of the application, available free of charge but which was slightly adapted by the author in the purpose of the ease of use.

We remind here that the adaptations were: converting the application in order that it could be used in Windows (the original application was developed for Linux), converting the application so that it becomes a console application, not a command

line application and the optimal rewriting in C++ of some methods (dynamic allocation instead of static allocation, we change the parameters of some functions, in those cases where they had more than four parameters, we changed some structures in classes and we wrote some Object-oriented modules). We added some new necessary methods, like the one for calculating standard deviation. Additionally, because k -folding method is the most frequently used cross-validation technique, we evaluate the accuracy of LAD using one random 10-fold cross-validation and for this purpose, we introduced some new methods.

- There are several software applications available for LAD, but at present it is not a largely used methodology. It might be that the integration of LAD in a collection of data learning algorithms like WEKA would make for a wise option;
- Results obtained with LAD proved very interesting in isolated cases and in particular fields, such as medicine, but they also demonstrated several drawbacks. One of the most significant is the uncontrolled generation of patterns;
- Considering the advances of technology, any Object-oriented implementation for LAD would not be sensible;
- Within [13] we have discussed about the economic implication that such methodology as LAD can have. In fact, it is true with any classification matter that the lower the costs, the more efficient the method.

The second direction this study aimed at considering was the comparative research of LAD and several algorithms known within the specialized fields. Of course, we also had to consider the fact that LAD remains a methodology used at a limited level and quite new in the field. Nevertheless, within the comparative research it was proved that:

- For the selected datasets, LAD offered even better results than in the case of other methods used at larger scale (such as MLP). Percentages obtained in the classification come to prove this aspect;
- From the perspective of the execution duration, LAD holds the highest grounds. It's certain that they are much quicker methodologies, but LAD offers a more decent execution time, compared to other results obtained in the research. For instance, MLP lasts shorter, but the end results are poorer;
- From the perspective of the applied notions, LAD does not require so much mathematical background knowledge, or yet, better said, it involves the knowledge of quite simple notions, such as the logic behind the Boolean functions, their simplification or solving techniques for set covering problems.

An important contribution of the author in this chapter was that of having undergone a computational evaluation of the LAD methodology and made comprehensive comparisons with other specific machine learning techniques. The overall conclusion is that research in the field of LAD methodology has made great progress, while it counts among classification methods rated as being highly intelligent. Our experiments led to the conclusion that the most considerable advantage of the methodology studied is that it provides self-explanatory/with justification classification, when compared to other methods. In order to obtain the best results possible, we evaluated the accuracy of the classification models in this chapter using one random 10-fold cross-validation procedure. All results provided are to be found in [11].

Another contribution was viewing LAD in economic parameters A. Băicoianu, S. Dumitrescu [13]. This was not difficult because the performance of the present methodology has been so far certified by numerous fruitful applications.

Further research in the field aims at assessing the time needed for designing a LAD model, together with finding means for reducing it as much as possible, while ensuring accurate results. Following this stage, we can also focus on the development of LAD models for various other types of medical issues which can use data mining techniques for risk evaluation.

At the same time, we intend to integrate results obtained with LAD to WEKA, since, as mentioned before, WEKA is a suitable tool for developing new machine learning schemes. Furthermore, WEKA can be extended as to include the elementary learning schemes designed not only for research but also for various other educational purposes. Achieving this aspect means that the use of LAD methodology as a classification tool will become much easier and user-friendly.

Additionally, as a perspective on this presented data mining tool, we would like to extend our previous work on Haskell [12] to some pattern mining algorithms. We intend to use specific libraries, like HLearn - Haskell based library for machine learning [82] - in some specific problems of data mining, particularly on LAD. An interesting perspective would be to find a Haskell alternative solution for practical problems basically solved with LAD methodology.

Among the observed advantages of LAD following experiments, we recall:

- the quality of patterns provided;
- the time necessary to achieving the experiments;
- the accuracy of results (quite low standard deviation);
- the confusion matrix obtained indicates minimal classification errors;
- the methodology does not depend on many parameters or constant values (like the other algorithms discussed here), hence the possibility to obtain different results within repeated experiments diminishes.

Among the disadvantages of LAD we recall:

- there is no automatic tool to support the LAD methodology;
- the existent code for the methodology is tetchy and difficult to run or to optimize, thus extending the application is a costly and time, inefficient process;
- from tests achieved on the datasets tested, it was noticed the LAD application provides unsatisfactory results when there are invalid recordings at the level of data sets (e.g. "Parkinsons Data Set", "seismic-bumps Data Set" [83] - the author avoided choosing such datasets);
- in the case of test repetition (for instance 20/30 times 10 fold cross validation), the time required for execution is of too great an amount (hours/days) - hence the conclusion that the application should be largely optimized;
- in the case of a great number of thresholds, the classification time for LAD proves exponential growth.

We recommend the LAD methodology for datasets with medical content, where invalid/incomplete information lacks entirely. The diffusion of the LAD methodology within the medical field is partly explainable by the fact that LAD generates justifiable/explainable patterns that is patterns which can be thoroughly understood by specialists in the field of medicine. In this field, such methodology is important, since it entails an automatic system of patient distribution or a healthy/unhealthy distribution of patients. This clearly results in reduced time for diagnosis. At the same time, it is certain that achieving such meaningful "patterns" is not a priority in other fields, where the percentage of accurate classification is more important.

4.3 Summary and future work

Within this chapter, we described the methodology of LAD and discussed its main components. Moreover, we shortly summarized the concept which lies at the basis of various well-known classification algorithms used as reference and which served at evaluating accuracy within the validation procedures of our computational experiments. We have also described a set of ground-breaking and opportunistic applications of an optimized design of LAD model for classification.

It is to be noticed that research of LAD has made great progress since early original publications on the topic. Though a decade ago research mainly focused on theoretical enhancements and on generic computational implementation, more recently the focus has switched to the applied application of such methodologies as LAD in fields of utmost priority, namely medicine. This particular application of LAD dawned in 2002-2003 when the results of a collective research study carried out together with clinicians from a healthcare foundation [3, 4] were officially published.

Numerous studies showed that the accuracy obtained by means of LAD models is fairly comparable to that of the best rated methods in data analysis, while results obtained with LAD methodology closely relate or even better those obtained by means of other similar methods. In the meanwhile, the value of LAD was in more than one occasion reconfirmed and enhanced in medical applications. The array of medical applications of LAD proposes that LAD has the potential of being further developed in order for it to support and provide new combinatorial understanding of a wide variety of studies, including SNP data analysis or studies on genetic disorders. Further on, such findings would help integrate results from different genomic, genetic and proteomic platforms.

Chapter 5

Conclusions

The aim of this thesis was to study various types of discrete/combinatorial optimization problems which have practical applications in various scientific fields. The entire range of studied problems stem from concrete applications to be usually encountered in real life situations. In regard to this particular selection of problems, we have set out to depict a variety of ground-breaking and opportunistic applications of optimization for the design of LAD models for classification and regression.

Below, you will find a brief review of the study and the most important aspects which confirm it as a relevant contribution to various scientific fields of knowledge and research.

The major contributions of this thesis in the area of rectangular two dimensional cutting problems can be found in papers authored by Iacob P., Marinescu D. and Băicoianu A. [34], D. Marinescu, A. Băicoianu [48], D. Marinescu, A. Băicoianu [49], D. Marinescu, A. Băicoianu [50], [51], D. Marinescu, A. Băicoianu [52], D. Marinescu, A. Băicoianu [53]. Each of these papers presents some original points of view and optimized algorithms on two dimensional cutting problems.

In Section "The generating cutting-covering solutions using Euclid's algorithm" our objective was to find some receipts for solving the cutting and covering problem, using a polynomial method based on Euclid's algorithm for finding the greatest common divisor. The complexity of this method is based on Euclid's algorithm and Lamé's theorem. We define the properties for our algorithm, we construct the set of receipts and we establish that the construction in one direction of the cutting-covering receipt is a geometrical construction of Euclid's algorithm, see Theorem 1. We were able to conclude the complexity of the new method based on the number of divisions in Euclid's algorithm using Lamé's theorem. The novelty of this problem is given by using the Euclid's algorithm for solving this kind of problems. The results within this section can be found in the paper authored by Iacob P., Marinescu D. and Băicoianu A. [34].

In Section "The determination of the guillotine restrictions for a rectangular covering model" we emphasize the cutting and covering problems with guillotine restrictions.

Using the graph representation of the cutting or covering pattern [47, 44] we presented in this section another analytic method for the testing of the guillotine restriction based on the decomposition of a graph in connex components. In [51] we used the graph representation of the cutting and covering pattern to prove the connection between guillotine restrictions and the connex components of the graph. We started from this connection and we presented in this section an algorithm which can be used to verify the guillotine restrictions in a two-dimensional covering model. The algorithm is originally developed by the authors of the paper. We introduced the notions of rectangular covering model, guillotine restrictions, downward adjacency, rightward adjacency and we considered the graph of downward adjacency and the graph of rightward adjacency for defining the cuts for the rectangular cutting and covering problem. The results from Theorems 13 and 14 suggest an algorithm for the verification of the guillotine restrictions, using the decomposition of graphs G'_d or G'_r in connex components, defined in [44]. The novelty for this section is given by the manner of defining the cuts and the algorithm. The correctness of the algorithm follows from the Theorems 13 and 14, that make the connection between a guillotine cut and the decomposition of the graph G'_d or G'_r in connex components. The algorithm for finding the connex components has the complexity $O(m)$, where m is the number of the arches [17, 19]. So the complexity of V-CUT or H-CUT procedures defined is also $O(m)$. It follows that the complexity of PREORDER procedure for a rectangular covering model of k items with guillotine restrictions is $O(km)$. We note that the results obtained within this section complete the results obtained in [47] and detailed results are found in [51]. Also, the next section completes this section, giving some extra information about the algorithm and a particular example with all iterations.

In section "The determination of the guillotine restrictions for a rectangular cutting-stock pattern" we consider a two-dimensional rectangular cutting stock problem in case of a cutting pattern with gaps. First we presented two new graph representations of the cutting pattern, weighted graph of downward adjacency and weighted graph of rightward adjacency. Using this kind of representation we propose a method to verify guillotine restrictions of the pattern which can be applied for cutting-stock pattern with gaps but also for the covering pattern without gaps and overlapping. The results from the Theorem 22 suggest an algorithm for the verification of the guillotine restrictions, in case of a cutting-stock pattern with gaps. The defined algorithm has for input the weighted graphs G_d or G_r attached to a rectangular cutting pattern, the output is the s-pictural representation of the cutting pattern like a formula in a Polish prefixed form. The algorithm constructs the syntax tree for the s-pictural representation of the cutting pattern, starting from the root to the leaves (*procedure PRORD*). For every vertex of the tree it verifies if it is possible to make a vertical (*procedure VCUT*) or horizontal cut (*HCUT procedure*), using an algorithm for decomposition of a graph in two components. We note that we can apply this algorithm also in the case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. The correctness of the algorithm follows from the Theorem 22, that makes the connection between a guillotine cut and the decomposition of a graph in two subgraphs. The procedure *PREORD*(

represents a preorder traversal of a graph, so the complexity is $O(k)$ [19], where k is the number of the cutting items. Also, in the procedure *VCUT*, respectively *HCUT* we traverse a subgraph of the initial graph. So, the complexity of the algorithm is $O(k^2)$. All the results exposed here are presented in [52].

We note that all the studied problems within the present chapter point out some new approaches of solving the cutting and rectangular problems. We defined various algorithms for this problem and we solved different particular types of two dimensional rectangular cutting and covering problems.

The major contributions of this thesis in the area of three dimensional bin packing problems can be found in papers [56, 59, 57, 58]. Each of these papers presents some original points of view and optimized algorithms on three dimensional bin packing problems.

The section "A topological order for a rectangular three dimensional bin packing problem" is an extension of one of the previous work of D. Marinescu, P. Iacob and K. Kiss-Jakab [55] regarding the two dimensional covering problem to a rectangular three dimensional bin packing problem, where a bin is packed with a set of rectangular boxes, without gaps or overlapping. We present a kind of topological sorting algorithm for this problem, of linear complexity, **OVERDIAG-3D Algorithm**. By extending the two dimensional covering model [44], we defined in this section three kinds of adjacency relations, adjacency in the direction of Ox , Oy and Oz , definitions in 23, 25 and 27. Starting with these three kinds of adjacency we define three kind of graphs: the graph of adjacency in direction Ox , OY and Oz and we gave a concrete example of a packing model. The novelty of this problem is given, on the one hand, by the mathematical models that we introduce and, on the other hand, by the fact that we used these extended types of graphs for this kind of three dimensional bin packing problems. Also, we discovered some important properties, see 32, 33. Due the Theorem 33 it is possible to represent simultaneously these obtained graphs by a single adjacency matrix, a matrix with elements from the set $\{0, 1, 2, 3\}$. For any packing model we defined a network, a graph of compound adjacency, 34 and we prove that the graph is acyclic, see 35. To determine a topological order we used a new algorithm, **OVERDIAG-3D** which is an extension of a topological sorting algorithm presented in [19]. This algorithm is based on the particularity of the compound graph defined, respectively on the form of the resulted matrix, attached to the graph. The authors' achievements within this section can be found in paper [56]. It completes the results obtained in [44] and [55]. A plan for loading of the boxes in the bin is obtained using a topological sorting algorithm of the vertices of this acyclic graph. Here we turn our attention to the practical example discussed in the previous section. All the results obtained and discussed in this section are in [59].

In section "The determination of the guillotine restrictions for a rectangular three dimensional bin packing pattern" we discussed the rectangular three dimensional bin packing problem, where a bin is loaded with a set of rectangular boxes, without overlapping. One of the most popular restriction for the solution of the three dimensional bin

packing problem is the guillotine restriction, see 40. Our objective here was to find a method for verifying if a solution of the three dimensional bin packing problem has the guillotine constraints or not. For this purpose we used a weighed graph representation 43 of a solution of the problem, the generalization of this kind of representation obtained by us for two dimensional cutting stock problem in [48, 49, 50]. Theorem 45 introduces some properties for the weighed graphs attached to the pattern. The results from the previous theorem suggest an algorithm for the verification of the guillotine restrictions, in case of a bin-packing pattern with gaps but without overlapping [57]. We remark that we can apply this algorithm also in case of a cutting-stock pattern without gaps and, of course, in the case of covering pattern with or without gaps. An extended example is discussed here and the algorithm iterations are highlighted. The prefix Polish notation for the resulted syntactic tree is given and the algorithm's complexity was studied [58].

The major contributions of this thesis in the area of LAD can be found in papers of A. Băicoianu [11], A. Băicoianu, A. Vasilescu, R. Pândaru [12], A. Băicoianu, S. Dumitrescu [13].

Clearly, the accuracy of a LAD model is directly related to the quality of the patterns in the model (prevalence and homogeneity). We used the results from [4, 3, 41] and we did our own experiments for investigating the performance of LAD. Also, the LAD algorithm was compared here with the main classification and regression algorithms used in the machine learning literature, and their implementation in WEKA [72].

For our final experiments, we used the public files from LADtools [41]. The personal contribution of the author in this area is the change made on this software. We remind here that the adaptations of the original application were: converting the application in order that it could be used in Windows (the original application was developed for Linux), converting the application so that it becomes a console application, not a command line application, the optimal rewriting in $C++$ of some methods, we added some new necessary methods, like the one for calculating standard deviation, we evaluate the accuracy of LAD using one random 10-fold cross-validation and for this purpose, we used two new methods.

We evaluated the accuracy of classification models in "Computational Experiments" section using one random 10-fold cross-validation. The other classification methods as well as the regression methods used for comparison are available in the WEKA package [36]. For the purpose of comparing our results with these of algorithms implemented in the WEKA package, we used the LAD accuracy definition, presented in [4]. Compared to the commonly used machine learning algorithms implemented in the publicly available WEKA software package, the implementation of LAD was shown here to be highly competitive classification algorithm. However, to fully benefit from the classification power and data mining capabilities of LAD, one must often go through a time consuming process. It is part of future research plans to assess the time needed to construct the LAD model and to minimize it as much as possible, with the best results. Also, is in our future plan to develop LAD models for other types of medical problems, that were resolved with data mining tools. Moreover, we intend to develop a LAD algorithm for WEKA. If

we resolve this issue, then the use of LAD algorithm in classification will become easier. Additionally, as a perspective on this presented data mining tool, we would like to extend our previous work on Haskell [12] to some pattern mining algorithms. We intend to use specific libraries, like `HLearn` - a Haskell based library for machine learning [74] - in some specific problems of data mining/machine learning, particularly on LAD [12]. An interesting perspective would be to find a `Haskell` alternative solution for practical problems basically solved with LAD.

Appendices

Appendix 1

Further on, we shall discuss the code designed in Octave by the author for the logical regression. More information on the Octave language code together with extra details on its setup process can be found at:

<http://www.gnu.org/software/octave/download.html> and
<http://www-mdp.eng.cam.ac.uk/web/CD/engapps/octave/octavetut.pdf>.

We will also mention a few general features of Octave, since some of them stand as important reasons for having chosen this language code in order to implement some of the classification algorithms discussed in Chapter 4, in view of their further comparison with LAD:

- GNU Octave is a high-level language used in numerical calculations;
- It is considered to be a less enhanced clone of MATLAB, while the key feature is its availability free of charge;
- The system can be used interactively or for running command files;
- This language code is portable on numerous operation systems;
- The Octave language code can be called directly from $C++$, and the other way round;
- This language code comes with support for complex numbers and matrices;
- It provides a wide array of matrix functions;
- Most of the times, the Octave language is exemplified in the command-line, but there are also various graphic work interfaces which come with it.

For further details on logical regression, see Chapter 4. The implementation was modular, so that algorithm testing for the datasets selected might be less intricate.

The first step was to randomly mix all selected data, by means of the shuffled function. This function deals with the random permutations of the lines in a matrix, by means of the random function of the Octave code, in order to generate index permutations.

```
function shuffled = shuffle (matrix)
    randomIndices = randperm(rows(matrix));
    shuffled = matrix(randomIndices, :);
endfunction
```

Next, the function `augmented` was described. This adds a first column filled with 1 to a given matrix. See logical regression algorithm [30, 31].

```
function augmented = augment_matrix (m)
    augmented = [repmat(1, rows(m), 1), m];
endfunction
```

We then use a function for reading input data. This function reads the data in the file and creates a matrix:

```
function [X, y] =
readData (filename, inputSize=given_number, outputSize=given_number)
    data = dlmread(filename, ' ');
    data = data(:, 1:columns(data)-1);
# the last value is always zero, garbage due to read
# shuffle data
    data = shuffle(data);
# put input values into X
    X = data(:, 1:inputSize);
# prepend a column filled with 1
    X = augment_matrix(X);
# put outputSize columns in yy
    yy = data(:, 1+inputSize:inputSize+outputSize);
# find locations of nonzero values in y
    [i, j] = find(yy);
    y = zeros(rows(data), 1);
    y(i) = j .- 1;
endfunction
```

It is necessary to use another function for the implementation of the predictive pattern it is preferred that this function is vectorized. It is also quite mandatory to have a function for calculating the error rate:

```
function probability = h(X, theta)
    probability = 1 ./ ( 1 .+ exp(-X * theta) );
endfunction
```

```
function error = J (X, theta, y)
    m = rows(X);
```

```

predicted = h(X, theta);

error = 0;
error += sum(log(predicted(y==1)));
error += sum(log(1 - predicted(y==0)));
error *= -1/m;

endfunction

```

The last function written for the logical regression is `doAll`, which divides the dataset into a given fraction in this specific situation it is 70% training data and 30% test data. Several values will be tested for the λ regularization hyper parameter, as well as for the α learning rate. If necessary, the evolution of the J error function will be graphically represented, while checking if any of these values decreases as they repeat:

```

function [theta, errors, X, y] =
doAll (filename, alpha=0.1, maxIters = 100)
    [X, y] = readData(filename);
    featuresCount = columns(X);
    #rand("seed", 7);
    theta = rand(featuresCount, 1) - 1;
    errors = []; #no error so far
    iters = 0;
    do
        iters += 1;

        theta -= alpha * X' * ( h(X, theta) - y );

        error = J(X, theta, y);
        errors = [errors, error];
    until iters == maxIters;
endfunction

```

Another algorithm we found to be very interesting compared to LAD was the multilayer perceptron. Alongside testing its results with WEKA, we decided to undergo its implementation in Octave. Further details on the algorithm are to be found in Chapter 4. The written implementation in Octave obeys the MLP principles. We felt that making a few comments on the code might be of use for grasping its deeper meaning:

```

function read()

# data read
data = load('breast-cancer-wisconsin.data');

```

```
z = data(:, 1:256);\% 1593 * 256
d= data(:, 257:266);\% 1593 * 9

# data permutatio
[z, d] = F1_permutation(z, d);
#impartire date
[z_train, d_train, z_test, d_test] = F2_splitData(z, d);

z_train = F3_addMinusOnes(z_train);
z_test = F3_addMinusOnes(z_test);

# minimum 2 neurons
# hidden layer dimension, number of neurons
# number of neurons from each hidden layer
J = 8;
K = 10;

# the way for finding the weights
[V, W] = algorithm(z_train, d_train, J, K);

# depending of the weights we calculate the percentage
percentage = test(z_test, d_test, V, W, J, K)
endfunction

function [newX, newY] = F1_permutation(X, y)

# random permutation of the lines
random_i = randperm(rows(X));
newX = X(random_i, :);
newY = y(random_i, :);
endfunction

function [X_train, y_train, X_test, y_test] = F2_splitData(X, y)
# split the data, 70\% for training set
percentage = 70;
m = rows(y);
nrTrain= int32 (percentage/100*m);

X_train = X(1:nrTrain, :);
y_train = y(1:nrTrain, :);

X_test = X((nrTrain+1):m, :);
```

```
y_test = y((nrTrain+1):m,:);
endfunction

function newZ = F3_addMinusOnes(z)
# adding the column with 1
newZ = [z, (-1)*ones(size(z)(1,1), 1)];
endfunction

function E = F4_error(E, d_k, o_k)
    E = E + 1/2(d_k - o_k);
endfunction

function f = sigmoid(net)
f = 1./ (1.+ exp(-net));
endfunction

function [V, W] = algorithm(z_train, d_train, J, K)

# training set
P = rows(z_train);
# the number of columns
I = columns(z_train);

niu = 0.5;\%learning rate
Emax = 0.001;\%minimum value of error

# the first layer of weights
W = rand(K, J) / 100;
# the second layer of weights
V = rand(J, I) / 100;

# each line from our testing data
p = 1;
q = 1;
E = 1;

# the last line from hidden layer
y(J, 1) = -1;

while(E > Emax)
    E = 0;
    for p = 1:P
```

```

z = z_train(p, :);
d = d_train(p, :);

for j=1:J-1
    y(j, 1) = sigmoid(V(j, :) * z');
endfor

for k=1:K
    o(p, k) = sigmoid(W(k, :) * y);
endfor

E = E + (1/2)*sum((d - o(p, :)).^2);

delta_o = (d(1, :) .- o(p, :)) .* (1 .- o(p, :)) .* o(p, :);
delta_y = y .* (1 - y) .* (W' * delta_o');

\% update the weights
W = W + niu * delta_o' * y';

V = V + niu * delta_y * z;
endfor

E = sqrt(E)/(P*K)
all_errors(q, 1) = E;

q = q + 1;
endwhile
q
plot(all_errors);
endfunction

function percentage = test(z_test, d_test, V, W, J, K)

P = size(z_test)(1, 1);

y(J, 1) = -1;

# number of predicted correct output
corect = 0;
for p = 1:P
    z = z_test(p, :);
    d = d_test(p, :);

```

```
for j = 1:J-1
    y(j, 1) = sigmoid(V(j, :) * z');
endfor

for k = 1:K
    o(p, k) = sigmoid(W(k, :) * y);
endfor

poz_d = find( d == max(d) );

poz_o = find( o(p, :) == max( o(p, :) ) );

if ( poz_d == poz_o )
    corect = corect + 1;
endif
endfor

# correct percentage
percentage = (corect * 100) / P;
endfunction
```

The results obtained with these two algorithms are: logical regression recognition rate between 87.0005% and 90.55%, MLP 91.10% and 94.766%. The time is not so important for our tests, but they were close enough.

Appendix 2

This Appendix is meant to highlight the changes the author underwent on the existing LAD processing tool, a software application which was initially implemented by E. Mayoraz. For further details, please refer to [60].

We remind here that the adaptations of the original application were: converting the application in order that it could be used in Windows (the original application was developed for Linux), converting the application so that it becomes a console application, not a command line application and the optimal rewriting in *C++* of some methods (dynamic allocation instead of static allocation, we change the parameters of some functions, in those cases where they had more than four parameters, we changed some structures in classes and we wrote some Object-oriented programming modules).

Since the modules are fairly consistent, we will only chose some of them in order to exemplify our purpose.

The header "basic.h" was mainly modified so that it could allow running Windows. Thus, we introduced: `if defined (_WIN32)`, and `math.h`, for the mathematical functions. Such implementations considered to be necessary to the task were introduced by the author. For instance, a generic function (such as template) for interchanging two values:

```
template< void Exchange (T &X, T &Y)
{
    if (& X == & Y)
        return;

    T Z = X;
    X = Y;
    Y = Z; }
```

Were also defined:

```
#define ForBoolean(B)
    for (boolean B = false; B <= true; B++)
#define ForBooleanDown(B)
    for (boolean B = true; B >= false; B--)
```

Both of them were used in order to ease to code writing process.

The header "cutPtsSet.H" was consistently modified. This contained defining elements for cutpoints. The original version of the file is to be found in the specific documentation. The code we are using for LAD testing is (we introduce here just a piece from it):

```

#ifndef cutPtsSet_h
#define cutPtsSet_h
#include "../Basics/basic.h"
#include "../Matrices/Matrix.h"
#include "../Matrices/setCovering.h"
#include "../Matrices/spareM.h"
#include "multiDS.h"
#include <iostream>
extern const char nonMonotonicAttr;
extern const char positiveAttr;
extern const char negativeAttr;
#define _val_GEQ_cp 1
#if 1
#define cutPtsSet_Plus1(x) ((x)+1)
#define cutPtsSet_Div2(x) ((x)>>1)
#define cutPtsSet_Tim2(x) ((x)<<1)
// else if T is float or double
#else
#define cutPtsSet_Plus1(x) (x)
#define cutPtsSet_Div2(x) ((x)/2.0)
#define cutPtsSet_Tim2(x) ((x)*2.0)
#endif

tcT class multiDS; // forward

tcT class cutPtsSet : private Matrix<T>
{
public:

cutPtsSet ()
: Matrix<T>(0),
origin(0),
weight(0),
var(0),
sigmaCP(0),
span(0)
{ }

```

```

cutPtsSet (multiDS<T>& d, float& confidence,
           const char method=0, const char weighting=0)

:
Matrix<T>(0),
origin(0),
weight(0),
var(0),
sigmaCP(0),
span(0)
{
if ( method==0 )
{
resize(d.ds[0].dim()*d.distinct(), d.ds[0].dim());
weight=0.0;
Matrix<T> valcat(d.distinct(),2);
int nbTh=0;
for (int attr=0; attr<d.ds[0].dim(); attr++)
{
sigmaCP(attr) = ( attr ? sigmaCP(attr-1) : 0);

int cl,clPtr;
for (cl=0, clPtr=0; cl<d.nbCats(); clPtr+=d.ds[cl].distinct(), cl++)
{
valcat.s(clPtr,d.ds[cl].distinct(),0,1) = d.ds[cl][attr];
valcat.s(clPtr,d.ds[cl].distinct(),1,1) = (T)(cl);
}
valcat.sort();
int total_rows=valcat.m(), first_d_k;
for(first_d_k=0;
    first_d_k<total_rows && known(valcat(first_d_k,0));
    first_d_k++)

;
if(first_d_k < total_rows)
{
int last_d_k;
for(last_d_k=first_d_k;
    last_d_k+1<total_rows && unknown(valcat(last_d_k+1,0));
    last_d_k++)

;
valcat.s(first_d_k,total_rows-last_d_k-1) =
    valcat.s(last_d_k+1,total_rows-last_d_k-1);

```

```
total_rows -= last_d_k - first_d_k + 1;
}
span(attr) = float (valcat(total_rows-1,0) - valcat(0,0));

boolean mixGroup=false;

T prevThresh=(T) (valcat(0,0));
T vall;
boolean newCutPtsSet;
for (int i=1; i<total_rows; i++)
{
newCutPtsSet=false;
if ( valcat(i-1,0) != valcat(i,0) )
{
vall = valcat(i-1,0);
newCutPtsSet = ( valcat(i-1,1)!=valcat(i,1) || mixGroup );
mixGroup=false;
}
else if ( valcat(i-1,1)!=valcat(i,1) ) // && vc(i-1,0)==vc(i,0)
{
newCutPtsSet = ( prevThresh != valcat(i,0) );
mixGroup=true;
}
if ( newCutPtsSet )
{
Array<T>::operator()(nbTh) = vall + valcat(i,0);
if ( weighting==2 )
weight(nbTh) = float(valcat(i,0)-vall)/(2.0*span(attr));
var(nbTh) = attr;
sigmaCP(attr)++;
nbTh++;
prevThresh=valcat(i,0);
}
}
}
resize_(nbTh);
weight.resize_(nbTh);
var.resize_(nbTh);
}
else if ( method==1 )
{
resize(d.ds[0].dim()*d.nbPairs(), d.ds[0].dim());
```

```

Matrix<T> my_cut_pts(d.nbPairs());
int nbTh=0;
for (int attr=0; attr<d.ds[0].dim(); attr++)
{
sigmaCP(attr) = ( attr ? sigmaCP(attr-1) : 0);

Matrix<int> singleton(1,1,attr);
multiDS<T> proj(d,singleton);
proj.sort().checkMult();

int ptr=0;
T ll,rr;
for (int clL=0; clL<proj.nbCats()-1; clL++)
for (int clR=clL+1; clR<proj.nbCats(); clR++)
for (int indL=0; indL<proj.ds[clL].distinct(); indL++)
for (int indR=0; indR<proj.ds[clR].distinct(); indR++)
if (
    (ll=proj.ds[clL](indL,0))!=(rr=proj.ds[clR](indR,0)) &&
    known(ll) && known(rr) &&
    ( ll<rr ? d.monotone(attr)!=negativeAttr
      : d.monotone(attr)!=positiveAttr )
)
my_cut_pts(ptr++) = ll+rr;
if ( ptr )
{
my_cut_pts.s(0,1,0,ptr).sort();
T mini,maxi;
dataSet<T> merged;
merged = proj.merge(false);
merged.sort();
int i;
for(i=0; i<merged.distinct() && unknown(merged(i,0)); i++)
{}
if (i==merged.distinct())
span(attr)=1;
else
{
mini=merged(i,0);
for(i=merged.distinct()-1; i>=0 && unknown(merged(i,0)); i--)
{}
maxi=merged(i,0);
span(attr) = maxi - mini;
}
}

```



```

}
Array<T>::operator() (nbTh) = my_cut_pts(0);
var(nbTh) = attr;
sigmaCP(attr)++;
nbTh++;
for (int pth=1; pth<ptr; pth++)
{
if ( my_cut_pts(pth)!=my_cut_pts(pth-1) )
{
Array<T>::operator() (nbTh) = my_cut_pts(pth);
var(nbTh) = attr;
sigmaCP(attr)++;
nbTh++;
}
}
}
else
span(attr)=0;
}
resize_(nbTh);
weight.resize_(nbTh);
var.resize_(nbTh);
}

Array<int> list(d.listOfPairs());
float minConfi=1.0;
for (int pair=0; pair<list.m(); pair++)
{
float maxGap=0.0;
for (int pcp=0; pcp<nbCP(); pcp++)
{
int vari=attribute(pcp);
T val1 = minimum( d.ds[list(pair,0)](list(pair,1),vari),
                  d.ds[list(pair,2)](list(pair,3),vari));
T val2 = maximum( d.ds[list(pair,0)](list(pair,1),vari),
                  d.ds[list(pair,2)](list(pair,3),vari));
if ( between(val1,val2,pcp) &&
      ( val1<val2 ? d.monotone(vari)!=negativeAttr
        : d.monotone(vari)!=positiveAttr ) &&
      span(vari)>0.0
    )
maximize(maxGap,

```

```

        float ((minimum(operator()(pcp)-cutPtsSet_Tim2(val1),
                               cutPtsSet_Tim2(val2)-operator()(pcp)) /
                (2.0 * span(vari)))));
    }
    minimize(minConfi,maxGap);
    }
    minConfi-=1E-5;
    if ( confidence > minConfi )
    flog<< endl << "The required confidence of " << setprecision(3)
    << setw(6) << confidence << " has been reset to " << setw(6)
    << (confidence=minConfi) << endl << flush;

// compute the weights
if ( weighting==1 ) // correlation with output
{
for (int pCP=0; pCP<nbCP(); pCP++)
{
int attr=attribute(pCP), set
        ;
Matrix<double> cardi(2,d.nbCats(),0.0);
for (set
        =0; set
        <d.nbCats(); set
        ++)
{
for (int ind=0; ind<d.ds[set
        ].distinct(); ind++)
switch ( compare(d.ds[set
        ](ind,attr),pCP,confidence) )
{
case +1:
{
cardi(1,set
        )+=d.ds[set
        ].multiplicity(ind);
break;
}
case 0:
cardi(0,set
        )+=d.ds[set
        ].multiplicity(ind);

```

```
}
}
if ( cardi.sum().min()(0) = 0.0 )
{
weight(pCP) = -1E10;
}
else
{
cardi /= cardi.sum().t();
double entropyPos=0.0;
double entropyNeg=0.0;
for (set
    =0; set
    <d.nbCats(); set
    ++)
{
entropyPos += ( cardi(1,set
    )>0.0 ?
    cardi(1,set
    )*log(cardi(1,set
    )) : 0.0 );
entropyNeg += ( cardi(0,set
    )>0.0 ?
    cardi(0,set
    )*log(cardi(0,set
    )) : 0.0 );
}
weight(pCP) = float(maximum(entropyPos,entropyNeg));
}
}
}
else if ( weighting==2 && method )
{
weight=1E10;
for (int attr=0; attr<d.ds[0].dim(); attr++)
{
Matrix<int> singleton(1,1,attr);
multiDS<T> proj(d,singleton);
proj.sort().checkMult();
for (int clL=0; clL<proj.nbCats()-1; clL++)
for (int clR=clL+1; clR<proj.nbCats(); clR++)
for (int indL=0; indL<proj.ds[clL].distinct(); indL++)
```

```

for (int indR=0; indR<proj.ds[clR].distinct(); indR++)
{
int val1 = proj.ds[clL](indL,0);
int val2 = proj.ds[clR](indR,0);
if ( val1 != val2 && known(val1) && known(val2))
{
int first=firstCPG( minimum(val1,val2),
                    attr,confidence);
int last = lastCPL( maximum(val1,val2),
                   attr,confidence);

if ( first>=0 )
for (int pth=first; pth<=last; pth++)
minimize(weight(pth),float(minimum(
absolute(cutPtsSet_Tim2(val1)-operator()(pth)),
absolute(cutPtsSet_Tim2(val2)-operator()(pth))))*
         float(proj.ds[clL].multiplicity(indL))*
         float(proj.ds[clR].multiplicity(indR)));
}
}
for (int pth=firstIndex(attr);
     pth<firstIndex(attr)+nbCP(attr); pth++)
if ( weight(pth)<1E10 )
weight(pth)/(= (2.0*span(attr)));
else
weight(pth)=0.0;
}
}
else if ( weighting==3 ) // global separability
{
for (int attr=0; attr<d.ds[0].dim(); attr++)
{
Matrix<int> singleton(1,1,attr);
multiDS<T> proj(d,singleton);
proj.sort().checkMult();
for (int clL=0; clL<proj.nbCats()-1; clL++)
for (int clR=clL+1; clR<proj.nbCats(); clR++)
for (int indL=0; indL<proj.ds[clL].distinct(); indL++)
for (int indR=0; indR<proj.ds[clR].distinct(); indR++)
{
int val1 = proj.ds[clL](indL,0);
int val2 = proj.ds[clR](indR,0);
if ( val1 != val2 && known(val1) && known(val2))

```

```

{
int first=firstCPG( minimum(val1,val2),
                    attr,confidence);
int last = lastCPL( maximum(val1,val2),
                    attr,confidence);
if ( first>=0 )
for (int pth=first; pth<=last; pth++)
weight(pth) += float(minimum(
                    absolute(cutPtsSet_Tim2(val1)-operator()(pth)),
                    absolute(cutPtsSet_Tim2(val2)-operator()(pth)))) *
                    float(proj.ds[clL].multiplicity(indL)) *
                    float(proj.ds[clR].multiplicity(indR));
}
}
weight.s(0,1,firstIndex(attr),nbCP(attr))/(2.0*span(attr));
}
}
}

~cutPtsSet ()
{}
tcT friend std::ostream& operator
<< (std::ostream& s, const cutPtsSet<T>& myself);
private:
cutPtsSet (int nbCP, int nbVars)
: Matrix<T>
(nbCPts),
origin(0),
weight(nbCPts),
var(nbCPts), sigmaCP(nbVars), span(nbVars)
{ }

cutPtsSet<T>& resize (int nbCPts, int nbVars)
{
Matrix<T>::resize(nbCPts);
weight.resize(nbCPts);
var.resize(nbCPts);
sigmaCP.resize(nbVars);
span.resize(nbVars);
if ( origin.n() )
origin.resize(nbCPts);
return *this;
}

```

```
}

int index (int cpIndex) const
{
#ifdef _CHECK_INDEX
if ( cpIndex<0 || cpIndex>=nbCP() )
throw whereOutOfRange( "int cutPtsSet<T>::index(int cpIndex) const",
                        "cpIndex", cpIndex, 0, nbCP()-1);
#endif

return (var(cpIndex) ? cpIndex - sigmaCP(var(cpIndex)-1) : cpIndex);
}

int firstIndex (int vari) const // Returns index of the first CP of VAR.
{
#ifdef _CHECK_INDEX
if ( vari<0 || vari>=nbVars() )
throw whereOutOfRange( "int cutPtsSet<T>::firstIndex(int vari) const",
                        "vari", vari, 0, nbVars()-1);
#endif

return (vari ? sigmaCP(vari-1) : 0);
}

int lastIndex (int vari) const
{
#ifdef _CHECK_INDEX
if ( vari<0 || vari>=nbVars() )
throw whereOutOfRange( "int cutPtsSet<T>::
lastIndex(int vari) const",
                        "vari", vari, 0, nbVars()-1);
#endif

return sigmaCP(vari)-1;
}

Matrix<T>origin;
Matrix<float>weight;
Matrix<int>var;
Matrix<int>sigmaCP;
Matrix<float>span;
// int nbAttr; // # of attributes with at least one cut point.
};
```

```

tcT std::ostream& operator <<
(std::ostream& s, const cutPtsSet<T>& myself)
{
s.setf(std::ios::fixed | std::ios::showpoint)
;
s << std::endl;
int curCP=0;
for (int v = 0; v < myself.nbVars(); v++)
if ( myself.nbCP(v) )
{
s << "v" << setw(2) << v+1 << ":";
for (int ptr=0; ptr<myself.nbCP(v); ptr++)
{
if ( ptr && !(ptr%5) )
s << std::endl << "    ";
s << std::setw(6) << ++curCP << ":"
<< std::setw(6) << std::setprecision(1)
<< myself.original(ptr,v)
<< std::setw(7) << std::setprecision(3)
<< myself.weight(myself.firstIndex(v)+ptr);
}
s << endl;
}
return s << "#total :" << myself.nbCP() << std::endl;
}
#endif

```

Generally, we introduced new Object-oriented programming elements and generic programming elements in *C++*, as well as various manipulators for clear and efficient formatting designs.

The file "Matrix.c" was modified in order to highlight matrix specific operators. In the same purposes - code reusability and code extensibility - we also implemented some new operators. Such an example of written operator is:

```

binMatrix& operator = (const binMatrix& a);
binMatrix& operator = (const binArray& a);
binMatrix& operator = (const boolean a);

```

The "Chrono.c" class was also modified. For instance, the class constructor was rewritten. In its original version this was:

```

chrono::chrono() {
    struct tms buf;

```

```

times(&buf);
starttime = laptime = (double)(buf.tms_utime)/CLK_TCK;
}

```

In our new version this is:

```

chrono::chrono()
{
#ifdef _WIN32
starttime = laptime = (double)clock() / CLOCKS_PER_SEC;
#else

struct tms buf;
times(&buf);
starttime = laptime = (double)(buf.tms_utime)/CLK_TCK;
#endif
}

```

The application we have modified contains five smaller projects: LAD, LAD.BIN, LAD.PAT, LAD.R2B, LAD.THE. Each of the individual components was partially or integrally modified, depending both on the errors reported when running Windows and on the purpose we had initially intended the application for. One of our main guiding criteria was the optimality of the written code. When running the application the result is a simple user menu, where the necessary functions for LAD processing are called.

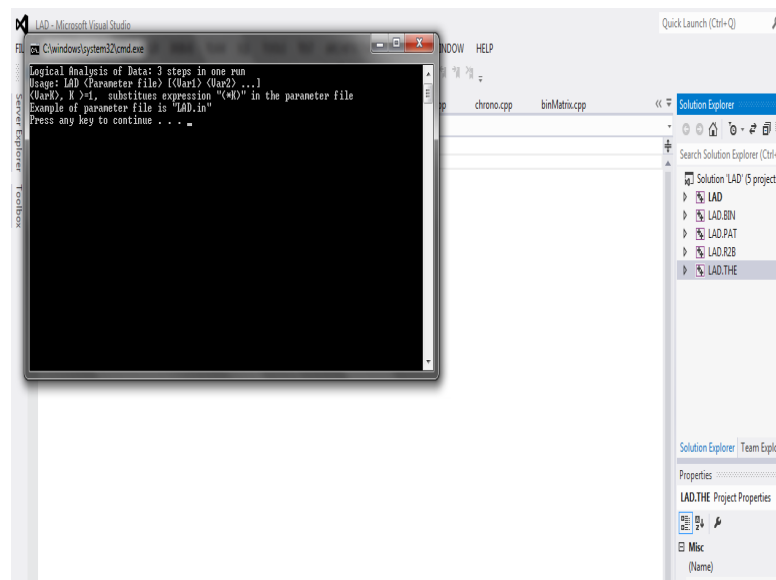


Figure 1: The application

We mention that we evaluate the accuracy of LAD using one random 10-fold cross-validation. For this purpose, here is the code for two methods:

```

void divide(Matrix<int>& partition, int K)
{
int maxCard=0;
for (int set=0; set<nbCats(); set++)
maximize(maxCard, ds[set].card());
partition.resize(maxCard+K+1, nbCats());
partition=0;
partition.s(0)=K;
Matrix<int> smallerPartition(1,K,1);
int nbSmaller = K;

for (int set=0; set<nbCats(); set++)
{
const int theFloor = floor(float(ds[set].card())/K);
partition.s(1,K,set,1) = theFloor;

int toAdd = ds[set].card() - K * theFloor;
if ( toAdd>0 && nbSmaller<=toAdd )
{
partition.s(1,K,set,1) += smallerPartition.t();
toAdd -= nbSmaller;
nbSmaller = K;
smallerPartition = 1;
}
if ( toAdd>0 )
{
int skipped=0;
for (int k=0; k<K; k++)
if ( smallerPartition(k) && partition(k+1,set)==theFloor &&
randomR() < float(toAdd)/(nbSmaller - skipped++) )
{
partition(k+1,set)++;
smallerPartition(k)=0;
nbSmaller--;
skipped--;
if ( !(--toAdd) )
break;
}
}
}
}

```

```

int finger=0;
for (int p=1; p<=K; finger+=partition(p,set), p++)
partition.s(K+1+finger,partition(p,set),set,1) = p;
for (int data=0; data<ds[set].card(); data++)
Exchange(partition(K+1+data,set),
partition(K+1+randomI(ds[set].card()),set));
}
}

```

```

void setFold(multiDS<T>& data,
const Matrix<int>& partition, const int fold)
{
if ( data.nbCats() != nbCats() )
{
delete data.ds;
data.ds = new dataSet<T> [nbCats()];
data.nbSets=nbCats();
data.list.resize();
}
const int K = partition(0,0);
data.label.resize(label);
data.normalized=normalized;
for (int set=0; set<nbCats(); set++)
{
const int cardin =(fold>0 ? partition(fold,set) :
ds[set].card()-partition(-fold,set));
Matrix<T> d(cardin,dim());
Matrix<int> multiple(1,cardin,1);
Matrix<int> labelle(1,cardin);
int ptrD=0;
int ptrDS=0;
int mu=1;
for (int ptrP=0; ptrP<ds[set].card(); ptrP++, mu++)
{
if ( mu > ds[set].multiplicity(ptrDS) )
{
mu=1;
ptrDS++;
}
if ( fold>0 ? partition(K+1+ptrP,set)==fold :

```

```
partition(K+1+ptrP, set) != (-fold) )
{
d.s(ptrD)=ds[set](ptrDS);
labelle(ptrD)=ds[set].label(ptrDS);
ptrD++;
}
}
data.ds[set] =
dataSet<T>(d,multiple,labelle,ds[set].monotone);
}
}
```

Appendix 3

This Appendix is meant to highlight and sustain the results from Chapter "Optimization in Logical Analysis of Data", section "Computational Experiments".

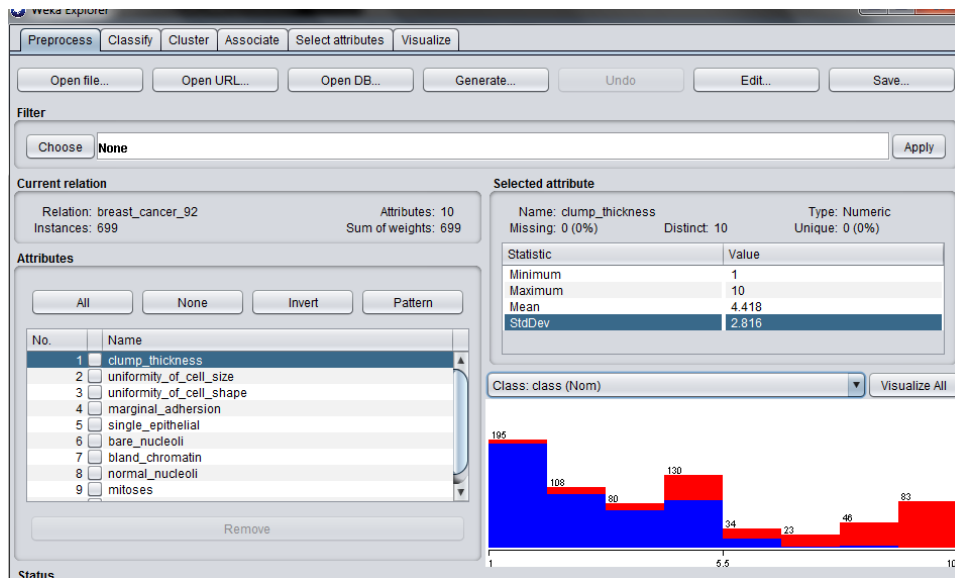
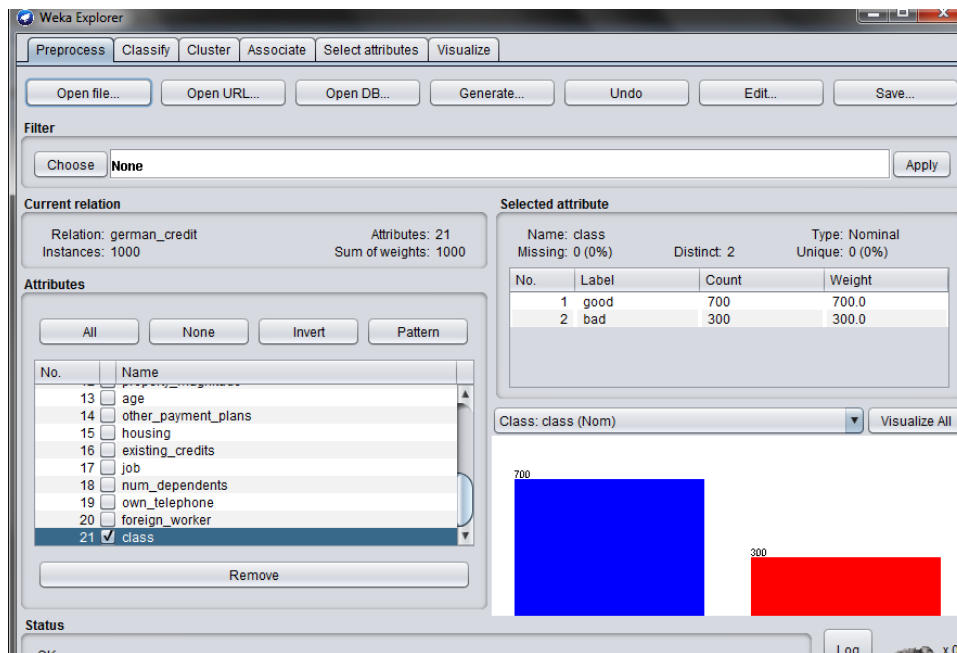
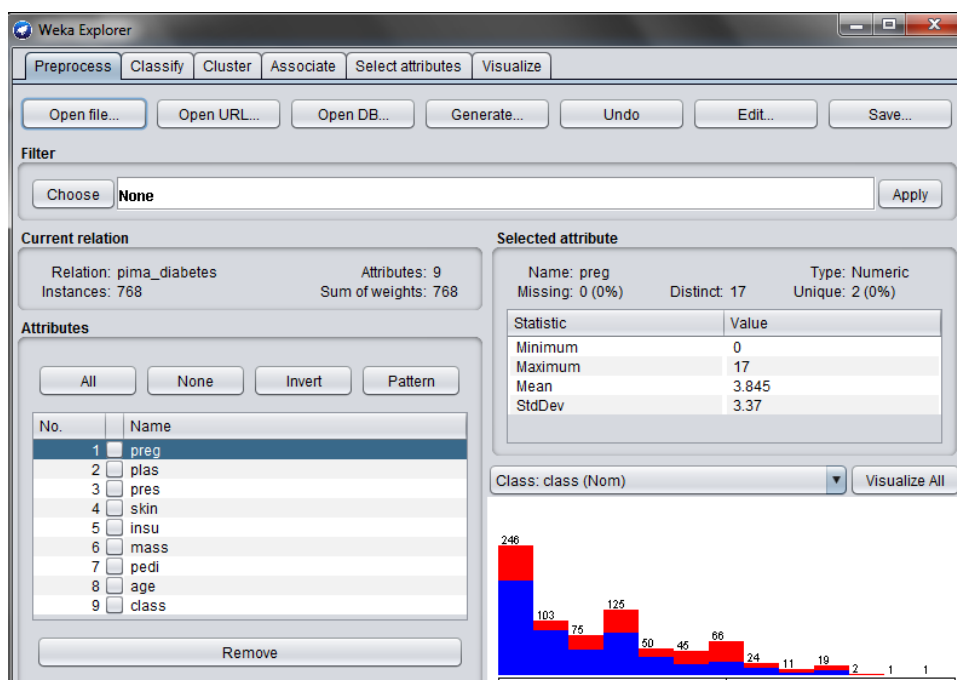
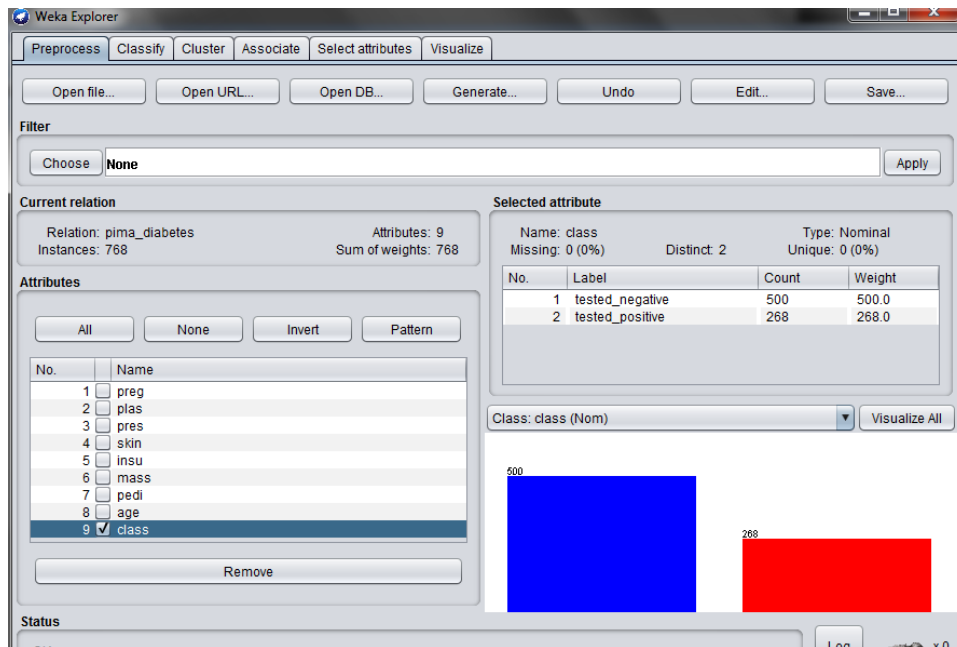
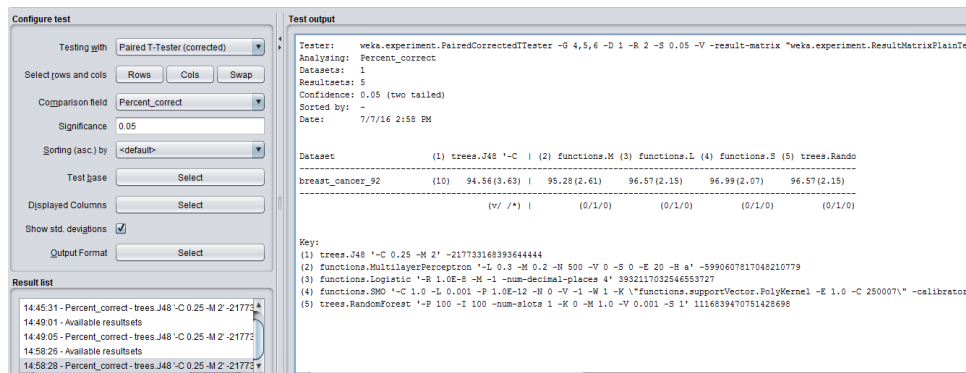


Figure 2: Dataset **breast-cancer-wisconsin.arff** in WEKA

Figure 3: Dataset **credit-g.arff** in WEKAFigure 4: Dataset **diabetes.arff** in WEKA

Figure 5: Dataset **diabetes.arff** StatisticsFigure 6: Results in Experimenter for **breast-cancer-wisconsin.arff**

Configure test

Testing with: Paired T-Tester (corrected)

Select rows and cols: Rows Cols Swap

Comparison field: Mean_absolute_error

Significance: 0.05

Sorting (asc) by: <default>

Test base: Select

Displayed Columns: Select

Show std. deviations:

Output Format: Select

Test output

Tester: weka.experiment.PairedCorrectedTester -G 4,5,6 -D 1 -R 2 -S 0.05 -V -result-matrix "weka.experiment.ResultMatrixPlainText"

Analysing: Mean_absolute_error

Datasets: 1

Resultsets: 5

Confidence: 0.05 (two tailed)

Sorted by: -

Date: 7/7/16 3:43 PM

Dataset: (1) trees.J48 -C 1 (2) functions. (3) functions. (4) functions. (5) trees.Rand

	(1)	(2)	(3)	(4)	(5)
breast_cancer_s2	(10)	0.07 (0.04)	0.05 (0.02)	0.05 (0.02)	0.03 (0.02) * 0.06 (0.02)
	(v/ **)	1	(0/1/0)	(0/1/0)	(0/0/1)

Key:

- (1) trees.J48 "-C 0.25 -M 2" -21773316839364444
- (2) functions.MultilayerPerceptron "-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a" -5990607817048210779
- (3) functions.Logistic "-R 1.0E-8 -M -1 -num-decimal-places 4" 3932117032546553727
- (4) functions.SMO "-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -E 1.0 -C 250007\" -calibrat
- (5) trees.RandomForest "-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1" 1116839470751428698

Result list

- 15:14:29 - Available resultsets
- 15:14:30 - Percent_correct -trees.J48 -C 0.25 -M 2 -21773316839364444
- 15:42:42 - Mean_absolute_error -trees.J48 -C 0.25 -M 2 -21773316839364444

Figure 7: Results in Experimenter for **breast-cancer-wisconsin.arff** - 10 repetitions

Configure test

Testing with: Paired T-Tester (corrected)

Select rows and cols: Rows Cols Swap

Comparison field: Percent_correct

Significance: 0.05

Sorting (asc) by: <default>

Test base: Select

Displayed Columns: Select

Show std. deviations:

Output Format: Select

Test output

Tester: weka.experiment.PairedCorrectedTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -S"

Analysing: Percent_correct

Datasets: 1

Resultsets: 5

Confidence: 0.05 (two tailed)

Sorted by: -

Date: 7/7/16 4:16 PM

Dataset: (1) trees.Ra | (2) trees (3) funct (4) funct (5) funct

	(1)	(2)	(3)	(4)	(5)
german_credit	(10)	76.40	70.50 *	75.10	71.50 * 75.20
	(v/ **)	1	(0/0/1)	(0/1/0)	(0/1/0)

Key:

- (1) trees.RandomForest "-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1" 1116839470751428698
- (2) trees.J48 "-C 0.25 -M 2" -21773316839364444
- (3) functions.SMO "-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -E 1.0 -C 250007\" -calibrat
- (4) functions.MultilayerPerceptron "-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a" -5990607817048210779
- (5) functions.Logistic "-R 1.0E-8 -M -1 -num-decimal-places 4" 3932117032546553727

Result list

- 16:16:12 - Available resultsets
- 16:16:14 - Percent correct -trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Figure 8: Results in Experimenter for **credit-g.arff** - 10 repetitions

batchSize	<input type="text" value="100"/>
binarySplits	<input type="button" value="False"/>
collapseTree	<input type="button" value="True"/>
confidenceFactor	<input type="text" value="0.25"/>
debug	<input type="button" value="False"/>
doNotCheckCapabilities	<input type="button" value="False"/>
doNotMakeSplitPointActualValue	<input type="button" value="False"/>
minNumObj	<input type="text" value="2"/>
numDecimalPlaces	<input type="text" value="2"/>
numFolds	<input type="text" value="3"/>
reducedErrorPruning	<input type="button" value="False"/>
saveInstanceData	<input type="button" value="False"/>
seed	<input type="text" value="1"/>
subtreeRaising	<input type="button" value="True"/>

Figure 9: Explorer/DefaultParameters/J48

Classifier

Choose **J48 - C 0.25 - M 2**

Test options

Use training set

Supplied test set

Cross-validation Folds **10**

Percentage split % **66**

(Nom) class

Result list (right-click for options)

13:02:23 - trees.J48

Classifier output

```

Correctly Classified Instances      661      94.5637 %
Incorrectly Classified Instances    38       5.4363 %
Kappa statistic                    0.8799
Mean absolute error                 0.0694
Root mean squared error             0.2229
Relative absolute error             15.352 %
Root relative squared error         46.8927 %
Total Number of Instances          699

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
      0.956   0.075   0.961     0.956   0.958     0.880  0.955    0.955    2
      0.925   0.044   0.918     0.925   0.921     0.880  0.955    0.902    4
Weighted Avg.   0.946   0.064   0.946     0.946   0.946     0.880  0.955    0.937

=== Confusion Matrix ===

  a  b  <-- classified as
438 20 | a = 2
 18 223 | b = 4

```

Figure 10: Results in Explorer for **breast-cancer-wisconsin.arff** J48 - 10-fold

batchSize	<input type="text" value="100"/>
buildCalibrationModels	<input type="button" value="False"/>
c	<input type="text" value="1.0"/>
calibrator	<input type="button" value="Choose"/> Logistic -R 1.0E-8 -M -1 -num-decimal-places 4
checksTurnedOff	<input type="button" value="False"/>
debug	<input type="button" value="False"/>
doNotCheckCapabilities	<input type="button" value="False"/>
epsilon	<input type="text" value="1.0E-12"/>
filterType	<input type="button" value="Normalize training data"/>
kernel	<input type="button" value="Choose"/> PolyKernel -E 1.0 -C 250007
numDecimalPlaces	<input type="text" value="2"/>
numFolds	<input type="text" value="-1"/>
randomSeed	<input type="text" value="1"/>
toleranceParameter	<input type="text" value="0.001"/>

Figure 11: Explorer/DefaultParameters/SMO

Classifier

Choose **SMO - C 1.0 - L 0.001 - P 1.0E-12 - N 0 - V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel - E 1.0 - C 250007" - calibrator "weka.classifiers.functions.**

Test options

Use training set

Supplied test set

Cross-validation Folds **10**

Percentage split % **66**

(Nom) class

Result list (right-click for options)

13:02:23 - trees.J48

13:31:13 - functions.SMO

Classifier output

```

Correctly Classified Instances      678      96.9957 %
Incorrectly Classified Instances    21      3.0043 %
Kappa statistic                    0.9337
Mean absolute error                 0.03
Root mean squared error            0.1733
Relative absolute error             6.6471 %
Root relative squared error        36.4672 %
Total Number of Instances          699

=== Detailed Accuracy By Class ===
           TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
0.974   0.037   0.980   0.974   0.977   0.934   0.968   0.972   2
0.963   0.026   0.951   0.963   0.957   0.934   0.968   0.928   4
Weighted Avg.   0.970   0.034   0.970   0.970   0.970   0.934   0.968   0.957

=== Confusion Matrix ===
  a  b  <-- classified as
446 12 | a = 2
 9 232 | b = 4

```

Figure 12: Results in Explorer for **breast-cancer-wisconsin.arff** SMO - 10-fold

bagSizePercent	<input type="text" value="100"/>
batchSize	<input type="text" value="100"/>
breakTiesRandomly	<input type="button" value="False"/>
calcOutOfBag	<input type="button" value="False"/>
debug	<input type="button" value="False"/>
doNotCheckCapabilities	<input type="button" value="False"/>
maxDepth	<input type="text" value="0"/>
numDecimalPlaces	<input type="text" value="2"/>
numExecutionSlots	<input type="text" value="1"/>
numFeatures	<input type="text" value="0"/>
numIterations	<input type="text" value="100"/>
outputOutOfBagComplexityStatistics	<input type="button" value="False"/>
printClassifiers	<input type="button" value="False"/>
seed	<input type="text" value="1"/>
storeOutOfBagPredictions	<input type="button" value="False"/>

Figure 13: Explorer/DefaultParameters/RandomForest

Classifier

Choose **RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1**

Test options

Use training set

Supplied test set

Cross-validation Folds

Percentage split %

(Nom) class

Result list (right-click for options)

13:02:23 - trees.J48

13:31:13 - functions.SMO

13:34:43 - trees.RandomForest

Classifier output

```

Correctly Classified Instances      675          96.5665 %
Incorrectly Classified Instances    24           3.4335 %
Kappa statistic                    0.9243
Mean absolute error                0.0612
Root mean squared error            0.1663
Relative absolute error            13.5381 %
Root relative squared error        34.9858 %
Total Number of Instances          699

```

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.969	0.041	0.978	0.969	0.974	0.924	0.989	0.994	2
	0.959	0.031	0.943	0.959	0.951	0.924	0.989	0.973	4
Weighted Avg.	0.966	0.038	0.966	0.966	0.966	0.924	0.989	0.986	

=== Confusion Matrix ===

```

a b <-- classified as
444 14 | a = 2
10 231 | b = 4

```

Status

Figure 14: Results in Explorer for **breast-cancer-wisconsin.arff** Random Forest - 10-fold

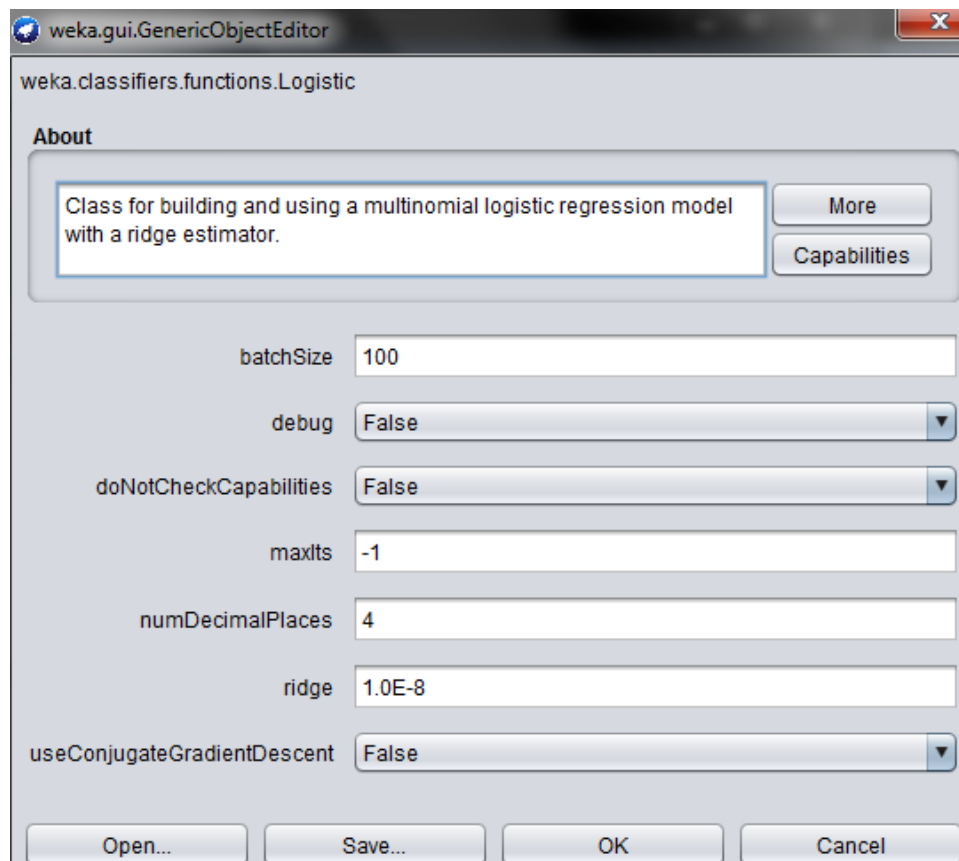


Figure 15: Explorer/DefaultParameters/Logistic

Classifier

Choose **Logistic -R 1.0E-8 -M -1 -num-decimal-places 4**

Test options

Use training set
 Supplied test set
 Cross-validation Folds
 Percentage split %

(Nom) class

Result list (right-click for options)

- 13:02:23 - trees.J48
- 13:31:13 - functions.SMO
- 13:34:43 - trees.RandomForest
- 13:37:16 - functions.Logistic

Classifier output

```

Correctly Classified Instances      675      96.5665 %
Incorrectly Classified Instances    24      3.4335 %
Kappa statistic                    0.924
Mean absolute error                0.0486
Root mean squared error            0.1668
Relative absolute error            10.7595 %
Root relative squared error        35.0841 %
Total Number of Instances          699

=== Detailed Accuracy By Class ===
              TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
0.974   0.050   0.974   0.974   0.974   0.924   0.993   0.997   2
0.950   0.026   0.950   0.950   0.950   0.924   0.993   0.984   4
Weighted Avg.   0.966   0.042   0.966   0.966   0.966   0.924   0.993   0.993

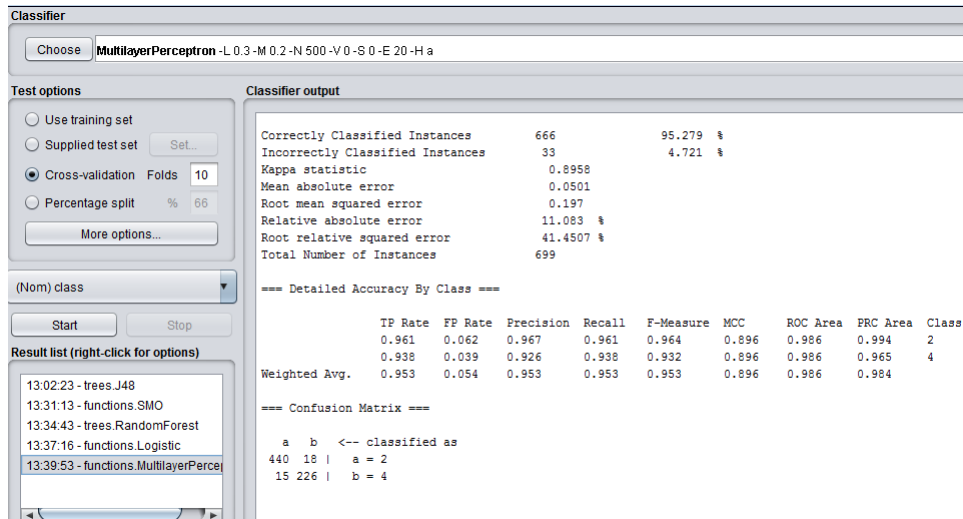
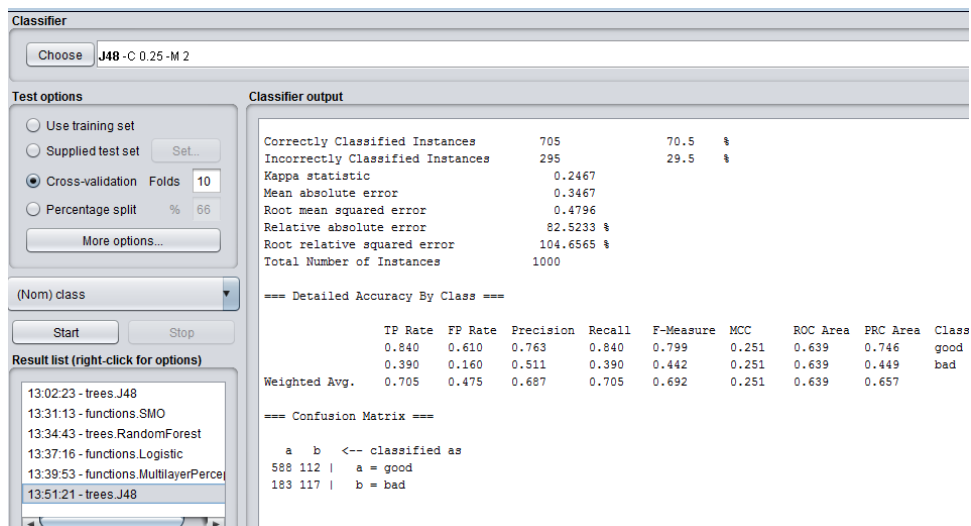
=== Confusion Matrix ===
  a  b  <-- classified as
446 12 | a = 2
 12 229 | b = 4

```

Figure 16: Results in Explorer for **breast-cancer-wisconsin.arff** Logistic - 10-fold

GUI	False
autoBuild	True
batchSize	100
debug	False
decay	False
doNotCheckCapabilities	False
hiddenLayers	a
learningRate	0.3
momentum	0.2
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	True
seed	0

Figure 17: Explorer/DefaultParameters/MLP

Figure 18: Results in Explorer for `breast-cancer-wisconsin.arff` MLP - 10-foldFigure 19: Results in Explorer for `credit-g.arff` J48 - 10-fold