# Acta Universitatis Sapientiae

# Informatica

Volume 10, Number 1, 2018

Sapientia Hungarian University of Transylvania
Scientia Publishing House

# Contents

Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 5–25

DOI: 10.2478/ausi-2018-0001

# Modular strategic SMT solving with SMT-RAT**

Gereon KREMER
RWTH Aachen University
Aachen, Germany
email:
gereon.kremer@cs.rwth-aachen.de

Erika ÁBRAHÁM
RWTH Aachen University
Aachen, Germany
email:
abraham@cs.rwth-aachen.de

**Abstract.** In this paper we present the latest developments in `SMT-RAT`, a tool for the automated check of quantifier-free real and integer arithmetic formulas for satisfiability. As a distinguishing feature, `SMT-RAT` provides a set of solving modules and supports their strategic combination. We describe our CArL library for arithmetic computations, the available modules implemented on top of CArL, and how modules can be combined to satisfiability-modulo-theories (SMT) solvers. Besides the traditional SMT approach, some new modules support also the recently proposed and highly promising model-constructing satisfiability calculus approach.

## 1 Introduction

The problem of checking the satisfiability of first-order logic formulas appears in many different areas like, e.g., program verification or synthesis approaches like planning or scheduling. On the one hand, solving arithmetic formulas has

* *This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No H2020-FETOPEN-2015-CSA 712689.*

**Computing Classification System 1998:** I.2.8
**Mathematics Subject Classification 2010:** 68-04
**Key words and phrases:** satisfiability modulo theories, polynomial arithmetic, strategic combination

deep historical roots in mathematical logic and *symbolic computation*. On the other hand, the last decades led to fruitful developments also in computer science, resulting in efficient *SAT* and *satisfiability-modulo-theories* (*SMT*) solvers. Whereas at the beginning SMT solvers focused on theories for equality logic and uninterpreted functions, bit-vectors, arrays and floating-point arithmetic, recently notable achievements were made also for arithmetic theories. For the support of polynomial constraints over the reals interesting symbiosis evolved between symbolic computation and satisfiability checking, learning from each other and mutually integrating successful techniques from the respective areas.

In this paper we describe recent developments for our SMT solver `SMT-RAT` [17, 16], which is mainly developed for checking the satisfiability of quantifier-free real and integer arithmetic formulas. A distinguishing feature of `SMT-RAT` is that it offers a library of decision procedure modules, which share a common interface structure such that they can be strategically combined based on user-defined specifications to efficient SMT solvers.

The majority of the decision procedures that are implemented in those `SMT-RAT` modules stem from the area of symbolic computation, for which implementations in *computer algebra systems* are available. Examples for such *theory solving* procedures are the simplex method, methods using Gröbner bases, the subtropical satisfiability method, the virtual substitution method or the cylindrical algebraic decomposition method. Thus it is natural to think of invoking their implementations provided by different computer algebra systems. However, there are several obstacles that hinder their direct embedding in SMT solvers. Firstly, SMT solvers for arithmetic problems need modules to check the consistency of sets of polynomial constraints in an *incremental* fashion, meaning that after the consistency of a constraint set has been determined, the consistency of an extended set needs to be checked. Such incremental consistency checks need to be executed frequently, therefore it is important that they are not done independently but re-use information from previous checks as much as possible. Secondly, in case of inconsistency, these modules need to return an *explanation* for unsatisfiability, e.g. in the form of an inconsistent subset of the constraints. Thirdly, SMT solvers explore the space of possible solutions in an enumerative manner, accompanied by smart propagation, resolution and learning procedures to avoid unnecessary work in unsatisfiable parts of the search space. Once it is detected that the currently considered part of the search space does not contain satisfying solutions, *backtracking* takes place, which requires also backtracking ability for the theory solver. Unfortunately, implementations in computer algebra systems do not provide these functionalities. Therefore, we implemented adaptations for several such methods as `SMT-RAT` modules that satisfy the above requirements.

These implementations in `SMT-RAT` required support for basic arithmetic computations like algorithms for polynomial division or calculating the greatest common divisor of two (multivariate) polynomials. Today's commonly used computer algebra systems like `Maple`, `Mathematica`, `Singular`, `GAP` or `Reduce` offer a rich set of highly efficient algorithms on polynomials [28], while also putting the focus on the user interface and graphical capabilities. Therefore, we experimented with their usage for our `SMT-RAT` modules. However, we experienced a major communication bottleneck at the interface between `SMT-RAT` and external tools: the frequent exchange of large constraint sets is very time consuming. Additionally, as the two communication sides use different data types to represent numbers, arithmetic expressions and constraints, the frequent communication caused a serious overhead also for datatype conversion.

An optimal solution can be offered by a (preferably free and open-source) *library* for arithmetic computations, which offers an object-oriented, generic and modular data structure for polynomials or, even better, formulas over polynomial constraints. Though a few libraries exist that try to bridge this gap, for example `GiNaC` [6] and `CoCoALib` [1], they all have various downsides that led us to implement our own `C++` library `CArL`. It is clearly not one of our short- or medium-term goals to compete with the performance of these algorithms on arbitrary inputs, when implementing a polynomial arithmetic library from scratch. However, we hope that in the long-term our tool will provide helpful support to other research groups in the SMT community and beyond.

Previously we reported on our software developments in [17] in 2012 and in [16] in 2015. The novel contributions of this paper are (i) the introduction of the `CArL` library and (ii) the description of `SMT-RAT` with a special focus on three new solver modules in the latest release.

The rest of this paper is structured as follows. In Section 2 we give a short introduction to SAT and SMT solving and briefly explain the main ideas of some relevant algebraic decision procedures. In Section 3 we introduce our arithmetic library `CArL`, followed by a description of our SMT solver `SMT-RAT` and its new modules in Section 4. Finally we provide some experimental results in Section 5 before we conclude the paper in Section 6.

## 2 Preliminaries

Traditionally, *satisfiability checking* aims at the automated check of the satisfiability of quantifier-free first-order logic formulas over some theories, whereas recent developments extend the functionalities for satisfiability checking to, e.g., quantified formulas or optimization. In this paper we focus on checking the satisfiability of quantifier-free *real arithmetic* formulas, which are Boolean combinations of constraints comparing polynomials over real-valued variables to zero.

**SAT solving** The success story of satisfiability checking started with *SAT solving* for propositional logic. The main strength of SAT solving is a highly efficient heuristic combination of enumeration, propagation, resolution and learning [19, 43]. The input is a propositional logic formula, which is a Boolean combination (using operators for negation $\neg$, conjunction $\wedge$, disjunction $\vee$ etc.) of Boolean variables called *propositions*. The input formula is first transformed into *conjunctive normal form* (*CNF*) in linear time and space on the cost of additional variables using Tseitin's transformation [51]. The result is a conjunction of disjunctions of possibly negated propositions; propositions and negated propositions are called *literals* and their disjunctions *clauses*. Enumeration is used to explore possible solutions, *deciding* which values for which propositions should be tried first. After each such decision, which is actually a guess for satisfying variable values, *propagation* is applied to detect certain implications of the current assignments and thus to reduce the number of "wrong guesses". For example, if the CNF contains a clause $(a \vee b)$ and the value *false* is decided for $a$ then propagation assigns *true* to $b$ in order to satisfy the clause. However, propagation cannot always avoid running into an unsatisfiable assignment. For example, the previous assignments would lead to the violation of the clause $(a \vee \neg b)$. When such a *conflict* is detected, *resolution* is applied to determine a reason for the conflict; in our example, resolving the two clauses $(a \vee b)$ and $(a \vee \neg b)$ would result in the resolvent $(a)$. *Learning* the reason for the conflict will protect the future search from running into conflicts with the same reason.

**SMT solving** The impressive success of SAT solving led to the idea to extend the technology to check satisfiability also for quantifier-free formulas over different theories. *Satisfiability modulo theories* (*SMT*) *solving* started for equalities and uninterpreted functions, theories relevant for program verification (arrays, bit-vectors, floating-point arithmetic etc.) and linear real and integer arithmetic. Nowadays, powerful tools exist that can handle also harder theories like, e.g., nonlinear arithmetic formulas. Some popular SMT solvers for arithmetic theories are, e.g., `AProVE` [29], `CVC4` [4], `MathSAT5` [12], `raSAT` [39], `veriT` [7], `Yices2` [22], `Z3` [20] and our solver `SMT-RAT` [16].

SMT solving typically works in a *lazy* fashion, meaning that the solver prioritizes to satisfy the Boolean structure of the formula first and check consistency in the theory afterwards. To do so, the *Boolean skeleton* or *Boolean abstraction* of the input formula is generated by replacing each theory constraint by a fresh proposition, resulting in a propositional logic formula, which can be checked for satisfiability by a SAT solver. If the skeleton is unsatisfiable then the input formula is unsatisfiable, too. Otherwise, if the SAT solver has determined a Boolean solution for the skeleton then suitable *theory solvers* are invoked to check whether all constraints with true abstraction propositions and the negations of all constraints with false abstraction propositions are together consistent. If this is the case then a satisfying solution for the input formula is found. Otherwise, the theory solvers need to provide an *explanation* for the inconsistency, typically by returning an inconsistent subset of the considered constraints. Learning the Boolean abstraction of this explanation refines the Boolean abstraction, avoiding Boolean solutions with the same theory conflict in future search. Besides such *full lazy* approaches, *less lazy* variants check theory consistency more frequently (usually after the full propagation of each Boolean decision).

**MCSAT** A recent technique called *model constructing satisfiability calculus* (*MCSAT*) [21] generalizes the above approach by defining a set of derivation rules, which includes besides Boolean decision, propagation and conflict resolution also their counterparts for the theory. Especially, MCSAT provides the possibility to guess not only truth values for the theory constraints but also values for the theory variables, and use some theory propagation techniques to drive the search for further theory variable values away from unsatisfiable parts of the state space. For example, if we decide to try the value 0 for a theory variable $x$ and a constraint $x > y$ should hold according to the Boolean search then we need to guess a negative value for $y$, e.g. $-1$. However, if also the constraint $x^2 > y^2$ should hold then we have run into a theory conflict, because for $x = 0$ there is no value for $y$ that would satisfy both constraints. In such cases, the theory conflict needs to be *explained* by a lemma, which in the optimal case generalizes the current conflict and helps to exclude from further search not only the current assignment but also others with similar reasons for being unsatisfying. For our example, the solver could explain the conflict by returning $x^2 > y^2 \rightarrow x > 0$.

**Theory decision procedures** As mentioned above, SMT solvers use, besides SAT solvers, also theory solvers, which need to check *sets* or *conjunctions* of theory constraints for consistency. Furthermore, for efficiency reasons, in the less lazy setting theory solvers should work *incrementally*, meaning that if a set of constraints is found satisfiable and the set is extended by adding some further constraints then the theory solver should not check the consistency of the extended set from anew but re-use previous results as much as possible. Additionally, for inconsistent constraint sets the theory solvers must be able to provide *explanations*.

For this purpose, we use decision procedures for arithmetic theories developed in symbolic computation and implemented in computer algebra systems. The symbiosis of these methods with SAT solving is fruitful because these methods are good at checking sets of constraints for satisfiability but they are not designed for combinatorial checks in Boolean structures. In the following we describe some of these procedures in a nutshell.

- *Interval constraint propagation* (*ICP*) [27, 34] uses interval arithmetic to contract given variable domains under the assumption of certain constraints. For example, if $x \in [0, 2]$ and $y \in [1, 3]$ and $x = y$ should hold then ICP can imply that $x \in [1, 2]$ and $y \in [1, 2]$. ICP is very powerful and can be applied to nonlinear arithmetic involving also trigonometric and transcendental functions, but it is incomplete in general.

- The *simplex method* [18] is applicable to linear real arithmetic. Originally it was developed for optimization but in the SMT context we use it for satisfiability checking only. The main idea is to start from an initial variable assignment satisfying a set of equalities and modify this assignment step-wise to satisfy also additional variable bounds.

- The *Fourier-Motzkin variable elimination method* allows to perform quantifier elimination on sets of linear real arithmetic constraints. The idea is that if two constraints define a lower and an upper bound respectively on a variable than satisfiability requires the lower bound to be smaller (or equal, depending on the comparison operator) than the upper bound. For example, $2y < x$ and $x < w$ requires $2y < w$. Collecting these requirements for all lower-upper-bound pairs on $x$ allows to eliminate $x$ from the constraint system.

- The original idea of the *virtual substitution method* [55] is to use solution equations to solve multivariate low-degree constraints symbolically and substitute these solutions into the other constraints to eliminate variables. As these symbolic solutions might contain e.g. square roots, special virtual substitution rules are applied which produce standard arithmetic constraints after the substitution. The method requires the degree of the polynomials to be bounded and is thus incomplete in general.

- The *cylindrical algebraic decomposition* (*CAD*) *method* [13] is a decision procedure for real arithmetic. It is a quantifier elimination method, which decomposes the state space into a finite number of sign-invariant (or truth-invariant) regions, such that in each region either all points satisfy the input formula or none of them does so. Therefore, it is sufficient to take a single sample point from each region and check whether any of the sample points satisfies the formula. The CAD method is complete but in worst case it comes with doubly exponential solving effort.

- A recent incomplete but highly efficient method for finding solutions for real-arithmetic constraint sets is the *subtropical satisfiability method* [25]. It analyzes the exponent vectors of the monomials in the constraints and tries to find dominating monomials whose values can be made larger or smaller than all other monomial values. For example, $x^3y+x^2y^2+y<0$ is satisfiable because for any positive value for $y$ we can find a sufficiently small value for $x$ such that $x^3y$ becomes the dominating monomial that makes the polynomial $x^3y+x^2y^2+y$ negative.

- The incomplete *branch-and-bound method* [41] can be used to extend decision procedures for real arithmetic to check the satisfiability of integer arithmetic constraint sets. It first checks the real relaxation of the input constraints using some decision procedure for the reals, i.e., assuming the variables to be real-valued instead of integer-valued. If no solution exists in the real domain then there is no integer solution. If an integer solution is found by the real-valued search then the formula is satisfiable. Otherwise, if a real-valued solution $v$ is found for an integer variable $x$ then the search branches on values for $x$ that are either less or equal than the largest integer below $v$ or at least as large as than the smallest integer above $v$.

However, practical implementations of these procedures are usually not designed to work incrementally and they neither support the generation of explanations. Therefore, before their embedding in SMT solving they need to be adapted to satisfy these requirements.

## 3 CArL

For any project that aims to work on arithmetic formulas, some data structures are needed to represent numbers, polynomials and formulas. Data types for the exact representation of real numbers of arbitrary size in C++ are provided e.g. by the libraries gmp and cln. Similar support is available for other languages like Java or Python. However, general-purpose libraries for the representation of real-algebraic numbers, polynomials, polynomial constraints and algebraic formulas, and efficient implementations of polynomial computations – ranging from addition and multiplication to pretty complex operations like greatest common divisor or factorization – are much more rare.

There are manifold reasons for this support gap. First of all, the range of algorithms that work on polynomials is extremely large and diverse, such that it is futile to attempt to exhaustively implement all algorithms. Furthermore, different representations tend to provide vastly different performance on different inputs and thus the application domain must be taken into consideration.

One attempt to provide a fairly generic C++ library for polynomial arithmetic is GiNaC [6]. In contrast to computer algebra systems, GiNaC is designed as an open framework to be integrated in other tools, providing symbolic manipulations like arithmetic operations on polynomials. Its popularity reveals the urgent need for such a library. It is used for example for symbolic execution [3], probabilistic pointer analysis [11], and in the parametric probabilistic model checker PARAM [32].

However, GiNaC is not generic in the sense that it does not allow arbitrary coefficient types for the polynomials, and provides no possibility to influence the ordering of the variables and monomials in the polynomials. Both are crucial for the efficient implementation of many algorithms, for example decision procedures based on Gröbner bases [10] or the cylindrical algebraic decomposition method [13]. Furthermore, GiNaC lacks thread safety, thus it cannot be used safely in parallelized applications.

Another C++ library is CoCoALib [1] which also provides many arithmetic operations on polynomials, though it is mostly tailored to the computation of Gröbner bases. Unfortunately, all CoCoALib polynomials are elements of some polynomial ring with a fixed variable ordering and polynomials of different rings are not directly compatible. This is a major obstacle whenever fresh variables are introduced or a certain operation is only performed on a small subset of the variables or on a different variable ordering. Furthermore CoCoALib does not offer all operations that are needed for methods like the cylindrical algebraic decomposition.

After having experimented with available libraries, we decided to develop a free and open-source *Computer Arithmetic Library* `CArL`[1] from scratch in `C++` to overcome these problems. The focus of `CArL` lies on efficient generic data types and algorithms for *polynomials and arithmetic formulas*, but also includes *bit vectors* and *uninterpreted variables and functions*.

Most of the data structures in `CArL` can be instantiated with different *number types*; it ships support for `gmp`, `cln` and native integers, as well as wrappers for `MPFR` and `Z3` rationals. For algebraic methods like the cylindrical algebraic decomposition method, `CArL` implements *real-algebraic numbers* – either in interval representation or using an encoding based on Thom's lemma. Furthermore, `CArL` implements an extension of the templated ***`boost`*** *intervals*, which allows open and closed bounds, and implements methods which are essential for interval constraint propagation techniques.

The library offers a variety of methods for computations with polynomials. Some of them provide basic functionalities like for example to get the list of variables of a polynomial, to check whether a polynomial is univariate, to iterate over the terms of a polynomial, to apply addition, subtraction, multiplication, substitution, comparison and evaluation, to normalize polynomials or to compute their derivatives. For univariate polynomials `CArL` can compute Cauchy and other bounds on the real zeros, Sturm sequences and their sign variations, resultants and sub-resultants, discriminants and real root isolations. Further methods for multivariate polynomials implement for example test for definiteness, sum of squares decomposition, polynomial (pseudo-)division and (pseudo-)remainder computation, factorization, computations of co-prime factors of coefficients, or S-polynomials. `CArL` even features its own implementation of Gröbner bases with a particular focus on the support of incrementality. Additionally to polynomial expressions, for methods like the virtual substitution `CArL` also offers data types for fractions and square root expressions.

To easily borrow further advanced functionality from other libraries or compare against their implementation, `CArL` integrates `CoCoALib` and `GiNaC`, offering alternative implementations for e.g. polynomial factorization, multivariate polynomial greatest common divisor and Gröbner bases computations. A preliminary integration of `Maple` – given an existing `Maple` installation – is also available.

Moreover, `CArL` bundles a lot of utility functionality that we deem useful when implementing any kind of tool similar to an SMT solver.

[1]Available at `https://github.com/smtrat/carl`.

# 4 SMT-RAT

The efficient solving of some variant of the satisfiability problem is a cornerstone for many techniques in formal verification and numerous industrial applications. We focus on the satisfiability modulo theories (SMT) problem that combines Boolean satisfiability with one or more theories, for example nonlinear arithmetic. A number of open-source solvers exist that tackle this class of problems with great success, for example `CVC4` [4], `raSAT` [52], `veriT` [7], `Yices2` [22] or `Z3` [20].

Our goal is to combine various techniques for SMT solving and study the interaction of different approaches. This requires a framework that allows a user to compose an SMT solver from individual *modules* easily in a very flexible way. Though the aforementioned solvers all have different solution techniques and more or less powerful mechanisms to combine them, we want this strategic combination of modules to be very transparent to the user. This idea is at the core of our SMT solving library `SMT-RAT`[2].

`SMT-RAT` is a library meant for, but not limited to, SMT solving. Though it can be used *as-is* to compose a stand-alone SMT solver – and this is how we use it most of the time – it is intended to be employed in either other SMT solvers or even be used for other solving tasks. The focus on modularity and composability yields a framework where the individual solver modules have a well-defined interface and are completely decoupled otherwise. This allows for an easy extension of `SMT-RAT` by new solving techniques without knowledge about the overall architecture and other modules. Undergraduate students routinely implement new theory solver modules or extend existing ones for various logics in practical courses or as thesis projects [14, 37, 47, 50, 40, 49, 42, 45, 46, 53, 56, 30, 33, 44, 31].

## 4.1 Previous SMT-RAT solver modules

`SMT-RAT` holds a rich collection of solver modules that can be grouped into *preprocessing modules* applying simplifications, *decision procedure modules* that implement satisfiability checking procedures and *meta modules*, though the border is sometimes blurry.

**Preprocessing modules** Oftentimes input problems have a certain structure that allows for some simplifications. `SMT-RAT` contains a collection of preprocessing techniques to exploit such possibilities. The `ESModule` searches for

[2]Available at `https://github.com/smtrat/smtrat`.

variables occurring linearly in equations that must hold and eliminate them. The `GBPPModule` is based on Gröbner bases and uses nonlinear equalities to simplify inequalities. Certain patterns of circular inequalities can be exploited with the `ICEModule` to eliminate variables. Some encodings of a multiple choice scenario can be simplified to Boolean decisions using the `MCBModule`. The `PFEModule` uses bounds on arithmetic variables to identify and remove sign-invariant factors of polynomials. Finally, the `SymmetryModule` identifies and breaks symmetries on both Boolean and theory variables.

**Decision procedure modules** At the core of most modern SMT solvers, and `SMT-RAT` is not different here, is a SAT solver. `SMT-RAT` uses `MiniSat` [24] as its `SATModule`, which employs CDCL(T)-style SAT solving and forwards theory calls to its backends. As theory solvers, `SMT-RAT` offers implementations of several complete and incomplete decision procedures, each of them encapsulated in a module.

The theory of *bit-vectors* is handled by the `BVModule` [42] that encodes bit-vector constraints to propositional logic similar to [26]. Another variant of theory constraints supported by `SMT-RAT` are *pseudo-Boolean* constraints – arithmetic constraints over Boolean variables. These are transformed by the `PBPPModule` [30] to either propositional or arithmetic formulas.

*Linear arithmetic* can be solved using the `FouMoModule`, which implements a variant of Fourier-Motzkin variable elimination, or the `LRAModule` which implements the simplex method in the spirit of [23], including support for linear integer arithmetic. Additionally the `CubeLIAModule` provides an incomplete but fast test for linear integer arithmetic inspired by [8].

The focus of our research lies however on *nonlinear arithmetic*. The cylindrical algebraic decomposition method is implemented in the `CADModule` [16, 41, 54], the only complete `SMT-RAT` module for nonlinear real arithmetic, which is complemented by different incomplete solver modules. The `GBModule` [38] uses a variant Gröbner bases to determine satisfiability over the reals. The virtual substitution method with various optimizations is implemented in the `VSModule` [15]. Another very popular method is interval constraint propagation that is available through the `ICPModule` [49]. All these modules can also work on nonlinear integer arithmetic problems using the branch-and-bound technique, though all of them are incomplete due to the undecidability of nonlinear integer arithmetic. Additionally the `IntBlastModule` [42, 41] encodes bounded nonlinear integer problems in bit-vector arithmetic, similar to what most other solvers do for this kind of problems.

**Meta modules** These solver modules do not implement any solving technique by themselves, but connect other modules. Meta modules extend what the user can do with the strategy framework: they allow to keep the strategy formalism comparably simple, as more complicated and technical components can be implemented as meta modules that encapsulate a specific strategic feature.

One such example is the `FPPModule` which applies a given strategy of preprocessing techniques to simplify a formula multiple times until a fixed-point is reached and no further simplifications can be done.

## 4.2 New modules

There are three new decision procedure modules in the latest `SMT-RAT` release.

**Subtropical satisfiability** The `STropModule` implements a quick check for satisfiability following the idea of [25] as briefly explained in Section 2. This method is very fast but incomplete: either it finds a satisfying solution or returns `unknown`, but it is not able to determine unsatisfiability.

**Cylindrical algebraic decomposition** The new version of `SMT-RAT` offers a complete re-implementation of the CAD method in the solver module `NewCADModule`. The advantage of this new CAD module lies in its data stuctures. The CAD method consists of two phases (projection and construction), both of them spanning a tree-structured search. Compared to the original `CADModule` the data structures for the projection and construction phases are more modular and allow for more flexibility. For example, the sample point construction can be performed in any heuristic order and more advanced optimizations e.g. for equational constraints could be integrated.

**MCSAT-style SMT Solving** As briefly mentioned in Section 2, an approach called *model-constructing satisfiability calculus* (*MCSAT*) [21] was proposed recently, firstly instantiated for nonlinear arithmetic which is called NLSAT [36]. Given its great success on nonlinear arithmetic problems, we developed a new `SMT-RAT` module to support MCSAT-style SMT solving based on the cylindrical algebraic decomposition.

As of now, `SMT-RAT` features the core solving engine for MCSAT-style solving and explanation functions based on CAD in the spirit of NLSAT and a first version using Fourier-Motzkin variable elimination for linear problems. Current work includes a more powerful implementation based on Fourier-Motzkin that also handles certain nonlinear cases, explanations based on virtual substitution for conflicts of bounded degree as described in [2] and an implementation of the OneCell [9] approach. As for CDCL(T)-style SMT solving, we work on combining the different explanation functions in a meaningful way.

The current version of `SMT-RAT` can be compiled to an MCSAT-style SMT solver using an NLSAT-style explanation and it seems to work reliably on the SMT-LIB benchmark set. It is rather premature still and should be considered work-in-progress. Because MCSAT-style reasoning requires a close interaction between the SAT solver and the theory module, our MCSAT module is currently integrated in the SAT module and can be activated through a dedicated module configuration. In a later release we will improve the modularity for MCSAT support.

## 4.3 Strategic combination of solver modules

The fundamental idea of `SMT-RAT` is to use a *strategic combination* of solver modules for SMT solving. A *module* encapsulates a single solving technique and multiple modules can be composed to form a *strategy*. The *manager* takes care of parsing the input formula and executing the strategy on it, possibly exploiting opportunities to execute multiple branches of the strategy in parallel.

Every module works on a set of *received formulas* $C_{rcv}$ that are the input to the solving technique. The module can be asked to check the consistency of the (conjunction of the) received formulas, to which the module can return `sat`, `unsat` or `unknown`. In case of satisfiability, the module may be asked to construct a *satisfying assignment* while unsatisfiability must be proven with an *infeasible subset* of $C_{rcv}$. Common extensions like the generation of *theory lemmas* or *multiple* infeasible subsets are supported as well.

While working on some received formula, a module may ask other modules for help by adding formulas to the set of *passed formulas* $C_{pass}$ and ask its *backends* to decide upon the satisfiability of this (sub-)problem. The backends of a module are defined by the strategy and can be annotated with *conditionals* that specify when a backend should be used. These conditionals could check whether the current formula is linear, argues over bit-vectors or contains weak inequalities. When a module calls its backends, the manager collects all backends whose conditionals evaluate to true on $C_{pass}$ and executes them sequentially or in parallel, according to the strategy specification. For these backend modules, $C_{rcv}$ is identical to $C_{pass}$ of the calling module.

We illustrate a possible strategy in Figure 1 that can solve formulas over bit-vectors and arithmetic theories. It starts with the meta-module `FPPModule` that uses the strategy `PPStrategy` to employ a series of preprocessing mod-

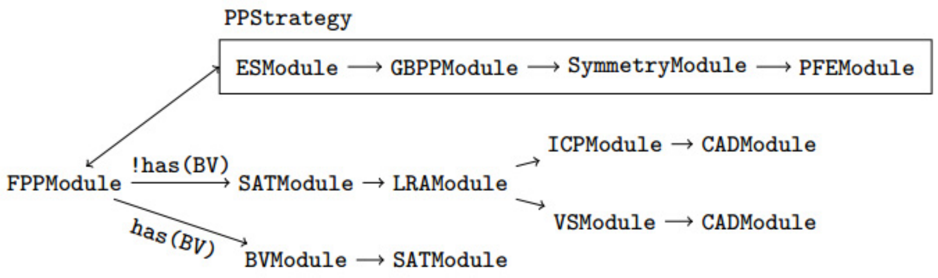


Figure 1: An example `SMT-RAT` strategy

ules: `ESModule`, `GBPPModule`, `SymmetryModule` and `PFEModule`. The preprocessed input is then forwarded either to the `BVModule` or to a `SATModule` that forwards theory calls to the `LRAModule` module. The `LRAModule` module tries to determine satisfiability and if it fails then it forwards the formula to its backends that use interval constraint propagation (`ICPModule`) or the virtual substitution (`VSModule`) method which are both incomplete and use the cylindrical algebraic decomposition method (`CADModule`) as a fallback. Note that the interval constraint propagation and the virtual substitution methods are called unconditionally and are thus both executed in parallel, the result of the first one to finish being returned to the `LRAModule` module.

All modules are thread-safe and can be used multiple times, for example the `SATModule` and the `CADModule` each have two instances that are completely independent of each other. In particular, different instances can be executed using different *configurations* that specify certain heuristics. The strict separation of procedures into modules that are sealed from each other is a great asset as it allows a high degree of flexibility and modularity and also significantly simplifies the implementation of new modules. This means, however, also that all modules should accept all possible input formulas as input. Thus if a module is called with a formula whose solution is not supported by the module than it should either call backend modules or return `unknown`.

## 5 Experimental results

The presented software, both `CArL` and `SMT-RAT`, are meant to be used within other projects in a community that emphasizes performance, albeit not as much as correctness. It is therefore important that it performs reasonably well on practical problems, in particular as competitors exist. We want to point out that we do not aim to be the fastest solver for any given logic for two reasons: firstly, we consider `SMT-RAT` a framework to allow for low-threshold research on novel SMT-related research; secondly, beating all the other solvers in a particular logic would exhaust too many resources on our side. Nevertheless, we need to perform reasonably well such that using `CArL` or `SMT-RAT` makes sense at all. We therefore present a couple of experiments for both `CArL` and `SMT-RAT` to give a feeling for the level of performance that can be expected.

## 5.1 Computations with polynomials

First we compare the implementation of multivariate polynomials from `CArL`, `CoCoALib` and `GiNaC`. We start with basic operations to compare, multiply and divide multivariate polynomials. Furthermore we compute the pseudo-remainder and resultant of multivariate polynomials and substitute individual variables by polynomials.

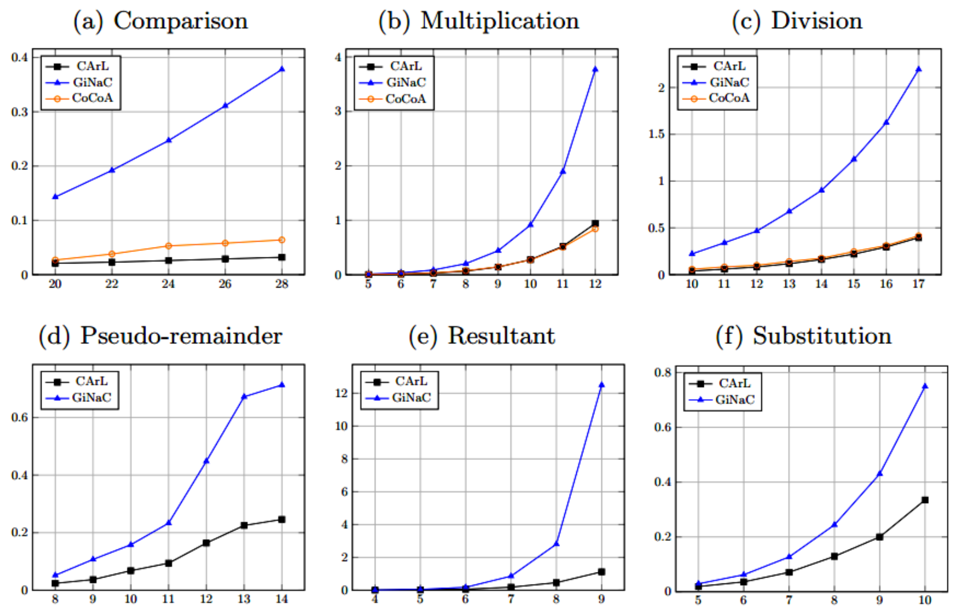


Figure 2: Experimental results for polynomial computations

Figure 2 shows some experimental results for these operations. We constructed a reasonably large set of random inputs (100 for multiplication, pseudo-remainder and resultant, 1000 for comparison, division and substitution) for every operation of the degree depicted on the $x$ axis and give the cumulative computation time in seconds. Note that all three libraries run on the exact same inputs and the conversions from one representation to another are not included in the results. We have been unable to find an implementation for the pseudo-remainder, resultant and substitution in `CoCoALib` and therefore only compare with `GiNaC` in these cases. We can see that `CArL` significantly outperforms `GiNaC` on all operations shown here and is comparable to `CoCoALib`.

Note that some more challenging algorithms like multivariate greatest common divisor of multivariate factorization are not implemented in `CArL` directly, but instead `CArL` provides a seamless integration of either `GiNaC` or `CoCoALib`. Comparisons of these methods are therefore not meaningful.

## 5.2 SMT solving

Based on the finding that at least our fundamental polynomial procedures are reasonably fast, we want to investigate whether `SMT-RAT` is competitive with other state-of-the-art solvers. Past publications have shown that `SMT-RAT` usually performs pretty good, in particular on nonlinear real arithmetic [52, 16] and nonlinear integer arithmetic [41, 35].

Three methods we have recently worked on are the `CADModule`, `STropModule` and our version of MCSAT-style solving. All three of them are targeted towards nonlinear real arithmetic and we show an overview about the current status in Figure 3 on the SMT-LIB [5] `QF_NRA` benchmark set, which contains 11354 problem instances from 10 different applications. The table shows the number of instances that could be solved (as `sat` or `unsat`) and those that could not be solved due to time or memory limits (as `resout`), in our case 60 seconds and 4 GB. The `STropModule` can not be applied to certain problems which is shown as `unknown`.

For the first solver the `STropModule` is used as the sole theory solver, while it can use other theory modules as backends in the second configuration, namely the `ICPModule`, `VSModule` and `CADModule`. As for the `CADModule`, we analyzed the impact of exploiting incrementality and different heuristics. `CADModule`$_{naive}$ uses no incrementality while `CADModule`$_A$ and `CADModule`$_B$ only differ in the order used for projecting polynomials. Using the exact same basic data structures as the CAD from `CADModule`, the `MCSATModule` imple-

| | `sat` | `unsat` | `solved` | `unknown` | `resout` |
|---|---|---|---|---|---|
| `STropModule` | 1372 | 2605 | 3977 | 5620 | 1757 |
| `STropModule` + Backends | 4260 | 4289 | 8549 | – | 2805 |
| `CADModule` $_{naive}$ | 2872 | 2699 | 5571 | – | 5783 |
| `CADModule` $_{A}$ | 4263 | 3873 | 8136 | – | 3218 |
| `CADModule` $_{B}$ | 4271 | 3803 | 8074 | – | 3280 |
| `MCSATModule` | 4297 | 4455 | 8752 | – | 2602 |

Figure 3: Experimental SMT solving results for different strategies on `QF_NRA`

ments a variant of the NLSAT approach. Note that these are preliminary results and none of the solvers uses our preprocessing techniques yet because we currently focus on these methods on their own. As a rough comparison, the leading solvers solved almost 9950 of the `QF_NRA` benchmarks at last year's SMT competition [48], though in 20 minutes instead of 60 seconds.

## 6 Conclusion

The implementation of formal approaches to handle arithmetic problems is highly challenging and extremely time consuming. In this paper we presented our `CArL` library for arithmetic computations, whose development required a serious effort. We also presented the latest version of `SMT-RAT`, whose development started in 2009. It required six years of work till we were able to participate in the SMT competition in 2015 the first time. Since then, our solver was enriched by further important modules like MCSAT-support based on the CAD method and a module for the subtropical satisfiability. All this work resulted in free and open-source software libraries that can be used not only in `SMT-RAT` but also in other software projects. Further optimizations like reduced projection in the CAD for equality constraints and further modules like MCSAT support based on the virtual substitution method are currently being implemented and will hopefully further strengthen applicability and efficiency.

Our hope is that other research groups can make use of `SMT-RAT` for their own research in their own tools. We have seen time and time again that SMT solvers are used as black-boxes and thus researchers cannot understand or modify the inner workings of the solver in question. We want to provide the opportunity to change that and make the customization and extension of a reasonably good solver for a specific class of problems accessible to non-experts that are so far forced to use a monolithic black-box solver.

## References

[1] J. Abbott, A. M. Bigatti, CoCoALib: A C++ library for computations in commutative algebra ... and beyond. *Proc. of ICMS'10* (2010), vol. 6327 of *LNCS*, Springer, pp. 73–76. ⇒7, 12

[2] E. Ábrahám, J. Nalbach, G. Kremer, Embedding the virtual substitution method in the model constructing satisfiability calculus framework. *Proc. of SC-square'17* (2017), vol. 1974 of *CEUR Workshop Proceedings*, CEUR-WS.org. ⇒16

[3] A. Albarghouthi, A. Gurfinkel, O. Wei, M. Chechik, Abstract analysis of symbolic executions. *Proc. of CAV'10* (2010), vol. 6174 of *LNCS*, Springer, pp. 495–510. ⇒12

[4] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, C. Tinelli, CVC4. *Proc. of CAV'11* (2011), vol. 6806 of *LNCS*, Springer, pp. 171–177. ⇒9, 14

[5] C. Barrett, P. Fontaine, C. Tinelli, The Satisfiability Modulo Theories Library (SMT-LIB). `http://www.SMT-LIB.org`, 2016. ⇒20

[6] C. Bauer, A. Frink, R. Kreckel, Introduction to the GiNaC framework for symbolic computation within the C++ programming language, *Journal of Symbolic Computation* **33,** 1 (2002) 1–12. ⇒7, 12

[7] T. Bouton, D. C. B. de Oliveira, D. Déharbe, P. Fontaine, veriT: An open, trustable and efficient SMT-solver, *Proc. of CADE-22* (2009), vol. 5663 of *LNCS*, Springer, pp. 151–156. ⇒9, 14

[8] M. Bromberger, C. Weidenbach, Fast cube tests for LIA constraint solving. *Proc. of IJCAR'16* (2016), Springer, pp. 116–132. ⇒15

[9] C. W. Brown, M. Košta, Constructing a single cell in cylindrical algebraic decomposition, *Journal of Symbolic Computation* **70** (2015) 14–48. ⇒17

[10] B. Buchberger, Gröbner bases: Applications. in: *The Concise Handbook of Algebra.* Kluwer Academic Publishers, 2002, pp. 265–268. ⇒12

[11] P.-S. Chen, Y.-S. Hwang, R. D.-C. Ju, J.-K. Lee, Interprocedural probabilistic pointer analysis, *IEEE Trans. Parallel Distrib. Syst.* **15,** 10 (2004) 893–907. ⇒ 12

[12] A. Cimatti, A. Griggio, B. Schaafsma, B., R. Sebastiani, The MathSAT5 SMT solver. *Proc. of TACAS'13*, vol. 7795 of *LNCS*. Springer, 2013, pp. 93–107. ⇒9

[13] G. E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Automata Theory and Formal Languages* (1975), vol. 33 of *LNCS*, Springer, pp. 134–183. ⇒11, 12

[14] F. Corzilius, *Virtual substitution in SMT solving*, Diploma thesis, RWTH Aachen University, 2011. ⇒14

[15] F. Corzilius, *Integrating Virtual Substitution into Strategic SMT Solving.* PhD thesis, RWTH Aachen University, 2016. ⇒15

[16] F. Corzilius, G. Kremer, S. Junges, S. Schupp, E. Ábrahám, SMT-RAT: An open source C++ toolbox for strategic and parallel SMT solving. *Proc. of SAT'15* (2015), vol. 9340 of *LNCS*, Springer, pp. 360–368. ⇒6, 7, 9, 15, 20

[17] F. Corzilius, U. Loup, S. Junges, S., E. Ábrahám, SMT-RAT: An SMT-compliant nonlinear real arithmetic toolbox. *Proc. of SAT'12* (2012), vol. 7317 of *LNCS*, Springer, pp. 442–448. ⇒6, 7

[18] G. B. Dantzig, *Linear programming and extensions.* Princeton University Press, 1963. ⇒10

[19] M. Davis, H. Putnam, A computing procedure for quantification theory. *Journal of the ACM* **7,** 3 (1960) 201–215. ⇒8

[20] L. de Moura, N. Bjørner, Z3: An efficient SMT solver. *Proc. of TACAS'08* (2008), vol. 4963 of *LNCS*, Springer, pp. 337–340. ⇒9, 14

[21] L. M. de Moura, D. Jovanovic, A model-constructing satisfiability calculus. *Proc. of VMCAI'13* (2013), vol. 7737 of *LNCS*, Springer, pp. 1–12. ⇒9, 16

[22] B. Dutertre, Yices 2.2. *Proc. of CAV'14* (2014), vol. 8559 of *LNCS*, Springer, pp. 737–744. ⇒9, 14

[23] B. Dutertre, L. M. de Moura, A fast linear-arithmetic solver for DPLL(T). *Proc. of CAV'06* (2006), vol. 4144 of *LNCS*, Springer, pp. 81–94. ⇒15

[24] N. Eén, N. Sörensson, An extensible SAT-solver. *Proc. of SAT'03* (2004), vol. 2919 of *LNCS*, Springer, pp. 502–518. ⇒15

[25] P. Fontaine, M. Ogawa, T. Sturm, T., X. T. Vu, Subtropical satisfiability. *Proc. of FroCoS'17* (2017), Springer, pp. 189–206. ⇒11, 16

[26] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, H. Zankl, SAT solving for termination analysis with polynomial interpretations. *Proc. of SAT'07* (2007), Springer, pp. 340–354. ⇒15

[27] S. Gao, M. Ganai, F. Ivančić, A. Gupta, S. Sankaranarayanan, E. M. Clarke, Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems. *Proc. of FMCAD'10* (2010), IEEE, pp. 81–90. ⇒10

[28] K. O. Geddes, S. R. Czapor, G. Labahn, *Algorithms for Computer Algebra.* Kluwer Academic Publishers, 1992. ⇒7

[29] J. Giesl, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, C. Otto, M. Plücker, P. Schneider-Kamp, T. Ströder, S. Swiderski, R. Thiemann, Proving termination of programs automatically with AProVE. *Proc. of IJCAR'14* (2014), vol. 8562 of *LNAI*, Springer, pp. 184–191. ⇒9

[30] M. Grobelna, SAT-modulo-theories solving for pseudo-Boolean constraints. Bachelor's Thesis, RWTH Aachen University, 2017. ⇒14, 15

[31] R. Haehn, Using equational constraints in an incremental CAD projection. Master's thesis, RWTH Aachen University, 2017. ⇒14

[32] E. M. Hahn, H. Hermanns, B. Wachter, L. Zhang, PARAM: A model checker for parametric Markov models. *Proc. of CAV'10* (2010), vol. 6174 of *LNCS*, Springer, pp. 660–664. ⇒12

[33] W. Hentze, Infeasible subsets for nonlinear SMT. Bachelor's Thesis, RWTH Aachen University, 2017. ⇒14

[34] S. Herbort, D. Ratz, Improving the efficiency of a nonlinear-system-solver using a componentwise Newton method. Tech. Rep. 2/1997, Inst. für Angewandte Mathematik, University of Karlsruhe, 1997. ⇒10

[35] D. Jovanović, Solving nonlinear integer arithmetic with MCSAT. *Proc. of VMCAI'17* (2017), Springer, pp. 330–346. ⇒20

[36] D. Jovanović, L. de Moura, Solving non-linear arithmetic. *Proc. of IJCAR'12* (2012), vol. 7364 of *LNAI*, Springer, pp. 339–354. ⇒16

[37] S. Junges, On Gröbner bases in SMT-compliant decision procedures. Bachelor's Thesis, RWTH Aachen University, 2012. ⇒14

[38] S. Junges, U. Loup, F. Corzilius, E. Ábrahám, E. On Gröbner bases in the context of satisfiability-modulo-theories solving over the real numbers. *Proc. of CAI'13* (2013), vol. 8080 of *LNCS*, Springer, pp. 186–198. ⇒15

[39] T. V. Khanh, X. Vu, M. Ogawa, raSAT: SMT for polynomial inequality. *Proc. of SMT'14* (2014), p. 67. ⇒9

[40] G. Kremer, Isolating real roots using adaptable-precision interval arithmetic. Master's thesis, RWTH Aachen University, 2013. ⇒14

[41] G. Kremer, F. Corzilius, E. Ábrahám, A generalised branch-and-bound approach and its application in SAT modulo nonlinear integer arithmetic. *Proc. of CASC'16* (2016), vol. 9890 of *LNCS*, Springer, pp. 315–335. ⇒11, 15, 20

[42] A. Krüger, Bitvectors in SMT-RAT and their application to integer arithmetics. Master's thesis, RWTH Aachen University, 2015. ⇒14, 15

[43] J. P. Marques-Silva, K. A. Sakallah, Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers* **48** (1999) 506–521. ⇒8

[44] J. Nalbach, Embedding the virtual substitution in the MCSAT framework. Bachelor's Thesis, RWTH Aachen University, 2017. ⇒14

[45] L. Netz, Using Horner schemes to improve the efficiency and precision of interval constraint propagation. Bachelor's Thesis, RWTH Aachen University, 2015. ⇒ 14

[46] L. Neuberger, Generation of infeasible subsets in less-lazy SMT-solving for the theory of uninterpreted functions. Bachelor's Thesis, RWTH Aachen University, 2015. ⇒14

[47] J. Redies, An extension of the GiNaCRA library for the cylindrical algebraic decomposition. Bachelor's Thesis, RWTH Aachen University, 2012. ⇒14

[48] SMT-COMP 2017 result summary. `http://smtcomp.sourceforge.net/2017/results-toc.shtml`, 2017. ⇒21

[49] S. Schupp, Interval constraint propagation in SMT compliant decision procedures. Master's thesis, RWTH Aachen University, 2013. ⇒14, 15

[50] D. Scully, Preprocessing for solving non-linear real-arithmetic formulas. Bachelor's Thesis, RWTH Aachen University, 2012. ⇒14

[51] G. S. Tseitin, On the complexity of derivation in propositional calculus. in: *Automation of Reasoning.* Springer, 1983, pp. 466–483. ⇒8

[52] V. X. Tung, T. Van Khanh, M. Ogawa, raSAT: An SMT solver for polynomial constraints. *Formal Methods in System Design* **51,** 3 (2017), 462–499. ⇒14, 20

[53] T. Viehmann, Projection operators for the CAD. Bachelor's Thesis, RWTH Aachen University, 2016. ⇒14

[54] T. Viehmann, G. Kremer, E. Ábrahám, Comparing different projection operators in the cylindrical algebraic decomposition for SMT solving. *Proc. of SC-square'17* (2017), vol. 1974 of *CEUR Workshop Proceedings*, CEUR-WS.org. ⇒15

[55] V. Weispfenning, Quantifier elimination for real algebra – the quadratic case and beyond. *Appl. Algebra Eng. Commun. Comput.* **8,** 2 (1997), 85–101. ⇒10

[56] T. Winkler, Using Thom's lemma for real algebraic numbers in the CAD. Bachelor's Thesis, RWTH Aachen University, 2016. ⇒14

*Received: May 15, 2018 • Revised: July 23, 2018*

Acta Univ. Sapientiae, Informatica 10, 1 (2018) 26–42

DOI: 10.2478/ausi-2018-0002

# Fruit recognition from images using deep learning

Horea MUREŞAN
Faculty of Mathematics and Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
email: horea94@gmail.com

Mihai OLTEAN
Faculty of Exact Sciences and Engineering
"1 Decembrie 1918" University of Alba Iulia
Alba Iulia, Romania
email: mihai.oltean@gmail.com

**Abstract.** In this paper we introduce a new, high-quality, dataset of images containing fruits. We also present the results of some numerical experiment for training a neural network to detect fruits. We discuss the reason why we chose to use fruits in this project by proposing a few applications that could use such classifier.

## 1 Introduction

The aim of this paper is to propose a new dataset of images containing popular fruits. The dataset was named Fruits-360 and can be downloaded from the addresses pointed by references [30] and [31]. Currently (as of 2018.05.22) the set contains 38409 images of 60 fruits and it is constantly updated with images of new fruits as soon as the authors have accesses to them. The reader is encouraged to access the latest version of the dataset from the above indicated addresses.

**Computing Classification System 1998:** I.2.6
**Mathematics Subject Classification 2010:** 68T45
**Key words and phrases:** deep learning, object recognition, computer vision, fruits dataset, image processing

Having a high-quality dataset is essential for obtaining a good classifier. Most of the existing datasets with images (see for instance the popular CIFAR dataset [29]) contain both the object and the noisy background. This could lead to cases where changing the background will lead to the incorrect classification of the object.

As a second objective we have trained a deep neural network that is capable of identifying fruits from images. This is part of a more complex project that has the target of obtaining a classifier that can identify a much wider array of objects from images. This fits the current trend of companies working in the augmented reality field. During its annual I/O conference, Google announced [32] that is working on an application named Google Lens which will tell the user many useful information about the object toward which the phone camera is pointing. First step in creating such application is to correctly identify the objects. The software has been released later in 2017 as a feature of Google Assistant and Google Photos apps. Currently the identification of objects is based on a deep neural network [33].

Such a network would have numerous applications across multiple domains like autonomous navigation, modeling objects, controlling processes or human-robot interactions. The area we are most interested in is creating an autonomous robot that can perform more complex tasks than a regular industrial robot. An example of this is a robot that can perform inspections on the aisles of stores in order to identify out of place items or understocked shelves. Furthermore, this robot could be enhanced to be able to interact with the products so that it can solve the problems on its own.

As the start of this project we chose the task of identifying fruits for several reasons. On one side, fruits have certain categories that are hard to differentiate, like the citrus genus, that contains oranges and grapefruits. Thus we want to see how well can an artificial intelligence complete the task of classifying them. Another reason is that fruits are very often found in stores, so they serve as a good starting point for the previously mentioned project.

The paper is structured as follows: in the first part we will shortly discuss a few outstanding achievements obtained using deep learning for fruits recognition, followed by a presentation of the concept of deep learning. In the second part we will present the framework used in this project - TensorFlow[27] and the reasons we chose it. Following the framework presentation, we will detail the structure of the neural network that we used. We also describe the training and testing data used as well as the obtained performance. Finally, we will conclude with a few plans on how to improve the results of this project.

## 2 Related work

In this section we review several previous attempts to use neural networks and deep learning for fruits recognition.

A method for recognizing and counting fruits from images in cluttered greenhouses is presented in [19]. The targeted plants are peppers with fruits of complex shapes and varying colors similar to the plant canopy. The aim of the application is to locate and count green and red pepper fruits on large, dense pepper plants growing in a greenhouse. The training and validation data used in this paper consists of 28000 images of over 1000 plants and their fruits. The used method to locate and count the peppers is two-step: in the first step, the fruits are located in a single image and in a second step multiple views are combined to increase the detection rate of the fruits. The approach to find the pepper fruits in a single image is based on a combination of (1) finding points of interest, (2) applying a complex high-dimensional feature descriptor of a patch around the point of interest and (3) using a so-called bag-of-words for classifying the patch.

Paper [17] presents a novel approach for detecting fruits from images using deep neural networks. For this purpose the authors adapt a Faster Region-based convolutional network. The objective is to create a neural network that would be used by autonomous robots that can harvest fruits. The network is trained using RGB and NIR (near infra red) images. The combination of the RGB and NIR models is done in 2 separate cases: early and late fusion. Early fusion implies that the input layer has 4 channels: 3 for the RGB image and one for the NIR image. Late fusion uses 2 independently trained models that are merged by obtaining predictions from both models and averaging the results. The result is a multi modal network which obtains much better performance than the existing networks.

On the topic of autonomous robots used for harvesting, paper [2] shows a network trained to recognize fruits in an orchard. This is a particularly difficult task because in order to optimize operations, images that span many fruit trees must be used. In such images, the amount of fruits can be large, in the case of almonds up to 1500 fruits per image. Also, because the images are taken outside, there is a lot of variance in luminosity, fruit size, clustering and view point. Like in paper [17], this project makes use of the

Faster Region-based convolutional network, which is presented in a detailed view in paper [16]. Related to the automatic harvest of fruits, article [14] presents a method of detecting ripe strawberries and apples from orchards. The paper also highlights existing methods and their performance.

In [10] the authors compile a list of the available state of the art methods for harvesting with the aid of robots. They also analyze the method and propose ways to improve them.

In [3] one can see a method of generating synthetic images that are highly similar to empirical images. Specifically, this paper introduces a method for the generation of large-scale semantic segmentation datasets on a plant-part level of realistic agriculture scenes, including automated per-pixel class and depth labeling. One purpose of such synthetic dataset would be to bootstrap or pre-train computer vision models, which are fine-tuned thereafter on a smaller empirical image dataset. Similarly, in paper [15] we can see a network trained on synthetic images that can count the number of fruits in images without actually detecting where they are in the image.

Another paper, [5], uses two back propagation neural networks trained on images with apple "Gala" variety trees in order to predict the yield for the upcoming season. For this task, four features have been extracted from images: total cross-sectional area of fruits, fruit number, total cross-section area of small fruits, and cross-sectional area of foliage.

Paper [9] presents an analysis of fruit detectability in relation to the angle of the camera when the image was taken. Based on this research, it was concluded that the fruit detectability was the highest on front views and looking with a zenith angle of $60^\circ$ upwards.

In papers [23, 1, 24] we can see an approach to detecting fruits based on color, shape and texture. They highlight the difficulty of correctly classifying similar fruits of different species. They propose combining existing methods using the texture, shape and color of fruits to detect regions of interest from images. Similarly, in [13] a method combining shape, size and color, texture of the fruits together with a k nearest neighbor algorithm is used to increase the accuracy of recognition.

One of the most recent works [22] presents an algorithm based on the improved ChanVese level-set model [4] and combined with the level-set idea and M-S mode [12]. The proposed goal was to conduct night-time green grape detection. Combining the principle of the minimum circumscribed rectangle of fruit and the method of Hough straight-line detection, the picking point of the fruit stem was calculated.

## 3 Deep learning

Deep learning is a class of machine learning algorithms that use multiple layers that contain nonlinear processing units [18]. Each layer uses the output from the previous layer as input. Deep learning[26] algorithms use more layers than shallow learning algorithms. Convolutional neural networks are classified as a deep learning algorithm. These networks are composed of multiple convolutional layers with a few fully connected layers. They also make use of pooling. This configuration allows convolutional networks to take advantage of bidimensional representation of data. Another deep learning algorithm is the recursive neural network. In this kind of architecture the same set of weights is recursively applied over some data. Recurrent networks have shown good results in natural language processing. Yet another model that is part of the deep learning algorithms is the deep belief network. A deep belief network is a probabilistic model composed by multiple layers of hidden units. The usages of a deep belief network are the same as the other presented networks but can also be used to pre-train a deep neural network in order to improve the initial values of the weights. This process is important because it can improve the quality of the network and can reduce training times. Deep belief networks can be combined with convolutional ones in order to obtain convolutional deep belief networks which exploit the advantages offered by both types of architectures.

In the area of image recognition and classification, the most successful results were obtained using artificial neural networks [6, 21]. This served as one of the reasons we chose to use a deep neural network in order to identify fruits from images. Deep neural networks have managed to outperform other machine learning algorithms. They also achieved the first superhuman pattern recognition in certain domains. This is further reinforced by the fact that deep learning is considered as an important step towards obtaining Strong AI. Secondly, deep neural networks – specifically convolutional neural networks – have been proved to obtain great results in the field of image recognition. We will present a few results on popular datasets and the used methods.

Among the best results obtained on the MNIST [28] dataset is done by using multi-column deep neural networks. As described in paper [7], they use multiple maps per layer with many layers of non-linear neurons. Even if the complexity of such networks makes them harder to train, by using graphical processors and special code written for them. The structure of the network uses winner-take-all neurons with max pooling that determine the winner neurons.

Another paper [11] further reinforces the idea that convolutional networks have obtained better accuracy in the domain of computer vision. The paper proposes an improvement to the popular convolutional network in the form of a recurrent convolutional network. Traditionally, recurrent networks have been used to process sequential data, handwriting or speech recognition being the most known examples. By using recurrent convolutional layers with some max pool layers in between them and a final global max pool layer at the end several advantages are obtained. Firstly, within a layer, every unit takes into account the state of units in an increasingly larger area around it. Secondly, by having recurrent layers, the depth of the network is increased without adding more parameters.

In paper [20] an all convolutional network that gains very good performance on CIFAR-10 [29] is described in detail. The paper proposes the replacement of pooling and fully connected layers with equivalent convolutional ones. This may increase the number of parameters and adds inter-feature dependencies however it can be mitigated by using smaller convolutional layers within the network and acts as a form of regularization.

## 4 Fruits-360 data set

In this section we describe how the data set was created and what it contains.

The images were obtained by filming the fruits while they are rotated by a motor and then extracting frames.

Fruits were planted in the shaft of a low speed motor (3 rpm) and a short movie of 20 seconds was recorded. Behind the fruits we placed a white sheet of paper as background.

However due to the variations in the lighting conditions, the background was not uniform and we wrote a dedicated algorithm which extract the fruit from the background. This algorithm is of flood fill type: we start from each edge of the image and we mark all pixels there, then we mark all pixels found in the neighborhood of the already marked pixels for which the distance between colors is less than a prescribed value. we repeat the previous step until no more pixels can be marked.

All marked pixels are considered as being background (which is then filled with white) and the rest of pixels are considered as belonging to the object. The maximum value for the distance between 2 neighbor pixels is a parameter of the algorithm and is set (by trial and error) for each movie.

Fruits were scaled to fit a 100x100 pixels image. Other datasets (like MNIST) use 28x28 images, but we feel that small size is detrimental when you have too similar objects (a red cherry looks very similar to a red apple in small images). Our future plan is to work with even larger images, but this will require much more longer training times.

To understand the complexity of background-removal process we have depicted in Figure 1 a fruit with its original background and after the background was removed and the fruit was scaled down to 100 x 100 pixels.

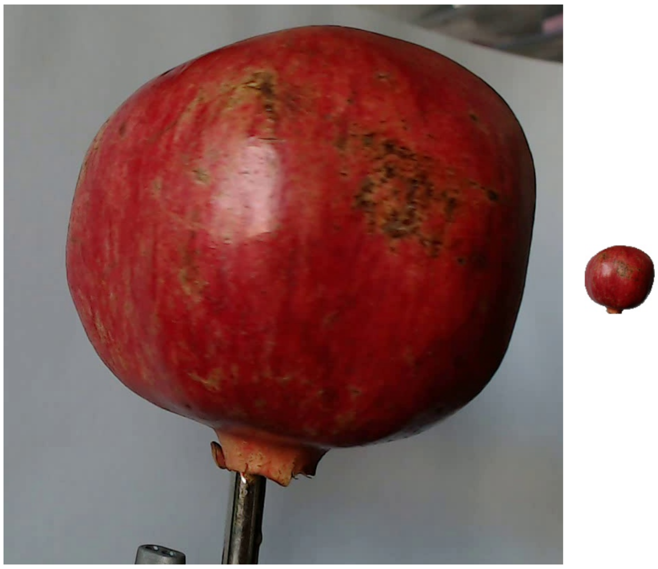


Figure 1: Left-side: original image. Notice the background and the motor shaft. Right-side: the fruit after the background removal and after it was scaled down to 100x100 pixels.

The resulted dataset has 38409 images of fruits spread across 60 labels. The data set is available on GitHub [30] and Kaggle [31]. The labels and the number of images for training are given in Table 1.

| **Label** | **Number of training images** | **Number of test images** |
|---|---|---|
| Apple Braeburn | 492 | 164 |
| Apple Golden 1 | 492 | 164 |
| Apple Golden 2 | 492 | 164 |
| Apple Golden 3 | 481 | 161 |
| Apple Granny Smith | 492 | 164 |
| Apple Red 1 | 492 | 164 |
| Apple Red 2 | 492 | 164 |
| Apple Red 3 | 429 | 144 |
| Apple Red Delicious | 490 | 166 |
| Apple Red Yellow | 492 | 164 |
| Apricot | 492 | 164 |
| Avocado | 427 | 143 |
| Avocado ripe | 491 | 166 |
| Banana | 490 | 166 |
| Banana Red | 490 | 166 |
| Cactus fruit | 490 | 166 |
| Carambula | 490 | 166 |
| Cherry | 492 | 164 |
| Clementine | 490 | 166 |
| Cocos | 490 | 166 |
| Dates | 490 | 166 |
| Granadilla | 490 | 166 |
| Grape Pink | 492 | 164 |
| Grape White | 490 | 166 |
| Grape White 2 | 490 | 166 |
| Grapefruit Pink | 490 | 166 |
| Grapefruit White | 492 | 164 |
| Guava | 490 | 166 |
| Huckleberry | 490 | 166 |
| Kaki | 490 | 166 |

*It continues on the next page.*

| **Label** | **Number of training images** | **Number of test images** |
|---|---|---|
| Kiwi | 466 | 156 |
| Kumquats | 490 | 166 |
| Lemon | 246 | 82 |
| Lemon Meyer | 490 | 166 |
| Limes | 490 | 166 |
| Litchi | 490 | 166 |
| Mandarine | 490 | 166 |
| Mango | 490 | 166 |
| Maracuja | 490 | 166 |
| Nectarine | 492 | 164 |
| Orange | 479 | 160 |
| Papaya | 492 | 164 |
| Passion Fruit | 490 | 166 |
| Peach | 492 | 164 |
| Peach Flat | 492 | 164 |
| Pear | 492 | 164 |
| Pear Abate | 490 | 166 |
| Pear Monster | 490 | 166 |
| Pear Williams | 490 | 166 |
| Pepino | 490 | 166 |
| Pineapple | 490 | 166 |
| Pitahaya Red | 490 | 166 |
| Plum | 447 | 151 |
| Pomegranate | 246 | 82 |
| Quince | 490 | 166 |
| Raspberry | 490 | 166 |
| Salak | 490 | 162 |
| Strawberry | 492 | 164 |
| Tamarillo | 490 | 166 |
| Tangelo | 490 | 166 |

Table 1: Number of images for each fruit. There are multiple varieties of apples each of them being considered as a separate object. We did not find the scientific/popular name for each apple so we labeled with digits (e.g. apple red 1, apple red 2 etc).

# 5 Neural network structure and utilized framework

For this project we used a convolutional neural network. This type of network makes use of convolutional layers, pooling layers, ReLU layers, fully connected layers and loss layers. In a typical CNN architecture, each convolutional layer is followed by a Rectified Linear Unit (ReLU) layer, then a Pooling layer then one or more convolutional layer and finally one or more fully connected layer.

A characteristic that sets apart the CNN from a regular neural network is taking into account the structure of the images while processing them. A regular neural network converts the input in a one dimensional array which makes the trained classifier less sensitive to positional changes.

Convolutional layers are named after the convolution operation. In mathematics convolution is an operation on two functions that produces a third function that is the modified (convoluted) version of one of the original functions. The resulting function gives in integral of the pointwise multiplication of the two functions as a function of the amount that one of the original functions is translated [25].

A convolutional layer consists of groups of neurons that make up kernels. The kernels have a small size but they always have the same depth as the input. The neurons from a kernel are connected to a small region of the input, called the receptive field, because it is highly inefficient to link all neurons to all previous outputs in the case of inputs of high dimensions such as images. For example, a 100 x 100 image has 10000 pixels and if the first layer has 100 neurons, it would result in 1000000 parameters. Instead of each neuron having weights for the full dimension of the input, a neuron holds weights for the dimension of the kernel input. The kernels slide across the width and height of the input, extract high level features and produce a 2 dimensional activation map. The stride at which a kernel slides is given as a parameter. The output of a convolutional layer is made by stacking the resulted activation maps which in turned is used to define the input of the next layer.

In TensorFlow [27] framework (which we utilized in the numerical experiments), a convolutional layer is defined like this:

```
conv2d(
    input,
    filter,
    strides ,
    padding ,
    use_cudnn_on_gpu=True ,
    data_format='NHWC' ,
    dilations=[1, 1, 1, 1] ,
    name=None
)
```

Applying a convolutional layer over an image of size 32 X 32 results in an activation map of size 28 X 28. If we apply more convolutional layers, the size will be further reduced, and, as a result the image size is drastically reduced which produces loss of information and the vanishing gradient problem. To correct this, we use padding. Padding increases the size of a input data by filling constants around input data. In most of the cases, this constant is zero so the operation is named zero padding. "Same" padding means that the output feature map has the same spatial dimensions as the input feature map. This tries to pad evenly left and right, but if the number of columns to be added is odd, it will add an extra column to the right. "Valid" padding is equivalent to no padding.

The strides causes a kernel to skip over pixels in an image and not include them in the output. The strides determines how a convolution operation works with a kernel when a larger image and more complex kernel are used. As a kernel is sliding the input, it is using the strides parameter to determine how many positions to skip.

ReLU layer, or Rectified Linear Units layer, applies the activation function max(0, x). It does not reduce the size of the network, but it increases its nonlinear properties.

Pooling layers are used on one hand to reduce the spatial dimensions of the representation and to reduce the amount of computation done in the network. The other use of pooling layers is to control overfitting. The most used pooling layer has filters of size 2 x 2 with a stride 2. This effectively reduces the input to a quarter of its original size.

Fully connected layers are layers from a regular neural network. Each neuron from a fully connected layer is linked to each output of the previous layer. The operations behind a convolutional layer are the same as in a fully connected layer. Thus, it is possible to convert between the two.

Loss layers are used to penalize the network for deviating from the expected output. This is normally the last layer of the network. Various loss function exist: softmax is used for predicting a class from multiple disjunct classes, sigmoid cross-entropy is used for predicting multiple independent probabilities (from the [0, 1] interval).

The input that we used consists of standard RGB images of size 100 x 100 pixels.

The neural network that we used in this project has the structure given in Table 2.

| **Layer type** | **Dimensions** | **Outputs** |
|---|---|---|
| Convolutional | 5 x 5 x 4 | 16 |
| Max pooling | 2 x 2 — Stride: 2 | - |
| Convolutional | 5 x 5 x 16 | 32 |
| Max pooling | 2 x 2 — Stride: 2 | - |
| Convolutional | 5 x 5 x 32 | 64 |
| Max pooling | 2 x 2 — Stride: 2 | - |
| Convolutional | 5 x 5 x 64 | 128 |
| Max pooling | 2 x 2 — Stride: 2 | - |
| Fully connected | 5 x 5 x 128 | 1024 |
| Fully connected | 1024 | 256 |
| Softmax | 256 | 60 |

Table 2: The structure of the neural network used in this paper.

The first layer is a convolutional layer which applies 16 5 x 5 filters. On this layer we apply max pooling with a filter of shape 2 x 2 with stride 2 which specifies that the pooled regions do not overlap. This also reduces the width and height to 50 pixels each. The second convolutional layer applies 32 5 x 5 filters which outputs 32 activation maps. We apply on this layer the same kind of max pooling as on the first layer, shape 2 x 2 and stride 2. The third convolutional layer applies 64 5 x 5 filters. Following is another max pool layer of shape 2 x 2 and stride 2. The fourth convolutional layer applies 128 5 x 5 filters after which we apply a final max pool layer. Because of the four max pooling layers, the dimensions of the representation have each been reduced by a factor of 16, therefore the fifth layer, which is a fully connected layer, has 5 x 5 x 16 inputs. This layer feeds into another fully connected layer with 1024 inputs and 256 outputs. The last layer is a softmax loss layer with 256 inputs. The number of outputs is equal to the number of classes.

In order to create our convolutional neural network we used TensorFlow [27]. This is an open source framework for machine learning created by

Google for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays called tensors. The main components in a TensorFlow system are the client, which uses the Session interface to communicate with the master, and one or more worker processes, with each worker process responsible for arbitrating access to one or more computational devices (such as CPU cores or GPU cards) and for executing graph nodes on those devices as instructed by the master. TensorFlow offers some powerful features such as: it allows computation mapping to multiple machines, unlike most other similar frameworks; it has built in support for automatic gradient computation; it can partially execute subgraphs of the entire graph and it can add constraints to devices, like placing nodes on devices of a certain type, ensure that two or more objects are placed in the same space etc.

## 6 Numerical experiments

The dataset was split in 2 parts: training set – which consists of 28736 images of fruits and testing set – which is made of 9673 images.

The data was bundled into TFRecords file (specific to TensorFlow). This is a binary file that contains protocol buffers with a feature map. In this map it is possible to store information such as the image height, width, depth and even the raw image. Using these files we can create queues in order to feed the data to the neural network. By calling the method $shuffle\_batch$ we provide randomized input to the network. The way we used this method was providing it example tensors for images and labels and it returned tensors of shape batch size x image dimensions and batch size x labels. This helps greatly lower the chance of using the same batch multiple times for training, which in turn improves the quality of the network.

On each image from the batch we applied some preprocessing in order to augment the data set. The preprocessing consists of randomly altering the hue and saturation, and applying random vertical and horizontal flips.

For the hue and saturation we use the tensorflow methods: $random\_hue$ and $random\_saturation$. To further improve the accuracy of the network we converted each image from the batch to grayscale and concatenated it to the image. Thus the data that is fed into the network will have the size 100 x 100 x 4.

In order to be able to detect fruits from images we used the previously described neural network which was trained over 40000 iterations with batches of 50 images selected at random from the training set. Every 50 steps we calculated the accuracy using cross-validation. This showed steady improving of the network until reaching 100% accuracy on cross-validation. For the testing phase, we used the testing set and the calculated accuracy was 96.3%.

Some of the incorrectly classified images are given in Table 3.

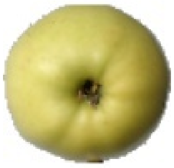
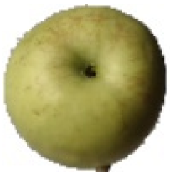
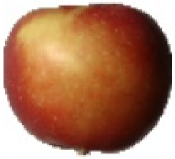
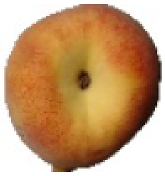
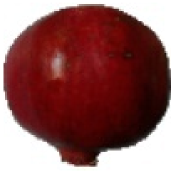
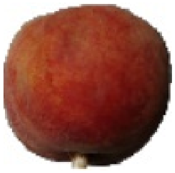
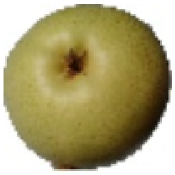
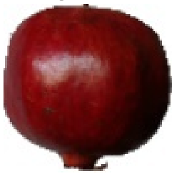
| | | | |
|---|---|---|---|
| Apple Golden 2  Apple Golden 3 | Apple Golden 3  Granny Smith (Apple) | Braeburn(Apple)  Apple Red 2 | Peach  Apple Red Yellow |
| Pomegranate  Nectarine | Peach  Apple Red 1 | Pear  Apple Golden 2 | Pomegranate  Braeburn(Apple) |

Table 3: Some of the images that were classified incorrectly. On the top we have the correct class of the fruit and on the bottom we have the class that was given by the network.

## 7 Conclusions

This project tries to set up a start to an area that is less explored at the current time. During this project we were able to explore part of the deep learning algorithms and discover strengths and weaknesses. We gained knowledge on deep learning and we obtained a software that can recognize fruits from images. We hope that the results and methods presented in this paper can be further expanded in a bigger project.

From our point of view one of the main objectives for the future is to improve the accuracy of the neural network. This involves further experimenting with the structure of the network. Various tweaks and changes to any layers as well as the introduction of new layers can provide completely different results. Another option is to replace all layers with convolutional layers. This has been shown to provide some improvement over the networks that have fully connected layers in their structure. A consequence of replacing all layers with convolutional ones is that there will be an increase in the number of parameters for the network [20]. Another possibility is to replace the rectified linear units with exponential linear units. According to paper [8], this reduces computational complexity and add significantly better generalization performance than rectified linear units on networks with more that 5 layers. We would like to try out these practices and also to try to find new configurations that provide interesting results.

In the near future we plan to create a mobile application which takes pictures of fruits and labels them accordingly.

Another objective is to expand the data set to include more fruits. This is a more time consuming process since we want to include items that were not used in most others related papers.

## Acknowledgments

A preliminary version of this dataset with 25 fruits was presented during the Students Communication Session from Babeş-Bolyai University, June 2017.

## References

[1] S. Arivazhagan, N. Shebiah, S. Nidhyanandhan, L.Ganesan, Fruit recognition using color and texture features, *Journal of Emerging Trends in Computing and Information Sciences,* **1,** 2 (2010) 90–94. ⇒29

[2] S. Bargoti, J. Underwood, Deep fruit detection in orchards, *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3626–3633. ⇒28

[3] R. Barth, J. Ijsselmuiden, J. Hemming, E. Van Henten, Data synthesis methods for semantic segmentation in agriculture: A Capsicum annuum dataset, *Computers and Electronics in Agriculture* **144** (2018) 284–296. ⇒29

[4] T. F. Chan, L. Vese, Active contours without edges. *IEEE Trans. Image Process* **10,** (2001) 266–277. ⇒29

[5] H. Cheng, L. Damerow, Y. Sun, M. Blanke, Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks, *Journal of Imaging*, **3,** 1 (2017) 6. ⇒29

[6] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber, Flexible, high performance convolutional neural networks for image classification, *Twenty-Second International Joint Conference on Artificial Intelligence*, Barcelona, pp. 1237–1242, AAAI Press, 2011. ⇒30

[7] D. C. Cireşan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Providence, pp. 3642–3649, 2012. ⇒30

[8] D. Clevert, T. Unterthiner, S. Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) *CoRR* abs/1511.07289, 2015. ⇒40

[9] J. Hemming, J. Ruizendaal, J. W. Hofstee. E J. Van Henten, Fruit detectability analysis for different camera positions in sweet-pepper *Sensors* **14,** 4 (2014) 6032–6044. ⇒29

[10] K. Kapach, E. Barnea, R. Mairon, Y. Edan, O. Ben-Shahar, Computer vision for fruit harvesting robots state of the art and challenges ahead, *Journal of Imaging* **3,** 1 (2017) 4–34. ⇒29

[11] M. Liang, X. Hu, Recurrent convolutional neural network for object recognition, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Boston. 2015. pp. 3367–3375. ⇒31

[12] D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, *Commun. Pure Appl. Math.* **42** (1989) 577-685. ⇒29

[13] P. Ninawe, S. Pandey, A completion on fruit recognition system using *k*-nearest neighbors algorithm, *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* **3,** 7 (2014) 2352–2356. ⇒ 29

[14] S. Puttemans, Y. Vanbrabant, L. Tits, T. Goedem, Automated visual fruit detection for harvest estimation and robotic harvesting, *Sixth International Conference on Image Processing Theory, Tools and Applications,* 2016. ⇒29

[15] M, Rahnemoonfar, C. Sheppard, Deep count: fruit counting based on deep simulated learning, *Sensors* **17,** 4 (2017) 905. ⇒29

[16] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, *Advances in Neural Information Processing Systems*, 2015, 91–99. ⇒29

[17] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez & C. McCool, DeepFruits: A fruit detection system using deep neural networks, *Sensors* **16,** 8 (2016) 1222. ⇒28

[18] J. Schmidhuber, Deep learning in neural networks: An overview, *Neural Networks* **61** (2015) 85–117. ⇒30

[19] Y. Song, C. Glasbey, G. Horgan, G. Polder, J. A. Dieleman, G. Van Der Heijden, Automatic fruit recognition and counting from multiple images, *Biosystems Engineering* **118** (2)14 203–215. ⇒28

[20] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for simplicity: The all convolutional net, *CoRR* abs/1412.6806, 2014. ⇒31, 40

[21] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, advances in neural information processing systems, *Twenty-Eight International Conference on Neural Information Processing Systems*, Montreal, Canada, 2015, pp. 2377–2385. ⇒30

[22] J. Xiong, Z. Liu, R. Lin, R. Bu, Z. He, Z. Yang, C. Liang , Green grape detection and picking-point calculation in a night-time natural environment using a charge-coupled device (CCD) vision sensor with artificial ullumination, *Sensors* **18,** 4, (2018) 969. ⇒29

[23] H. M. Zawbaa, M. Abbass, M. Hazman, A. E. Hassenian, Automatic fruit image recognition system based on shape and color features, in: *Advanced Machine Learning Technologies and Applications.* (eds.: Hassanien A. E., Tolba M. F., Taher Azar A.) AMLTA 2014. Series: *Communications in Computer and Information Science* vol. 488, 2014, 278–290. ⇒29

[24] D. Li, H. Zhao, X. Zhao, Q. Gao, L. Xu, Cucumber detection based on texture and color in greenhouse, *International Journal of Pattern Recognition and Artificial Intelligence* **31** 1754016 (2017) 17 pag. ⇒29

[25] Convolution in Mathematics. https://en.wikipedia.org/wiki/Convolution. Last visited on 26.05.2018 ⇒35

[26] Deep Learning article on Wikipedia. `https://en.wikipedia.org/wiki/Deep_learning`. Last visited on 05.05.2018 ⇒30

[27] TensorFlow. `https://www.tensorflow.org`. Last visited on 05.05.2018 ⇒27, 35, 37

[28] MNIST. `http://yann.lecun.com/exdb/mnist`. Last visited on 05.05.2018 ⇒30

[29] CIFAR-10 and CIFAR-100 Datasets. `https://www.cs.toronto.edu/~kriz/cifar.html`. Last visited on 05.05.2018 ⇒27, 31

[30] Fruits 360 Dataset on GitHub. `https://github.com/Horea94/Fruit-Images-Dataset`. Last visited on 05.05.2018 ⇒26, 32

[31] Fruits 360 Dataset on Kaggle. `https://www.kaggle.com/moltean/fruits`. Last visited on 28.05.2018 ⇒26, 32

[32] Britta O'Boyle, Chris Hall, What is Google Lens and how do you use it? Last visited on 05.05.2018 ⇒27

[33] Google Lens on Wikipedia, `https://en.wikipedia.org/wiki/Google_Lens`. Last visited on 05.05.2018 ⇒27

*Received: May 15, 2018 • Revised: June 25, 2018*

Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 43–57

DOI: 10.2478/ausi-2018-0003

# On the use of model transformation for the automation of product derivation process in SPL

Nesrine LAHIANI
LRDSI Laboratory
C.S. Department
Saad Dahlab University
Blida, Algeria
email:
`lahiani.nesrine@gmail.com`

Djamal BENNOUAR
LIMPAF Laboratory
C.S. Department
Akli Mohand Oulhadj University
Bouira, Algeria
email:
`djamal.bennouar@univ-bouira.dz`

**Abstract.** Product Derivation represents one of the main challenges that Software Product Line (SPL) faces. Deriving individual products from shared software assets is a time-consuming and an expensive activity. In this paper, we (1) present an MDE approach for engineering SPL and (2) propose to leverage model-to-model transformations (MMT) and model-to-text (MTT) transformations for supporting both domain engineering and application engineering processes. In this work, we use ATL as a model-to-model transformation language and Acceleo as a model-to-text transformation language.The proposed approach is discussed with e-Health product line applications.

## 1 Introduction

Companies are more and more forced to customize their software products for completely different customers. In practice they often clone an existing system

**Computing Classification System 1998:** C4, D.2.11 D.2.13
**Mathematics Subject Classification 2010:** 68N99
**Key words and phrases:** product derivation, software product lines, model-driven engineering, domain specific language

and adapt it to the customer's needs. In such scenarios software product lines promise benefits, for example, reduced maintenance effort, improved quality, and customizability. However, introducing new development processes into a company is risky and might not pay off . The other advantage is that this fairly recent software development paradigm allows companies to create efficiently a variety of complicated products with a short lead-time.

In a software product line context, software products are developed in two phases, i.e. a domain engineering process and an application engineering process. Domain engineering a basis is provided for the actual development of the individual products. During application engineering individual products are derived from the product line, i.e. constructed using a subset of the shared software artifacts. If necessary, additional or replacement product-specific assets may be created.

The key activity in application engineering is Product Derivation. It addresses the construction of a concrete product from the product line core assets, which includes the derivation of application artifacts from domain artifacts, for instance the derivation of Application Requirements from Domain Requirements, the derivation of the Application Architecture from the Domain Architecture and the derivation of Application Components from Domain Components.

In this context, this paper proposes a model-driven product derivation approach based on Model-Driven Engineering principles [20, 21] . We use (1) metamodels to represent domain concerns such as application, architectural, or technological; (2) models that conform to metamodels to designate particular products of product lines; (3) model transformation programs to derivate members of the line from an initial model. Transformation programs are composed by transformation rules. Each transformation rule is responsible for producing a part of the final product. To derive a complete product, we have to assemble the rules in a precise ordering that determines the order in which the individual parts are produced and assembled. To express configurable variability we use feature models. Our feature model represents variation points and variants according to user needs. From the feature model, a product Designer defines a Configuration with his choices. Consequently, our big challenge is to produce adapted transformation programs to contain rules able to derive products with the desired user choices.

The remainder of this paper is structured as follows: Section 2 discusses related work, while Section 3 introduces the terminology and concepts used in this work.In Section 4 we present an overview of our approach for product derivation. Section 5 illustrates the application of our approach on a case study. Finally, Section 6 presents the conclusions.

## 2 Related work

In this section, we cite the state-of-the-art related to product derivation approaches. Perovich et al. in [19] employ model-driven techniques to transform a feature model to specific product architectures. However, the domain design is specified in terms of ATL transformation rules, therefore the transformation processes is not completely automated. Such an approach is complex and makes the SPL architecture design process difficult.

An approach is proposed in [4, 5] to derive the architecture of a product by selectively copying elements from the SPL architecture based on a product-specific feature configuration. The SPL architecture model contains variability to cover all products aspects. This approach deals only with the derivation of the high level product architecture. The mapping between features and the components realizing their implementation is done through an implementation model. A prototype that implements the derivation as a model transformation is described in the Atlas Transformation language.

Tawhid et al. in [22] proposed to derive an UML model of a specific product from the UML model of a product line based on a given feature configuration is enabled through the mapping between features from the feature model and their realizations in the design model. The mapping technique proposed aims to minimize the amount of explicit feature annotations in the UML design model of SPL. Implicit feature mapping is inferred during product derivation from the relationships between annotated and non-annotated model elements as defined in the UML metamodel and well-formed rules. The transformation is realized in ATL.

Gonzlez-Huerta et al. in [11] presented a set of guidelines for the definition of pattern-based quality-driven architectural transformations in a Model-Driven SPL development environment. These guidelines rely both on a multimodel that represents the product line from multiple viewpoints as well as on a derivation process that makes use of this multimodel to derive a product architecture that meets the quality requirements.

Parra et al.[18] propose an approach for feature-based architecture composition in component-based software product lines. To fill the gap between features and software components, authors rely on the definition of aspect-like composition models that link every particular feature with several software components. The approach detects that one feature requires a second feature when the pointcut that define the variation point for the first feature references source code elements referred to by the aspect that defines the second feature. The approach can detect when one feature excludes a second feature, when the pointcuts (that define the variation points) for both features refer to the same source code elements.

All the preceding approaches constitute good effort to provide a smooth transition from feature models to product architectures. In addition, some approaches such as [19, 22] use model-driven techniques to transform a feature model into product architectures. However, as the domain design is specified in terms of ATL transformation rules [3], the transformation processes cannot be fully automated. Other similar efforts that rely on aspect-oriented techniques[18] to derive product architectures from feature selections .Although, the derivation at higher levels of abstraction, that is from generic to concrete product line architectures, is poorly addressed.

## 3 Terminology and basic concepts

In this Section we describe the main terminology and basic concepts of the different areas involved in our approach.

### 3.1 Product lines

Definition (Product lines). *A Software Product Lines can be defined as is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and are developed from a common set of core assets in a prescribed way [6].*

Definition (Feature modeling). Feature modeling is the activity of identifying externally visible characteristics of products in a domain and organizing them into a model called a feature model. The feature modeling described in this section is based on that of [7].

Definition (Product derivation ) We focus in this paper at application engineering known also as product derivation (PD). PD has been defined in many different ways. McGregor in [16] describes the process as *"Product derivation is the focus of a software product line organization and its exact form contributes heavily to the achievement of targeted goals"*.

Deelstra et al. in [8] define product derivation by *"A product is said to be derived from a product family if it is developed using shared product family artifacts. The term product derivation therefore refers to the complete process of constructing a product from product family software assets"*.

### 3.2 Model-driven engineering

DEFINITION (MDE) Kent defines Model Driven Engineering (MDE) by extending MDA with the notion of software development process (i.e., MDE emerged later as a generalization of the MDA for software development) [12]. MDE refers to the systematicuse of models as primary engineering artifacts throughout the engineering lifecycle. All the definition of MDE are based on the concept of model, meta-model, and model transformation.

DEFINITION (META-MODEL) A model is frequently considered an instance conforming a meta-model. Based on [14] *"a meta-model is a model of a modeling language where the languageis specified"*.

DEFINITION (MODEL TRANSFORMATION) Performing a model transformation by taking one or more models as the input and producing one or more models as the output requires a clear understanding of the abstract syntax and the semantics of the source and the target models. Metamodelling is a key concept in MDA that defines the abstract syntax of the models and the inter-relationships between the model elements [13].

The common setting for all transformation languages is such that the model to be transformed (source model) is supplied as a set of class and association instances conforming to the source metamodel. The result of transformation is the target model - the set of instances conforming to the target metamodel. Therefore the transformation has to operate on instance sets specified by a class diagram.

## 4 Model-driven product derivation approach

In this section, we present an overview of our approach for product derivation. It is founded on the principles and techniques of software product lines and model driven engineering. Figure 1 illustrates the main elements of our approach and their respective relationships.

### 4.1 Domain engineering

**Domain analysis**. Domain analysis [17] or Feature modelling is the first activity to define the commonality and variability that can be expected to occur among the SPL members identified in the product line's scope. We use feature model to present the similarities and variations among the products identified in the product line's scope that can be expected to occur.

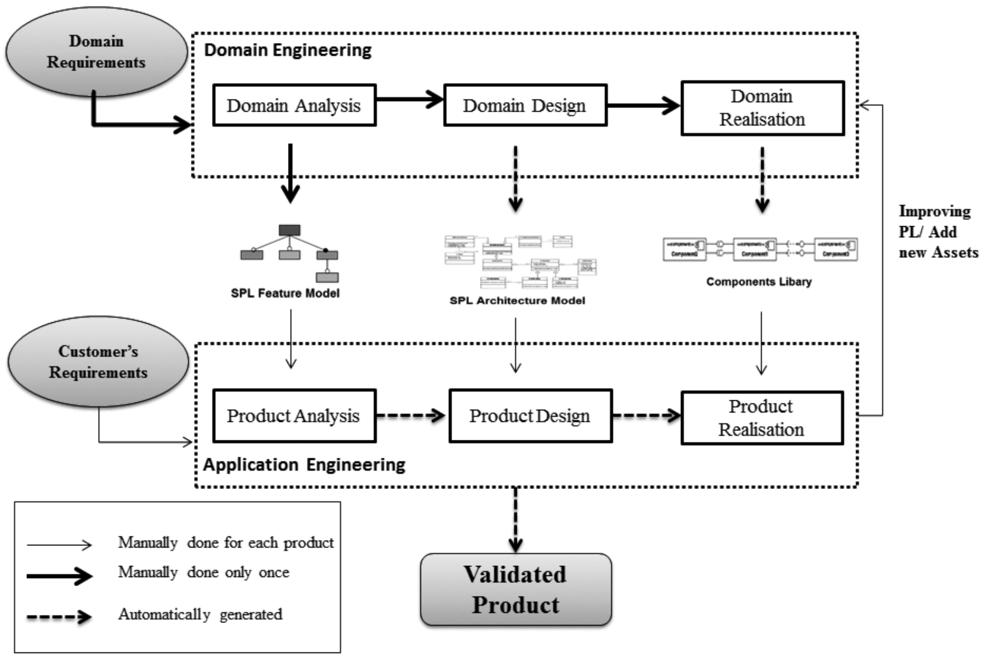


Figure 1: Overview of our approach

To build our metamodel we modify and simplify the metamodel proposed by Czarnecki et al. [7]; we depict it in Figure 2. All Features in the Feature Model have different names and may be composed of several members.

**Domain design**. At this stage the proposed derivation approach uses the mapping technique [15] in aim to map features to architecture model. After that, feature model is considered as an input parameter and then is processed by a model-to-model (M2M) transformation written in ATL (Atlas Transformation Language)[2] that creates an Architecture Model which is composed of a set of rules and helpers. The rules define the mapping between the source and target model. The helpers are methods that can be called from different points in the ATL transformation. This model describes all components that have to be included to implement this particular Application Feature Model. We need to create in the target model all the model element types that compose a component model as its shown in Figure 3.

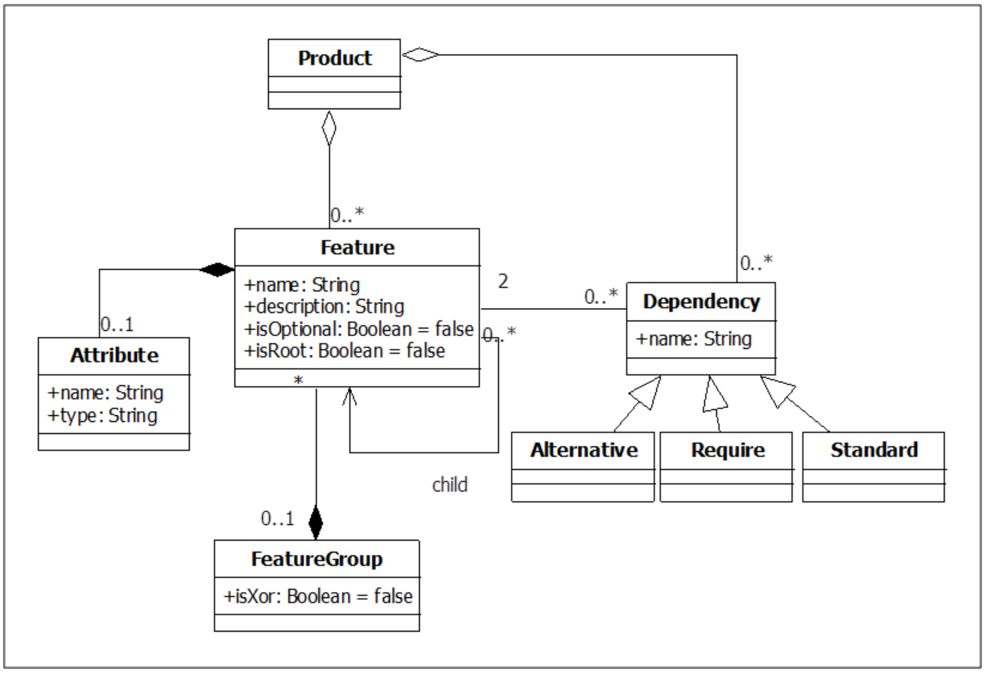


Figure 2: UML metamodel for feature models.

**Domain realization**. The goals of the domain realization sub-process are to provide the detailed design and the implementation of reusable software assets, based on the Architecture Model obtained in the domain design. In addition, domain realization incorporates configuration mechanisms that enable application realization to select variants and build an application with the reusable artifacts. The model obtained in Domain Design is then processed by a model-to-text (M2T) transformation which generates an equivalent textual configuration implemented using Acceleo language [1] to promote the generation of Java. This tool specializes in the generation of text files (code, XML, documentation) starting from models. Using Acceleo we can generate the source code based on templates and models expressed with EMF [9].

## 4.2 Application engineering

**Product analysis**. The main goal of product analysis is to document the requirements artifacts for a particular application and at the same time reuse, as much as possible, the domain requirements artefacts. A feature configura-

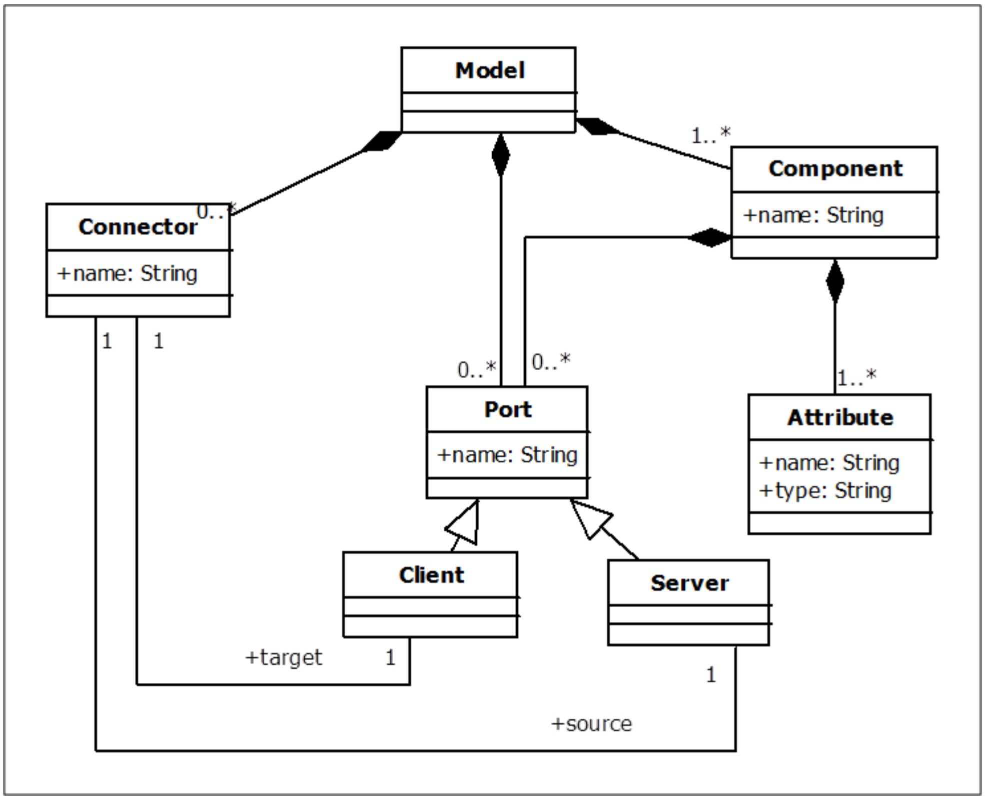


Figure 3: UML metamodel for Component models.

tion is the production of this activity which is a legal combination of features that specifies a particular product. This activity uses feature models as input to select the feature relevant for customers requirements to build the product and identify the specific-assets of the product. Once the selection is checked and validated by the product designer the output at this stage is a specialized version of feature model (application feature model).

**Product design**. The main goal of the product design activity is to produce the product architecture model. The product architecture model is defined for the particular product being developed, considering its desired features defined in the feature configuration model.

**Product realization**. The main goal of product realization is to build the actual product, taking in consideration the product architecture defined in the previous activity. The corresponding component implementations developed during the domain implementation must be used to obtain the implementation of the product.

## 5 Case study

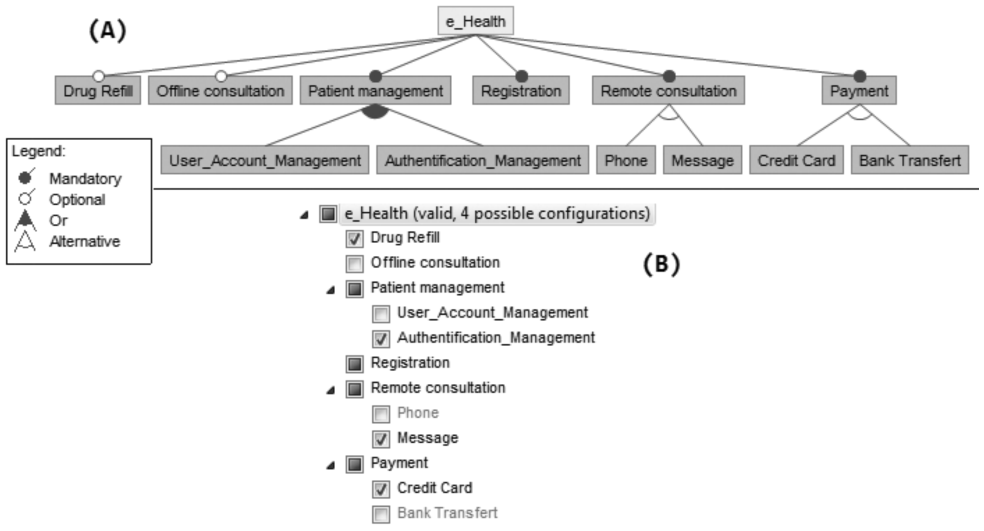
Health-related Internet technology applications delivering a range of clinical care, content, and connectivity, are referred to collectively as e-health. The most remarkable attribute of e-Health is that it is enabling the transformation of the health system from one that is barely focused on curing diseases in hospitals by health professionals, to a system focused on keeping citizens healthy by affording them with information to take care of their health whenever the need arises, and wherever they may be. E-health is promoted as a mechanism to bring growth, gain, cost savings, and process improvement to health care.

In this context of an e-Health application, we present in this section a simple case study to illustrate the overall process, from the feature model to the final product.

### 5.1 Domain engineering

The first activity is domain analysis where we define the feature model for e-Health Product Line, as Figure 4 (part A) shown doctors could connect via the application to follow up (1) remote consultation (via phone/message) and (2) manage patients accounts. Patient also must do (3) a registration so that he/she can consult and (4) pay using its own credit card or just by bank transfer which are alternative features only one could be chosen. Drug refill and offline consultation are two optional features that could be chosen or just left.

Second activity, the domain design where we build the feature-to-architecture transformation rule artifact is built. We use a model-to-model transformation we developed to create an initial version of this model from the feature model, only containing all defined features and their member relationship. We present a fragment of one of the rules using the ATL specialization of our metamodel illustrated in Figure 3, using textual notation. Final activity in domain engineering is the domain realization. The architecture-to-components transformation rule artifact is built. We use a model-to-text transformation we developed to create an initial version of this model from the architecture model, only containing all defined features and their member relationship



e_Health (valid, 4 possible configurations)
- [✓] Drug Refill
- [ ] Offline consultation
- ▣ Patient management
  - [ ] User_Account_Management
  - [✓] Authentification_Management
- ▣ Registration
- ▣ Remote consultation
  - [ ] Phone
  - [✓] Message
- ▣ Payment
  - [✓] Credit Card
  - [ ] Bank Transfert

Figure 4: e-Health product line: (a) Feature Model tree for e-Health applications (b) Feature Configuration Model for e-Health.

## 5.2 Application engineering

During product analysis we use FeatureIDE [10] an Eclipse plug-in for Feature-Oriented Software Development to create the feature configuration model defining the desired features in the new product being built. A feature configuration is a legal combination of features that specifies a particular product. We use a text-to-model transformation to obtain the model shown in Figure **??** ( part B),that illustrates the selected features as an instance of the metamodel shown in Figure 2.

During product design, the meta-transformation is used to generate from the feature-to-architecture transformation rule the Model Architecture artifact. The proposed model transformation approach takes as input the SPL source model and generates a product target model. Our transformation generating a concrete product model from a SPL model is implemented in (ATL). An ATL transformation is composed of a set of rules and helpers. The rules define the mapping between the source and target model, while the helpers are methods that can be called from different points in the ATL transformation. This transformation is then applied to the feature configuration model to automatically generate the product architecture. Here we show an example of an ATL helper and rules:

```
module MyRules; -- Module Template
create OUT: Components from IN: Features;
rule Componnent{
from
 e : Feature!Feature
to
 out : Component!Component (
          name <- e.name, )
}
rule Association{
from
 e : Feature!Dependency
to
 out : Component!Connector (
          name <- e.name, )
}
rule Attribute {
from
 e : Feature!Attribute
to
 out : Component!Attribute (
          name <- e.name,
          type <- e.type )
}
rule Port {
from
 e : Feature!Operation
to
 out : Component!Port (
          name <- e.name,
          type <- e.parameter->select(x|x.kind=#pdk_return)->
           asSequence()->first().type,
          parameters <- e.parameter->select(x|x.kind<>#pdk_return)->
          asSequence()
     )
}
```

As Figure 5 illustrates a fragment of the resulting PRODUCT ARCHITECTURE model generated by the rules, applied to the FEATURE CONFIGURATION MODEL shown in Figure 4. The e-Health product line application component is composed by the subcomponents generated by the rules. Final activity in application engineering is the application realization. At this stage,

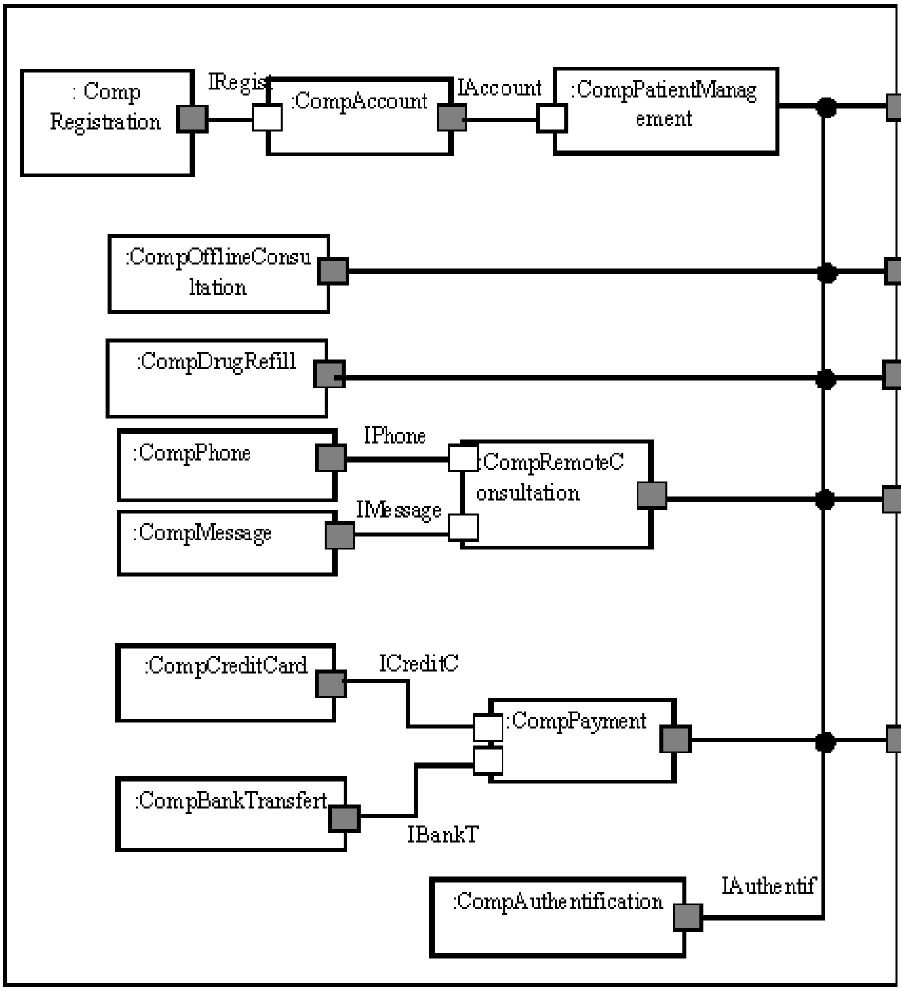


Figure 5: Component Model for e-Health product Line applications.

we write a program that generates Java code from our previously created architecture model using Acceleo, which navigates the model and creates the source code ( ∗.java files for Java). Our goal is to transform the features into java classes and Attribute into class properties, and finally generate set and get methods for class properties. here is the code used to create a bean for each of the classes defined in our target model:

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML')/]

[template public generate(aClass : Class)]
[file (aClass.name.concat('.java'), false)]
  public class [aClass.name.toUpperFirst()/] {
  [for (p: Property | aClass.attribute) separator('\n')]
    private [p.type.name/] [p.name/];
  [/for]

  [for (p: Property | aClass.attribute) separator('\n')]
    public [p.type.name/] get[p.name.toUpperFirst()/]() {
      return this.[p.name/];
    }
  [/for]

  [for (o: Operation | aClass.ownedOperation) separator('\n')]
    public [o.type.name/] [o.name/]() {

    }
  [/for]
  }
[/file]
[/template]
```

## 6 Conclusion

The main objective of a product line is reusability.Various assets are being used in software product lines. These assets have different values. Also, the values of them differ from the value of the profit obtained through using these assets is different. Derivation of a product from an SPL seems to be an easy step since its relied on reuse. Actually the product derivation represents one of the main challenges that SPL faces due to time-consuming. In this paper, we intended to reduce the development time of a product by automating the derivation by generating some java code using Acceleo in conjunction with ATL. The proposed transformation uses Feature-architecture mapping technique by instantiating the initial feature model, an instance of feature model is constructed according to customers requirements. Then, separate features into two kinds: common and variable. The main idea is to create for each feature a component or a set of components combined in a specific way. Linking these created components together based on the relationships among features in the feature model is the last step of our process. Although testing is of main importance in the context of product lines due to high reuse, in this paper we do not cover testing activities and it is one of the limitations of our proposed approach. This paper has illustrated by means of e-Health application the overall process, from the feature model to the final product. As future work, we will add more features to e-Health product line application and also intend to build a new set of components. A possibility is to apply our approach on other product line applications as e-Vote and also add testing activity to the approach.

## References

[1] Acceleo Project, [Online]. Available:https: //eclipse.org/acceleo. ⇒49

[2] ATL Project, [Online]. Available: http: //www.eclipse.org/atl/. ⇒48

[3] J. Bèzivin, G. Dup, F. Jouault, G. Pitette, & J. E.Rougui, First experiments with the ATL model transformation language: Transforming XSLT into XQuery. *2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture* Vol. 37. 2003 ⇒46

[4] G. Botterweck, K. Lee, S. Thiel, Automating product derivation in software product line engineering, *Software Engineering*, Kaiserslautern, 2009, pp. 177–182. ⇒45

[5] G.Botterweck, L. OBrien, & S. Thiel. Model-driven derivation of product architectures. *Proc. of the twenty-second IEEE/ACM international conference on Automated software engineering*, 2007, pp. 469–472. ⇒45

[6] P. Clements, L. Northrop. *Software Product Lines: Practices and Patterns.* The SEI series in software engineering. Addison–Wesley, Boston, 2002. ⇒46

[7] K. Czarnecki, S. Helsen, U. Eisenecker. Staged configuration using feature models. *Int. Conf. on Software Product Lines*, Springer Berlin Heidelberg, 2004, pp.266–283. ⇒46, 48

[8] S. Deelstra, M. Sinnema, J. Bosch, Product derivation in software product families: a case study. The Journal of Systems and Software, **74** (2005) 173–194. ⇒ 46

[9] Eclipse Modeling Framework, [Online]. Available: http://www.eclipse.org/emf/. ⇒49

[10] FeatureIDE,[Online] Available: `http://wwwiti.cs.unimagdeburg.de/iti_db/research/featureide/`. ⇒52

[11] J. Gonzàlez-Huerta, E. Insfran, S. Abrahao, J. D. McGregor, Architecture derivation in product line development through model transformations, *22nd Int. Conf. on Information Systems Development*, 2013, pp. 371–384. ⇒45

[12] S. Kent, Model driven engineering, *Int. Conf. on Integrated Formal Methods, Lecture Notes in Comp. Sci.*, **2335** (2002) pp. 286–298. ⇒47

[13] A. G. Kleppe, J. B. Warmer, W. Bast, *MDA Explained: the Model Driven Architecture: Practice and Promise*, Addison-Wesley Professional. 2003. ⇒47

[14] I. Kurtev. *Adaptability of model transformations*, PhD Thesis, University of Twente Research Information. ⇒47

[15] N. Lahiani, D. Bennouar, A software product line derivation process based on mapping features to architecture, *Proc. of the Int. Conf. on Advanced Communication Systems and Signal Processing ICOSI*, 2015. ⇒48

[16] J. McGregor, Goal-driven product derivation, Journal of Object Technology, **8** 5 (2009) 7–19. ⇒46

[17] L. Northrop, P. Clements, With F. Bachmann, J. Bergey, G. Chastek, S. Cohen, P. Donohoe, L. Jones, R. Krut, R. Little, J. McGregor, L. OBrien, *A framework for software product line practice, version 5.0*, Software Enginering Institut, 2012. ⇒47

[18] C. Parra, A. Cleve, X.Blanc, L. Duchien, Feature-based composition of software architectures. *European Conf. on Software Architecture*, Springer, Berlin, Heidelberg, 2010, pp. 230–245. ⇒45, 46

[19] D. Perovich , P. O. Rossel, M. C.Bastarrica, Feature model to product architectures: Applying MDE to software product lines, *Software Architecture, & European Conf. on Software Architecture*, WICSA/ECSA 2009, Joint Working IEEE/IFIP Conf., IEEE 2009, pp. 201–210. ⇒45, 46

[20] D. C. Schmidt. Guest editors introduction: model-driven engineering. *IEEE Comput.* **39** 2 (2006) 25–31. ⇒44

[21] T. Stahl, M. Voelter, K. Czarnecki, *Model-Driven Software Development: Technology, Engineering, Management*, John Wiley & Sons 2006. ⇒44

[22] R. Tawhid , D. C. Petriu. Product model derivation by model transformation in software product lines. *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 14th IEEE International Symposium*, IEEE 2011, pp. 72–79. ⇒45, 46

*Received: January 14, 2018 • Revised: April 5, 2018*

Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 58–72

DOI: 10.2478/ausi-2018-0004

# A survey on sentiment classification algorithms, challenges and applications

Muhammad Rizwan Rashid RANA
University Institute of Information Technology
Pir Mehr Ali Shah Arid Agriculture University
Rawalpindi, Pakistan
email: `rizwanrana315@gmail.com`

Asif NAWAZ
University Institute of Information Technology
Pir Mehr Ali Shah Arid Agriculture University
Rawalpindi, Pakistan
email: `asif.nawaz@uaar.edu.pk`

Javed IQBAL
Department of Computer Science
University of Engineering and Technology
Taxila, Pakistan
email: `javed1797@hotmail.com`

**Abstract.** Sentiment classification is the process of exploring sentiments, emotions, ideas and thoughts in the sentences which are expressed by the people. Sentiment classification allows us to judge the sentiments and feelings of the peoples by analyzing their reviews, social media comments etc. about all the aspects. Machine learning techniques and Lexicon based techniques are being mostly used in sentiment classification to predict sentiments from customers reviews and comments. Machine learning techniques includes several learning algorithms to judge the sentiments i.e Navie bayes, support vector machines etc whereas Lexicon Based techniques includes SentiWordnet, Wordnet etc. The main target of this survey is to give nearly full image of sentiment classification techniques. Survey paper provides the comprehensive overview of recent and past research on sentiment classification and provides excellent research queries and approaches for future aspects

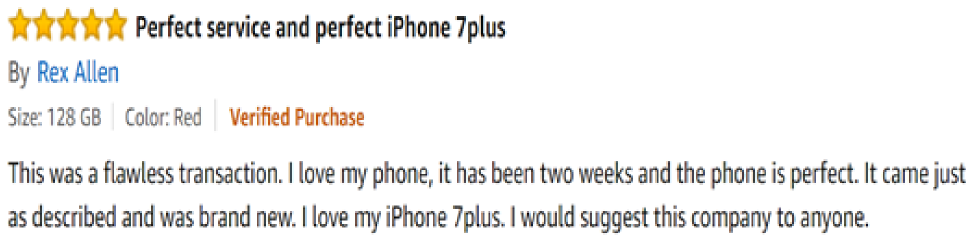
**Computing Classification System 1998:** I.2.6, K.3.0, K.3.m
**Mathematics Subject Classification 2010:** 68R15
**Key words and phrases:** sentiment classification, supervised learning, opinion mining, machine learning, lexicon approaches, unsupervised learning

## 1 Introduction

The fast growth of World Wide Web (WWW) is constantly increasing the online communication. A huge number of customers reviews or suggestions on everything are present on the web and these customers reviews and suggestions are increasing day by day. Micro blogging like twitter, Facebook etc. are powerful channels for peoples sentiments, thoughts, ideas and feelings. It is now estimated that, in just 60 seconds, over 400,000 Twitter posts are being shared, about 300,000 Facebook statuses updates, about 25,000 items purchased from Amazon, over 5million YouTube videos viewed and about 2.7 million Google searches are being made among many other things [1] . Users rating and reviews, which have been found on many ecommerce and market websites is a good source that helps peoples to build opinion about specific products. As a result of this phenomenon, increasing numbers of opinions and thoughts are being spread and published over the internet

Before the rise of internet to answer the question of What do people think about any product or anything else, Surveys and polls are distributed in the form of paper in peoples. With the expeditious development of internet and the popularity of Micro-blogging sites like Facebook, Twitter etc enables an alternative option for getting opinions from large population [38]. Now a days Web become the necessity for people to share their ideas, experiences and opinions as well as seeking others experiences and opinions [47]. Millions of ideas and experiences are shared every day. It is impossible for peoples to read all ideas and experiences. About 2.7 million Google searches was made. A query Artificial Intelligence returns 98,400,000 results. This whole scenario demands fast, effective and accurate technique to track sentiments, opinions and ideas that are flowing on internet. Sentiment classification is the key component of such techniques [4, 41, 11].

Sentiments represent the viewpoint of customer such as like (positive), dislike (negative) and may be neutral viewpoint [7]. Sentiment classification also called opinion mining is the process to automatically determine the sentiment category to which the textual content belongs [46]. We can categorize the reviews, comments and document mainly in two types. These are numeric sentiment and categorical sentiment. Common example of a numeric sentiment is rating system in ecommerce sites. Using this rating system company judge the response of peoples. Categorical sentiment is a method to classify the comments or reviews in different categories. These categories are binary (positive and negative), ternary (positive, negative and neutral) and multiple categories (Anger, happy, sad etc.) [40]. Opinions are become the necessary parts in all human activities. There are two types of reviewing sites, generic reviewing sites and specified reviewing sites. Generic reviewing sites include sites like amazon.com, epinions.com, rottentomatoes.com etc and Specified reviewing sites includes tripadvisor.com, yelp.com. Both of these reviewing sites have a significant effect in our decision meaning process. These decisions includes buying a camera, smart phone etc, making investments on any product etc, choosing schools, decision about any movie etc etc. Before the Internet, Other sources such as friends, relatives etc affect the human decision process. Positive review is showing in Figure 1 and negative review is showing in Figure 2.

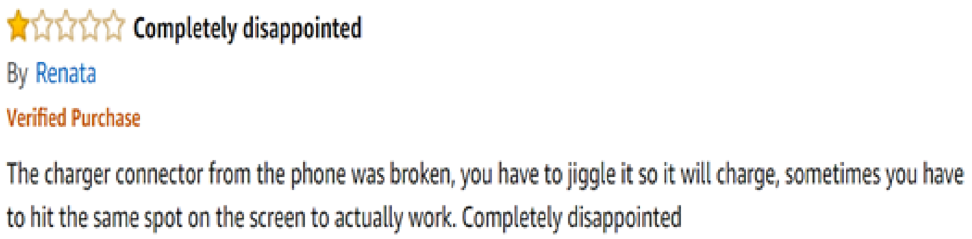

★★★★★ Perfect service and perfect iPhone 7plus

By Rex Allen

Size: 128 GB | Color: Red | Verified Purchase

This was a flawless transaction. I love my phone, it has been two weeks and the phone is perfect. It came just as described and was brand new. I love my iPhone 7plus. I would suggest this company to anyone.

Figure 1: Positive review


★☆☆☆☆ Completely disappointed

By Renata

Verified Purchase

The charger connector from the phone was broken, you have to jiggle it so it will charge, sometimes you have to hit the same spot on the screen to actually work. Completely disappointed

Figure 2: Negative review

The meaning of word sentiment itself is still very wide. Opinion mining mostly focuses on opinions which communicate or involve positive or negative sentiments from reviews, comments etc. These user reviews are very useful to organizations for making intelligent decisions about product purchasing and also helpful for merchants in knowing their products progress in the market

[29]. Nowadays everyone shares their views and experiences online. For example, if somebody wants to buy a mobile phone, such as the Samsung mobile, and he or she dont know about this mobile phone. He or she can use the internet, open mobile phone web site and read customer reviews about the product and then he or she can make a decision in the light of provided user reviews. This manual process is named as text mining, opinion mining or sentiment analysis. History calculation sentiment of document is the task of marketing team of any company. Humans have no trouble in reading the movie review, product review and any political comment and categorized it in positive, negative or neutral class. We humans use a technique of reading and understand the underlying meaning of a sentence but when there are a millions of reviews then its a time consuming task to read all reviews and categorized it in positive, negative or neutral class.

In the wake of digital age, thousands of movies are directed per year and peoples share millions of reviews and comments about these movies on the internet. Usually a movie reviews peoples share their comments about the movie. As it is the time consuming task so it is very hard for humans to judge the tone of these reviews and classify it in positive or negative category. There is a strong demand for automatically analyzing and summarizing the opinions expressed in natural language text. We can automatically analyze and classify the reviews using machine learning techniques. Sentiment classification can be helpful for customers who need to research the sentiment of product before purchase or companies that need to watch the general public sentiment of their brands.

Sentiment classification or opinion mining has been studied at three different levels of classification [28] these include sentiment classification at document level, sentence level and at aspect level. Classification of whole document in a positive or negative is called document level classification. Sentence-level sentiment classification techniques read document sentence by sentence and decide whether each sentence gives a positive, negative or neutral opinion for a service, product etc. Aspect level or entity level sentiment classification is the most modern technique which classifies the reviews or comments on the basis of aspects or entities

## 2 Literature review

In the last decade, lot of research work has been carried out in sentiment classification. Techniques of sentiment classification (i.e judging tone of the text) have been performed for a variety of applications over a wide range of classification algorithms. A Review of some existing techniques from literature is provided in the following section. There exists a four different techniques for sentiment classification as shown in Figure 3.

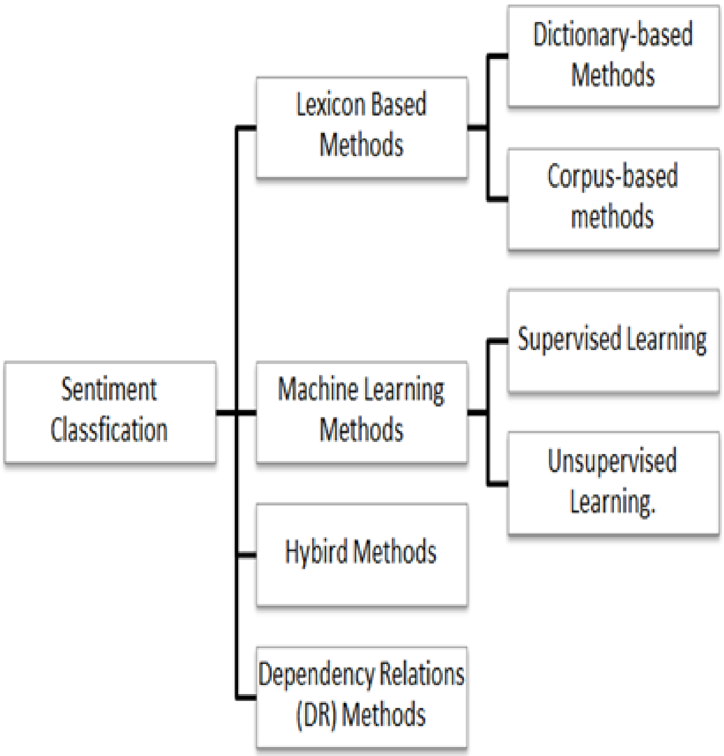


Figure 3: Sentiment classification techniques

## 2.1 Lexicon based methods

Lexicon based methods adopt a lexicon to perform aspect based sentiment analysis. These methods can work by counting, analyzing and weighting opinion words. These methods are further divided in two broad classes. They are dictionary-based methods and corpus-based methods

### 2.1.1 Dictionary based methods

Dictionary- based methods used the lexicon database to judge the tone of text. Popular lexicon databases are WordNet, Sentiwordnet etc. WordNet groups the words into synsets (synonym sets) and the semantic relation between synsets [32]. WordNet is used on adjectives in order to find their semantic orientation [20]. Process first count the number of synonym links for adjectives such as bad,good etc. Another paper used the WordNet to create semantic lexicon. They used the antonym relation of adjective and WordNet synonym [10]. This idea is used for constructing another lexicon named as SentiWordNet. SentiWordNet provides the three types of sentiment scores of each words. These types are positive score, negative score and objective score. Another paper uses the three different lexical relations in WordNet [2]. These lexical relations consists on antonymy, hyponymy and synonymy). It takes the adjective from epinions.com reviews and mapped to the star rating. Paper proposed method uses the breadth first search adjective on WordNet synonymy graph with unknown sentiment and then distance-weighted nearest neighbor algorithm is to calculate the weights of two average rating of two nearest neighbor as related adjective. Different bootstrapping method using WordNet is proposed in [43]. Algorithms take the known sentiment orientations as input and generate the set of synonyms (Synsets) as output. The new generated syssets are then used to calculate the polarities of words. Constrained symmetric nonnegative matrix factorization (CSNMF) technique with sentiment lexicon generation is used in [14]. Proposed method words on two steps. In first step dictionary is used to find the candidate sentiment and in second step corpus is used to assign the polarity score to each word.

### 2.1.2 Corpus based methods

One of the earliest ideas that use the Corpus-based method was presented by Hazivassiloglou and McKeown [13]. Proposed idea used the seed adjective and corpus to find other sentiment adjectives in corpus. This technique also used some linguistic rules. One of the rule is about conjunction AND. According to this rule where ever conjoined adjective comes, they have same orientation. For example if there is a sentence Today I am happy and delighted Here if happy is a positive then definitely delighted is also a positive. Basic idea behind conjunction AND rule is peoples always express same sentiments on both side of AND. Other rules are OR, BUT, NEITHERNOR etc.

Above approach was extended by introducing call coherency [21]. Call coherency includes intra-sentential consistency and inter-sentential consistency. Intra- sentential consistency exists within sentence and inter-sentential consistencies exist between neighboring sentences. This technique is used to find the domain dependent sentiment words. Later this technique was also used in [19]. There are many words in same domain have different orientations in different way of writing [9]. Same word is uses as positive in one context and in second context it will be used as negative. For example in mobile domain lets takes two reviews. First review is Mobile have long battery life and second review It takes long time to open contacts. In first review long is used in positive sense and in second review long is used in negative sense. Author presents the solution to solve this problem. First find the aspects and sentiment words or opinion words from text then use both aspects and sentiment word in pair like (aspect, sentiment word). For example (contacts,long). To predict which pair is positive and which is negative, the call con coherency will also used.

In 2011, authors argue the technique to study the lengthening of words (e.g thanksssssssssss) in social media sites [5]. Usually in comments and tweets are many lenghty words present. According to authors these lengthy words shows the high sentiments in comments and proposed a automatic the technique for finding the sentiments. Connotation lexicon is very much changed from simple sentiment lexicon [12]. Using Connotation lexicon paper achieved the better results as compare to other lexicons

## 2.2 Machine learning

Machine learning is further divided in Supervised Learning and Unsupervised Learning. In supervised learning, output dataset is necessary, we train algorithm on output dataset and get the desired outputs whereas in unsupervised learning we dont have any output datasets, instead the data is clustered into different classes [45].

### 2.2.1 Supervised learning

Pang et al. applied Support Vector Machine, Navie Bayes and Maximum entropy with different feature extracting techniques on movie reviews [34]. Experimental results shows the SVM have best performance with unigram text representation. It has been noted that without POS tagging information accuracy of naive bases and maximum entropy increases but it decreases the performance of SVM. Liu et al. argued the sentiment classification system that uses Nave Bayes Classifier and Map Reduce framework [26]. Paper uses machine leaning algorithm Nave Bayes Classifier to classify the sentiments in two classes positive and negative. Paper also uses Map Reduce framework with Naive Bayes Classifier to get better results. Map Reduce framework usually use to analyze extremely large datasets such as tweets collections, movie reviews etc. Experimental results show the accuracy of Naive Bayes classifier on large data sets is 82%.

Dhande and Patnaik uses neural network with Naive Bayes classifier because in many complex real world situations Naive Bayes cannot work well [8]. An experimental result shows that, when we are using Naive Bayes Classifier then Accuracy will be 62.35. So for getting better results Paper uses the Neural Network with Naive Bayes Classifier. Results show that accuracy of sentimental analysis increased up to 80.65 by combining Neural Network with Naive Bayes Classifier. Shaziya et al. takes the dataset of movie reviews and apply two well knows classifier on this dataset. These classifiers are Naive Bayes classifier (NB) and Support Vector Machine (SVM) [37]. The dataset is preprocessed and various filters have been applied to reduce the feature set. Papers use the Feature selection method for getting most valuable words and use Information Gain, and Gain Ratio methods for getting distinctive word. Using these methods we get the most value data from dataset. An experimental result shows that Navies Bayes results are better than SVM results. Accuracy of Naive Bayes classifier is 86.1% for positive reviews and 84.1% for negative reviews and accuracy for SVM classifier is 84.7% for positive reviews and 84.4% for negative reviews.Support Vector Machine (SVM) is more efficient classifier than Naive Bayes in many cases.

Manek et al. adopted SVM classifier with Gini Index feature selection method for sentiment classification for large movie review data set [28]. Experimental results show that Gini Index feature selection method is better in terms of accuracy and performance. Paper achieves the accuracy 78% using SVM classifier on large dataset of movie reviews. Paper [15] implement the three sentiment analysis algorithms for identifying the sentiments (positive or negative) from reviews. Experiential results are then compared with the numerical ratings of hotels. Dataset of One million reviews with numerical rating is collected from Tripadvisor. Results shows that predicted rating from sentiment analysis algorithms are very close to actual ratings of the hostel. Sentiment classification using Bayesian Classifier was implemented in [36]. Experimental results show that bayesian classifier works well on large dataset as well as small dataset.

### 2.2.2 Unsupervised learning

Unsupervised learning is the second type of Machine Learning. They proposed a technique to find the potential sentiment phrase by explicit aspects in its surrounding. Surrounding of any aspect is measured using syntactic dependencies. All potential sentiment phrases are examined and the phrase which shows positive or negative sentiments is retained. Semantic orientation and polarity is calculated by unsupervised technique. Unsupervised technique that is used are named as relaxation labeling. In paper [30], the probabilistic latent semantic indexing (pLSI) [17] was used to develop Topic-Sentiment Mixture (TSM) model which reveal latent topics including sentiment classes as additional topics. The dynamic nature of social media data whereby sentiments and topics constantly change means that sentiment/topic models also need to be updated over time. This is addressed by the dynamic JST [24] which captures both topic and sentiment dynamics by assuming that the current sentiment-topic specific word distributions are generated according to the word distributions.

Consequently the Dependency Sentiment-LDA model, which relaxes the sentiment independent assumption, was introduced by Li et al [23]. In this model the sentiments of the words in a document are viewed to form a Markov chain, where the sentiment of a word is dependent on the previous one. Although topic modeling approaches to sentiment classification do not require labeled data, they still rely on sentiment lexicons as the source of prior sentiment knowledge. Like with purely lexicon-based methods, their performance was shown to be dependent on both the coverage and quality of the lexicons used by Lin and He [25]. However, the lexicon-based methods offer greater flexibility to incorporate linguistically derived contextual knowledge making for a more transparent and accessible approach to sentiment classification.

## 2.3 Hybird methods

Sentiment classification was also observed to improve when multiple classifiers, formed from machine learning and lexicon-based methods, are used to classify a document [35]. The hybrid method also helps overcome certain limitations of the combined methods. For instance, in a system called P Senti lexicon knowledge was used to filter out non sentiment-bearing words from the feature set subsequently used for machine learning [22].

Evaluation of P Senti shows the hybrid approach achieved better performance compared to pure lexicon-based and better cross-domain portability compared to pure machine learning. In another work, a small amount of training data for machine learning was compensated with lexicon knowledge [31]. In some other work, machine learning was applied to optimize sentiment scores prior to lexicon based sentiment classification [42]. This approach has the tendency to produce domain adapted lexicons which in turn improve sentiment classification. It is noteworthy, however, that although the hybrid approach can help overcome certain limitations of either of the combined methods (lexicon-based or machine learning) alone, it can also combine challenges from both methods. For instance, it often requires both labeled data, which can be difficult to obtain, as well as a sentiment lexicon.

### 2.4 Dependency relationship (DR) techniques

DR can be used to generalize the changing relationship of opinion words and aspects. Paper [3] enlisted the DR to get paired aspect-opinion by using movie reviews. By using dependency relationship parser, the parsed words in a sentence are joined by definite dependency relationship. By using dependency sequence, encouraging results in various research fields by employing distinct approaches to point product features and their kindred point of view from various language reviews. Different feature selection techniques have been used besides with machine learning approaches like bigrams and unigrams [34]. Paper [16] deployed syntactic relations between words in different sentnces for the organization of document sentiments. Agarwal et al. (2015) used ConceptNet ontologybased dependency relations to extract features from text. They also used a method called mRMR which is basically a feature selection scheme to remove redundant information. Paper [39] proposed a technique that retrieves product aspects and opinions by taking signifies and linguistics information based on dependency relationship.

## 3 Applications

Sentiment classification is a large field that contains the vast range of applications being discussed in past research. In last decade, a lot of research has been done to examine the influence of media on the business world. The Internet has turned into a vast source of all kind of knowledge for everybody. Using internet there is an opportunity to discover the perspectives of people in general about organization methodologies, political developments, business world etc. In short, numerous applications of sentiment classification have emerged in domains of daily life like sentiment classification for the business world, political reviews, movie reviews etc. Some of these applications are OpinionMiner [27], Opinion observer [18] and OpinionFinder [44].

There are hundreds of companies that develop sentiment classification tools for their themselves and for their clients. These companies include Oracle, Azamon, Google, Hawlett-Packart, SAS and Facebook. Some small companies also build sentiment classification tool for theor clients. They are Lexalytics, Semantria, Synapsify, ThriveMetrics, Etuma and MeshLabs. Facebook's Gross National Happiness application developed by facebook to predict the happiness of peoples on facebook, by countries. This application works by checking positive and negative words from peoples statuses [33]. For political parties tracking sentiments of peoples are very important. A site named as Sentex.com tracks sentiments of political parties and political topics and provides sentiment classification in ternary category (positive, negative and neutral) [6].

## 4 Challenges

There are several challenges in judging sentiments form reviews, comments etc. Usually in reviews there is inconsistent and erratic data. People have various ways of expressing sentiments; sometime they use shorthand and lots of abbreviations. Usually they cannot use proper grammar in reviews. We judge positive or negative opinions from reviews using opinion words and phrases are usually used to express. These phrases and opinion words may be used in positive and negative situations. For example good is for positive and not good is for negative. Judgment of positive and negative sentiments from review depends on context of what is around it. There are very less words that will always attach a positive or negative sentiment to an expression. Comments and reviews also contain irony and hidden emotions. The task of judging sentiments is also a challenging task, due subjective sentences and also ambiguity naturally found in opinionated text. Ambiguity words are the same meaning words which come in more than one time in same sentence. Ambiguity becomes a serious problem when it come with irony and convey words. Take for example the sentence A great mobile, yeah right!this may look like a positive review, but it may be taken as a negative review. One of the major issue in Lexicon based approaches is need of lexicons for other languages. Lexicons are only available for some popular languages like English, Arabic, Chinese etc but for unpopular languages there is no lexicons available. Also lexicons of Arabic, Chinese languages are limited in term of words, they cannot cover all words of these languages

## 5 Conclusion

Sentiment classification comes forth as a challenging field with lots of fence as it involves natural language processing and hidden emotions. It has a wide variety of applications that could benefit from its results, such as movie reviews, product reviews, news analytics, and marketing, question answering, knowledge bases and so on. There are various areas in sentiment classification field where lots of improvement is needed with existing techniques. This survey gives a brief insight about sentiment classification, types of sentiment classification and comparison of existing techniques. The interest of peoples in languages other than English in this sentiment classification is growing day by day as there is still a lack of resources and researches concerning these languages. Building resources, used in sentiment classification tasks, is still needed for many natural languages. Survey also highlights some major challenges about judging sentiments and future work is to overcome these challenges to get better results

## References

[1] M. Aminu, *Contextual lexicon-based sentiment analysis for social media*, PhD Thesis, Université Robert Gordon University, Aberdeen, 2016. ⇒59

[2] A. Andreevskaia, S. Bergler, When specialists and generalists work togethe overcoming domain dependence in sentiment tagging, *Proc. ACL08 HLT*, 2008, pp.290–298. ⇒63

[3] A. Andronic, F. Arleo, R. Arnaldi, A. Beraudo, E. Bruna, D. Caffarri, Z. Conesa del Valle et al., Heavy-flavour and quarkonium production in the lhc era: from protonproton to heavy-ion collisions, *The European Physical Journal* (2016) **76**: 107. ⇒67

[4] R. N. Behera, R. Manan, S. Dash, Ensemble based hybrid machine learning approach for sentiment classification –A Review, *International Journal of Computer Applications* **146**, 6 (2016) 31–36. ⇒59

[5] S. Brody, N. Diakopoulos, Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs, *Conference on Empirical Methods in Natural Language Processing*, 2007, pp. 562–570. ⇒64

[6] Y. Choi, H.Lee, Data properties and the performance of sentiment classification for electronic commerce applications, *Information Systems Frontiers* **19** (2017) 1–20. ⇒68

[7] K. T. Devendra, S. K. Yadav, Fast retrieval approach of sentimental analysis with implementation of bloom filter on Hadoop, *International Conference on Computational Techniques in Information and Communication Technologies*, 2016, pp.529–551. ⇒59

[8] L. Dey, S. Chakraborty, A. Beepa, S. Tiwari, Sentiment analysis of review datasets Using nave bayes and k-nn classifier, *Information Engineering and Electronic Business* (2016) 54–62. ⇒65

[9] X. Ding, B. Liu, P. S. Yu, A holistic lexicon-based approach to opinion mining, *Proc. of the 2008 international conference on web search and data mining, ACM*, 2008, pp. 231–240. ⇒64

[10] A. Esuli, F. Sebastiani, Determining term subjectivity and term orientation for opinion mining, *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006. ⇒63

[11] Y. Fei, Simultaneous support vector selection and parameter optimization using support vector machines for sentiment classification, *Software Engineering and Service Science (ICSESS), 7th IEEE Int. Conference*, 2016, pp. 59–62. ⇒59

[12] S. Feng, R. Bose, Y. Choi, Connotation lexicon: A dash of sentiment beneath the surface meaning, *Proc. 51st Annual Meeting of the Association for Computational Linguistics*, 2013, pp- 1774–1784. ⇒64

[13] V. Hatzivassiloglou, K. R. McKeown, Predicting the semantic orientation of adjectives, *Proc. 35th Annual Meeting of the Association for Computational Linguistics*, 1997, pp. 174–181. ⇒63

[14] W. Haywood, J. Ricky, J. B. Holcomb, E. A. Gonzalez, Z. Peng, S. Pati, P. W. Park, W. Wang, A. M. Zaske, T. Menge, R. A. Kozar, Modulation of syndecan-1 shedding after hemorrhagic shock and resuscitation, *PloS*, 2011. ⇒63

[15] W. He, X. Tian, R. Tao, W. Zhang, G. Yan, V. Akula, Application of social media analytics: a case of analyzing online hotel reviews, *Online Information Review* (2017) 921–935. ⇒65

[16] D. T. Hess, Akio Matsumoto, Sung-Oog Kim, H. E. Marshall, J. S. Stamler, Protein s-nitrosylation: purview and parameters, *Nature Reviews Molecular Cell Biology* (2005). ⇒67

[17] T. Hofmann, Probabilistic latent semantic indexing, *Proc. 16th Int. Conference on World Wide Web*, 1999, pp. 50–57. ⇒66

[18] W. Jin, H. H. Ho, R. K. Srihari, Opinionminer: a novel machine learning system for web opinion mining and extraction, *Proc. 15th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2009, pp. 1195–1204. ⇒67

[19] N. Kaji, M. Kitsuregawa, Building lexicon for sentiment analysis from massive collection of html documents, *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007. ⇒64

[20] J. Kamps, M. Marx, R. J. Mokken, M. D. Rijkel, Using wordnet to measure semantic orientations of adjectives, *LREC*, 2004, pp. 1115–1118. ⇒63

[21] H. Kanayama, T. Nasukawa, Fully automatic lexicon expansion for domain-oriented sentiment analysis, *Proc. 2006 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics*, 2006, pp. 355–363. ⇒64

[22] A. Mudinas, D. Zhang, M. Levene, Combining lexicon and learning based approaches for concept-level sentiment analysis, *Proc. of the First Int. Workshop on Issues of Sentiment Discovery and Opinion Mining*, 2012, pp. 51–58. ⇒66

[23] S. Li, C. R. Huang, G. Zhou, S. Y. M. Lee, Employing personal/impersonal views in supervised and semi-supervised sentiment classification, *Proc. 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 414–423. ⇒66

[24] F. Li, M. Huang, X. Zhu, Sentiment analysis with global topics and local dependency, *Association for the Advancement of Artificial Intelligence* **10** (2010) 1371–1376. ⇒66

[25] C. Lin, Y. H. Lee, Joint sentiment/topic model for sentiment analysis, *Proc. 18th ACM Conference on Information and Knowledge Management*, 2009, pp. 375–384. ⇒66

[26] B. Liu, E. Blasch, Y. Chen, D. Shen, G. Chen, Scalable sentiment classification for big data analysis using naive bayes classifier, *IEEE Int. Conference on Big Data*, 2013, pp. 99–104. ⇒65

[27] B. Liu, H. Minqing, C. Junsheng, Opinion observer: analyzing and comparing opinions on the Web, *Proc. 14th Int. Conference on World Wide Web*, 2005, pp. 342–351. ⇒67

[28] A. Manek, P. Deepa, C. Mohan, K.Venugopal, Aspect term extraction for sentiment analysis in large movie reviews using gini index feature selection method and svm classifier, *World wide web* **20** (2016). ⇒61, 65

[29] C. Mate, Product aspect ranking using sentiment analysis: a survey, *, Int. Research Journal of Engineering and Technology* **(**2014). ⇒61

[30] Q. Mei, X. Ling, M. Wondra, H. Su, C. Zhai, Topic sentiment mixture: modeling facts and opinions in weblogs, *Proc. 16th Int. Conference on World Wide Web*, 2007, pp. 171–180. ⇒66

[31] P. Melville, W. Gryc, R. D. Lawrence, Sentiment analysis of blogs by combining lexical knowledge with text classication, *Proc. 15th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2012, pp. 163–173. ⇒67

[32] G. M. Miller, Wordnet: a lexical database for english, *Communications of the Association for Computing Machinery* (1995) 39–41. ⇒63

[33] A. Ortigosa, J. Martin, R. Carro, Sentiment analysis in facebook and its application to e-learning, *Computers in Human Behavior* **31** (2014) 527–541. ⇒68

[34] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up?: sentiment classification using machine learning techniques, *Proc. ACL-02 Conference on Empirical Methods in Natural Language Processing*, 2002, pp. 79–86. ⇒64, 67

[35] R. Prabowo, M. Thelwal, Sentiment analysis: a combined approach, *Journal of Informetrics* **3** (2009) 143–157. ⇒66

[36] M. R. R. Rana, M. A. Akbar, T. Ahmad, Sentiment classification of customer reviews using bayesian classifier, *Asian Journal of Engineering, Sciences & Technology* **7** (2017). ⇒65

[37] H. Shaziya, G. Kavitha, R. Zaheer, Text categorization of movie reviews for sentiment analysis, *Int. Journal of Innovative Research in Science, Engineering and Technology* (2015) 11255–11262. ⇒65

[38] S. P. Sivasubramanian, N. Suganya, Sentiment analysis on micro-blogs, *Int. Innovative Research Journal of Engineering and Technology* **2** (2017) 46–51. ⇒ 59

[39] G. Somprasertsri, P. Lalitrojwong, Mining feature-opinion in online customer reviews for opinion summarization, *J. UCS* **16** (2010) 938–955. ⇒67

[40] J. Steinberger, T. Brychcin, M. Konkol, Sentiment and social media analysis, *Proc. 5th Workshop on Computational Approaches to Subjectivity*, 2014. ⇒59

[41] B. N. Supriya, V. Kallimani, S. Prakash, C. B. Akki, Twitter sentiment analysis using binary classification technique, *Int. Conference on Nature of Computation and Communication*, 2016, pp. 391–396. ⇒59

[42] M. Thelwall, K. Buckley, G. Paltoglou, Sentiment strength detection for the social web, *Journal of the Association for Information Science and Technology* **63** (2012) 163–173. ⇒67

[43] P. D. Turney, M. L. Littman, Measuring praise and criticism: inference of semantic orientation from association, *ACM Transactions on Information Systems (TOIS)* (2003) 315–346. ⇒63

[44] T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, Opinionfinder: a system for subjectivity analysis, *HLT-Demo '05 Proc. of HLT/EMNLP on Interactive Demonstrations*, 2005, pp. 34–35. ⇒67

[45] Q. Ye, Z. Zhang, R. Law, Sentiment classification of online reviews to travel destinations by supervised machine learning approaches, *Expert systems with applications* **36** (2009) 6527–6535. ⇒64

[46] N. Zainuddin, A. Selamat, V. Kekan, Sentiment analysis using support vector machine, *Computer, Communications, and Control Technology (I4CT), 2014 Int. Conference*, 2014, pp. 333–337. ⇒59

[47] A. Zubiaga, I. S. Vicente, P. Gamallo, J. R. P. Campos, I. A. Loinaz, N. Aranberri, A. Ezeiza, V. F. Fernandez, Overview of tweetlid: tweet language identification, *TweetLID SEPLN*, 2014, pp. 1–11. ⇒59

*Received: May 30, 2018 • Revised: July 15, 2018*

Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 73–85

DOI: 10.2478/ausi-2018-0005

# Exact fit problem generator for cutting and packing, revisiting of the upper deck placement algorithm

Levente FILEP
Sapientia University
Department of Economic Sciences
Miercurea Ciuc
email:
`fileplevente@uni.sapientia.ro`

László ILLYÉS
Sapientia University
Department of Economic Sciences
Miercurea Ciuc
email:
`illyeslaszlo@uni.sapientia.ro`

**Abstract.** Problem generators are practical solutions for generating a set of inputs to specific problems. These inputs are widely used for testing, comparing and optimizing placement algorithms. The problem generator presented in this paper fills the gap in the area of 2D Cutting & Packing as the sum of the area of the small objects is equal to the area of the Large Object and has at least one perfect solution. In this paper, the already proposed Upper Deck algorithm is revisited and used to test the proposed generator outputs. This algorithm bypasses the dead area problem that occurs in most of all well-known strategies of the 2D Single Knapsack Problem where we have a single large rectangle to cover with small, heterogeneous rectangle shapes, whom total area exceeds the large object's area. The idea of placing the small shapes in a free corner simplifies and speeds the placement algorithm as only the available angles are checked for possible placements, and collision detection only requires the checking of corners and edges of the placed shape. Since the proposed generator output has at least one exact solution, a series of optimization performed on the algorithm is also presented.

**Computing Classification System 1998:** I.2.8. [Problem Solving, Control Methods, and Search]: Subtopic - Heuristic methods
**Mathematics Subject Classification 2010:** 68T20, 90B40
**Key words and phrases:** Cutting stock problem, problem generator, genetic algorithm

## 1 Introduction

Cutting and Packing (C&P) problems are of great interest in operation research. From storage filling to memory allocation, the knapsack problem appears in many forms. As these problems are NP-complete (non-deterministic polynomial time), meta-heuristic approaches, such as Genetic Algorithms (GA), are popular as they provide good-enough solutions.

Problem generators are seen as a tool to overcome limitations imposed by the absence of appropriate test problems [14] in the field of C&P. The problem generator proposed and presented in this paper aims to fill the gap in the area by providing problems with at least one exact solution which, besides benchmarking, can be used to improve upon existing algorithms.

The upper deck placement GA, which was presented by one of the authors in the 2nd ESICUP meeting [6], is revisited in this paper and improvement attempts are made using the output of the proposed problem generator.

### 1.1 Problem generators

There are many problem generators developed for C&P. For our problem of two-dimensional rectangle cutting problem, the existing generator is the 2DCPackGen [14] developed by Silva, Oliveira and Wsher, downloadable from [17]. Since each of these generates shapes whose summed areas exceeds the surface area there is no best-known solution to these. While this is not a problem, since the aim of meta-heuristic algorithms is to provide a good enough solution for this kind of problems, it is a problem if we need to know of better solutions to further optimize the placing algorithms. Using brute force to try out all possible combinations to obtain the best possible solution its not a practical option. On the 2ndESICUP meeting, Alvarez-Valdez [1] defined the problems with perfect solutions as Jigsaw problems. T. Inamichi, et al. [9] proposed algorithms for perfect packing problems. Mumford and Wang [13] uses a perfect guillotine-cut example to check the performance of their algorithms. The first approach to the Jigsaw generator problem was from Illys and Fbin [7], but it was not implemented. This paper presents such an exact fit or in other words a perfect matching 2D, non-guillotine cutting stock problem generator.

### 1.2 Cutting stock problems

According to the typology of cutting stock problems [16], our work falls into the two-dimensional Single Knapsack Problem (2D-SKP) category. Previous algorithms to solve this problem are the Bottom Left algorithm (BL) [10], the improved BL algorithm [12], the BL Fill algorithm [4]. There are other, newer algorithms proposed and presented, but these are the first best approaches.

## 2 Proposed problem generator

In contrast to existing generators which generate shapes with a given distribution, usually gamma, until the combined area of these exceeds the cutting stock area, our generator starts with the cutting stock area and uses expanding rectangles placed at random locations to achieve complete coverage of this area. As opposed to other methods of randomly cutting an area into smaller pieces, the presented method was chosen due to ease of implementation in an arbitrary programming language.

For easy use, the generator is provided a graphical interface where the input parameters can be set: the width and height of the cutting stock area, the initial seed used for the pseudo-random number generator, the minimum and maximum shape extensions (in other words the growth of the shapes) and a cutoff ratio for random shape location. The width and height determine the unit square size of the stock cutting area, where each generated shape size is a multiple of this unit area.

The generation process starts with a given size area represented as a $W \times H$ (width, height) grid on which small shapes are placed and expanded in each step. These steps are repeated by the generator until these shapes cover the entire surface area.

Each step begins by randomly selecting an $X$ and $Y$ location on the grid. If the chosen coordinates are outside of any existing shape, in other words, it's a free cell, then a new $1 \times 1$ size shape is created at the location and marked for the next operation. If the coordinates are inside an existing shape then, that shape is marked.

Using the previously marked shape, if possible, this is expanded in a random direction: up, down, left or right. The number of expansion attempts of the shape is dictated by the input parameters.

It is trivial to observe that as the number of free unit squares decreases it's harder to hit any new unoccupied cells by generating random $X, Y$ locations, consequently the algorithm's progression is slowing down. The $\mathsf{FaCR}$ cutoff ratio represents the ratio between the free and total cells after which instead of choosing random locations, the generator selects one from the remaining free cells, marks it as a new shape and expands this until possible. This operation is repeated until all cells are occupied by a shape, thus completing the generation process and achieving a perfect matching problem. This measure is needed to ensure that the generation process ends in a timely manner. From experimental results a $0.1 - 0.25$ cutoff value is ideal. Figure 1 illustrates some of the steps and the end result.

**Algorithm 1:** Pseudo-code for problem generator

```
Data: Input parameters:
        $W$,$H$: width and height of the cutting stock area
        $FaCR$: Free area Cutoff Ratio
        $minEs$, $maxEs$: minimum and maximum number of shape
        expansion
Result: shape list
begin
    $TotalArea \leftarrow W * H$;
    $FreeArea \leftarrow 0$;
    while $(FreeArea/TotalArea) > FaCR$ do
        Generate random $X, Y$ coordinates // 0 ⩽ X ⩽ W, 0 ⩽ Y ⩽ H
        if $Grid[X, Y]$ is empty then
            Generate new $1 \times 1$ $Shape[i]$ at $Grid[X, Y]$ location;
            Mark $Grid[X, Y]$ as part of $Shape[i]$;
        $Es \leftarrow Random()$ // $minEs \leqslant Es \leqslant maxEs$
        Expand $Shape[i]$ $Es$ number of times;
        Recalculate $FreeArea$;
    while FreeArea ¿ 0 do
        Get $X, Y$ of next free unit square;
        Generate new $1 \times 1$ $Shape[i]$ at $Grid[X, Y]$ location;
        $Es \leftarrow Random()$ // $minEs \leqslant Es \leqslant maxEs$
        Expand $Shape[i]$ $Es$ number of times;
        Recalculate $FreeArea$;
```

## 3 Upper deck placement genetic algorithm

Placement algorithms are used to place small objects, known as shapes in our case rectangles on a Large Object, known as cutting stock in our case a rectangle too. The sides of the small pieces are parallel to the stock plate, the pieces do not overlap each other and do not exceed the dimension of the stock plate.

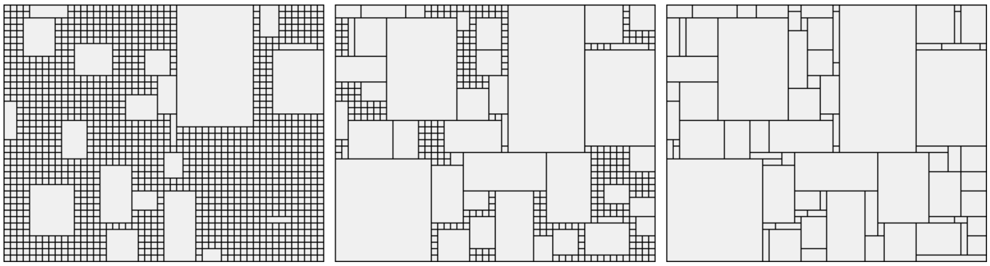


Figure 1: Shape generation process, early, mid and final stage

The genetic algorithm version of the upper deck algorithm was proposed at a previous conference [6], therefore a full version of this is not described here. Instead, we reflect on the differentiating parts from a conventional placement GA, namely the shape placement, angle generation part and the genetic operators of this. Before this paper, only the little objects order of placement was coded in permutation chromosomes. The rotation possibility was also addressed by Hopper and Turton in [3] but was not encoded in the gene, while the deck was not addressed at all.

Each input shapes is provided with a unique identification id. Even if two shapes are identical they are given separate ids and treated as distinct. This sacrifices some memory for computational speed gains since avoids the necessity of counting and checking the correct number of placed shapes against the input at each genetic operation.

## 3.1 Chromosome structure

Each gene in the chromosome represents a shape's placement properties. These include: unique identification `id`, a set of Boolean value indicating whether the shape is placed or not at the stored angle number and its rotation.

## 3.2 Heuristic

The strong point of the algorithm consists of the utilized heuristic. The algorithm maintains a list of the available angles where the next shape can be placed. This significantly lowers the number of possibilities needed to be checked for a possible shape placement, thus increasing the computational efficiency of the algorithm. The notation of the available angles is in the order of appearance and clockwise as shown in Figure 2. As new shapes are placed, depending on the placement, the number of angles can increase or decrease, but the notation is always maintained.

By definition [6], the upper angles are defined as the angle created by the intersection of either two of the cutting stock area walls, the edges of previously placed shapes or a combination of these. Therefore these are the concave angles. This definition has practical reasons behind it. As an example let's consider the filling a storage area that has a single entrance, where placing crates in concave angles would be unpractical.

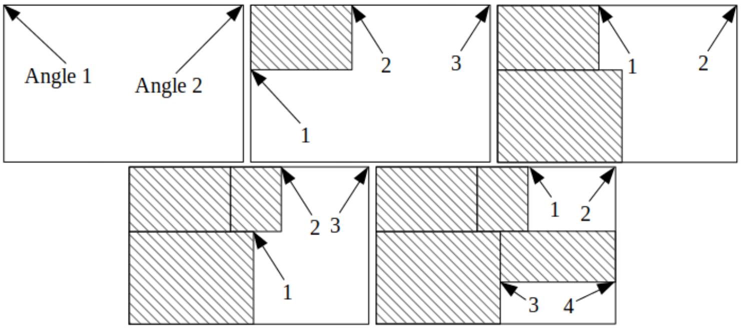


Figure 2: Angle notation on various stages of placement

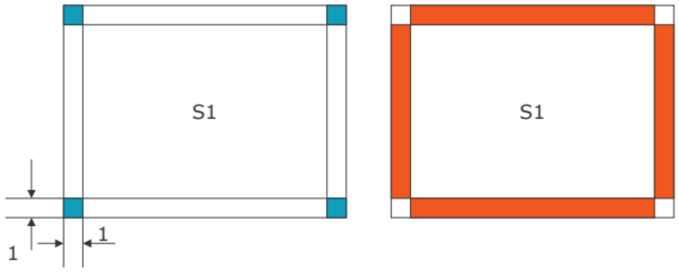


Figure 3: Collision detection: corners first, then along the edges

## 3.3 Collision detection

For computational speed improvement, as proposed prior [6], a raster grid is used to detect the possibility of placement. This sacrifices additional memory in favor of computational speed. The collision detection in this case, as illustrated in Figure 3, only involves checking the corners first and then checking against the borders in the raster grid. Another advantage is that the number of already placed shapes does not impact the detection speed since this depends only on the number of angles which has to be checked and on the length of the edges (width, height) of the shape being tested.

### 3.4 Genetic operators

Roulette Based Fitness (RBF) selection and Partially mixed-cyclic (PMX-CX) crossover are used. The fitness value is $F = 1 - R$, where $R$ is the waste percentage

$$R = \frac{Area_{free}}{Area_{total}},$$

$R \in [0, 1]$, therefore $F \in [0, 1]$. A higher value of $F$ indicates a better solution than a lower one. Mutation can either change the rotation of an angle or switch two genes. Since shape $id$ is unique, their change it's not allowed.

## 4 Results and improvements

The first test of the algorithm was using an already presented problem [5, 8]. The problem is presented in more detail in Section 4.4. The initial tests were conducted using a population of $100$ individuals, $1000$ generations, $0.15$ crossover and $0.05$ mutation. Both crossover and mutation values express the fraction of affected population by the genetic operators in a $[0..1]$ interval. The algorithm generated a best fitness value of $0.969$ with an average best fitness value of $0.941$ out of $100$ runs. The fitness value was collected after each $100$th generation. This result is illustrated in Figure 5a by the straight line.

However, using an input of 558 shapes listed in Appendix Input list, 120x110, 558 shapes with an cutting stock area of $120 \times 110$ from our generator, as indicated by the dotted line on the same figure, the algorithm convergence, in our opinion, was extremely low over the $1000$ generations. Arguably the difference is due to the exact fit solution generated. In this case, each of the shapes not placed has a far bigger impact on the fitness value compared to the previous test where the number of shapes used has a far greater total area than the cutting stock. To improve on this limitation, instead of placing a shape at a random angle, we modified the algorithm to try all possible angles of placement available at a given step. The following sections describe the improvements obtained using this idea.

### 4.1 Crossover and mutation operation improvement

Since each gene stores the placed angle number for the shape, in case of crossover and mutation operations some of these will fall outside of the available angle numbers at a given step. While this is not necessarily an issue because with generation progression these individuals are eliminated, in practice, we found that this results in unreasonable slow convergence over the 1000 generations. To overcome this limitation all mutated chromosomes are reevaluated, the placement angles of the shapes are, if needed, recalculated and the genes updated. In such case, the shape is attempted to be placed again at the available angles. This in term, requires that the rest of the chromosome to be updated as well. The operation requires an additional computational effort, however, the result is a significant convergence improvement as seen in Figure 5b represented by the straight line compared to the result shown in Figure 5a with the dotted line.

We also applied the concept of elite individuals [15], the aim of which is to ensure that individuals carrying the best solutions propagate to the next generation. The result of this improvement is also incorporated in the result illustrated in Figure 5b with the straight line.

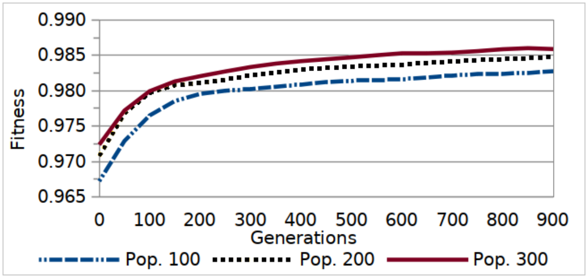
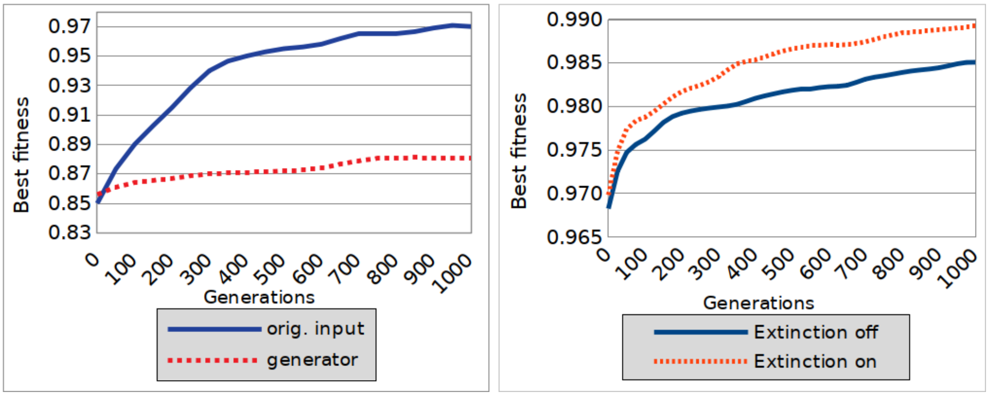


Figure 4: Population size influence on fitness

### 4.2 Initial population and maximum generation size

After we achieved a satisfying conversion, we investigated the quality of the starting population and the size of maximum generations. As there is an exact fit solution to the input used but also noticing the fitness convergence, we tried to improve the quality of the initial population by applying the above idea of testing all possible angles when trying to place a given shape. This resulted in the starting population's best fitness value improvement by an average value



(a) initial results (b) improved algorithm

Figure 5: Comparison between the original and the improved algorithm

of 0.1122 (11.22%). This result is observable in the difference of the starting fitness values between Figure 5a and Figure 5b.

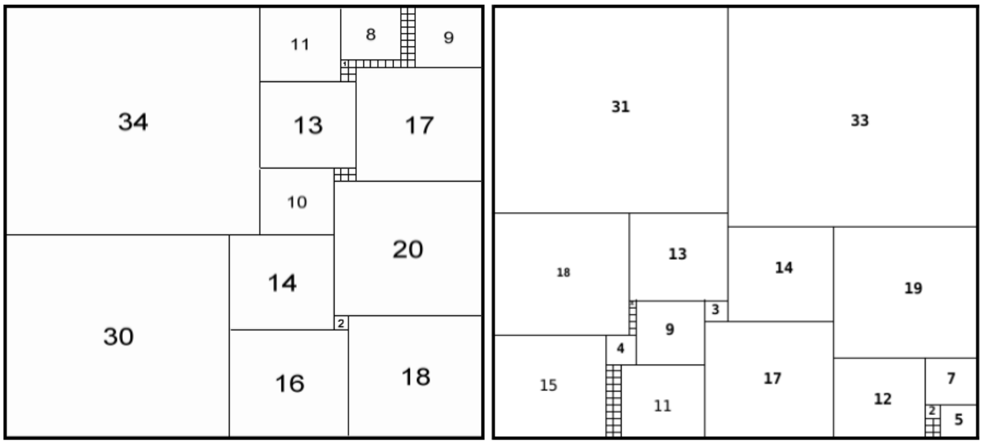
Testing the population size influence on the algorithm's result quality we considered that the average fitness value increase is beneficial up to a population of 200 − 250 for the current set of input. Illustrated in Figure 4, the result was obtained out of 10 runs using the 558 input shape list and the fitness value collected every 100th generation. As a result, we considered a population of 250 with a maximum of 1000 generations for the further tests.

## 4.3 Mass extinction

For even better convergence to the known solution, the notion of mass extinction [11] which shows some promise in regards to certain types of genetic algorithms [2] was also tested. Using preliminary tests, parameter values of 0.25 and 250 are used, meaning that a quarter of the population is eliminated each 250th generations. After the extinction process, the crossover probability is dynamically increased to double the value to allow for population regeneration. In this case, the chance increases for weaker fitness value individuals to produce offsets. The theoretical benefit of the extinction process is that the newly created individuals increase the population diversity, thus allowing the population to explore more of the search space or if being stuck in a local maxima this operation increases the chance of breaking out. The result obtained from a 100 consecutive runs, as illustrated in Figure 5b with the dotted line, shows a slight improvement of the average best fitness values by 0.00418 (0.418%) with a lightly faster convergence.

### 4.4 Revisiting the 64x64 particular covering problem

With the improved algorithm, we revisited the particular covering problem [5, 8] of placing most squares from a list of $1 \times 1$, $2 \times 2$, to $46 \times 46$ on a $64 \times 64$ grid, trying to get the most cell coverage. Here we managed to improve on the existing solution presented in the old paper where the best result obtained left 35 free cells, illustrated in figure Figure 6a, while the new solution, illustrated in figure Figure 6b, has only 32 free. The result was obtained from 100 runs and the best fitness was achieved twice, in runs 24 and 71.



(a) Previous result with 35 free space (b) Improved result with 32 free space

Figure 6: Improvement made to the particular placement problem

## 5 Conclusions and future work

In this paper, we presented an exact fit 2D problem generator which fills a gap in the problem generators area. Preliminary testing was done using the revisited upper deck placement genetic algorithm. Using the input from the generator a series of improvements made to the algorithm was investigated and presented. These improved both the quality of the starting population as well as the convergence of the GA. We also revisited the $64 \times 64$ particular covering problem where we managed to improve on the previous result. Further development plans for the presented generator include the option to generate guillotine stock problems as well as three-dimensional problems.

## Acknowledgements

This work was partially supported by the Collegium Talentum 2017 Programme of Hungary.

## References

[1] R. Alvarez-Valdez, F. Parreño, J.M. Tamarit, A Tabu Search Algorithm for two-dimensional, non-quillotine problems, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74

[2] H. David Mathias, R. Vincent, An empirical study of crossover and mass extinction in a genetic algorithm for pathfinding in a continuous environment, *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, 2016, pp. 4111–4118. ⇒81

[3] E. Hopper, B. C. H. Turton, A genetic algorithm for a 2D industrial packing problem, *Computers & Industrial Engineering* **37,** (1999), 375–378. ⇒77

[4] E. Hopper, B. C. H. Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, *European Journal of Operational Research* **128,** 1 (2001) 34–57. ⇒75

[5] L. Illyés, Genetic algorithms for a particular covering problem, *International Conf. on Economic Cybernetics*, Bucuresti, Romania, 2004. ⇒79, 82

[6] L. Illyés, Upper Angle Placement Algorithm with Genetic Approach for 2D Rectangle Knapsack Problem, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74, 77, 78

[7] L. Illyés, Cs. Fábián, "Jigsaw" problem generator for 2D rectangle single large object for non-guillotine and guillotine cutting, *Proc. WSCSP2005, Workshop on Cutting Stock Problems*, Miercurea-Ciuc, Romania, 2005, pp. 83–89. ⇒74

[8] L. Illyés, L. Pál, Generalized particular covering problem with genetic algorithms, *AMO - Advanced Modeling and Optimization*, **7,** 1, 2005, 1–7. ⇒ 79, 82

[9] T. Inamichi et. al., Branch-and-bound algorithms for regular strip packing and perfect packing problems, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74

[10] S. Jakobs, On genetic algorithms for packing polygons, *European Journal of Operational Research* **88,** (1996), 165–181. ⇒74

[11] B. Jaworski, Kuczkowski, R. Śmierzchalski, Extinction Event Concepts for the Evolutionary Algorithms, *Przeglad Elektrotechniczny*, **1,** 88 (2012), 252–255. ⇒ 81

[12] D. Liu, H. Teng, An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles, *European Journal of Operational Research* **112,** (1999), 413–420. ⇒75

[13] C. L. Mumford, P. Y. Wang, A genetic simulated annealig approach to aacking, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74

[14] E. Silva, J. F. Oliveira, G. Wäsher, A problem generator for two-dimensional rectangular cutting and packing problems, *ESICUP.* (2007), https://paginas.fe.up.pt/~esicup/problem_generators. ⇒74

[15] M. Villalobos-Arias, C. Coello Coello, O. Hernndez-Lerma, Asymptotic convergence of some metaheuristics used for multiobjective optimization, *FOGA 2005: Foundations of Genetic Algorithms*, Aizu-Wakamatsu, Japan, 2005, pp. 95–111. ⇒80

[16] G. Wäscher, H. Hauner, H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research* **183,** 3 (2007), 1109–1130. ⇒74

[17] *EURO Special Interest Group on Cutting and Packing*, `https://paginas.fe.up.pt/~esicup/problem_generators` ⇒74

## Appendix

## Input list, 120x110, 558 shapes

The following table contains the list of input shapes used for testing the GA. The values in column "#" represents the id of the shape, columns "W." and "H." its width and height, while column "Nr." indicates the quantity of this.

| # | W. | H. | Nr. | # | W. | H. | Nr. | # | W. | H. | Nr. | # | W. | H. | Nr. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 34 | 36 | 4 | 5 | 6 | 71 | 7 | 3 | 3 | 106 | 10 | 7 | 2 |
| 2 | 1 | 2 | 19 | 37 | 4 | 6 | 5 | 72 | 7 | 4 | 3 | 107 | 10 | 8 | 3 |
| 3 | 1 | 3 | 9 | 38 | 4 | 7 | 2 | 73 | 7 | 5 | 4 | 108 | 10 | 12 | 1 |
| 4 | 1 | 4 | 9 | 39 | 4 | 8 | 1 | 74 | 7 | 6 | 4 | 109 | 10 | 13 | 2 |
| 5 | 1 | 5 | 13 | 40 | 4 | 9 | 2 | 75 | 7 | 7 | 2 | 110 | 10 | 16 | 1 |
| 6 | 1 | 6 | 5 | 41 | 4 | 10 | 2 | 76 | 7 | 8 | 1 | 111 | 11 | 3 | 1 |
| 7 | 1 | 7 | 2 | 42 | 5 | 1 | 7 | 77 | 7 | 9 | 1 | 112 | 11 | 5 | 3 |
| 8 | 1 | 8 | 4 | 43 | 5 | 2 | 8 | 78 | 7 | 12 | 3 | 113 | 11 | 6 | 1 |
| 9 | 1 | 11 | 1 | 44 | 5 | 3 | 6 | 79 | 7 | 13 | 2 | 114 | 11 | 10 | 1 |
| 10 | 2 | 1 | 15 | 45 | 5 | 4 | 10 | 80 | 8 | 1 | 1 | 115 | 11 | 11 | 1 |
| 11 | 2 | 2 | 18 | 46 | 5 | 5 | 7 | 81 | 8 | 2 | 6 | 116 | 11 | 12 | 1 |
| 12 | 2 | 3 | 19 | 47 | 5 | 6 | 6 | 82 | 8 | 3 | 1 | 117 | 12 | 4 | 2 |
| 13 | 2 | 4 | 11 | 48 | 5 | 7 | 5 | 83 | 8 | 4 | 1 | 118 | 12 | 5 | 2 |
| 14 | 2 | 5 | 7 | 49 | 5 | 8 | 1 | 84 | 8 | 5 | 3 | 119 | 12 | 7 | 1 |
| 15 | 2 | 6 | 7 | 50 | 5 | 9 | 3 | 85 | 8 | 6 | 1 | 120 | 12 | 8 | 2 |
| 16 | 2 | 7 | 2 | 51 | 5 | 10 | 1 | 86 | 8 | 7 | 1 | 121 | 12 | 10 | 1 |
| 17 | 2 | 8 | 3 | 52 | 5 | 11 | 1 | 87 | 8 | 8 | 2 | 122 | 13 | 3 | 1 |
| 18 | 2 | 9 | 1 | 53 | 5 | 12 | 1 | 88 | 8 | 9 | 1 | 123 | 13 | 4 | 1 |
| 19 | 2 | 15 | 1 | 54 | 6 | 1 | 5 | 89 | 8 | 10 | 2 | 124 | 13 | 5 | 1 |
| 20 | 3 | 1 | 18 | 55 | 6 | 2 | 6 | 90 | 8 | 12 | 1 | 125 | 13 | 6 | 2 |
| 21 | 3 | 2 | 22 | 56 | 6 | 3 | 3 | 91 | 8 | 13 | 1 | 126 | 13 | 7 | 1 |
| 22 | 3 | 3 | 14 | 57 | 6 | 4 | 9 | 92 | 9 | 1 | 1 | 127 | 13 | 12 | 1 |
| 23 | 3 | 4 | 8 | 58 | 6 | 5 | 9 | 93 | 9 | 2 | 2 | 128 | 14 | 4 | 1 |
| 24 | 3 | 5 | 4 | 59 | 6 | 6 | 5 | 94 | 9 | 3 | 1 | 129 | 14 | 5 | 2 |
| 25 | 3 | 6 | 11 | 60 | 6 | 7 | 3 | 95 | 9 | 4 | 3 | 130 | 15 | 2 | 1 |
| 26 | 3 | 7 | 1 | 61 | 6 | 8 | 3 | 96 | 9 | 5 | 2 | 131 | 15 | 6 | 1 |
| 27 | 3 | 9 | 2 | 62 | 6 | 9 | 3 | 97 | 9 | 7 | 2 | 132 | 15 | 9 | 2 |
| 28 | 3 | 10 | 1 | 63 | 6 | 10 | 1 | 98 | 9 | 8 | 1 | 133 | 16 | 4 | 1 |
| 29 | 3 | 11 | 2 | 64 | 6 | 11 | 1 | 99 | 9 | 9 | 2 | 134 | 16 | 5 | 1 |
| 30 | 3 | 13 | 3 | 65 | 6 | 12 | 1 | 100 | 9 | 10 | 1 | 135 | 16 | 6 | 1 |
| 31 | 3 | 20 | 1 | 66 | 6 | 14 | 1 | 101 | 9 | 11 | 2 | 136 | 19 | 9 | 1 |
| 32 | 4 | 1 | 10 | 67 | 6 | 15 | 2 | 102 | 9 | 15 | 1 | 137 | 22 | 9 | 1 |
| 33 | 4 | 2 | 9 | 68 | 6 | 19 | 1 | 103 | 10 | 2 | 1 | | | | |
| 34 | 4 | 3 | 12 | 69 | 7 | 1 | 6 | 104 | 10 | 4 | 2 | | | | |
| 35 | 4 | 4 | 10 | 70 | 7 | 2 | 7 | 105 | 10 | 6 | 1 | | | | |

*Received: March 15, 2018 • Revised: July 13, 2018*

Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 86–109

DOI: 10.2478/ausi-2018-0006

# Hierarchical clustering with deep Q-learning

Richárd FORSTER
Eötvös University
email: forceuse@inf.elte.hu

Ágnes FÜLÖP
Eötvös University
email: fulop@caesar.elte.hu

**Abstract.** Following up on our previous study on applying hierarchical clustering algorithms to high energy particle physics, this paper explores the possibilities to use deep learning to generate models capable of processing the clusterization themselves. The technique chosen for training is reinforcement learning, that allows the system to evolve based on interactions between the model and the underlying graph. The result is a model, that by learning on a modest dataset of $10,000$ nodes during $70$ epochs can reach $83,77\%$ precision for hierarchical and $86,33\%$ for high energy jet physics datasets in predicting the appropriate clusters.

## 1 Introduction

Different datasets should be clusterized with specific approaches. For real world networks, hierarchical algorithms, like the Louvain method, provides an efficient way to produce the clusters, that will represent elements, that have a strong connection. In high energy physics clusterization can be done by finding jets using for example the $k_t$ jet clustering. Here a cluster will be a narrow cone of hadrons and other particles produced by the hadronization of a quark or gluon in a heavy ion experiment. Fusing these algorithms a more generic

**Computing Classification System 1998:** I.1.4
**Mathematics Subject Classification 2010:** 58A20
**Key words and phrases:** jet, cluster algorithm, hierarchical clustering, deep q-learning, neural network, multi-core, keras, cntk, louvain

process can be conceived, that was studied in [13]. As a consequence building a graph from the available particles, the same hierarchical clustering can be computed, like on other network related datasets. The graph of the two different input differs only in how the edges are represented, while networks have some strength associated to their connections, the weight of the edges between the different nodes, for particles this weight will be the distance between the elements.

Further generalizing the approach, in this paper a deep learning method is described based on reinforcement learning, that allows the system, to learn to clusterize the input graphs without any external user interaction, relying only on the agent's experience on the graph. This way a single algorithm can be used for the different kind of datasets without any additional specificity in the computational process. In the context of present paper this will lead to a generic model, that is capable to predict the clusterization steps of the elements in different hierarchical clustering tasks.

The evaluation is provided on real world network data taken from the U.S. Census 2010 database. Further data was generated for jet physics using the AliRoot framework [29]. A comparison is given on modularity level of the clusterization between the standard Louvain method and a modified version using the generated model for predicting future communities. Early results shows, that the neural network is capable to achieve an average precision on the Census test dataset of $83,77\%$ and $84,12\%$ for jet dataset, learning for only $70$ epochs on a training set consisting from information collected only on the hierarchical dataset.

## 2 Hierarchical clustering

This section contains a brief introduction of the used hierarchical clustering algorithm and how it was fused with the processes of the jet algorithms from physics to achieve a generalized clustering solution.

### 2.1 Jet

A jet is a narrow cone of hadrons and other particles which were produced by the hadronization processes of a quark or gluon plasma in a particle physics or heavy ion experiment[27]. The constituent particles are carrying a color charge, such as quarks, they cannot exist in free form due to QCD confinement. This theory allows the colorless states only. The color-free particles formed during the fragmentation process form the jet, because the fragments all tend to travel in the same direction, creating a narrow jet of particles (Figure 1). Jets are measured in particle detectors and researched to determine the properties of the original quarks. Jets are produced in QCD hard scattering processes, evolving high transverse momentum quarks or gluons, which is called collectively partons in the partonic picture[25]. Perturbative QCD calculations may have colored partons in the final state, but the colorless hadrons are detected in the observed experimental. It can be understand what happened in the detector, if all outgoing colored partons must first undergo parton showering and then combination of the created partons into hadrons.

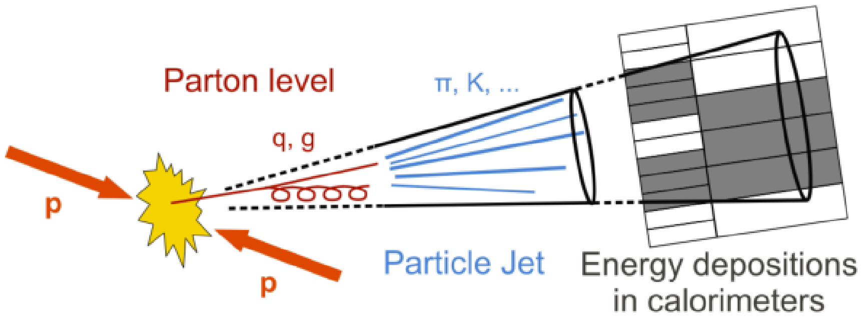


Figure 1: Structure of jet

## 2.2 Jet clusterisation

The jet clustererisation means when we research the jet momentum owing to the final state particles in the calorimeter [30]. For more accurate understanding the conversion results, we can take into account a muon systems. They form the clusters all together. Two problems should be noted in theoretical research: the infrared (IR) safety and collinear safety. The infrared safe means, that the measured object does not depend on the low energy theoretical physics. The collinear(C) safety is understood, when a parton is substituted by a collinear pair of partons, then the jet clustering outcome does not change. Then the jet can be observed by perturbative technic to apply the experiment, because the jet does not modulate when the particles radiates a very soft objects, or fails to two collinear particles. By theoretical consideration in the case of the infrared divergence the integral of Feynman diagram diverges due to the constituent objects hold very small energy which goes to zero. When the system consists massless particles, it can apply an infrared cutoff and it approximates to zero. The divergence stays finite quantity in the experimental data. Then the infrared safe and collinear safe jet reconstruction algorithm can be evaluated for the measurement which satisfies the theoretical considerations or it is used to a given order due to the IRC safe method. Because the jet mass and energy depend on the jet radius, therefore these quantities can be determined more precisely, if the jet radius becomes larger. In the case of the smaller size the cluster consists of more hadronised particles.

**Substructure of jet** It can be one jet which including more than one group of gaussian- distributed clusters. Substructure is possible a non gaussian component, which is compliances to an offset. It can also contains another gaussian group of clusters, ie. second hard jet. Three different types can be defined:

I: Subjet from uncorrelated sources, overlapping the hard jet which is thought or clustered together with it. This is soft process, derived from proton-leftovers, initial state radiation, beam-rests and/or scatterings, e.g. pileup (PU) and underlying event (UE).

II: Subjet from correlated sources, clustered together with the hard jet considered, coming from the same primary vertex, but another branch of the Feynman diagram.

III: Subjet from correlated sources, deriving from the decay of a single boosted particle, clustered together into a single jet.

## 2.3 Jet algorithm

During the last 40 years several jet reconstruction algorithms have been developed for hadronic colliders [31, 1]. The first ever jet algorithm was published by Sterman and Weinberg in the 1970's [34]. The cone algorithm plays an important role when a jet consists of a large amount of hadronic energy in a small angular region. It is based on a combination of particles with their neighbours in $\eta - \varphi$ space within a cone of radius $R = \sqrt{(\Delta\varphi^2 + \Delta\eta^2)}$. However the sequential recombination cluster algorithms combine the pairs of objects which have very close $p_t$ values. The particles merge into a new cluster through successive pair recombination. The starting point is the lowest $p_t$ particles for clustering in the $k_t$ algorithm, but in the anti-$k_t$ recombination algorithm it is the highest momentum particles.

The jet clustering involves the reconstructed jet momentum of particles, which leaves the calorimeter together with modified values by the tracker system.

### 2.3.1 Cone algorithm

The Cone algorithm is one of the regularly used methods at the hadron colliders. The main steps of the iteration are the following [34]: the seed particle $i$ belongs to the initial direction, and it is necessary to sum up the momenta of all particle $j$, which is situated in a circle of radius $R$ ($\Delta R_{ij}^2 = (y_i - y_j)^2 + (\varphi_i - \varphi_j)^2 < R^2$) around $i$, where $y_i$ and $\varphi_i$ are the rapidity and azimuth of particle $i$.

The direction of the sum is applied as a new seed direction. The iteration procedure is repeated as long as the direction of the determined cone is stable.

It is worth noting what happens when two seed cone overlaps during the iteration. Two different groups of cone algorithms are discussed: One possible solution is to select the first seed particle that has the greatest transverse momentum. Have to find the corresponding stable cone, i.e. jet and delete the particles from the event, which were included in the jet. Then choose a new seed, which is the hardest particle from the remaining particles, and apply to search the next jet. The procedure is repeated until there is no particle that has not worked. This method avoids overlapping.
Other possibility is the so called "overlapping" cones with the split-merge approach. All the stable cones are found, which are determined by iteration from all particles. This avoids the same particle from appearing in multiple cones. The split-merge procedure can be used to consider combining pair of cones. In this case more than a fraction $f$ of the transverse momentum of the softer cones derives from the harder particles; otherwise the common particles assigned to the cone, which is closer to them. The split-merge procedure applies the initial list of protojets, which contains the full list of stable cones:

1. Take the protojet with the largest $p_t$ (i.e. hardest protojet), label it $a$.
2. Search the next hardest protojet that shares particles with $a$ (i.e. overlaps), label it $b$. If no such protojet exists, delete $a$ from the list of protojets and add it to the list of final jets.
3. Determine the total $p_t$ of the particles, which is shared between the two protojets, $p_{t,shared}$.
    - If $p_{t,shared} > f$, where $f$ is a free parameter, it is called the overlap threshold, replace protojets $a$ and $b$ with a single merged protojet.
    - Otherwise the protojets are scattered, for example assigning the shared particles to the protojet whose axis is closer.
4. Repeat from step 1 as long as there are protojets left.

A similar procedure to split-merge method is the so called split-drop, where the non-shared particles, which fall into the softer of two overlapping cones are dropped, i.e. are deleted from the jets altogether.

### 2.3.2 Sequential recombination jet algorithm

They go beyond just finding jets and implicitly assign a clustering sequence to an event, which is often closely connected with approximate probabilistic pictures that one may have for parton branching. The current work focuses on the $k_t$ algorithm, whose parallelization was studied in [11] and [12].

**The $k_t$ algorithm for hadrons** In the case of the proton-proton collision, the variables which are invariant under longitudinal boots are applied. These quantities which were introduced by [5] and the distance measures are longitudinally invariant as the following:

$$d_{ij} = \min\left(p_{ti}^2, p_{tj}^2\right) \Delta R_{ij}^2, \qquad \Delta R_{ij} = (y_i - y_j)^2 + (\varphi_i - \varphi_j)^2 \tag{1}$$

$$d_{iB} = p_{ti}^2. \tag{2}$$

In this definition the two beam jets are not distinguished.

If $p = -1$, then it gives the "anti-$k_t$" algorithm. In this case the clustering contains hard particles instead of soft ones. Therefore the jets extend outwards around hard seeds. Because the algorithm depends on the energy and angle through the distance measure, therefore the collinear branching will be collected at the beginning of the sequence.

## 2.4 The Louvain algorithm

The Louvain method [4], is a multi-phase, iterative, greedy hierarchical clusterization algorithm, working on undirected, weighted graphs. The algorithm processes through multiple phases, within each phase multiple iterations until a convergence criteria is met. Its parallelization was explored in [22], that was further evolved into a GPU based implementation as was detailed in [10]. The modularity is a monotonically increasing function, spreading across multiple iterations, giving a numerical representation on the quality of the clusters. Because the modularity is monotonically increasing, the process is guaranteed to terminate. Running on a real world dataset, termination is achieved in not more than a dozen iterations.

### 2.4.1 Modularity

On a set, $S = C_1, C_2, ..., C_k$, containing every community in a given partitioning of $V$, where $1 \leq k \leq N$ and $V$ is the set of nodes, $N$ is the number of nodes. Modularity $Q$ is given by the following [26]:

$$Q = \frac{1}{2W} \sum_{i \in V} e_{i \to C(i)} - \sum_{C \in S} \left( \frac{deg_C}{2W} \cdot \frac{deg_C}{2W} \right), \tag{3}$$

where $deg_C$ is the sum of the degrees of all the nodes in community $C$, $e_{i \to C(i)}$ is the sum of weights of all edges connecting node $i$ to all nodes in community $C(i)$ and $W$ is the sum of the weight of all the edges.

Modularity has multiple variants, like the ones described in [37, 2, 3]. Yet the one defined in Eq. (3) is the more commonly used.

## 2.5 Hierarchical $k_t$ clustering

In [13] it was studied how to do hierarchical clustering, following the rules of the $k_t$ algorithm. First the list of particles has to be transformed into a graph, with the particles themselves appointed as nodes. The distance between the elements is a suitable selection for a weight to all edges between adjacent particles. But as it eventually leads up to $n * (n - 1)/2$ links, where $n$ is the number of nodes, a better solution is to make connections between nearest neighbours and to the second to nearest. If the particle's nearest "neighbour" is the beam, it will be represented with an isolated node. While the Louvain algorithm relies on modularity gain to drive the computation, the jet clustering variant doesn't have the modularity calculation, as it is known that the process will end, when all particles are assigned to a jet.

The result of this clustering will still be a dendogram, where the leafs will represent the jets.

## 3 Basic artificial neural networks

Since the beginning of the 1990s the artificial neural network (ANN) methods are employed widely in the high energy physics for the jet reconstruction and track identification [9, 18]. These methods are well-known in offline and online data analysis also.

Artificial neural networks are layered networks of artificial neurons (AN) in which biological neurons are modeled. The underlying principle of operation is as follows, each AN receives signals from another AN or from environment, gathers these and creates an output signal which is forwarded to another AN or the environment. An ANN contains one input layer, one or more hidden layers and one output layer of ANs. Each AN in a layer is connected to the ANs in the next layer. There are such kind ANN configurations, where the feedback connections are introduced to the previous layers.

## 3.1 Architecture

An artificial neuron is denoted by a set of input signals $(x_1, x_2, \dots x_n)$ from the environment or from another AN. A weight $w_i$ $(i = 1, \dots n)$ is assigned to each input signal. If the value of weight is larger than zero then the signal is excited, otherwise the signal is inhibited. AN assembles all input signals, determines a net signal and propagates an output signal.

### 3.1.1 Types of artificial networks

Some features of neural systems which makes them the most distinct from the properties of conventional computing:

- The associative recognition of complex structures
- Data may be non-complete, inconsistent or noisy
- The systems can train, i.e. they are able to learn and organize themselves
- The algorithm and hardware are parallel

There are many types of artificial neural networks. In the high energy particle physics the so-called multi-layer perception (MLP) is the most widespread. Here a functional mapping from input $x_k$ to output $z_k$ values is realised with a function $f_{z_k}$:

$$z_k = f_{z_k}\left(\sum_{j=1}^{m+1} w_{k_j} f_{y_j}\left(\sum_{i=1}^{n+1} v_{ji} x_i\right)\right),$$

where $v_{ji}$ are the weights between the input layer and the hidden layer, and $w_{kj}$ are the weights between the hidden layer and the output layer. This type of ANN is called feed-forward multi-layer ANN.

It can be extended into a layer of functional units. In this case an activation function is implemented for the input layer. This ANN type is called functional link ANN. The output of this ANN is similar such as previously ANN, without it has additional layer, which contains $q$ functions $h_l(x_1 \dots x_n)(l = 1 \dots q)$.

The weights between the input layer and the functional layer are $u_{li} = 1$, if $h_l$ depends on $x_i$, and $u_{li} = 0$ otherwise. The output of this ANN is:

$$z_k = f_{z_k}\left(\sum_{j=1}^{m+1} w_{kj} f_{y_j}\left(\sum_{l=1}^{q+1} v_{jl} h_l(x_1 \ldots x_n)\right)\right).$$

The functional link ANNs provides better computational time and accuracy then the simple feed-forward multi-layer ANN.

**Application in high-energy physics** The first application, which was published in 1988, discussed a recurrent ANN for tracking reconstruction [8]. A recurrent ANN was also used for tracking reconstruction in LEP experiment [28]. An article published about a neural network method which was applied to find efficient mapping between certain observed hadronic kinematical variables and the quark-gluon identify. With this method it is able to separate gluon from quark jets originating from the Monte-Carlo generated $e^+e^-$ events [21]. A possible discrimination method is presented by the combination of a neural network and QCD to separate the quark and gluon jet of $e^+e^-$ annihilation [6].

The neural network clusterisation algorithm was applied for the ATLAS pixel detector to identify and split merged measurements created by multiple charged particles [20]. The neural network based cluster reconstruction algorithm which can identify overlapping clusters and improves overall particle position reconstruction [32].

Artificial intelligence offers the potential to automate challenging data-processing tasks in collider physics. To establish its prospects, it was explored to what extent deep learning with convolutional neural networks can discriminate quark and gluon jets [19].

## 4 Q-learning

Q-learning is a model-free reinforcement learning technique [35]. The reinforcement learning problem is meant to be learning from interactions to achieve a goal. The learner and decision-maker is called the agent. The thing it interacts with is called the environment, that contains everything from the world surrounding the agent. There's a continuous interaction between, where the agent selects an action and the environment responds by presenting new situations (states) to the agent. The environment also returns rewards, special numerical values that the agent tries to maximize over time. A full specification of an environment defines a task, that is an instance of the reinforcement learning problem. Specifically, the agent and environment interact at each of a sequence of discrete time steps $t = 0, 1, 2, \ldots$. At each time step $t$, the agent receives the environment's state, $s_t \in S$, where $S$ is the set of possible states, and based on that it selects an action, $a_t \in \mathfrak{A}(s_t)$, where $\mathfrak{A}(s_t)$ is the set of all available actions in state $s_t$. At the next time step as a response to the action, the agent receives a numerical reward, $r_{t+1} \in \mathfrak{R}$ , and finds itself in a new state, $s_{t+1}$ (Figure 2).

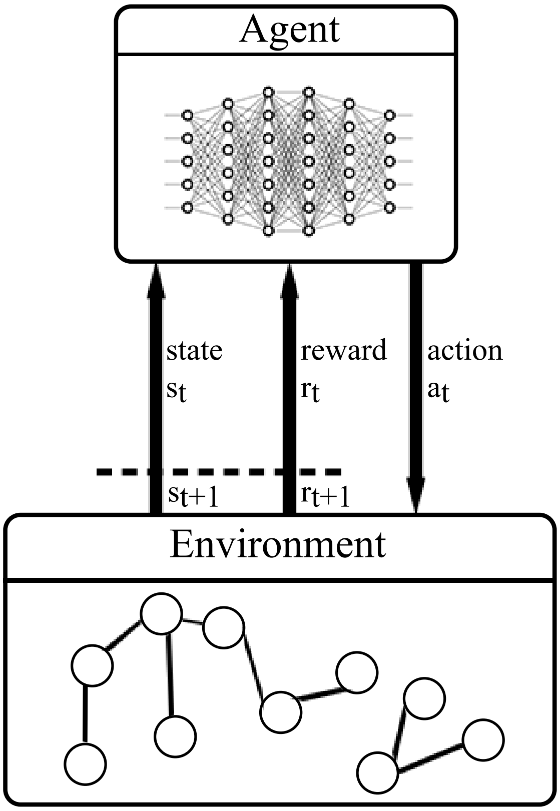


Figure 2: The agent-environment interaction in reinforcement learning

At every time step, the agent implements a mapping from states to probabilities of selecting the available actions. This is called the agent's policy and is denoted by $\pi_t$, where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods specify how the agent changes this using its experience. The agent's goal is to maximize the total amount of reward it receives over the long run.

## 4.1 Goals and rewards

The purpose or goal of the agent is formalized in terms of a special reward passed from the environment. At each time step, the reward is a simple number, $r_t \in \Re$. The agent's goal is to maximize the total reward it receives.

## 4.2 Returns

If the rewards accumulated after time step $t$ is denoted by $r_{t+1}, r_{t+2}, r_{t+3}, \ldots$, what will be maximized by the agent is the expected return $R_t$, that is defined as some function of the received rewards. The simplest case is the sum of the rewards: $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$, where $T$ is the final time step. This approach comes naturally, when the agent-environment interaction breaks into subsequences, or episodes. Each episode ends in a special terminal state, that is then being reset to a standard starting state. The set of all nonterminal states is denoted by $S$, while the set with a terminal state is denoted by $S^+$.

Introducing discounting, the agent tries to maximize the the sum of the discounted rewards by selecting the right actions. At time step $t$ choosing action $a_t$, the discounted return will be defined with Eq. (4).

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{4}$$

where $\gamma$ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate. It determines the present value of future rewards: a reward received at time step $t + k$ is worth only $\gamma^{k-1}$ times the immediate reward. If $\gamma < 1$, the infinite sum still is a finite value as long as the reward sequence $\{r_k\}$ is bounded. If $\gamma = 0$, the agent is concerned only with maximizing immediate rewards. If all actions influences only the immediate reward, then the agent could maximize equation 4 by separately maximizing each reward. In general, this can reduce access to future rewards and the return may get reduced. As $\gamma$ approaches 1, future rewards are used more strongly.

## 4.3 The Markov property

Assuming a finite set of states and reward values, also considering how a general environment responds at time $t+1$ to the action taken at time $t$, this response may depend on everything that has happened earlier. In this case only the complete probability distribution can define the dynamics:

$$\Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\}, \tag{5}$$

for all $s'$,$r$, and all possible values of the past events: $s_t, a_t, r_t, \dots, r_1, s_0, a_0$. If the state has the Markov property the environment's response at $t+1$ depends only on the state and action at $t$ and the dynamics can be defined by applying only Eq. (6).

$$\Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}, \tag{6}$$

for all $s'$, $r$, $s_t$, and $a_t$. Consequently if a state has the Markov property, then it's a Markov state, only if (6) is equal to (5) for all $s'$, $r$, and histories, $s_t, a_t, r_t, \dots, r_1, s_0, a_0$. In this case, the environment has the Markov property.

## 4.4 Markov cecision process

A reinforcement learning task satisfying the Markov property is a Markov decision process, or MDP. If the state and action spaces are finite, then it is a finite MDP. This is defined by its state and action sets and by the environment's one-step dynamics. Given any state, action pair, $(s, a)$, the probability of each possible next state, $s'$, is

$$P^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}.$$

Having the current state and action, $s$ and $a$, with any next state, $s'$, the expected value of the next reward can be computed with

$$R^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}.$$

These quantities, $P^a_{ss'}$ and $R^a_{ss'}$, completely specify the most important aspects of the dynamics of a finite MDP.

## 4.5 Value functions

Reinforcement learning algorithms are generally based on estimating value functions, that are either functions of states or state-action. They estimate how good a given state is, or how good a given action in the present state is. How good it is, depends on future rewards that can be expected, more precisely, on the expected return. As the rewards received depends on the taken actions, the value functions are defined with respect to particular policies. A policy, $\pi$, is a mapping from each state, $s \in S$, and action, $a \in A(s)$, to the probability $\pi(s, a)$ of taking action $a$ while in state $s$. The value of a state $s$ under a policy $\pi$, denoted by $V^\pi(s)$, is the expected return when starting in $s$ and following $\pi$. For MDPs $V^\pi(s)$ is defined as

$$V^\pi(s) = E_\pi\{R_t|s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right\},$$

where $E_\pi$ is the expected value given that the agent follows policy $\pi$. The value of the terminal state is always zero. $V^\pi$ is the state-value function for policy $\pi$. Similarly, the value of taking action $a$ in state $s$ under a policy $\pi$, denoted by $Q^\pi(s, a)$ is defined as the expected return starting from $s$, taking the action $a$, and following policy $\pi$:

$$Q^\pi(s, a) = E_\pi\{R_t|s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right\}.$$

$Q^\pi$ is the action-value function for policy $\pi$.

$V^\pi$ and $Q^\pi$ can be estimated from experience. If an agent follows policy $\pi$ and maintains an average of the actual return values in each encountered state, then it will converge to the state's value, $V^\pi(s)$, as the number of times that state is encountered approaches infinity. If in a given state every action has a separate average, then these will also converge to the action values, $Q^\pi(s, a)$.

## 4.6 Optimal value functions

To solve a reinforcement learning task, a specific policy needs to be found, that achieves a lot of reward over the long run. For finite MDPs, an optimal policy can be defined. Value functions define a partial ordering over policies. A policy $\pi$ is defined to be better than or equal to a policy $\pi'$ if its expected return is greater than or equal to $\pi'$ for all states. Formally, $\pi \geq \pi'$ if and only if $V^\pi(s) \geq V^{\pi'}(s)$ for all $s \in S$. At least one policy exists, that is better than or equal to all other policies and this is the optimal policy. If more than one exists, the optimal policies are denoted by $\pi^*$. The state-value function among them is the same, called the optimal state-value function, denoted by $V^*$, and defined as

$$V^*(s) = \max_\pi V^\pi(s),$$

for all $s \in S$. The optimal action-value functions are also shared, denoted by $Q^*$, and defined as

$$Q^*(s, a) = \max_\pi Q^\pi(s, a),$$

for all $s \in S$ and $a \in A(s)$. For the state-action pair $(s, a)$, this gives the expected return for taking action $a$ in state $s$ and following an optimal policy. Thus, $Q^*$ can be defined in terms of $V^*$ as follows:

$$Q^*(s, a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}.$$

## 5 Clustering with deep Q-learning

The Deep Q-learning (DQL) [23, 24] is about using deep learning techniques on the standard Q-learning (Section 4).

Calculating the Q state-action values using deep learning can be achieved by applying the following extensions to standard reinforcement learning problems:

1. Calculate Q for all possible actions in state $s_t$,
2. Make prediction for Q on the new state $s_{t+1}$ and find the action $a_{t+1} = \max_a a \in A(s_{t+1})$, that will yield the biggest return,
3. Set the Q return for the selected action to $r + \gamma Q(s_{t+1}, a_{t+1})$. For all other actions the return should remain unchanged,
4. Update the network using back-propagation and mini-batches stochastic gradient descent.

This approach in itself leads to some additional problems. The *exploration-exploitation issue* is related to which action is taken in a given state. By selecting an action that always seems to maximize the discounted future reward, the agent is acting greedy and might miss other actions, that can yield higher overall reward in the long run. To be able to find the optimal policy the agent needs to take some exploratory steps at specific time steps. This is solved by applying the $\epsilon$-*greedy algorithm* [35], where a small probability $\epsilon$ will choose a completely random action.

The other issue is the problem of the *local-minima* [36]. During training multiple states can be explored, that are highly correlated and this may make the network to learn replaying the same episode. This can be solved, by first storing past observations in a *replay memory* and taking random samples from there for the mini-batch, that is used to replay the experience.

## 5.1 Environment

The environment provides the state that the agent will react to. In case of clustering the environment will be the full input graph. The actual state the necessary information required to compute the Louvain method, packaged into a Numpy stack. These include the weights, degrees, number of loops, the actual community and the total weight of the graph. Each state represents one node of the graph with all of its neighbors. The returned rewards for each state will be based on the result of the actual Louvain clusterization, which means during training the environment will compute the real clusters. If the action selected by the agent leads to the best community, that will have a positive reward set to $10000$ and in any other case the returned value will be $-1000$. After stepping, the next state will contain the modified community informations.

The agent's action space is finite and predefined and the environment also has to reflect this. Let the cardinality of the action space be noted for all $s \in S$ states by $|A(s)|$ For this reason, the state of the environment contains information about only $|A(s)|$ neighbors. This can lead to more nodes, than how many really is connected to a given element. In this case the additional dummy node's values are filled with extremals, in the current implementation with negative numbers. One limitation of the actual solution is that if the number of neighbors are higher, than $|A(s)|$, then only the first $|A(s)|$ neighbors will be considered, in the order in which they appear in dataset. The first "neighbor" will be currently evaluated node, so in case the clusterization will not yield any better community, the model should see, that the node stays in place.

To help avoid potential overflow during the computation, weights of the input graph are normalized to be between $0.000001$ and $1$.

## 5.2 Agent

The agent acts as the decision maker, selecting the next community for a given node. It takes the state of the environment as an input and gives back the index of the neighbor that is considered to be providing the best community.

### 5.2.1 Implementation in Keras

Keras [38] is a Python based high-level neural networks API, compatible with the TensorFlow, CNTK, and Theano machine learning frameworks. This API encourages experimentation as it supports rapid development of neural networks. It allows easy and fast prototyping, with a user friendly, modular, and extensible structure. Both *convolutional networks* and *recurrent networks* can be developed, also their combinations are also possible in the same agent. As all modern neural network API it both runs on CPU and GPU for higher performance.

The core data structure is a model, that is a collection of layers. The simplest type is the *Sequential model*, a linear stack of layers. More complex architectures also can be achieved using the Keras functional API.

The clustering agent utilizes a sequential model:

```
from keras.models import Sequential

model = Sequential()
```

Stacking layers into a model is done through the *add* function:

```
from keras.layers import Dense

model.add(Dense(128, input_shape=(self.state_size,
            self.action_size), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
```

The first layer will handle the input and has a mandatory parameter defining its size. In this case *input_shape* is provided as a 2-dimensional matrix, where *state_size* is the number of parameters stored in the state and *action_size* is the number of possible actions. The first parameter tells how big the output dimension will be, so in this case the input will be propagated into a 128-dimensional output.

The following two layers are hidden layers (Section 3) with $128$ internal nodes, with *rectified linear unit* (ReLU) activation. The rectifier is an activation function given by the positive part of its argument: $f(x) = x^+ = \max(0, x)$, where $x$ is the input to a neuron. The rectifier was first introduced to a dynamical network in [16]. It has been demonstrated in [14] to enable better training of deeper networks, compared to the widely used activation function prior 2011, the logistic sigmoid [15].

During training overfitting happens, when the ANN goes to memorize the training patterns. In this case the network is weak in generalizing on new datasets. This appears for example, when an ANN is very large, namely it has too many hidden nodes and hence, there are too many weights which need to be optimized.
The dropout for the hidden layers is used to prevent overfitting on the learning dataset. Dropout is a technique that makes some randomly selected neurons ignored during training. Their contribution to the activation of neurons on deeper layers is removed temporally and the weight updates are not applied back to the neurons. If neurons are randomly dropped during training, then others will have to handle the representation, that is required to make predictions, that is normally handled by the dropped elements. This results in multiple independent internal representations for the given features [33]. This way the network becomes capable of better generalization and avoids potential overfitting on the training data.

The output so far will still be a matrix with the same shape as the input. This is flatten into a 1-dimensional array by adding the following layer:

```
model.add(Flatten())
```

Finally to have the output provide the returns on each available actions, the last layer changes the output dimension to *action_size*:

```
model.add(Dense(self.action_size, activation='linear'))
```

Once the model is set up, the learning process can be configured with the *compile* function:

```
model.compile(loss='mse', optimizer=Adam(lr=self.learning_rate)),
```

where *learning_rate* has been set to $0.001$. For the loss function *mean squared error* is used, optimizer is an instance of *Adam* [17] with the mentioned learning rate. The discount rate for future rewards have been set to $\gamma = 0.001$. This way the model will try to select actions, that yield the maximum rewards in the short term. While maximizing the reward in long term can eventually lead to a policy, that computes the communities correctly, choosing it this small makes the model learn to select the correct neighbors faster.

To make a prediction on the current state, the *predict* function is used:

```
model.predict(state.reshape(1, self.state_size,
                               self.action_size))
```

For Keras to work on the input state, it always have to be reshaped into dimensions $(1, state\_size, action\_size)$, while the change always has to keep the same number of state elements.

## 6 Results

Evaluation of the proposed solution is done by processing network clustering on undirected, weighted graphs. These graphs contain real network information, instead of evaluating on physics related datasets (Section 2.3), as it is more suitable for the original Louvain method. Because of this, the modularity can be used as a sort of metric to measure the quality (Subsection 2.4) of the results. Additionally the number of correct predictions and misses are used to describe the deep Q-learning (Section 5) based method's efficiency.

Numerical evaluations are done by generating one iteration on the first level of the dendogram as the top level takes the most time to generate as it is based on all the original input nodes. The GPU implementation of the Louvain method being used was first described in [10].

### 6.1 Dataset

The proposed model, as well as the Louvain clustering works on undirected, weighted graphs. Such graphs can be generated from U.S. Census 2010 and Tiger/Line 2010 shapefiles, that are freely available from [40] and from jet physics information simulated by the AliRoot framework.

#### 6.1.1 Census dataset

The Census dataset contains the following:

- the vertices are the Census Blocks;
- there's an edge between two vertices if the corresponding Census Blocks share a line segment on their border
- each vertex has two weights:
  - Census2010 POP100 or the number of people living in that Census Block
  - Land Area of the Census Block in square meters
- the edge weights are the pseudo-length of the shared borderlines.
- each Census Block is identified by a point, that is given longitudinal and latitudinal coordinates

A census block is the smallest geographical unit used by the United States Census Bureau for tabulation of 100-percent data. The pseudo-length is given by $\sqrt{(x^2 + y^2)}$, where $x$ and $y$ are the differences in longitudes and latitudes of each line segment on the shared borderlines. The final result is multiplied by $10^7$ to make the edge weights integers. For clusterization the node weights are not used.

The matrices used for evaluation contains the information related to New York, Oregon and Texas (Table 1), that was arbitrarily selected from the SuiteSparse Matrix Collection [39]. The graph details can be found in [7].

| | New York | Oregon | Texas |
|---|---|---|---|
| Nodes | $350,169$ | $196,621$ | $914,231$ |
| Edges | $1,709,544$ | $979,512$ | $4,456,272$ |

Table 1: Size of the Census datasets

### 6.1.2 Jet dataset

AliRoot is the Off-line framework for simulation, reconstruction and analysis for CERN's ALICE experiment. The simulation covers all processes of primary collisions and generates the newly created particles, follows through their transportation and calculates the hits in each component.

For the current work the selected dataset is based on the points detected by the system's TPC. The Time Projection Chamber (TPC) detector is the main tracking component of ALICE. Particles passing through this detector ionizes the gas molecules inside and these ionization points are registered [29].

The datasets were simulated with the framework's PbPbbench test application. Three events were generated, the sizes are detailed in Table 2.

| | Event1 | Event2 | Event3 |
|---|---|---|---|
| Nodes | $140,535$ | $139,162$ | $67,778$ |
| Edges | $140,535$ | $139,162$ | $67,778$ |

Table 2: Size of the Jet datasets

Due to the limitations of the proposed solution as was described in Subsection 5.1, in all cases only 4 neighbors are kept for each nodes during the computation.

## 6.2 Precision of the neural network

The precision depends on how well the model can generalize the learned information. In clustering this will highly depend on the structure of the graph. The model described in Section 5 have been trained on a training set built from the Oregon graph. The first 10000 nodes based on the order how they are first mentioned in the original dataset was taken as a subgraph and the Louvain method was applied on it, generating the communities. In each step a matrix was built, where the lines represents the nodes and the columns contains the necessary values for the computation (current community, weight, degree). Learning phase was running for 70 epochs. The ratio of the good and bad predictions for Census data are shown in Table 3 and for jet data are shown in Table 4.

| | New York | Oregon | Texas |
|---|---|---|---|
| Positive | 310,972 | 172,028 | 750,563 |
| Negative | 39,197 | 24,593 | 163,668 |

Table 3: Positive/negative predictions of the model for Census data

The deep learning solution's precision in average is 83,77%. Specifically on the datasets it's respectively 87,4%, 85,7% and 78,2%. Precision can be further increased by running the training for more epochs or by further tune the hyper-parameters. Looking at the number of connected components of each graph, New York has 3, Oregon 1 and Texas 1. The highest number of connected components lead to the highest precision, since subgraphs are containing less nodes, than in graphs with less components. As a consequence of this the model performs better with smaller subgraphs, than on more complex structures.

| | Event1 | Event2 | Event3 |
|---|---|---|---|
| Positive | 122,405 | 123,436 | 56,391 |
| Negative | 18,130 | 15,726 | 11,387 |

Table 4: Positive/negative predictions of the model dor jet data

The average precision in this case is 86,33%. For each event it's respectively 87,1%, 88,7% and 83,2%. This could be further increased by teaching the model on training sets that also contains information on the jet structure.

### 6.3 Modularity comparison

The Louvain method assumes nothing of the input graph. The clusterization can be done without any prior information of the groups being present in the network. The modularity (Subsection 2.4.1) for the Census data is presented (Table 5) for all 3 test matrices for both the Louvain algorithm and the deep Q-learning based solution. The same results are presented for the jet data in Table 6.

| | New York | Oregon | Texas |
|---|---|---|---|
| Louvain | 0.82 | 0.68 | 0.76 |
| DQL | 0.72 | 0.58 | 0.59 |

Table 5: Modularities achieved by Louvain and with the DQL solution for Census data

The modularities showing similar results to the precision of the network: the New York graph has a modularity less with $14,53\%$ compared to the Louvain computation, while Oregon is less with $13,8\%$ and Texas is less with $17,36\%$. This proves, that by loosing from the precision, the qualities of the clusters do not degrade more than, what is lost on the precision.

| | Event1 | Event2 | Event3 |
|---|---|---|---|
| Louvain | 0.76 | 0.78 | 0.81 |
| DQL | 0.66 | 0.69 | 0.67 |

Table 6: Modularities achieved by Louvain and with the DQL solution for jet data

The changes in modularities is comparable to that of the Census dataset results, proving that the generic model learning only on hierarchical data can also work with comparable efficiency on jet related datasets.

## 7 Summary

In this paper a reinforcement learning based model was devised to use a community processor during the generation the dendogram in clusterization tasks. The training dataset was based on the Louvain method's processing and was learned using deep learning from one of the test graphs. The model detailed in Section 5 predicted the next level of the dendogram with $83,77\%$ of precision for the Census dataset and $86.33\%$ for the jet dataset, achieving that, while only being learned for $70$ epochs and on $10000$ nodes, that were selected from the same graph. Reinforcement learning produces a model, that can generalize based on the training data, as such without a broader selection of structures it cannot reach higher efficiency.

## 8 Future work

Increasing the model's effectiveness, the learning process should be extended, so it will generalize on any kind of graph with arbitrary number of children. Also the training set needs to extended with data from graphs with different structures and complexity. The model should be further evaluated on datasets with different structural complexities. A coherent method to test the performance of the new clusterization should be explored. Thanks to the GPU based deep learning frameworks, the learning rate increases in accordance with the power of the underlying GPU, but how the inferencing performance affects the clustering should be further evaluated.

## References

[1] A. Ali, G. Kramer, Jets and QCD: A historical review of the discovery of the quark and gluon jets and its impact on QCD *Eur. Phys. J. H.* **36** (2011) 245–326. [arXiv:1012.2288 [hep-ph]]. ⇒89

[2] D. Bader, J. McCloskey, Modularity and graph algorithms, *SIAM AN10 Minisymposium on Analyzing Massive Real-World Graphs*, 2009, pp. 12-16. ⇒92

[3] J. W. Berry, B. Hendrickson, R. A. LaViolette, C. A. Phillips, Tolerating the community detection resolution limit with edge weighting, *Phys. Rev. E* **83,** 5 (2011) 056119. ⇒92

[4] V. D. Blondel, J-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, *Journal of Statistical Mechanics: Theory and Experiment* **10** (2008) P10008 ⇒91

[5] S. Carani, Yu. L Dokshitzer, M. H. Seymour, B. R. Webher, Longitudinally-invariant $k_\perp$-clustering algorithms for hadron-hadron collisions, *Nuclear Physics B* **406** (1993) 187–224. ⇒91

[6] I. Csabai, F. Czakó, Z. Fodor, Quark- and gluon-jet separations using neural networks, *Phys. Rev. D* **44** 7 (1991) R1905–R1908. ⇒94

[7] T. Davis, Y. Hu, The University of Florida Sparse Matrix Collection, Mathematical Software, Vol 38, Issue 1, 2011, pp 1:1–1:25. ⇒104

[8] B. Denby, Neural networks and cellular automata in experimental high energy physics, *Computer Physics Communications* **49** (1988) 429–448. ⇒94

[9] B. Denby, Neural networks in high energy physics: a ten year perspective, *Computer Physics Communications* **119** (1999) 219. ⇒92

[10] R. Forster, Louvain community detection with parallel heuristics on GPUs, *20th Jubilee IEEE International Conference on Intelligent Engineering Systems* **20** (2016) doi: 10.1109/INES.2016.7555126 ⇒91, 103

[11] R. Forster, A. Fülöp, Jet browser model accelerated by GPUs, *Acta Univ. Sapientiae Informatica* **8,** 2 (2016) 171–185. ⇒91

[12] R. Forster, A. Fülöp, Parallel $k_t$ jet clustering algorithm, *Acta Univ. Sapientiae Informatica* **9,** 1 (2017) 49–64. ⇒91

[13] R. Forster, A. Fülöp, Hierarchical $k_t$ jet clustering for parallel achitectures, *Acta Univ. Sapientiae Informatica* **9,** 2 (2017) 195–213. ⇒87, 92

[14] X. Glorot, A. Bordes, Y- Bengio, Deep sparse rectifier neural networks, *Proc 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011*, Fort Lauderdale, FL, USA. Volume 15 of JMLR:W&CP 15. ⇒101

[15] J. Han. C. Moraga, The influence of the sigmoid function parameters on the speed of backpropagation learning, *IWANN '96 Proc. of the Int. Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*, 1995, pp. 195-201. ⇒102

[16] R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, H. S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature* **405** (2000) 947-951. ⇒101

[17] D. P. Kingma, J. B. Adam, A method for stochastic optimization, 2014, arXiv:1412.6980 ⇒102

[18] H. Kolanoski, Application of artifical neural networks in particle physics, *Nuclear Instruments and Methods in Physics Research A* **367** (1995) 14–20. ⇒92

[19] P. T. Komiske, E. M. Metodiev, M. D. Schwartz, Deep learning in color: towards automated quark/gluon jet discrimination, *J. High Energy Physics* (2017) 110. ⇒94

[20] K. J. C. Leney, A neural-network clusterisation algorithm for the ATLAS silicon pixel detector, *J. of Physics: Conbnference Series* **523** (2014) 012023. ⇒94

[21] L. Lönnblad, C. Peterson, T. Rögnvaldsson, Using neural networks to identify jets, *Nuclear Physics* **B349** (1991) 675–702. ⇒94

[22] H. Lu, Mahantesh Halappanavar, A. Kalyanaraman, Parallel heuristics for scalable community detection, *Parallel Computing* **47** (2015) 1937. ⇒91

[23] V. Mnih et al., Playing Atari with deep reinforcement learning, 2013, arXiv:1312.5602 ⇒99

[24] V. Mnih et al., Human-level control through deep reinforcement learning, *Nature*, 2015, doi:10.1038/nature14236 ⇒99

[25] T. Muta, *Foundation of Quantum Chrodinamics*, World Scientific Press, 1986. ⇒88

[26] M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks, *Phys. Rev. E* **69** 2 (2004) 026113. ⇒92

[27] M. E. Peskin, D. V. Schroeder, *Quantum Field Theory*, Westview Press, 1995. ⇒87

[28] C. Peterson, Track finding with neural networks, *Nuclear Instruments and Methods* **A279** (1988) 537. ⇒94

[29] D. Rohr, S. Gorbunov, A. Szostak, M. Kretz, T. Kollegger, T. Breitner, T. Alt, ALICE HLT TPC Tracking of Pb-Pb Events on GPUs, *Journal of Physics: Conference Series* **396** (2012) doi:10.1088/1742-6596/396/1/012044 ⇒87, 104

[30] G. P. Salam, Towards jetography, *Eur. Phys. J.* C**67** (2010) 637-686. ⇒88

[31] S. Salur, Full Jet reconstruction in heavy ion collisions, *Nuclear Physics A* **830,** 1-4 (2009) 139c–146c. ⇒89

[32] K. E. Selbach, Neural network based cluster reconstruction in the ATLAS pixel detector, *Nuclear Instruments and Methods in Physics Research A* **718** (2013) 363–365. ⇒94

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *JMLR* **15** (2014) 1929-1958. ⇒102

[34] G. Sterman, S. Weinberg, Jets from quantum chromodynamics, *Phys. Rev. Lett.* **39** (1977) 1436. ⇒89, 90

[35] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, A Bradford Book, 1998, ISBN: 978-0262193986 ⇒94, 99

[36] G. Swirszcz, W. M. Czarnecki, R. Pascanu, Local minima in training of neural networks, 2016, arXiv:1611.06310 ⇒100

[37] V. A. Traag, P. Van Dooren, Y. Nesterov, Narrow scope for resolution-limit-free community detection, *Phys. Rev. E* **84,** 1 (2011) 016114. ⇒92

[38] ∗ ∗ ∗ Keras: The Python Deep Learning library ⇒101

[39] ∗ ∗ ∗ SuiteSparse Matrix Collection ⇒104

[40] ∗ ∗ ∗ United States Census Bureau ⇒103

*Received: April 5, 2018 • Revised: July 30, 2018*

Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 110–132

DOI: 10.2478/ausi-2018-0007

# Low and high grade glioma segmentation in multispectral brain MRI data

László SZILÁGYI
Sapientia University
Târgu Mureş, Romania
email: `lalo@ms.sapientia.ro`

David ICLĂNZAN
Sapientia University
Târgu Mureş, Romania
email: `iclanzan@ms.sapientia.ro`

Zoltán KAPÁS
Sapientia University
Târgu Mureş, Romania
email: `zoltankapas@yahoo.com`

Zsófia SZABÓ
Sapientia University
Târgu Mureş, Romania
email: `sz_zsokaa@yahoo.com`

Ágnes GYŐRFI
Sapientia University
Târgu Mureş, Romania
email: `gyorfiagnes@ms.sapientia.ro`

László LEFKOVITS
Sapientia University
Târgu Mureş, Romania
email: `lefkolaci@ms.sapientia.ro`

**Abstract.** Several hundreds of thousand humans are diagnosed with brain cancer every year, and the majority dies within the next two years. The chances of survival could be easiest improved by early diagnosis. This is why there is a strong need for reliable algorithms that can detect the presence of gliomas in their early stage. While an automatic tumor detection algorithm can support a mass screening system, the precise segmentation of the tumor can assist medical staff at therapy planning and patient monitoring. This paper presents a random forest based procedure trained to segment gliomas in multispectral volumetric MRI records. Beside the four observed features, the proposed solution uses 100 further

**Computing Classification System 1998:** I.5.3
**Mathematics Subject Classification 2010:** 68T10
**Key words and phrases:** machine learning, random forest, image segmentation

features extracted via morphological operations and Gabor wavelet filtering. A neighborhood-based post-processing was designed to regularize and improve the output of the classifier. The proposed algorithm was trained and tested separately with the 54 low-grade and 220 high-grade tumor volumes of the MICCAI BRATS 2016 training database. For both data sets, the achieved accuracy is characterized by an overall mean Dice score $> 83\%$, sensitivity $> 85\%$, and specificity $> 98\%$. The proposed method is likely to detect all gliomas larger than $10\,\mathrm{mL}$.

## 1 Introduction

Most brain tumors are diagnosed in a certain advanced stage, when the symptoms convince the patient to go to the doctor. The average brain tumor patient lives 14 months after the diagnosis. The development of imaging devices and computers make it possible to elaborate intelligent automated procedures that would allow for regular screening of a larger population and detecting most tumors in an earlier phase. Beside establishing the diagnosis, the automatic segmentation and quantitative analysis can assist therapy planning and evolution tracking of the tumor. However, automatic tumor segmentation is not only utmost important task, but also a very challenging one, because of the high variety of anatomical structures and low contrast of current imaging techniques, which make the difference between normal regions and the tumor hardly recognizable for the human eye [6].

Magnetic resonance imaging (MRI) is the preferred imaging device in brain tumor screening, due to its better contrast and relatively fine resolution. However, it also bears difficulties like the possible presence of intensity inhomogeneity [27], and the relative intensity values that vary from device to device and from patient to patient [19]. The MICCAI Brain Tumor Segmentation Challenge, organized yearly since 2012, intensified the research in this topic and led to several important solutions, which are usually assisted by the use of prior information, and employ various image processing and pattern recognition methodologies. Asman et al. [2] applied a non-parametric intensity analysis in combination with a segmentation based on multiple atlases. Ghanavati et al. [5] provided a solution using the AdaBoost classifier to distinguish tumor voxels from normal ones using features based on intensity, texture, and symmetry. Hamamci et al. [7] proposed a cellular automata driven method that produces segmentation based on level sets. Sachdeva et al. [22] deployed a content based active contour model relying on intensity and texture features extracted from the histogram and co-occurrence matrix of the MRI data. Njeh et al. [18] introduced a graph cut based solution that performs distribution matching, which is highly efficient because of using rather global than pixelwise information. Zhang et al. [29] proposed a support vector machine based procedure to follow the evolution of brain tumors over time. Tustison et al. [26] combined random forests with symmetry based features to segment brain tumors. Szilágyi et al. [25] provided a semi-supervised framework for the fuzzy c-means clustering algorithm to produce accurately segmented tumors. Kanas et al. [13] combined a clustering based preprocessing with a multi-parametric random walker segmentation. Havaei et al. [8] developed an automatic brain tumor segmentation procedure based on deep neural networks that exploits both local and global contextual features simultaneously. Pereira et al. [20] proposed a convolutional neural network solution exploiting small kernels and successfully applied it for brain tumor segmentation. Menze et al. [17] combined a Gaussian mixture model with the expectation maximization (EM) algorithm to achieve an accurate segmentation. Another Gaussian mixture based accurate solution was given by Juan-Albarracín et al. [12]. Islam et al. [11] employed multifractional Brownian motion features to provide patient-independent characterization of tumor tissues and applied the AdaBoost algorithm for tissue segmentation. Shin et al. [23] proposed deep convolutional neural networks and successfully combined it with transfer learning. Huang et al. [9] provided a brain tumor segmentation framework employing local independent projection-based classification. Lê et al. [15] proposed a brain tumor segmentation procedure based on a tumor growth model. Pinto et al. [21] employed extremely random trees to provide a hierarchical solution to the low-grade glioma segmentation problem. Zaouche et al. [28] provided a semi-supervised low-grade glioma segmentation based on specially designed spatial edge filters and maximum likelihood optimization. For further information on current brain tumor segmentation techniques, there are available recent reviews [6, 10].

In a previous paper [14] we have presented a preliminary study on the use of random forests in the detection and segmentation of high-grade gliomas. Each voxel was characterized by a 16-element feature vector, including minimum, maximum, and median values computed from the neighborhood of the voxel. The procedure proposed in that study was evaluated using the 220 high-grade tumor volumes from the MICCAI BRATS 2016 data set. The best overall Dice Score was found 81%. As a further development [24], we proposed a random forest solution trained and tested using 104-element feature vectors that included various computed morphological and Gabor wavelet features. Our main goal in this paper is to perform a detailed evaluation of the solution based on 104-element feature vector using all low-grade (LG) and high-grade

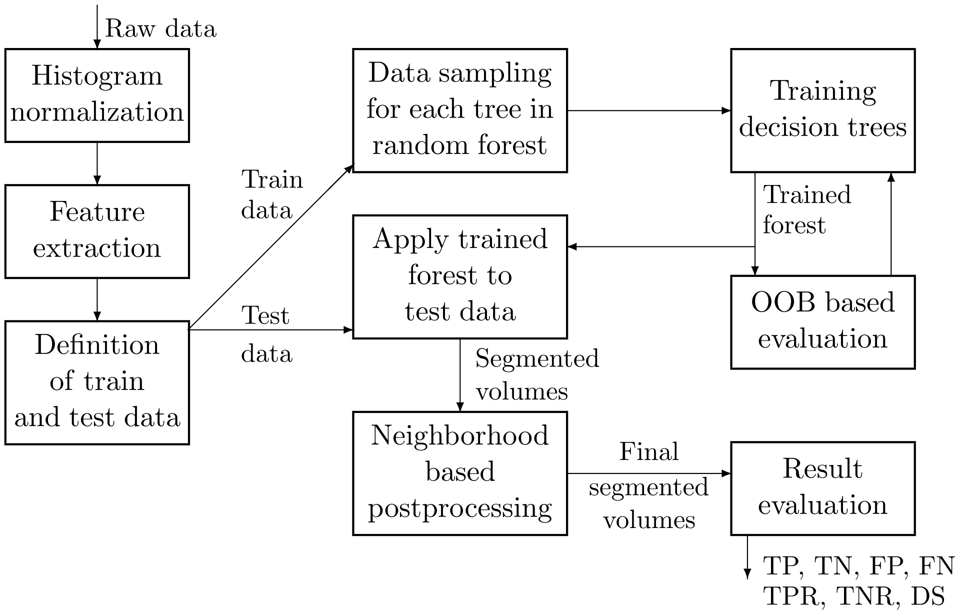


Figure 1: Block diagram of the proposed method.

(HG) tumor volumes of the MICCAI BRATS 2016 train database.

The rest of this paper is structured as follows: Section 2 gives details on the proposed methodology. Section 3 exhibits and discusses the achieved results. Finally, Section 4 concludes the investigation.

## 2 Materials and Methods

Our goal was to elaborate an accurate segmentation procedure based on a machine learning algorithm, applicable for both LG and HG brain tumor volumes separately. This paper presents results obtained using a random forest approach, combined with histogram normalization, Gabor feature extraction for texture characterization, and a neighborhood-based post-processing. The trees of the random forest are trained to separate the whole tumor from normal tissues. The structure of the elaborated segmentation procedure is presented in Fig. 1.

| Property | LG volumes | HG volumes |
|---|---|---|
| Number of data volumes | 54 | 220 |
| Average size of whole tumor ($mL$) | 101.1 | 110.6 |
| Minimum size of whole tumor ($mL$) | 18.9 | 8.5 |
| Maximum size of whole tumor ($mL$) | 265.3 | 318.4 |
| Total number of negative voxels | 72.2M | 311.5M |
| Total number of positive voxels | 5.46M | 24.32M |

Table 1: Main attributes of the input data set

## 2.1 MICCAI BRATS data sets

Fully anonymized multimodal MR image data was obtained from the MICCAI 2016 Challenge on Multimodal Brain Tumor Segmentation [16]. This database contains multi-contrast MR scans of 274 glioma patients, out of which 54 having low-grade and 220 having having-grade glioma lesions. For each patient, multimodal ($T1$, $T2$, FLAIR, and post-Gadolinium $T1$ ($T1C$)) MR images were recorded and linearly co-registered to the T1 contrast image. All data volumes were skull stripped, and interpolated to $1$ $mm$ isotropic resolution. Each record contains approximately $1.5$ millions of true tissue voxels, out of which the rate of positives ranges from $0.5\%$ to $20\%$. Manual annotation produced by human experts is provided for all voxels. Further technical details of the data set are given in Table 1.

## 2.2 Data preprocessing

A major drawback of MR imaging consists in the lack of a standard scale of image intensities. This is why we need to map the histogram of each data channel of BRATS volumes onto a uniform scale. Although literature contains various recommendation is this issue [19, 4], we opted to employ a simple linear transform $x \to \alpha x + \beta$ to all intensities, where parameters $\alpha$ and $\beta$ were established separately for each volume and each data channel such a way that the 25-percentile and 75-percentile values became $600$ and $800$, respectively. Further on, a minimum and a maximum intensity barrier was enforced at $200$ and $1200$, respectively. This approximately corresponds to a $10$-bit resolution in each data channel.

Most voxels of the MRI records have valid nonzero intensity in all data channels. However, there are voxels with one or more missing values. We considered

| Neighborhood | $3 \times 3 \times 3$ | $3 \times 3$ | $5 \times 5$ | $7 \times 7$ | $9 \times 9$ | $11 \times 11$ | Total |
|---|---|---|---|---|---|---|---|
| Average | 4 | 4 | 4 | 4 | 4 | 4 | 24 |
| Maximum | 4 | | | | | | 4 |
| Minimum | 4 | | | | | | 4 |
| Median | | 4 | 4 | 4 | 4 | 4 | 20 |
| Gradient | | | | 16 | | | 16 |
| Gabor wavelet | | | | | | 32 | 32 |
| Total | 12 | 8 | 8 | 24 | 8 | 40 | 100 |

Table 2: Inventory of computed features. All four data channels were involved equally.

that the region of interest (ROI) in the BRATS volumes includes all voxels that have at least one nonzero value in any of the observed data channels. Missing values were replaced by the mean intensity value of existing neighbors within the 26-element immediate spatial neighborhood, or the grand mean of the given data channel whenever no neighbors with correct intensity were found in the neighborhood.

Although the four observed features of each voxel bear a lot more information than any one of them, there is an acute need to extend the feature vectors with further computed features. A total number of 100 computed features were added to the feature vector describing each voxel, according to the inventory given in Table 4. For each of the four observed intensities (T1, T2, T1C, FLAIR), six average, five median, one minimum, one maximum, four gradient values, and further eight Gabor features were extracted. All computed feature values were linearly scaled into the $[200, 1200]$ interval. This way, together with the four observed features, each voxel is described by a 104-element feature vector. These feature vectors are used by the classification stage of the proposed segmentation procedure.

## 2.3 Data classification

Binary decision trees (BDT) of unlimited depth can describe any hierarchy of crisp (non-fuzzy) two-way decisions [1]. Given an input data set of vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,m}]^T$, a BDT can be employed to learn the classification that corresponds to any set of labels $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. The classification learned by the BDT can be perfect if there are no identical training vectors with different labels, that is, $\mathbf{x}_i = \mathbf{x}_j$

implies $\lambda_i = \lambda_j$, $\forall i, j \in \{1, 2, \dots, n\}$. The BDT is built during the training process. Initially the tree consists of a single node, the root, which has to make a decision regarding all $n$ train data vectors. If not all $n$ vectors have the same label, which is likely to be so, then the set of data is not homogeneous, and there is a need for a separation. The decision will compare a single chosen feature, the one with index $k$ ($1 \le k \le m$), of the input vectors with a certain threshold $\alpha$, and the comparison will separate the vectors into two subgroups: those with $x_{i,k} < \alpha$ ($i = 1 \dots n$), and those with $x_{i,k} \ge \alpha$ ($i = 1 \dots n$). The root will then have two child nodes, each corresponding to one of the possible outcomes of the above decision. The left child will further classify those $n_1$ input vectors, which satisfied the former condition, while the right child those $n_2$ ones that satisfied the latter condition. Obviously, we have $n_1 + n_2 = n$ with $n_1 > 0$ and $n_2 > 0$. For both child nodes, the procedure is the same as it was for the root. When at a certain point of the learning algorithm, all vectors being classified by a node have the same label $\lambda_p$, then the node is declared a leaf node, which is attributed to the class with index $p$. Another case when a node is declared leaf node is when all vectors to be separated by the node are identical, so there is no possible condition to separate the vectors. In this case, the label of the node is decided by the majority of labels, or if there is no majority, a label should be chosen from the present ones. In our application, this kind of rare leaves are labeled as tumor.

The separation of a finite set of data vectors always terminates in a finite number of steps. The maximum depth of the tree highly depends on the way of establishing the separation condition in each node. Our application uses an entropy based criterion to choose the separation condition. Whenever a node has to establish its separation criterion for a subset of vectors $\overline{\mathbf{X}} \subseteq \mathbf{X}$ containing $\overline{n}$ items with $1 < \overline{n} \le n$, the following algorithm is performed:

1. Find all those features which have at least two different values in $\overline{\mathbf{X}}$.
2. Find all different values for each feature and sort them in increasing order.
3. Set a threshold candidate at the middle of the distance between each consecutive pair of values for each feature.
4. Choose that feature and that threshold, for which the entropy-based criterion
$$E = \overline{n}_1 \log \frac{\overline{n}_1}{\overline{n}} + \overline{n}_2 \log \frac{\overline{n}_2}{\overline{n}} \tag{1}$$
gives the minimum value, where $\overline{n}_1$ ($\overline{n}_2$) will be the cardinality of the subset of vectors $\overline{\mathbf{X}}_1$ ($\overline{\mathbf{X}}_2$), for which the value of the tested feature is less than (greater or equal than) the tested threshold value.

After having the BDT trained, it can be applied for the classification of test data vectors. Any test vector is first fed to the root node, which according to the stored condition and the feature values of the vector, decides towards which child node to forward the vector. This strategy is followed then by the chosen child node, and the vector will be forwarded to a further child. The classification of a vector terminates at the moment when it is forwarded to a leaf node of the tree. The test vector will be attributed to the class indicated by the labeling of the reached leaf node.

Binary decision trees were trained to separate tumor voxels from negative ones. Due to practical reasons, negative voxels were randomly subsampled to $12\%$ for the BDT training process. Random forests were trained according to the following parameters:

1. The number of trees in the forest denoted by $n_T$. This parameter was usually set to $255$. Experiments proved this number of trees more than necessary for good accuracy.
2. The number of data vectors used to train each tree of the forest, denoted by $n_P$. Typical values of this parameter ranged from $10$ thousand to $500$ thousand.
3. The rate (percentage) of negative labeled data within the training set, denoted by $p_n$.
4. The threshold of positive votes $\theta_p$ (expressed in percentage) necessary to assign a voxel to the class of positives. Making a decision according to majority voting would mean using a $\theta_p = 50\%$ threshold, but slightly shifted values of $\theta$ may lead to better accuracy.

Ideal parameter settings were identified using the so-called out-of-bag (OOB) data, as recommended by Breiman in [3]. Testing on OOB data allowed us to preselect those forests that were likely to produce high accuracy, and discard those that were prone to severe misclassifications. The best performing trees achieved $93-95\%$ correct decisions, while the most accurately classifying forests scored $96-98\%$ in labeling the OOB data.

## 2.4 Post-processing

A posterior relabeling scheme was implemented as follows. The input data of the post-processing step consisted in the labels provided by the random forest to all voxels in the test volume. For each voxel, the number of tumor labeled neighbors ($\nu_T$) and the number of all neighbors ($\nu_{\text{All}}$) were extracted, using a predefined neighborhood. The final label of a voxel was set to tumor if and only if $\nu_T/\nu_{\text{All}} > \theta_n$. The optimal value of this threshold was established based on tests performed on OOB data. The ideal neighborhood to be employed in post-processing was identified as the cubic $11 \times 11 \times 11$ sized one for LG tumor volumes, and $9 \times 9 \times 9$ for HG tumor volumes.

## 2.5 Evaluation of accuracy

We employed the Dice score ($\mathsf{DS}$) as the main indicator of accuracy, defined as

$$\mathsf{DS} = \frac{2 \times \mathsf{TP}}{2 \times \mathsf{TP} + \mathsf{FP} + \mathsf{FN}} \in [0, 1] \ , \tag{2}$$

where $\mathsf{TP}$, $\mathsf{FP}$, and $\mathsf{FN}$ stand for the number of true positives, false positives, and false negatives, respectively. Fine accuracy is reflected by $\mathsf{DS}$ values close to 1, but in this brain tumor segmentation problem, $\mathsf{DS}$ values around $0.94$ are considered ideal [16], due to inter-rater differences that are present in the ground truth. Further on, sensitivity (or true positive rate, $\mathsf{TPR}$) defined as

$$\mathsf{TPR} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN}} \ , \tag{3}$$

specificity (or true negative rate) defined as

$$\mathsf{TNR} = \frac{\mathsf{TN}}{\mathsf{TN} + \mathsf{FP}} \ , \tag{4}$$

and the rate of correct decisions

$$\mathsf{ACC} = \frac{\mathsf{TP} + \mathsf{TN}}{\mathsf{TP} + \mathsf{TN} + \mathsf{FP} + \mathsf{FN}} \tag{5}$$

were used as secondary accuracy indicators, where $\mathsf{TN}$ represents the number of true negatives.

If we denote by $\mathsf{TP}_i$, $\mathsf{TN}_i$, $\mathsf{FP}_i$, and $\mathsf{FN}_i$, the true/false positives/negatives obtained at testing volume number $i$ ($i = 1 \dots p$, where $p$ is the number of volumes), then we define average Dice score as

$$\widetilde{\mathsf{DS}} = \frac{1}{p} \sum_{i=1}^{p} \mathsf{DS}_i = \frac{1}{p} \sum_{i=1}^{p} \frac{2 \times \mathsf{TP}_i}{2 \times \mathsf{TP}_i + \mathsf{FP}_i + \mathsf{FN}_i} \ , \tag{6}$$

and overall Dice score as:

$$\overline{\mathsf{DS}} = \frac{2 \times \sum\limits_{i=1}^{p} \mathsf{TP}_i}{2 \times \sum\limits_{i=1}^{p} \mathsf{TP}_i + \sum\limits_{i=1}^{p} \mathsf{FP}_i + \sum\limits_{i=1}^{p} \mathsf{FN}_i} \ . \tag{7}$$

Similarly, we will compute overall and average values for the sensitivity ($\overline{TPR}$, $\widetilde{TPR}$) and the specificity ($\overline{TNR}$,$\widetilde{TNR}$).

## 3 Results and Discussion

The proposed algorithm was validated using the total number of 220 HG and 54 LG tumor volumes separately. Both volume sets were ordered randomly and then divided into two groups which were called the even and the odd group. Volumes from the even (odd) group were segmented with random forests trained with all volumes from the odd (even) group. The main parameters of the algorithm were each given several values in a suitable range: train data size between 1k and 500k pixels per tree, rate of negative train data between 85% and 95%. The number of trees was set to 255, but smaller forests were also evaluated via omitting some of the trees. The rate of positive votes (given by the trees of the forest) that is necessary to declare a pixel positive, was also investigated in the range between 20% and 80%, to optimize the final decision. Accuracy results were obtained for all image volumes. Average, median, and overall values of main statistical accuracy indicators were computed as presented in Section 2.5. Table 3 summarizes the main accuracy indicator values for both LG and HG tumor volumes.

Figure 2 presents the overall Dice score values ($\overline{\text{DS}}$) obtained for the output of the random forest, plotted against the training data size, for various values of the negative data rate ($p_n$) situated between 86% and 95%. In case of LG data, the RF classifier performed best at $p_n = 93\%$, achieving $\overline{\text{DS}} > 81\%$. In case of HG data, the accuracy seems to saturate at $p_n = 94\%$, but the obtained Dice scores are lower. Larger training data size usually leads to better accuracy, but this tendency saturates around 300k pixels per tree in LG data. Figure 3 shows the overall Dice scores obtained after post-processing. Validating the labels given by the random forest to each pixel visibly helped more in HG volumes.

Figures 4 and 5 exhibit the same Dice scores as Figs. 2 and 3, but here the overall Dice scores are plotted against the rate of negatives in the training data $p_N$, and each curve stands for a certain training data size. These graphs also show us that the random forest classified LG data better than HG data, but the post-processing improved the overall accuracy in HG, thus the final overall Dice scores are close to each other. Post-processing improves the DS by approximately 2% in LG volumes and by 3% in HG volumes. The accuracy difference between the largest and smallest training data size is lower in HG

| Accuracy Indicator | LG volumes | | HG volumes | |
|---|---|---|---|---|
| | Before PP | After PP | Before PP | After PP |
| Overall $\overline{\text{DS}}$ | 81.0% | 83.8% | 81.1% | 83.6% |
| Average $\widetilde{\text{DS}}$ | 77.0% | 81.3% | 76.2% | 80.2% |
| Median DS | 81.0% | 84.6% | 80.3% | 85.5% |
| Overall $\overline{\text{TPR}}$ | 76.7% | 84.8% | 74.5% | 83.2% |
| Average $\widetilde{\text{TPR}}$ | 77.0% | 81.3% | 70.7% | 77.6% |
| Median TPR | 81.0% | 84.6% | 75.4% | 85.1% |
| Overall $\overline{\text{TNR}}$ | 99.03% | 98.64% | 99.28% | 98.86% |
| Average $\widetilde{\text{TNR}}$ | 99.04% | 98.64% | 99.28% | 98.86% |
| Median TNP | 99.33% | 98.84% | 99.55% | 99.25% |
| DS > 80% | 30 of 54 | 42 of 54 | 114 of 220 | 144 of 220 |
| DS > 85% | 16 of 54 | 26 of 54 | 89 of 220 | 111 of 220 |
| DS > 90% | 6 of 54 | 12 of 54 | 48 of 220 | 56 of 220 |

PP stands for post-processing.

Table 3: Main accuracy indicators

data, probably due to the largest number in data volumes. The optimal rate of negatives in the training data for HG volumes is $p_N = 93\%$ at any tested train data size, while for the LG volumes it varies between 92% and 94%. Below 10k training pixels per tree, there is a sudden drop in accuracy.

Figure 6 exhibits the final overall sensitivity values, for LG and HG volumes separately, plotted against the rate of negatives in the training data ($p_N$), for training data sizes ranging from 10k to 500k. Sensitivity values have a slight dropping tendency as $p_N$ rises. Sensitivity is higher in HG volumes, especially at smaller training data sizes.

Figure 7 presents the sensibility, specificity, and Dice score values obtained for individual LG and HG records, sorted in increasing order of the accuracy indictors. Sensitivity and Dice score values range between 30% and 100%, but they do not follow a uniform distribution. In both LG and HG data, there is a small subset (10—15%) of volumes which lead to mediocre resuts. Most of these volumes have a lot of missing data. The graphs presented in Fig. 7 suggest that accuracy indicators have a distribution that grants higher median value than the average. Numeric values listed in Table 3 confirm this suggestion. Specificity is over 95% in most of the cases, having its average and median value between 98.5% and 99%. It is important to have such a high specificity,

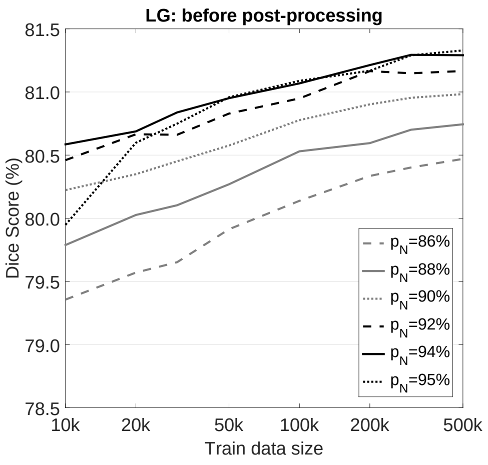


Figure 2: Overall Dice score plotted against the train data size, at various values of the rate of negative train data $p_n$, and $n_T = 255$ trees in each random forest, without post-processing. Left and right panel exhibit the results obtained for the 54 LG and 220 HG tumor volumes, respectively.

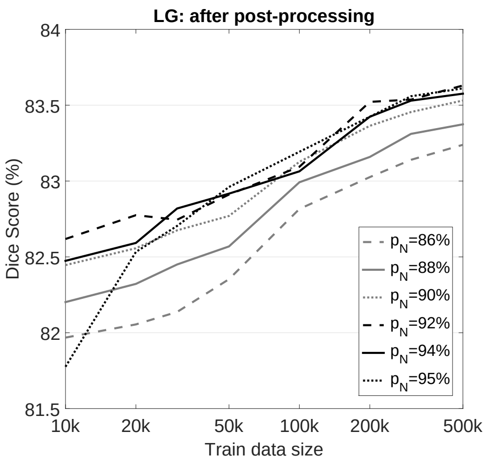


Figure 3: Improved values of the overall Dice scores exhibited in Fig. 2, obtained after post-processing.

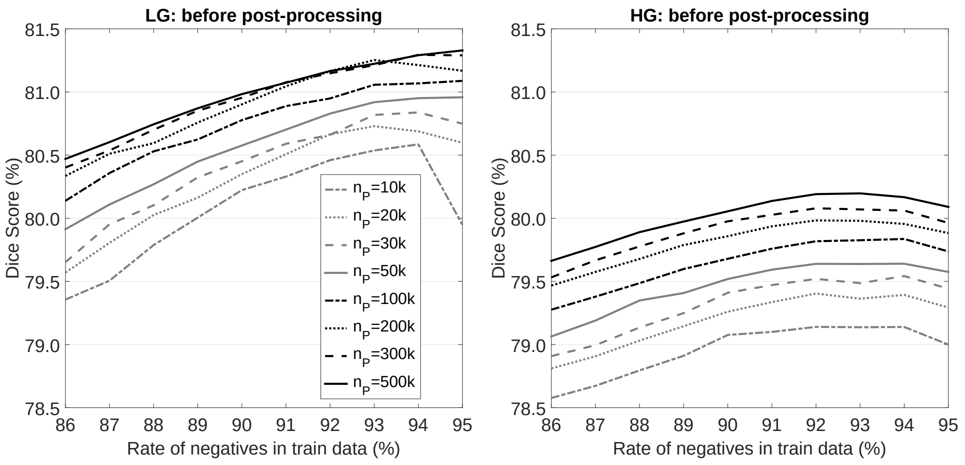


Figure 4: Overall Dice score plotted against the rate of negative train data $p_n$, at various values of the train data size, and $n_T = 255$ trees in each random forest, without post-processing. Left and right panel exhibit the results obtained for the 54 LG and 220 HG tumor volumes, respectively.

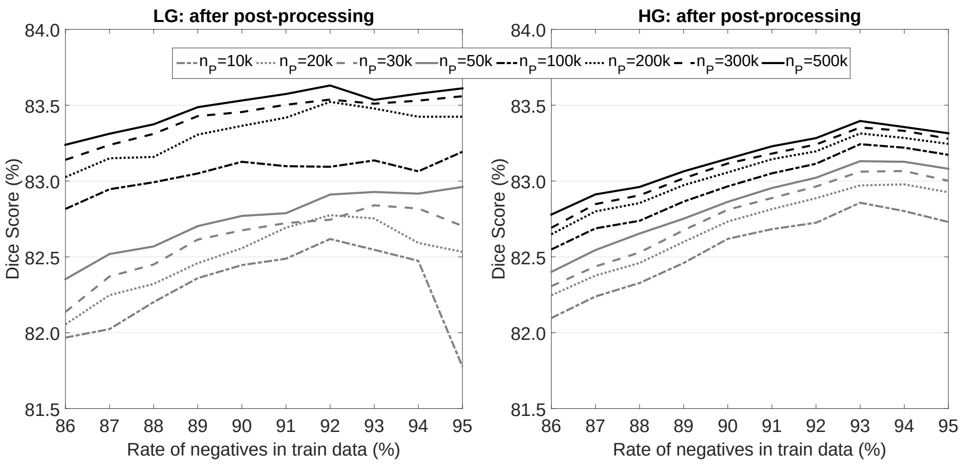


Figure 5: Improved values of the overall Dice scores exhibited in Fig. 4, obtained after post-processing.

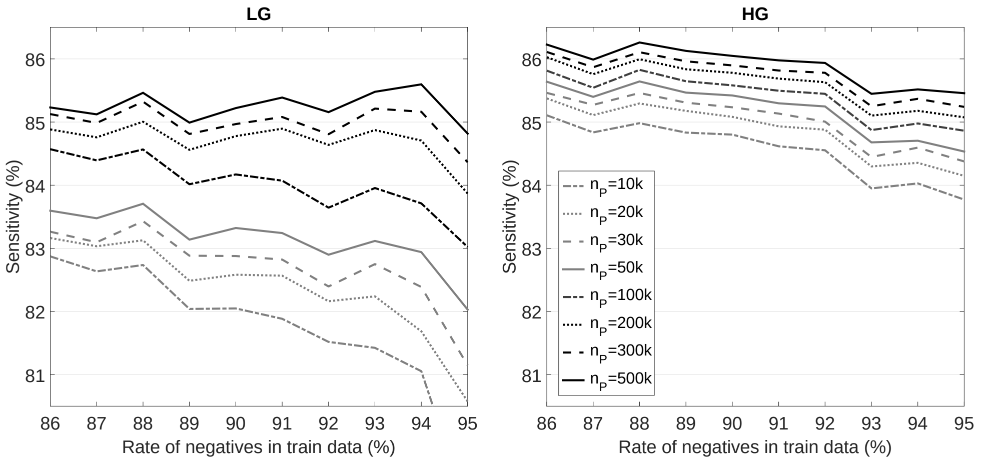


Figure 6: Overall Sensitivity plotted against the rate of negative train data $p_n$, at various values of the number of pixels used to train each random tree. These results were obtained using $n_T = 255$ trees in each random forest, after post-processing. Left and right panel exhibit the results obtained for the 54 LG and 220 HG tumor volumes, respectively.

because otherwise we would have a lot of false positives due to the high rate of negative data.

Figure 8 depicts the Dice score values obtained for individual MRI records, before and after post-processing, sorted in the increasing order of the Dice score. Left and right panels show LG ang HG volumes, respectively. Post-processing seems to help most volumes, especially those with lower Dice score value. When the random forest classifier produces a Dice score over 90%, post-processing may slightly reduce the accuracy. Figure 9 exhibits similar curves for the Sensitivity instead of the Dice score. Here the post-processing proves highly effective, as all Sensitivity values rise during the last processing step. Dice scores of individual volumes improve by 4% in average for both LG and HG volumes, while Sensitivity values rise by 4% and 7% for LG and HG volumes, respectively. Figure 10 plots individual Dice score values after post-processing vs. before post-processing, for LG volumes in the left panel and HG volumes in the right panel. Figure 11 uses the same representation for Sensitivity values instead of Dice scores. Each cross ($\times$) situated above the diagonal drawn in dashed line indicates a case where post-processing improved

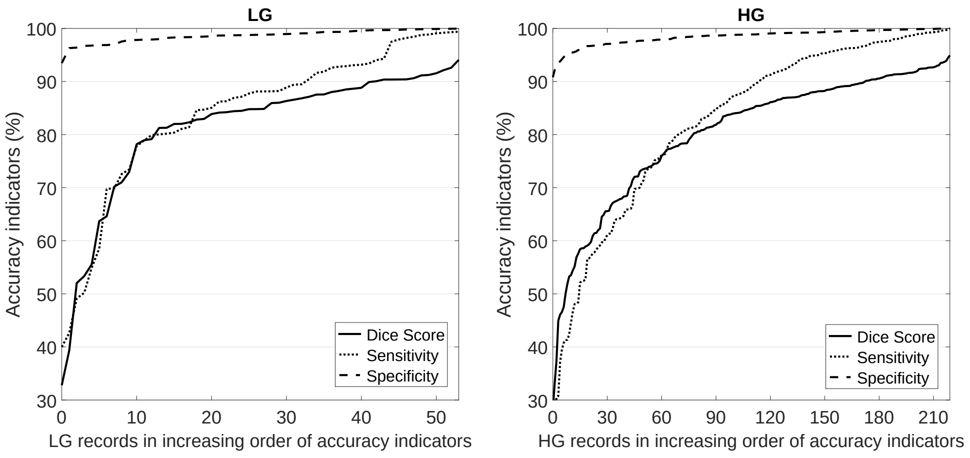


Figure 7: The three main accuracy indicators obtained for each LG (left panel) and HG (right panel) tumor volume separately, after post-processing. Accuracy indicator values were sorted in increasing order.

the value of the accuracy indicator. Figure 10 also contains crosses below the diagonal, which means that the Dice score does not always improve during post-processing. Even if the accuracy is damaged in some cases, the average effect of post-processing is beneficial (Table 3).

Figure 12 plots the individual Dice scores obtained for each LG and HG volume against the size of the tumor, showing the difference between the output of the random forest classifier (left column) and the final post-processed result (right column). There is a general rule that the larger the tumor the better the chances of detection and accurate segmentation. The identified linear trends show that the strongest effect of post-processing occurs in case of small tumors.

Figure 13 shows detailed final result obtained on two MRI slices originating from LG volumes. The top row shows a slice whose segmentation was successful, with a very reduced number of misclassified pixels. The bottom row exhibits a slice where the segmentation of the tumor was visibly mistaken, with a large area of false negatives (unidentified part of the tumor), but the presence of the tumor can be reliably detected based on this segmentation, because most of the tumor pixels were identified.

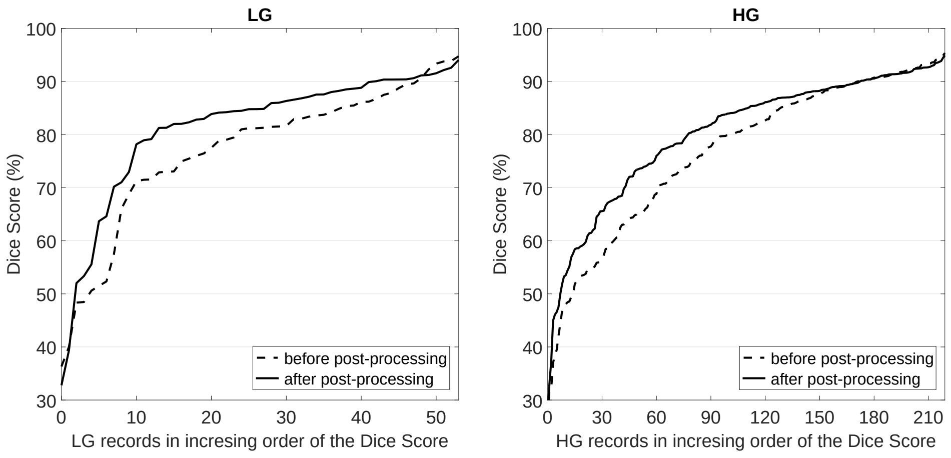


Figure 8: The effect of post-processing upon individual Dice score values obtained for LG (left panel) and HG (right panel) tumor volumes separately. Accuracy indicator values were sorted in increasing order.

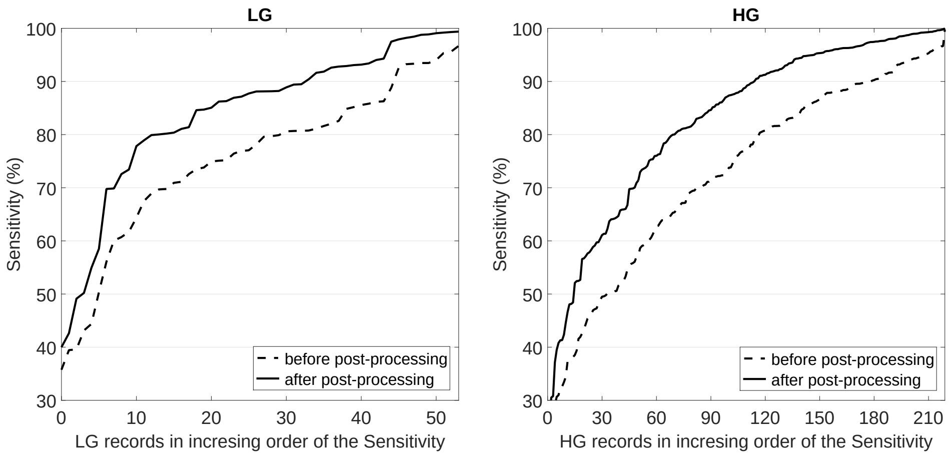


Figure 9: The effect of post-processing upon individual Sensitivity values obtained for LG (left panel) and HG (right panel) tumor volumes separately. Accuracy indicator values were sorted in increasing order.

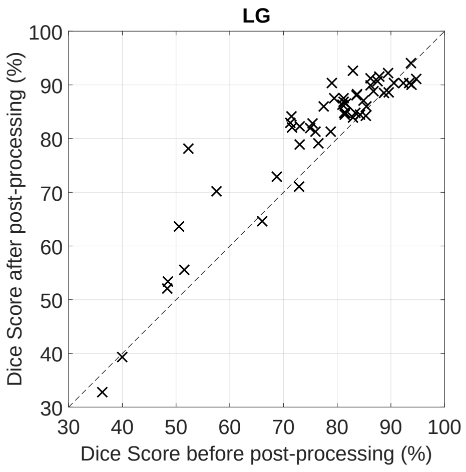
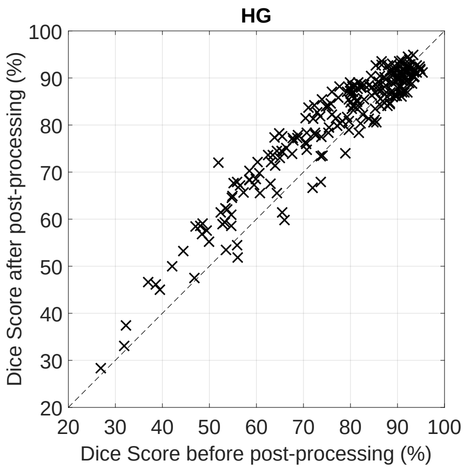


Figure 10: Dice scores obtained for individual LG (left panel) and HG (right panel) tumor volumes, plotted after post-processing vs. before post-processing.



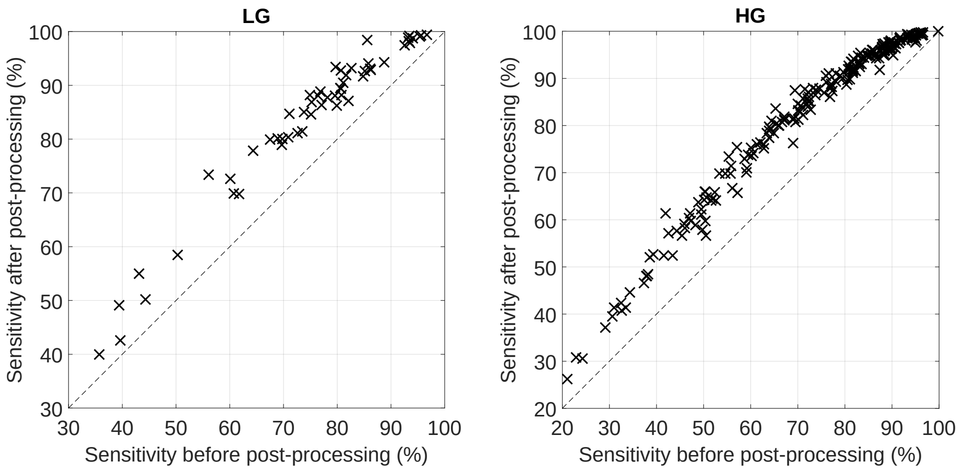


Figure 11: TPR values obtained for individual LG (left panel) and HG (right panel) tumor volumes, plotted after post-processing vs. before post-processing.

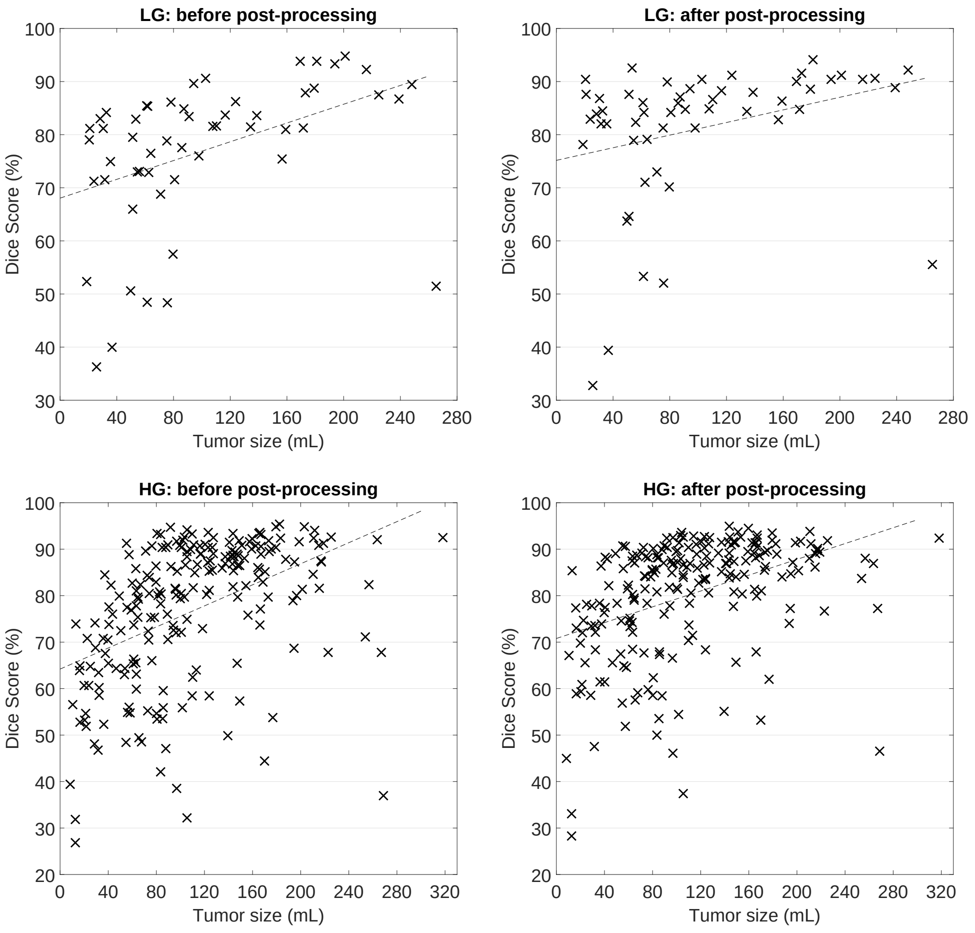


Figure 12: Dice score values obtained for individual LG (top row) and HG (bottom row) tumor volumes separately, represented against tumor size. The linear trend was also identified and indicated by the dashed lines. Left panel shows the benchmarks of the random forest's output, while the right panel the benchmarks of the post-processed segmentation.

The segmentation of a single volume ranges between 60 and 75 seconds, when executed on a single core of a PC with i7 processor running at 3.4 GHz frequency, which can be reduced below 20 seconds when executed in parallelized version on four cores. The largest computational burden represents the

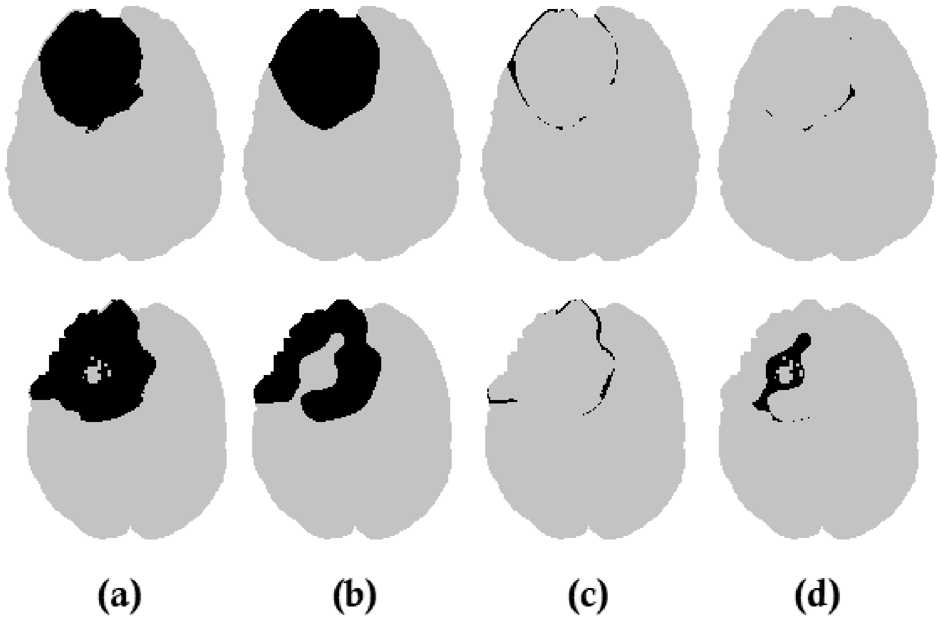


Figure 13: Detailed results presented for two slices: (a) the actual tumor (ground truth) is shown in black; (b) the segmented tumor; (c) false positives; (d) false negatives.

extraction of the 100 extra features for the approximately 1.5 million voxels of the volume.

The overall Dice score over 83.5% allows us to detect the presence of the tumor in a great majority of cases. However, the accuracy indicators can be further improved the following ways:

1. Using further texture features extracted from the neighborhood of each voxel.
2. Employing an effective feature selection scheme to eliminate useless features.
3. Implementing a more complex post-processing that investigates the contiguous ensembles of detected tumor voxels and discard small ones.

An objective comparison with existing methods enumerated in the Introduction is not an easily accomplishable task, as not all of them used the BRATS data set for evaluation, and even those which did, they did not evaluate all

| Feature name | | | Rate of |
|---|---|---|---|
| Data channel | Operation | Neighborhood | usage |
| T1 | Median | $11 \times 11$ | 99.38% |
| FLAIR | Average | $11 \times 11$ | 94.66% |
| FLAIR | Average | $9 \times 9$ | 53.30% |
| FLAIR | Maximum | $3 \times 3 \times 3$ | 32.53% |
| T2 | Average | $11 \times 11$ | 26.24% |
| T1c | Average | $11 \times 11$ | 20.63% |
| FLAIR | Average | $3 \times 3 \times 3$ | 19.43% |
| T2 | Maximum | $3 \times 3 \times 3$ | 18.35% |
| FLAIR | Average | $7 \times 7$ | 18.34% |
| T1c | Median | $11 \times 11$ | 17.00% |
| FLAIR | Median | $3 \times 3$ | 15.85% |
| FLAIR | Median | $11 \times 11$ | 14.45% |
| FLAIR | Average | $5 \times 5$ | 12.17% |
| T2 | Minimum | $3 \times 3 \times 3$ | 11.77% |
| T1 | Median | $11 \times 11$ | 10.75% |
| FLAIR | Median | $5 \times 5$ | 10.39% |

Table 4: Most frequently used features

the 54 available low-grade and/or 220 available high-grade tumor volumes. With respect to the methods involved in the comparison in [21], our proposed methodology seems competitive, and it will further improve with the implementation of the above listed ideas.

To optimize the efficiency of the random forest classifier, it is useful to know how many times the features are used in decision making during the testing phase, which are the most and least used features? The number of trees in the random forest was denoted by $n_T$. Let us denote the number of test voxels by $N_{test}$. Performing the whole classification requires to perform $n_T \times N_{test}$ tests on binary decision trees. Let us further suppose, that feature number $f$ is used at least once in $N_f$ tests, with $0 \leq N_f \leq n_T \times N_{test}$. Having all these assumed, the rate of usage of feature number $f$ is $N_f/(n_T \times N_{test})$. Table 4 lists those features, whose rate of usage exceeds $10\%$, in decreasing order of the rate of usage. There are three important things to remark:

1. Data channels FLAIR and T2 are apparently more useful than the other two.
2. The four observed features do not appear in the table, indicating their reduced usefulness in the decision making. However, they cannot be called useless, because all other computed features are extracted from them.
3. Eliminating the least used features from the system may reduce the computation burden without damaging the segmentation accuracy.

## 4 Conclusion

This paper presented an automatic tumor detection and segmentation algorithm employing random forests of binary decision trees. The proposed methodology reliably detects both LG and HG tumors if their volume exceeds 10 mL. It is likely to obtain finer segmentation accuracy in the future via implementing some of the above mentioned further development ideas. We will also concentrate on differentiating among the parts of the whole tumor (edema, tumor core, necrosis, active tumor), according to the grand truth provided by the BRATS database.

## Acknowledgements

This research was funded by the Sapientia Institute for Research Programs (KPI). The work of L. Szilágyi was additionally funded by the János Bolyai Research Fellowship program of the Hungarian Academy of Sciences. The work of Z. Kapás was additionally funded by the Szekely Forerunner Federation. The work of Zs. Szabó was additionally funded by the Accenture Student Research Scholarship program.

## References

[1] S. B. Akers, Binary decision diagrams, *IEEE Trans. Computers* **C-27,** 6 (1978) 509–516. ⇒115

[2] A. J. Asman, B. A. Landman, Out-of-atlas labeling: a multi-atlas approach to cancer segmentation, *Proc. IEEE International Symposium on Biomedical Imaging*, Barcelona, Catalunya, 2012, pp. 1236–1239. ⇒111

[3] L. Breiman, Random forests, *Machine Learning* **45,** 1 (2001) 5–32. ⇒117

[4] J. D. Christensen, Normalization of brain magnetic resonance images using histogram even-order derivative analysis, *Magn. Reson. Imaging* **21,** 7 (2003) 817–820. ⇒114

[5] S. Ghanavati, J. Li, T. Liu, P. S. Babyn, W. Doda, G. Lampropoulos, Automatic brain tumor detection in magnetic resonance images, *Proc. IEEE International Symposium on Biomedical Imaging*, Barcelona, Catalunya, 2012, pp. 574–577. ⇒111

[6] N. Gordillo, E. Montseny, P. Sobrevilla, State of the art survey on MRI brain tumor segmentation, *Magn. Reson. Imaging* **31** (2013) 1426–1438. ⇒111, 112

[7] A. Hamamci, N. Kucuk, K. Karamam, K. Engin, G. Unal, Tumor-Cut: segmentation of brain tumors on contranst enhanced MR images for radiosurgery applications, *IEEE Trans. Med. Imaging* **31** (2012) 790–804. ⇒111

[8] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P. M. Jodoin, H. Larochelle, Brain tumor segmentation with deep neural networks, *Med. Image Anal.* **35** (2017) 18–31. ⇒112

[9] M. Y. Huang, W. Yang, Y. Wu, J. Jiang, W. F. Chen, Q. J. Feng, Brain tumor segmentation based on local independent projection-based classification, *IEEE Trans. Biomed. Eng.* **61** (2014) 2633–2645. ⇒112

[10] J. E. Iglesias, M. R. Sabuncu, Multi-atlas segmentation of biomedical images: A survey, *Med. Image Anal.* **24** (2015) 205–219. ⇒112

[11] A. Islam, S. M. S. Reza, K. M. Iftekharuddin, Multifractal texture estimation for detection and segmentation of brain tumors, *IEEE Trans. Biomed. Eng.* **60** (2013) 3204–3215. ⇒112

[12] J. Juan-Albarracín, E. Fuster-Garcia, J. V. Manjón, M. Robles, F. Aparici, L. Martí-Bonmatí, J. M. García-Gómez, Automated glioblastoma segmentation based on a multiparametric structured unsupervised classification, *PLoS ONE* **10** 5 (2015) e0125143. ⇒112

[13] V. G. Kanas, E. I. Zacharaki, C. Davatzikos, K. N. Sgarbas, V. Megalooikonomou, A low cost approach for brain tumor segmentation based on intensity modeling and 3D random walker, *Biomed. Sign. Proc. Control* **22** (2015) 19–30. ⇒112

[14] Z. Kapás, L. Lefkovits, D. Iclănzan, Á. Győrfi, B. L. Iantovics, Sz. Lefkovits, S. M.. Szilágyi, L. Szilágyi, Automatic brain tumor segmentation in multispectral MRI volumes using a random forest approach, *Proc. Pacific-Rim Symposium on Image and Video Technology (PSIVT'17)*, *Lecture Notes in Artificial Intelligence* **10749** (2018) 137–149. ⇒112

[15] M. Lê, H. Delingette, J. Kalpathy-Cramer, E. R. Gerstner, T. Batchelor, J. Unkelbach, N. Ayache, Personalized radiotherapy planning based on a computational tumor growth model, *IEEE Trans. Med. Imaging* **36** (2017) 815–825. ⇒112

[16] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al., The multimodal brain tumor image segmentation benchmark (BRATS), *IEEE Trans. Med. Imaging* **34,** 10 (2015) 1993–2024. ⇒114, 118

[17] B. H. Menze, K. van Leemput, D. Lashkari, T. Riklin-Raviv, E. Geremia, E. Alberts, et al. , A generative probabilistic model and discriminative extensions for brain lesion segmentation – with application to tumor and stroke, *IEEE Trans. Med. Imaging* **35** (2016) 933–946. ⇒112

[18] I. Njeh, L. Sallemi, I. Ben Ayed, K. Chtourou, S. Lehericy, D. Galanaud, A. Ben Hamida, 3D multimodal MRI brain glioma tumor and edema segmentation: a graph cut distribution matching approach, *Comput. Med. Image Anal.* **40** (2015) 108–119. ⇒111

[19] L. G. Nyúl, J. K. Udupa, X. Zhang, New variants of a method of MRI scale standardization, *IEEE Trans. Med. Imaging* **19,** 2 (2010) 143–150. ⇒111, 114

[20] S. Pereira, A. Pinto, V. Alves, C. A. Silva, Brain tumor segmentation using convolutional neural networks in MRI images, *IEEE Trans. Med. Imaging* **35** (2016) 1240–1251. ⇒112

[21] A. Pinto, S. Pereira, D. Rasteiro, C. A. Silva, Hierarchical brain tumour segmentation using extremely randomized trees, *Patt. Recogn.* **82** (2018) 105–117. ⇒112, 129

[22] J. Sahdeva, V. Kumar, I. Gupta, N. Khandelwal, C. K. Ahuja, A novel content-based active countour model for brain tumor segmentation, *Magn. Reson. Imaging* **30** (2012) 694–715. ⇒111

[23] H. C. Shin, H. R. Roth, M. C. Gao, L. Lu, Z. Y. Xu, I. Nogues, J. H. Yao, D. Mollura, R. M. Summers, Deep nonvolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, *IEEE Trans. Med. Imaging* **35** (2016) 1285–1298. ⇒112

[24] Zs. Szabó, Z. Kapás, Á. Győrfi, L. Lefkovits, S. M. Szilágyi, L. Szilágyi, Automatic segmentation of low-grade brain tumor using a random forest classifier and Gabor features, *Proc. 14th International Conference on Fuzzy Systems and Knowledge Discovery*, Huangshan, China, 2018, pp. 1106–1113. ⇒112

[25] L. Szilágyi, L. Lefkovits, B. Benyó, Automatic Brain Tumor Segmentation in multispectral MRI volumes using a fuzzy c-means cascade algorithm, *Proc. 11th International Conference on Fuzzy Systems and Knowledge Discovery*, Zhangjiajie, China, 2015, pp. 285–291. ⇒112

[26] N. J. Tustison, K. L. Shrinidhi, M. Wintermark, C. R. Durst, B. M. Kandel, J. C. Gee, M. C. Grossman, B. B. Avants, Optimal symmetric multimodal templates and concatenated random forests for supervised brain tumor segmentation (simplified) with ANTsR, *Neuroinformatics* **13** (2015) 209–225. ⇒112

[27] U. Vovk, F. Pernuš, B. Likar, A review of methods for correction of intensity inhomogeneity in MRI, *IEEE Trans. Med. Imaging* **26** (2007) 405–421. ⇒111

[28] R. Zaouche, A. Belaid, S. Aloui, B. Solaiman, L. Lecornu, D. Ben Salem, S. Tliba, Semi-automatic method for low-grade gliomas segmentation in magnetic resonance imaging, *IRBM* **39** (2018) 116–128. ⇒112

[29] N. Zhang, S. Ruan, S. Lebonvallet, Q. Liao, Y. Zhou, Kernel feature selection to fuse multi-spectral MRI images for brain tumor segmentation, *Comput. Vis. Image Undestand.* **115** (2011) 256–269. ⇒112

*Received: Aug. 5, 2018 • Revised: Aug. 15, 2018*

# Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English. Information about each series can be found at http://www.acta.sapientia.ro.

# Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University    Sciendo by De Gruyter    Scientia Publishing House

**ISSN 1844-6086**
http://www.acta.sapientia.ro

# Information for authors

**Acta Universitatis Sapientiae, Informatica** publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/).

References should be listed alphabetically based on the Intructions for Authors given at the address http://www.acta.sapientia.ro.

Illustrations should be given in Encapsulated Postscript (eps) format.

One issue is offered each author free of charge. No reprints will be available.

**Contact address and subscription:**
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Idea Printing House
Director: Péter Nagy

**ISSN 1844-6086**
http://www.acta.sapientia.ro