Multivariate Investigation of NP-Hard Problems: Boundaries Between Parameterized Tractability and Intractability

Uéverton dos Santos Souza
Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/nº
São Domingos - Niterói - RJ, CEP: 24210-346, Brazil
Email: usouza@ic.uff.br

Advisor: Fábio Protti
Instituto de Computação - Universidade Federal Fluminense
Av. Gal. Milton Tavares de Souza, s/nº, São Domingos
Niterói - RJ, CEP: 24210-346, Brazil
Email: fabio@ic.uff.br

Co-advisor: Maise Dantas da Silva
Dep. Ciência e Tecnologia - Universidade Federal Fluminense
Rua Recife, s/n, Jardim Bela Vista,
Rio das Ostras - RJ, CEP: 28895-532, Brazil
Email: maise@vm.uff.br

Co-advisor: Dieter Rautenbach
Inst. Opt. und Operations Research - Universität Ulm
Helmholtzstraße 18 / Raum 1.68
89081 Ulm, Germany
Email: dieter.rautenbach@uni-ulm.de

***Abstract*—The main goal when using computing to solve a problem is to develop a mechanism to solve it efficiently. In general, this efficiency is associated with solvability in polynomial time. The theory of NP-completeness was developed to show which problems probably do not have polynomial time algorithms. However, many NP-hard and NP-complete problems must still be solved in practice; therefore it is natural to ask if each of these problems admits an algorithm whose non-polynomial time complexity is purely a function of some subset of its aspects. Questions about the existence of such algorithms are addressed within the theory of parameterized computational complexity developed by Downey and Fellows.**

In this thesis we present a multivariate investigation of the complexity of some NP-hard problems, i.e., we first develop a systematic complexity analysis of these problems, defining its subproblems and mapping which one belongs to each side of an "imaginary boundary" between polynomial time solvability and intractability. After that, we analyze which sets of aspects of these problems are *sources* of their intractability, that is, subsets of aspects for which there exists an algorithm to solve the associated problem, whose non-polynomial time complexity is purely a function of those sets. Thus, we use classical and parameterized computational complexity in an alternate and complementary approach, to show which subproblems of the given problems are NP-hard and latter to diagnose for which sets of parameters the problems are fixed-parameter tractable, or in FPT.

This thesis exhibits a classical and parameterized complexity analysis of different groups of NP-hard problems. The addressed problems are divided into four groups of distinct nature, in the context of data structures, combinatorial games, and graph theory: (I) *and/or* **graph solution and its variants;** (II) **flooding-filling games;** (III) **problems on P_3-convexity;** (IV) **problems on induced matchings.**

Keywords*—*Parameterized Complexity, And/Or Graph Solution, Flood-filling Games on Graphs, P_3-Convexity, Induced Matching.

I. Introduction

The question "$P = NP$?" is the most important open question in computer science, and the theory of NP-completeness was developed to show which problems probably do not have polynomial-time algorithms. Though it is nice to know that some problems do not have polynomial time algorithms unless $P = NP$, many NP-hard and NP-complete problems must still be solved in practice (especially those with real-world applications); therefore it is natural to ask if each of these problems admits an algorithm whose non-polynomial time complexity is purely a function of some subset of its aspects (in pratice many aspects of the problem often has bounded size or value). Questions about the existence of such algorithms are addressed within the theory of parameterized computational complexity developed by Downey and Fellows [1], [3], [4].

In the first part of the thesis, we present a detailed review of the theory of parameterized complexity, where the concepts and techniques involving fixed-parameter tractability and intractability are discussed.

The parameterized complexity theory became very popular in recent years and has become an important research topic at many universities around the world. In this sense, the Latin-American community can be somewhat outdated, with few Latin-American researchers working on this area. A major contribution of this thesis is to collaborate in popularizing the parameterized complexity nationwide. In this regard, as a consequence of our work, we can mention a narrowing of the relationship between researchers and the professors Frances Rosamond and Michael R. Fellows, one of the authors of the parameterized complexity. This relationship resulted in some studies developed with their co-authorship, and in particular in their visit to Latin-American in November 2014 and their subsequent participation as invited speakers in the 6th Latin American Workshop on Cliques in Graphs.

In addition, one of the goals of this thesis it is to make

an analysis on the sources of polynomial time intractability of some interesting problems. We develop a systematic complexity analysis of these problems, defining its subproblems and mapping which one belongs to each side of an "imaginary boundary" between polynomial-time solvability and intractability. After that, we analyze which sets of aspects of these problems are *sources* of their intractability, that is, subsets of aspects for which there exists an algorithm to solve the associated problem, whose non-polynomial time complexity is purely a function of these sets. Thus, we use classical and parameterized computational complexity in an alternate and complementary approach, to show which subproblems of the given problems are NP-hard and latter to diagnose for which sets of parameters the problems are fixed-parameter tractable, or in FPT.

This thesis exhibits a classical and parameterized complexity analysis of different groups of NP-hard problems. The problems studied are of distinct nature, and the concepts discussed have applications in many areas such as software engineering, distributed systems, artificial intelligence, bioinformatics, operational research, social networks, automation, game theory, among others. Morever, the proofs presented are of several types such as NP-hardness proofs, polynomial algorithms, structural characterizations, W[1]-hardness and W[2]-hardness proofs, FPT algorithms, polynomial kernel results, and infeasibility of polynomial kernels.

The first group of studied problems involve two important data structures used for modeling many real-word applications, *and/or graphs* and *x-y graphs*. An and/or graph is an acyclic digraph containing a source, such that every vertex $v \in V(G)$ has a label $f(v) \in \{\texttt{and}, \texttt{or}\}$. X-y graphs are a generalization of and/or graphs: every vertex v_i of an x-y graph has a label x_i-y_i meaning that v_i depends on x_i of its y_i out-neighbors. We investigate the complexity of finding a solution subgraph H of such digraphs, which must contain the source and obey the following rule: if a vertex is included in H then x_i of its out-edges must also be included in H, where an `and`-vertex has $x_i = y_i$, and an `or`-vertex has $x_i = 1$.

The second group of problems consists of variants of a one-player combinatorial game known as the Flood-Filling Game, which is played on a colored board and whose objective is to make the board monochromatic ("flood the board") with the minimum number of flood moves. A flood move consists of assigning a new color c_i to the a pivot tile p and also to all the tiles connected to p by a monochromatic path immediately before the move. The flood-filling game where all moves use the same pivot p is denoted by Flood-It. When the player can freely choose which tile will be the pivot of each move the game is denoted by Free-Flood-It.

The third group comprises some problems on P_3-convexity. More specifically we are interested in identifying either the minimum P_3-geodetic set or the minimum P_3-hull set S of a graph, from which the whole vertex set of G is obtained either after one or eventual iterations, respectively. Each iteration adds to a set S' of vertices all the vertices of $V(G) \setminus S'$ with two neighbors in S'.

The last group of problems studied in this thesis focus on a classical topic in graph theory. These problems are related to maximum matchings, maximum induced matchings, and the distance between them in a graph. The matching number $\nu(G)$ of G is the maximum cardinality of a matching in G, and a matching with $\nu(G)$ edges is a maximum matching of G. An induced matching is a set M' of edges of G at pairwise distance at least 2. The induced matching number $\nu_2(G)$ of G is the maximum cardinality of an induced matching in G, and an induced matching with $\nu_2(G)$ edges is a maximum induced matching. The distance between a maximum matching of a graph G and its maximum induced matching is the difference between the cardinality of these sets ($\nu(G) - \nu_2(G)$).

For each group of problems above, there is a chapter in this thesis devoted to it. A brief abstract of our work and the obtained results it is presented at the beginning of each chapter.

Below we present the list of papers developed and published throughout my PhD, related to this thesis.

1) Tractability and hardness of flood-filling games on trees. *Theoretical Computer Science*, v. 576, p. 102-116, 2015.
2) Maximum Induced Matchings close to Maximum Matchings. *Theoretical Computer Science*, 2015. (to appear)
3) Complexity Properties of Complementary Prism. *Journal of Combinatorial Optimization*, 2015. (to appear)
4) An algorithmic analysis of Flood-it and Free-Flood-it on graph powers. *Discrete Mathematics and Theoretical Computer Science*, v. 16, p. 279-290, 2014.
5) Revisiting the complexity of and/or graph solution. *Journal of Computer and System Sciences*, v. 79, p. 1156-1163, 2013.
6) Complexidade Parametrizada para Problemas em Grafos E/OU. *Pesquisa Operacional para o Desenvolvimento*, v. 4, p. 160-174, 2012.
7) The Flood-It game parameterized by the vertex cover number. *Electronic Notes in Discrete Mathematics*, LAGOS 2015.
8) On Graphs with Induced Matching Number Almost Equal to Matching Number. *Electronic Notes in Discrete Mathematics*, LAGOS 2015.
9) On P3-convexity of Graphs with Bounded Degree. *Lecture Notes in Computer Science*, v. 8546, p. 263-274, 2014. *10th Conference on Algorithmic Aspects of Information and Management (AAIM)*.
10) Parameterized Complexity of Flood-Filling Games on Trees. *Lecture Notes in Computer Science*, v. 7936, p. 531-542, 2013. *19th International Computing and Combinatorics Conference (COCOON)*.
11) Complexity of Geodetic Number Problem in Graphs with Maximum Degree 4. *13th Cologne-Twente Workshop on Graphs and Comb. Optimization (CTW)*, 2015.
12) The P3-Convexity in the Complementary Prism of a Graph *13th Cologne-Twente Workshop on Graphs and Comb. Optimization (CTW)*, 2015.
13) Parameterized And/Or Graph Solution. *12th Cologne-Twente Workshop on Graphs and Comb. Optimization (CTW)*, 2013.
14) Optimal Variability Selection in Product Line En-

gineering.[1] 24th Conference on Software Eng. and Knowledge Engineering (SEKE), 2012.

15) Inundação em Grafos. *16th Congreso Latino Iberoamericano de Investigación Operativa (CLAIO)*, 2012.

Other projects were developed and submitted along my PhD, such as:

16) P_3-Convexity Problems on Bounded-Degree and Planar Graphs. *Theoretical Computer Science*.
17) Tractability and Kernelization Lower Bound for And/Or Graph Solution. *Discrete Applied Mathematics*.
18) Complexity of Geodetic Number Problem in Graphs with Bounded Degree. *Journal of Computer and System Sciences*.
19) A Multivariate Investigation of Flood-It Game. *Discrete Applied Mathematics*.
20) Parameterized Problems on Complementary Prisms. *Discrete Mathematics*.

The main results of this work are briefly presented as follows.

II. Background

A *computational problem* is a question to be answered, typically containing several variables whose values are unspecified. An *instance* of a problem is created by specifying particular values for its variables. A problem is described by specifying both its instances and the nature of solutions for those instances.

A *decision problem* Π consists of a set D_Π of instances and a set $Y_\Pi \subseteq D_\Pi$ of yes-instances. A decision problem is described informally by specifying: (i) a generic instance in terms of its variables; (ii) a yes-no question stated in terms of the generic instance.

An *optimization problem* Π consists of a set D_Π of instances and a set $S_\Pi \subseteq D_\Pi$ of solutions such that for each $I \in D_\Pi$, there is an associated set $S_\Pi[I] \subseteq S_\Pi$ of solutions for I. An optimization problem is described informally by specifying: (i) a generic instance in terms of its variables; (ii) an objective function g to be calculated, and the properties that must be satisfied by any solution associated with an instance created from the generic instance. An optimal solution $S_\Pi[I]$ is a solution which maximizes/minimizes the value $g(S_\Pi[I])$.

An *algorithm* A for a problem Π is a finite sequence of instructions for some computer which solves Π. A *polynomial time algorithm* is defined to be one whose time complexity function is $O(p(n))$ for some polynomial function p, where n is used to denote the input length [2].

Definition 1: A problem Π belongs to class *P* if and only if Π can be solved in polynomial time by a deterministic algorithm.

Definition 2: A problem Π belongs to class *NP* if and only if for a given certificate (a string that certifies the answer to a computation), there is a deterministic algorithm which verifies its validity in polynomial time.

[1] In this paper the authors apply `and/or` graphs in software engineering problems.

Definition 3: Given two problems Π and Π', $\Pi \propto \Pi'$ (Π is reducible to Π' in polynomial time) if there exists an algorithm that, given an instance I of Π, constructs an instance I' of Π' in polynomial time in $|I|$ such that from a subroutine to I', a correct answer for I is output.

Definition 4: A problem Π' is *NP-hard* if for all problems $\Pi \in NP$, $\Pi \propto \Pi'$; if Π' is also in NP, then Π' is NP-complete.

It is easy to see that $\Pi \in P$ implies $\Pi \in NP$. If any single NP-hard problem can be solved in polynomial time, then all problems in NP can also be solved in polynomial time. If any problem in NP cannot be solved in polynomial time, then so neither can all NP-complete problems. An NP-complete problem Π, therefore, has the following property: $\Pi \in P$ if and only if $P = NP$. The question "$P = NP$?" is the most important open question in computer science.

An algorithm is *efficient* if its complexity function satisfies some criterion, e.g., the complexity function is a polynomial in the instance size. A problem is *tractable* if it has an efficient algorithm; otherwise, the problem is said to be *intractable*. As there are many possible criteria which can be used to define efficiency, there are many possible types of tractability and intractability [5].

A. Parameterized Tractability in Pratice

The theory of NP-completeness was developed to show which problems do not probably have polynomial time algorithms. Since the beginning of this theory in 1971, thousands of other problems have been shown to be NP-hard and NP-complete. Though it is nice to know that such problems do not have polynomial time algorithms unless $P = NP$, a inconvenient fact remains: these problems (especially those with real-world applications) must still be solved. Thus, the following question emerges:

"How does one solve NP-complete problems efficiently in practice?"

Firstly, we have two possibilities:

× - Try to construct a polynomial time algorithm (implies $P = NP$).
√ - Invoke some type of polynomial-time heuristic algorithm.

However, in pratice some set of aspects of the problem has bounded size or value. There are another approaches:

× - Invoke some type of "brute force" optimal-solution technique, that in effect runs in polynomial time because its time complexity function is $O(n^{f(k)})$, where n is used to denote the input length and k is some aspect with bounded size or value. However, when the instances to be solved are large, this approach may not be feasible.
√ - Invoke a non-polynomial time algorithm such that its non-polynomial time complexity is *purely* a function of some subset of aspects of the problem that are of bounded size or value in practice.

This last approach immediately suggests the following questions:

1) Given a problem and a subset of aspects of that problem, is there an algorithm for that problem whose non-polynomial time complexity is purely a function of those aspects?
2) Relatively to which aspects of that problem do such algorithms exist?

If a problem Π for a set K of its aspects admits such algorithms described in (1), i.e. solvable in $f(K).n^{O(1)}$ time, then $\Pi \in FPT$ with respect to K (the class of *fixed-parameter tractable* problems). Alternatively, one can show that such algorithm probably does not exist by establishing a intractability of this version of the problem.

B. Multivariate Investigation

According to Garey and Johnson [2], whenever we are confronted with a new problem, a natural first question to ask is: Can it be solved via a polynomial time algorithm? We can concentrate our efforts on trying to find a polynomial time algorithm as efficient as possible. However, if no polynomial time algorithm is apparent, an appropriate second question to ask is: "Is the problem NP-complete?". Suppose now we have just succeeded in demonstrating that our initial problem is NP-complete. Even though this answers the two questions which began our analysis, there are still many appropriate follow-up questions that could be asked. The problem we have been analyzing is often distilled from a less elegant applied problem, and some of the details that were dropped in the distillation process might alter the problem enough to make it polynomially solvable. If not, there still might be significant special cases that can be solved in polynomial time. Such possibilities can be investigated by analyzing subproblems of our original problem.

It should be apparent that, even though a problem Π is NP-complete, each of the subproblems of Π might independently be either NP-complete or polynomially solvable. Assuming that $P \neq NP$, we can view the subproblems of any NP-complete problem Π as lying on different sides of an imaginary "boundary" between polynomial time solvability and intractability [2].

Figure 1 [2] gives a schematic representation for one possible "current state of knowledge" about a collection of subproblems of a problem Π.

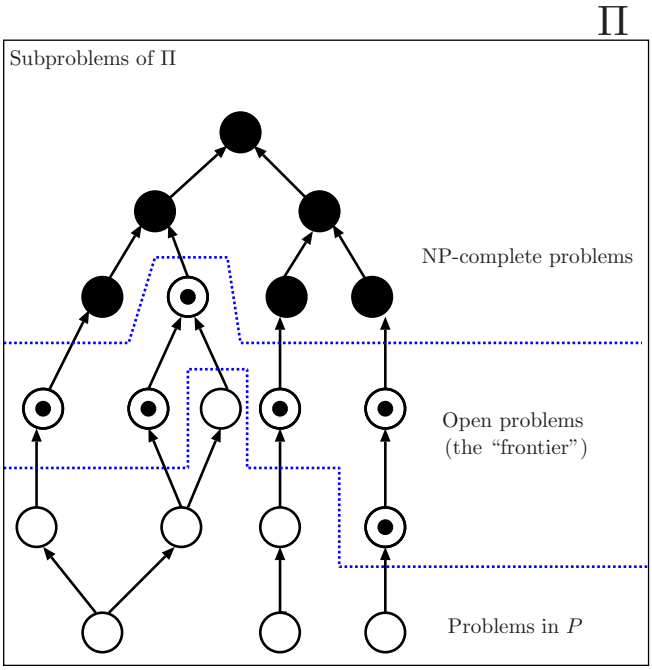

Fig. 1. One possible state of knowledge about subproblems of an NP-complete problem Π. An arrow from Π^1 to Π^2 signifies that Π^1 is a subproblem of Π^2.

In this thesis, our first goal when analyzing a problem is to determine which subproblems lie on each side.

1) Intractability Mapping: Any problem Π contains a domain D that is the set of all instances of Π. A problem Π' is a subproblem of Π if it asks the same question as Π, but only over a subset of the domain of Π.

Definition 5: Let Π be a problem with domain D and let $C = \{a_1, a_2, \ldots, a_\ell\}$ be a subset of aspects of Π. We denote by:

- *[*a_1=c'_1,a_2=c'_2,...,a_ℓ=c'_ℓ*]*-Π the subproblem of Π with domain D' such that each instance in D' has aspects $a_1, a_2, \ldots, a_\ell$ bounded by the constants $c'_1, c'_2, \ldots, c'_\ell$ respectively.
- *[*a_1,a_2,...,a_ℓ*]*-Π, or *[*C*]*-Π, the family of variants of Π such that every aspect in C is bounded by some constant.

Given an NP-hard problem Π and a subset C of its aspects, a systematic complexity analysis starts from the following steps.

1) Verify if [C]-Π is in P, or NP-hard.
2) If [C]-Π is in P, determine each minimal subset C' of C such that [C']-Π is in P.
3) If [C]-Π is NP-hard, determine for which values of the aspects in C the problem is solvable in polynomial time or remains NP-hard.

In a systematic complexity analysis of a problem Π, it is very common to identify subproblems of Π which can be solved in polynomial time. In general, it can be shown by some exhaustive algorithm in time $O(n^{f(k)})$, where n is used to denote the input length and k is some aspect with bounded size or value. Note that, when the instances to be solved are large, this approach may not be feasible in practice.

A parameter is a function which extracts a particular aspect or set of aspects of a problem from instances of that problem; it can also be considered as that set of aspects. As such, a parameter is both a mechanism for isolating an aspect of a problem and the "container" in which these aspects are packaged for subsequent manipulation [5].

Definition 6: A parameterized problem Π is described informally by specifying:

- A generic instance in terms of its variables.
- The aspects of an instance that comprise the parameter.

- A question stated in terms of the generic instance.

Definition 7: Let Π be a NP-hard problem and let $S = \{a_1, a_2, \ldots, a_\ell\}$ be a subset of aspects of Π. We denote by:

- $\Pi(a_1, a_2, \ldots, a_\ell)$, or $\Pi(S)$, the parameterized version of Π where the aspects in S are fixed as parameters.

Definition 8: [1] A parameterized problem $\Pi(S)$ belongs to the class XP if there exists an algorithm to solve $\Pi(S)$ in time $f(S).n^{g(S)}$, where n is used to denote the input length and f and g are arbitrary functions.

Observation 1: [S]-Π and $\Pi(S)$ are different problems. The instances of *[S]-*Π has the aspects in S with size bounded by constants, while in $\Pi(S)$ the parameters are just a mechanism for isolating aspects for subsequent manipulation (in this case, the aspects not necessarily have bounded size).

Lemma 1: Given an NP-hard problem Π and a subset S of its aspects, if *[S]-*Π remains NP-hard, then the parameterized problem $\Pi(S)$ is not in XP, unless $P = NP$.

Proof. If Π is in XP then by definition this problem is solved by an algorithm that runs in time $f(S)n^{g(S)}$ for some functions f and g. When the value of every aspect in S is fixed, the values of $f(S)$ and $g(S)$ become constants and this running time becomes polynomial in n. As this algorithm also solves $[S]$-Π and $[S]$-Π is NP-hard then $P = NP$.

Corollary 1: If $P \neq NP$, then $\Pi(S)$ is in XP if and only if *[S]-*Π is solvable in polynomial time.

Given a problem Π and some subset $S = \{s_1, \ldots, s_n\}$ of the aspects of Π, there are 3^n different basic families of variants of the problem, based on which of the aspects is declared as either:

1) an unrestricted part of the input,
2) part of the aggregate parameterization, or
3) a fixed constant (yielding part of the indexing of the family of parameterized problems).

Let Π be problem and let $S = \{s_1, \ldots, s_n\}$ be a subset of the aspects of Π. $[S_1]$-$\Pi(S_2)$ is the family of parameterized problems where the aspects in $S_1 \subseteq S$ are fixed constants and the aspects in $S_2 \subseteq S$ are aggregate parameters.

> A parameterized problem $\Pi(S)$ belongs to the class FPT, or *fixed-parameter tractable*, if there exists an algorithm to solve $\Pi(S)$ in time $f(S).n^{O(1)}$, where n is used to denote the input length and f is an arbitrary function.

Individual parameterized results are very good at establishing whether or not a given problem has an FPT-algorithm for a particular set of aspects of that problem. However, if one is interested in fully characterizing the set of FPT-algorithms for parameterized versions of a NP-hard problem, individual results are not sufficient because a fixed-parameter tractability (intractability) result says nothing about which subsets (supersets) of its aspects also render fixed-parameter tractability (intractability) [5]. In this case, it is necessary to make a systematic parameterized complexity analysis of the problem, determining the parameterized complexity relative to all non-empty subset of aspects of the problem.

A list of parameterized results produced by a systematic parameterized complexity analysis relative to some set of aspects S for a problem Π can be visualized as a *polynomial time intractability map* that shows which sets of aspects of the problem can be said to be responsible for (and hence are sources of) that problem's polynomial time intractability [5].

Definition 9: [5] Given a NP-hard problem Π and some subset S of aspects of Π, S is a source of polynomial-time intractability for Π, if $\Pi(S)$ is in FPT.

> " In parameterized complexity, the focus is not on *whether* a problem is hard, the theory starts from the assumption that most interesting problems are intractable when considered classically. The focus is on the question: *What makes the problem computationally difficult?* ". [1]

In this thesis, one of the goals it is to make an analysis on the sources of polynomial time intractability of some problems, that are "minimal" in the sense that their associated FPT-algorithms are not trivial extensions of other FPT-algorithms.

C. Parameterized Complexity

Classical complexity views a problem as an instance and a question, where the running time is specified by the input's size. However, when a problem comes from "real life" we always know more about the problem. The problem is planar, the problem has small width, the problem only concerns small values of the parameters. Thus, why not have a complexity theory which exploits these structural parameters? Why not have a complexity theory more fine-tuned to actual applications? [1]

The Parameterized Complexity Theory was proposed by Downey and Fellows [1] as a promising alternative to deal with NP-hard problems described by the following general form: given an object x and a nonnegative integer k, does x have some property that depends only on k (and not on the size of x)? In parameterized complexity theory, k is set as the *parameter*, considered to be *small* in comparison with the size $|x|$ of object x. It may be of high interest for some problems to ask whether they admit deterministic algorithms whose running times are exponential with respect to k but polynomial with respect to $|x|$.

As is common in complexity theory, we describe problems as languages over finite alphabets Σ. To distinguish them from parameterized problems, we refer to sets $\Pi \subseteq \Sigma^*$ of strings over Σ (nonempty) as classical problems.

Definition 10: Let Σ be a finite alphabet.

1) A parametrization of Σ^* is a mapping $k : \Sigma^* \to \mathbb{N}$ that is polynomial time computable.
2) A parameterized problem (over Σ) is a pair $(\Pi, k) = \Pi(k)$, consisting of a set $\Pi \subseteq \Sigma^*$ and a parametrization k of Σ^*.

Example. Let SAT denote the set of all satisfiable propositional formulas, where propositional formulas are encoded as strings over some finite alphabet Σ. Let $k : \Sigma^* \to \mathbb{N}$ be the parameterization defined by:

$$k = \begin{cases} number\ of\ variables\ of\ x, & (at\ least\ one\ variable), \\ 1, & otherwise. \end{cases} \quad (1)$$

If $\Pi(k)$ is a parameterized problem over the alphabet Σ, then we call strings $x \in \Sigma^*$ *instances* of $\Pi(k)$ and k, the the corresponding parameter. Usually, we represent a parameterized problem $\Pi(k)$ in the form:

Instance: $x \in \Sigma$.
Parameter: k
Problem: Decide whether $x \in \Pi(k)$.

In the same way that the notion of *polynomial time* is central to the classical formulation of computational complexity, a central notion to parameterized complexity is *fixed-parameter tractability*.

Definition 11: A parameterized problem $\Pi(k)$ is fixed-parameter tractable, or *FPT*, if the question "$x \in \Pi(k)$?" can be decided in running time $f(k).|x|^{O(1)}$, where f is an arbitrary function on nonnegative integers. The corresponding complexity class is called *FPT*.

III. Complexity of And/Or Graph Solution

The first group of studied problems involve two important data structures used for modeling many real-word applications, *and/or graphs* and *x-y graphs*. An and/or graph is an acyclic, edge-weighted directed graph containing a single source vertex such that every vertex v has a label $f(v) \in \{\texttt{and}, \texttt{or}\}$. A solution subgraph H of an and/or-graph must contain the source and obey the following rule: if an `and`-vertex (resp. `or`-vertex) is included in H then all (resp. one) of its out-edges must also be included in H. X-y graphs are defined as a natural generalization of and/or graphs. In this section we first present the results published in the paper [U. S. Souza, F. Protti, M. Dantas da Silva, Revisiting the complexity of and/or graph solution, J. Comput. Syst. Sci. 79:7 (2013) 1156-1163] where we have investigated the complexity of such problems under various aspects, including parameterized versions of it. However, this article kept the main open question still open: Is the problem of finding a solution subgraph of cost at most k (where k is a fixed parameter) in FPT? We finish this work finally presenting a positive answer to this question, via kernelization techniques. Also, using a framework developed by Bodlaender *et al*. (2009) and Fortnow and Santhanam (2011), based upon the notion of compositionality, we show that the above parameterized problem does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

An and/or graph is an acyclic digraph containing a source (a vertex that reaches all other vertices by directed paths), such that every vertex $v \in V(G)$ has a label $f(v) \in \{\texttt{and}, \texttt{or}\}$. In such digraphs, edges represent dependency relations between vertices: a vertex labeled `and` depends on all of its out-neighbors (conjunctive dependency), while a vertex labeled `or` depends on only one of its out-neighbors (disjunctive dependency).

We define x-y graphs as a generalization of and/or graphs: every vertex v_i of an x-y graph has a label x_i-y_i to mean that v_i depends on x_i of its y_i out-neighbors. Given an and/or graph G, an equivalent x-y graph G' is easily constructed as follows: sinks of G are vertices with $x_i = y_i = 0$; `and`-vertices satisfy $x_i = y_i$; and `or`-vertices satisfy $x_i = 1$.

And/or graphs were used for modeling problems originated in the 60's within the domain of Artificial Intelligence. Since then, they have successfully been applied to other fields, such as Operations Research, Automation, Robotics, Game Theory, and Software Engineering, to model cutting problems, interference tests, failure dependencies, robotic task plans, assembly/disassembly sequences, game trees, software versioning, and evaluation of boolean formulas. With respect to x-y graphs, they correspond to the *x-out-of-y model* of resource sharing in distributed systems.

In addition to the above applications, special directed hypergraphs named *F-graphs* are equivalent to and/or graphs. An F-graph is a directed hypergraph where hyperarcs are called *F-arcs* (for *forward arcs*), which are of the form $E_i = (S_i, T_i)$ with $|S_i| = 1$. An F-graph H can be easily transformed into an and/or graph as follows: for each vertex $v \in V(H)$ do $f(v)$=`or`; for each F-arc $E_i = (S_i, T_i)$, where $|T_i| \geq 2$, do: create an `and`-vertex v_i, add an edge (u, v_i) where $\{u\} = S_i$, and add an edge (v_i, w_j) for all $w_j \in T_i$.

The optimization problems associated with and/or graphs and x-y graphs are formally defined below.

MIN-AND/OR
Instance: An and/or graph $G = (V, E)$ where each edge e has an integer weight $\tau(e) > 0$.
Goal: Determine the minimum weight of a subdigraph $H = (V', E')$ of G (*solution subgraph*) satisfying the following properties:
- $s \in V'$;
- if a non-sink node v is in V' and $f(v)$=`and` then every out-edge of v belongs to E';
- if a non-sink node v is in V' and $f(v)$=`or` then exactly one out-edge of v belongs to E'.

MIN-X-Y
Instance: An x-y graph $G = (V, E)$ where each edge e has an integer weight $\tau(e) > 0$.
Goal: Determine the minimum weight of a subdigraph $H = (V', E')$ of G satisfying the following properties:
- $s \in V'$;
- for every non-sink node v_i in V', exactly x_i of its y_i out-edges belong to E'.

A. Main obtained results on the problem

Theorem 1: MIN-AND/OR remains NP-hard even for a very restricted family of and/or graphs where edges have weight one and `or`-vertices have out-degree at most two.

Theorem 2:
(a) The parameterized problem MIN-AND/OR$^0(k)$, whose domain includes and/or graphs allowing zero-weight edges, is W[2]-hard.
(b) MIN-X-Y(k) is *W[1]*-hard.
(c) MIN-AND/OR(k) is fixed-parameter tractable.

Using a framework developed by Bodlaender et al. (2009) and Fortnow and Santhanam (2011), based upon the notion of compositionality, we show that the above parameterized problem does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

Theorem 3: MIN-AND/OR(k) has no polynomial kernel unless $NP \subseteq coNP/poly$ and consequently $PH \subseteq \Sigma_3^p$.

IV. FLOODING-FILLING GAMES

The second group of studied problems consists of variants of a one-player combinatorial game known as the Flood-Filling Game, which is played on a colored board and whose objective is to make the board monochromatic ("flood the board") with the minimum number of flood moves. A flood move consists of assigning a new color c_i to the a pivot tile p and also to all the tiles connected to p by a monochromatic path immediately before the move. The flood-filling game where all moves use the same pivot p is denoted by Flood-It. When the player can freely choose which tile will be the pivot of each move the game is denoted by Free-Flood-It.

Flood-It game, originally played on a colored board consisting of an $n \times m$ grid, where each tile of the board has an initial color from a fixed color set. In the classic game, two tiles are *neighboring* tiles if they lie in the same row (resp. column) and in consecutive columns (resp. rows). A sequence C of tiles is a *path* when every pair of consecutive tiles in C is formed by neighboring tiles. A *monochromatic path* is a path in which all the tiles have the same color. Two tiles a and b are *m-connected* when there is a monochromatic path between them. In Flood-It, a move consists of assigning a new color c_i to the top left tile p (the *pivot*) and also to all the tiles m-connected to p immediately before the move. The objective of the game is to make the board monochromatic ("flood the board") with the minimum number of moves. Figure 2 shows a sequence of moves to flood a 3×3 grid colored with five colors.

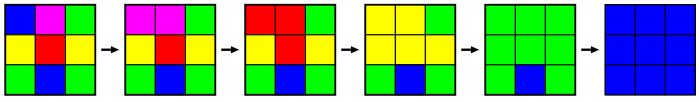

Fig. 2. An optimal sequence of moves to flood a 3×3 grid.

We consider these games when played on any graph with an initial coloring.

Many complexity issues on Flood-It and Free-Flood-It have recently been investigated. In [25], Arthur, Clifford, Jalsenius, Montanaro, and Sach show that Flood-It and Free-Flood-It are NP-hard on $n \times n$ grids colored with at least three colors. Meeks and Scott [28] prove that Free-Flood-It is solvable in polynomial time on $1 \times n$ grids and on 2-colored graphs, and also that Flood-It and Free-Flood-It remain NP-hard on $3 \times n$ grids colored with at least four colors. Up to the authors' knowledge, the complexity of Flood-It on $3 \times n$ grids colored with three colors remains as an open question. Clifford, Jalsenius, Montanaro, and Sach present a polynomial-time algorithm for Flood-It on $2 \times n$ grids. In [29], Meeks and Scott show that Free-Flood-It remains NP-hard on $2 \times n$ grids. Fleischer and Woeginger [27] proved that Flood-It is NP-hard on trees.

Flood-filling games in bioinformatics. Since the 90's, an increasing number of papers on biological applications have been dealt with as combinatorial problems. Vertex-colored graph problems have several applications in bioinformatics. The Colored Interval Sandwich Problem has applications in DNA physical mapping and in perfect phylogeny; vertex-recoloring problems appear in protein-protein interaction networks and phylogenetic analysis; the Graph Motif Problem was introduced in the context of metabolic network analysis; the Intervalizing Colored Graphs Problem models DNA physical mapping; and the Triangulating Colored Graph Problem is polynomially equivalent to the Perfect Phylogeny Problem.

Flood-Filling games on colored graphs are also related to many problems in bioinformatics. As shown in this paper, Flood-It played on trees is analogous to a restricted case of the Shortest Common Supersequence Problem. Consequently, these games inherit from the Shortest Common Supersequence Problem many applications in bioinformatics, such as: microarray production, DNA sequence assembly, and a close relationship to multiple sequence alignment. In addition, some disease spreading models, work in a similar way to flood-filling games.

We present below the formal definitions of the two flood-filling games studied in this chapter.

Flood-It (decision version)
Instance: A colored graph G with a pivot vertex p, an integer λ.
Question: Is there a sequence of at most λ flood moves which makes the graph monochromatic, using p as the pivot in all moves?

Free-Flood-It (decision version)
Instance: A colored graph G with a pivot vertex p, an integer λ.
Question: Is there a sequence of at most λ flood moves which makes the graph monochromatic?

A complete mapping of the complexity of flood-filling games on trees is made, charting the consequences of single and aggregate parameterizations. Furthermore, we show that Flood-It on trees and Restricted Shortest Common Supersequence (RSCS) are analogous problems, which proves some FPT and W[1]-hard of cases of Flood-it. In addition, we prove that Flood-It remains NP-hard when played on 3-colored trees, which closes an open question. We also present a general framework for reducibility from Flood-It to Free-Flood-It. Analyzing the behavior of these games when played on other classes of boards, we describe polynomial time algorithms, and some NP-hard cases. Finally, we show that Flood-it is fixed-parameter tractable when parameterized by the vertex cover number, and it has a polynomial kernel if the number of colors is a second parameter.

Let Π be a flood-filling game and let $S = \{s_1, \ldots, s_n\}$ be a subset of the aspects of Π. $[S_1]$-$\Pi(S_2)$ is the family of parameterized problems where the aspects in $S_1 \subseteq S$ are fixed constants and the aspects in $S_2 \subseteq S$ are aggregate parameters.

We consider the following aspects of the problem: c - number of colors; λ - number of moves; d - maximum distance

of the pivot; o - maximum orbit; k - number of leaves; r - number of bad moves, $r = (\lambda - c)$.

A. Main obtained results on the problem

Theorem 4:
(a) $[d]$-Flood-It on trees remains NP-hard when $d = 2$.
(b) $[d]$-Flood-It(c) is in FPT and admits a polynomial kernelization.

Theorem 5:
(a) Flood-It on trees and RSCS are analogous problems.
(b) Flood-It(c, k, λ) on trees is p-analogous to RSCS$(|\Sigma|, \ell, \Lambda)$.

We remark that the book [2] (see Appendix) reports that the SCS problem is solvable in polynomial time when each string has size two. This assertion is false, as we can see by Theorems 4 and 5.

Corollary 2: Flood-It(k, c) on trees is W[1]-hard.

A colored rooted tree is a pc-tree (phylogenetic colored tree) if no color occurs more than once in any path from the root to a leaf. A pc-tree T is a cpc-tree if each color occurs exactly once in any path from the root to a leaf. Restricting attention to phylogenetic colored trees, we have significant effects on problem complexity.

Theorem 6:
(a) Flood-It(k) on pc-trees with pivot root is W[1]-hard.
(b) Flood-It on trees remains NP-hard even when restricted to cpc-trees.
(c) Flood-It(r) on cpc-trees with pivot root is in FPT.
(d) Flood-It(k, r) on trees is in FPT.

Let G be a graph, $v \in V(G)$, and ℓ a positive integer. The graph $\psi(G, v, \ell)$ is constructed as follows: (i) create ℓ disjoint copies $G_1, \ldots, G_\ell$ of G; (ii) contract the copies $v_1, v_2, \ldots, v_\ell$ of v into a single vertex v^*. Let $\mathscr{F}$ be a class of graphs. Then:

$\psi(\mathscr{F}) = \{G \mid G = \psi(G', v, \ell)$ for some triple $(G' \in \mathscr{F}, v \in V(G'), \ell > 0)\}$.

Theorem 7: Flood-It on $\mathscr{F}$ is reducible in polynomial time to Free-Flood-It on $\psi(\mathscr{F})$.

Theorem 8:
(a) Flood-It on trees remains NP-hard even restricted to 3-colored trees.
(b) Free-Flood-It on trees remains NP-hard even restricted to 3-colored trees..

Theorem 9: In Free-Flood-It on pc-trees, there always exists an optimal free-flooding which is a flooding with pivot root.

Theorem 10:
(a) Flood-it is solvable in polynomial time on P_n^2, C_n^2, and $2 \times n$ circular grids.
(b) Free-Flood-it remains NP-hard on P_n^2, C_n^2, and $2 \times n$ circular grids.

Theorem 11: Flood-it on graphs is fixed-parameter tractable, FPT, when parameterized by the size of the minimum vertex cover (k).

Theorem 12: Flood-it on graphs admits a polynomial kernelization when parameterized by the size of the minimum vertex cover (k) and the number of colors (c).

V. P_3-CONVEXITY

The third group comprises some problems on P_3-convexity. More specifically we are interested in identifying either the minimum P_3-geodetic set or the minimum P_3-hull set S of a graph, from which the whole vertex set of G is obtained either after one or eventual iterations, respectively. Each iteration adds to a set S of vertices all the vertices of $V(G) \setminus S$ with two neighbors in S.

One of the most elementary models of the spreading of a property within a network – like sharing an idea or disseminating a virus – one can consider a graph G, a set S of vertices of G that initially possesses the property, and an iterative process whereby new vertices u are added to S whenever sufficiently many (usually two) neighbors of u are already in S. Similar models were studied in various contexts, such as statistical physics, social networks, marketing, and distributed computing.

Let $G = (V, E)$ be a graph. For $U \subseteq V$, let the interval $I[U]$ of U in G be the set $U \cup \{u \in V(G) \setminus U \mid |N_G(u) \cap U| \geq 2\}$. A set S of vertices of G is *P_3-geodetic* if $I[S]$ contains all vertices of G. The *P_3-geodetic number* $g_{P_3}(G)$ (or just $g(G)$) of a graph G is defined as the minimum cardinality of a P_3-geodetic set. The decision problem related to determining the P_3-geodetic number is known to be NP-complete for general graphs, and coincides with the well-studied 2-domination number [40], [43], [44].

A *P_3-hull* set U of G is a set of vertices such that:

- $U^0 = U$
- $U^k = I[U^{k-1}]$, for $k \geq 1$.
- $\exists\ k \geq 0 \mid U^k = V(G)$

We define $H_G(S) \subseteq V(G)$ as $I[S]^{k+1}$ such that $I[S]^{k+1} = I[S]^k$, $k \geq 0$. The cardinality of a minimum P_3-hull set of G is the *P_3-hull number* of G, denoted by $h_{p3}(G)$ (or just $h(G)$). Again, the decision problem related to determining the P_3-hull number of a graph is still a well known NP-complete problem [36].

We analyze the complexity of these problems when some parameters related to the maximum and minimum degree of a graph are known. In the following subsection we review some results on planar satisfiability problems. In Section 2 we present some results on finding a minimum P_3-hull set of graphs with bounded degree. Finally, in Section 3 we analyze complexity aspects of finding a minimum P_3-geodetic set on planar graphs with bounded degree.

A. Main obtained results on the problem

Theorem 13: Let c be a positive integer. If G is a graph with $\delta(G) \geq \frac{|V(G)|}{c}$, then

$$h_{P_3}(G) \leq 2 \left\lceil \frac{\log(2c)}{\log\left(\frac{2c^2}{2c^2-1}\right)} \right\rceil + 2c^3.$$

Corollary 3: A minimum P_3-hull set of a graph G with $\delta(G) \geq \frac{|V(G)|}{c}$ (for some constant c) can be found in polynomial time.

Theorem 14:
(a) P_3-HULL NUMBER remains NP-complete on planar graphs G with $\Delta(G) = 3$.
(b) A minimum P_3-hull set of a cubic graph can be found in polynomial time.

Theorem 15:
(a) P_3-GEODETIC NUMBER remains NP-complete on planar graphs G with $\Delta(G) = 3$.
(b) P_3-GEODETIC SET(k) is W[2]-hard.
(c) P_3-GEODETIC SET(k, Δ) is fixed-parameter tractable.

The complementary prism $G\bar{G}$ of G arises from the disjoint union of the graph G and its complement $\bar{G}$ by adding the edges of a perfect matching joining pairs of corresponding vertices of G and $\bar{G}$.

Theorem 16:
(a) To decide whether a compl. prism $G\bar{G}$ has a P_3-geodetic set of size k is NP-complete.
(b) A minimum P_3-hull set of a compl. prism $G\bar{G}$ can be found in polynomial time.

VI. PROBLEMS ON INDUCED MATCHINGS

The last group of problems studied in this thesis focus on a classical topic in graph theory. These problems are related to maximum matchings, maximum induced matchings, and the distance between them in a graph. The matching number $\nu(G)$ of G is the maximum cardinality of a matching in G, and a matching with $\nu(G)$ edges is a maximum matching of G. An induced matching is a set M' of edges of G at pairwise distance at least 2. The induced matching number $\nu_2(G)$ of G is the maximum cardinality of an induced matching in G, and an induced matching with $\nu_2(G)$ edges is a maximum induced matching. The distance between a maximum matching of a graph G and its maximum induced matching is the difference between the cardinality of these sets $(\nu(G)-\nu_2(G))$. We study graphs G with $\nu(G) - \nu_2(G) \leq k$, for some non-negative integer k.

Let G be a graph with $\nu(G) - \nu_2(G) \leq k$ for some non-negative integer k.

Let M_1 and M_2 be a maximum matching and a maximum induced matching of G, respectively, and let $H = (V(G), M_1 \Delta M_2)$.

If some component of H is a path with 2 edges, say $e_1 \in M_1$ and $e_2 \in M_2$, then $(M_1 \setminus \{e_1\}) \cup \{e_2\}$ is a maximum matching of G having more edges in common with M_2 than M_1. Iteratively applying this exchange operation to M_1, we may assume that no component of H is a path with 2 edges.

While maximum matchings can be found efficiently [57], it is algorithmically hard to find a maximum induced matching [61], [49]. It is even hard to approximate the induced matching number under substantial restrictions, and efficient exact and approximation algorithms have been proposed for several special graph classes (cf. [52], [59] for a detailed discussion). The fixed parameter tractability of induced matchings when parameterized by their cardinality was studied in [60], [59], [52]. While this problem is $W[1]$-hard in general, it was shown to be fixed parameter tractable for several restricted graph classes.

We study graphs where the matching number is not much larger than the induced matching number. Kobler and Rotics [56] showed that the graphs where these two numbers coincide, can be recognized efficiently. Cameron and Walker [51] extended this result and gave a complete structural description of these graphs. We review the results from [56], [51] and present shorter proofs. We study graphs G where $\nu(G) - \nu_2(G) \leq k$. We show that the recognition of these graphs can be done in polynomial time for fixed k and is fixed parameter tractable when parameterized by k for graphs of bounded maximum degree.

A. Main obtained results on the problem

Lemma 2: The components of H are

- isolated vertices,
- paths of length 1 whose edge belongs to $M_1 \setminus M_2$, and
- paths of length 3 whose two leaf edges belong to $M_1 \setminus M_2$ and whose middle edges belong to $M_2 \setminus M_1$.

Furthermore, H has exactly $\nu(G) - \nu_2(G)$ non-trivial components.

Theorem 17: For a fixed non-negative integer k, the graphs G with $\nu(G) - \nu_2(G) \leq k$ can be recognized in polynomial time.

Our next result states that the recognition of those graphs G with $\nu(G) - \nu_2(G) \leq k$ that are of bounded maximum degree is fixed parameter tractable when parameterized by k.

Theorem 18: Let Δ and k be positive integers. The graphs G with $\nu(G) - \nu_2(G) \leq k$ of maximum degree at most Δ can be recognized in $f(k, \Delta)n^c$ time where the constant c does not depend on Δ or k.

VII. CONCLUSIONS

In this thesis, a multivariate investigation of NP-hard problems has been carried out as a systematic application of classical and parameterized complexity techniques. This approach focused on drawing for each analyzed problem its boundaries between: (i) polynomial-time solvable and NP-hard subproblems; (ii) tractable and intractable parameterized versions. This strategy presents a more refined analysis of the complexity of problems, as well as their possibilities of solvability in practice. The idea is to map out when a problem Π becomes polynomial-time intractable and the sets of aspects that are responsible for this NP-hardness, i.e., the sets of aspects for which we can isolate its non-polynomial time complexity to solve Π as a purely function of them.

The tools used in this systematic analysis are first and foremost methods for algorithm design, or polynomial-time reductions (Karp reductions), to demonstrate polynomial time solvability or NP-hardness of a given problem, respectively. To show that a problem is fixed-parameter tractable, or in FPT, with respect to a set of parameters, we apply the bounded search tree and problem kernel (kernelization) methods. In another direction, we apply FPT-reductions (parametric reductions) to identify the level of parameterized intractability (W[t]-hardness, $t \geq 1$) of parameterized problems. Since every

problem in FPT has a kernelization algorithm, we also analyze whether an FPT problem has a kernel of polynomial size with respect to its parameters. We establish the infeasibility of polynomial kernels for parameterized problems by a framework developed by Bodlaender et al. [6] and Fortnow and Santhanam [7], based upon the notion of or-compositionality, which shows that a problem does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

This thesis make a multivariate investigation of different groups of NP-hard problems: (i) and/or graph solution and its variants; (ii) flooding-filling games; (iii) problems on P_3-convexity; (iv) problems on induced matchings.

1) **And/or graph solution and its variants.** We have proved that the problem MIN-AND/OR remains NP-hard even for and/or graphs where edges have weight one, or-vertices have out-degree at most two, and vertices with in-degree greater than one are within distance at most one of a sink; and that deciding whether there is a solution subtree with weight exactly k of a given x-y tree is also NP-hard. In a parameterized point of view, we have shown that MIN-AND/OR$^0(k)$ is W[2]-hard, and MIN-X-Y(k) is W[1]-hard. We also deal with the main question: "Is MIN-AND/OR$(k) \in$ *FPT*?". We answer positively to this question via a reduction to a problem kernel. Finally, we analyze whether MIN-AND/OR(k) admits a polynomial kernelization algorithm, and using the framework based upon the notion of or-compositionality, we show that MIN-AND/OR(k) does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

2) **Flood-filling games**. We analyze the complexity consequences of parameterizing Flood-it and Free-Flood-it by one or two of the following parameters: c - number of colors; λ - number of moves; d - maximum distance of the pivot (or diameter, in the case of Free-Flood-It); o - maximum orbit; k - number of leaves; r - number of bad moves. During our analysis we have shown that Flood-It on trees is analogous to Restricted Shortest Common Supersequence, and Flood-It remains NP-hard on 3-colored trees, closing an open question. We also present a general framework for reducibility from Flood-It to Free-Flood-It. Analyzing the computational complexity of these games on other classes of graphs such as powers of paths, power of cycles, circular grids, and graphs with bounded vertex cover, we conclude that: (i) Flood-it can be solved in polynomial time when played on P_n^2, C_n^2, and $2 \times n$ circular grids; (ii) Free-Flood-it is NP-hard when played on P_n^2, C_n^2; and $2 \times n$ circular grids. Finally, we prove that Flood-it on graphs is fixed-parameter tractable considering the size of a minimum vertex cover as the parameter; in addition, we show a polynomial kernelization algorithm for Flood-it when, besides the minimum vertex cover, the number of colors is also a parameter.

3) **Problems on P_3-convexity**. We prove that: (i) a minimum P_3-hull set of a graph G can be found in polynomial time when $\delta(G) \geq \frac{n(G)}{c}$ (for some constant c); (ii) deciding if the size of a minimum P_3-hull set of a graph is at most k remains NP-complete even on planar graphs with maximum degree four; (iii) a minimum P_3-hull set of a cubic graph can be found in polynomial time; (iv) a minimum P_3-hull set can be found in polynomial time in graphs with minimum feedback vertex set of bounded size and with no vertices of degree two; (v) deciding if the size of a minimum P_3-geodetic set of a planar graph with maximum degree three is at most k is NP-complete. Some trivial parameterized results on P_3-geodetic sets are also shown.

4) **Problems on induced matchings.** We present a short proof of a structural description of the graphs G where the matching number $\nu(G)$ equals the induced matching number $\nu_2(G)$, and use it to study graphs G with $\nu(G) - \nu_2(G) \leq k$. We show that the recognition of these graphs can be done in polynomial time for fixed k, and is fixed parameter tractable when parameterized by k for graphs of bounded maximum degree. Finally, we extend some of Cameron and Walker's results to k-matchings in graphs of sufficiently large girth.

REFERENCES

[1] R. G. Downey, M. Fellows. *Parameterized complexity*, Springer, 1999.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[3] J. Flum, M. Grohe. *Parameterized complexity theory*, Springer, 2006.

[4] R. Niedermeier. *Invitation to fixed-parameter algorithms*, Oxford University Press, 2006.

[5] H. T. Wareham, M. R. Adviser-Fellows. Systematic parameterized complexity analysis in computational phonology, PhD thesis, University of Victoria, 1999.

[6] Hans L. Boadlaender, Rodney G. Downey, Michael R. Fellows, Danny Hermelin, On problems without polynomial kernels, *Journal of Computer and System Sciences*, v. 75, p. 423-434, 2009.

[7] L. Fortnow, R. Santhanam, Infeasibility of instance compression and succinct PCPs for NP, *Journal of Computer and System Sciences*, v.77, p. 91-106, 2011.

[8] H. L. Boadlaender, Kernelization: New upper and lower bound techniques, in: J. Chen, F. V. Fomin (Eds.), Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC 2009, in: Lecture Notes in Computer Science, v. 5917, Springer Verlag, p. 17-37, 2009.

[9] Hans L. Boadlaender, Stéphan Thomassé, Anders Yeo, Kernel bounds for disjoint cycles and disjoint paths, *Theoretical Computer Science*, v. 412, p. 4570-4578, 2011.

[10] Hans L. Bodlaender, Bart M.P. Jansen, Stefan Kratsch, Kernel bounds for path and cycle problems, *Theoretical Computer Science*, doi:10.1016/j.tcs.2012.09.006, 2012

[11] Cao T., Sanderson, A. C., And/or net representation for robotic task sequence planning, *IEEE Trans. Systems Man Cybernet, Part C: Applications and Reviews*, v. 28, p. 204-218, 1998.

[12] Corandi R., Westfechtel B., Version models for software configuration management, *ACM Computing Surveys*, v. 30, p. 233-282, 1998.

[13] DeMello L. S. H., Sanderson A. C., A correct and complete algorithm for the generation of mechanical assembly sequences, *IEEE Trans. Robotics and Automation*, v. 7, p. 228-240, 1991.

[14] Gallo, G., Longo, G., Nguyen, S., Pallottino, S., Directed hypergraphs and applications, *Discrete Applied Mathematics*, v. 42, p. 177-201, 1993.

[15] Guo, J., Niedermeier, R., Invitation to Data Reduction and Problem Kernelization, *ACM SIGACT News*, v. 38, n. 1, p. 31-45, 2007.

[16] Neeldhara Misra, Venkatesh Raman, Saket Saurabh, Lower Bounds on Kernelization, *Discrete Optimization*, v. 8, p. 110-128, 2011.

[17] Karp R. M., Reducibility among combinatorial problems, in R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, 1972.

[18] Medeiros, R. P., Souza, U. S., Protti, F., Murta, L. G. P., Optimal Variability Selection in Product Line Engineering, *Proc. of the 24th International Conference on Software Engineering and Knowledge Engineering - SEKE 2012.*

[19] Morabito R., Pureza V., A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem, *Annals of Operation Research*, v. 179, p. 297-315, 2010.

[20] Sahni S., Computationally related problems, *SIAM Journal on Computing*, v. 3, n. 4, p. 262-279, 1974.

[21] Souza U. S., Protti F., Dantas da Silva M., Complexidade parametrizada para problemas em grafos E/OU, *Pesquisa Operacional para o Desenvolvimento*, v. 4, n. 2, p. 160-174, 2012.

[22] Souza, U. S., Protti, F., Dantas da Silva, M., Revisiting the Complexity of And/Or Graph Solution, *Journal of Computer and System Sciences*, v. 79, p. 1156-1163, 2013.

[23] Souza, U. S., Protti, F., Dantas da Silva, M., Parameterized And/Or Graph Solution, *Proc. of the 12th Cologne Twente Workshop on Graphs and Combinatorial Optimization - CTW 2013*, 205-208, 2013.

[24] Chee-Keng Yap, Some Consequences of Non-Uniform Conditions on Uniform Classes, *Theoretical Computer Science*, v. 26, p. 287-300, 1983.

[25] D. Arthur, R. Clifford, M. Jalsenius, A. Montanaro, and B. Sach, The Complexity of Flood-Filling Games, in Paolo Boldi and Luisa Gargano, editors, *Proceedings of FUN*, *Lecture Notes in Computer Science* 6099 (2010) 307–318, Springer.

[26] M. R. Fellows, M. T. Hallett, and U. Stege, Analogs & Duals of the MAST problem for Sequences & Trees, *Journal of Algorithms* 49:1 (2003) 192–216.

[27] R. Fleischer, G. J. Woeginger, An Algorithmic Analysis of the Honey-Bee Game, *Theoretical Computer Science* 452, pp. 75-87, 2012.

[28] K. Meeks and A. Scott, The Complexity of Flood-Filling Games on Graphs, *Discrete Applied Mathematics* 160 (2012) 959–969.

[29] K. Meeks and A. Scott, The Complexity of Free-Flood-It on $2 \times n$ Boards, arXiv:1101.5518v1 [cs.DS], January 2011.

[30] U. S. Souza, F. Protti, and M. Dantas da Silva, Parameterized Complexity of Flood-Filling Games on Trees. In: D.-Z. Du, G. Zhang, Editors, *Proceedings of COCOON 2013 - 19th International Computing & Combinatorics Conference*, Hangzhou, China, June 2013, *Lecture Notes in Computer Science* v. 7936, pp. 531-542.

[31] Souza, Protti and Dantas da Silva, *"Inundação em Grafos"*, 16th Congreso Latino Iberoamericano de Investigación Operativa & 44th Simpósio Brasileiro de Pesquisa Operacional, CLAIO/SBPO 2012.

[32] M.R. Cappelle, L. Penso, D. Rautenbach, Recognizing Some Complementary Products, Theoretical Computer Science, to appear.

[33] P. Balister, B. Bollobás, J.R. Johnson, M. Walters. Random majority percolation, *Random Struct. Algorithms*. v. 36, p. 315–340, 2010.

[34] J. Balogh, B. Bollobás. Sharp thresholds in Bootstrap percolation. *Physica A*, v. 326, p. 305–312, 2003.

[35] J.-C. Bermond, J. Bond, D. Peleg, S. Perennes. The power of small coalitions in graphs. *Discrete Appl. Math.*, v. 127, 399–414, 2003.

[36] C. C. Centeno, M. C. Dourado, L. D. Penso, D. Rautenbach, J. L. Szwarcfiter. Irreversible conversion of graphs. *Theor. Comput. Sci.*, v. 412, p. 3693–3700, 2011.

[37] C. C. Centeno, L. D. Penso, D. Rautenbach, V. G. P. de Sá. Immediate versus eventual conversion: comparing geodetic and hull numbers in P 3-convexity. In: *Graph-Theoretic Concepts in Computer Science*, Springer, p. 262-273, 2012.

[38] P.A. Dreyer Jr, F.S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion, *Discrete Appl. Math.*, v. 157, 1615–1627, 2009.

[39] S.A. Cook, The complexity of theorem-proving procedures, in: *Proc. 3rd Ann. ACM Symp. on Theory of Computing Machinery*, New York, p. 151-158, 1971.

[40] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater. Fundamentals of domination in graphs. Marcel Dekker, 1998.

[41] D. Lichtenstein, Planar satisfiability and its uses. *SIAM Journal on Computing*, v. 11, p. 329-343, 1982.

[42] "L. D. Penso, F. Protti, D. Rautenbach, U. S. Souza", "On P_3-convexity of Graphs with Bounded Degree", "10th International Conference on Algorithmic Aspects of Information and Management, AAIM 2014", 2014.

[43] A. Hansberg, L. Volkmann. On graphs with equal domination and 2-domination numbers. *Discrete Mathematics*, v. 308, n. 11, p. 2277-2281, 2008.

[44] C. C. Centeno, L.D Penso, D. Rautenbach, V. G. P. de Sá. Geodetic Number versus Hull Number in P_3-Convexity. *SIAM Journal on Discrete Mathematics*, v. 27, n. 2, p. 717-731, 2013.

[45] A. Brandstädt and C.T. Hoáng, Maximum induced matchings for chordal graphs in linear time, Algorithmica 52 (2008) 440-447.

[46] A. Brandstädt and R. Mosca, On distance-3 matchings and induced matchings, Discrete Appl. Math. 159 (2011) 509-520.

[47] K. Cameron, R. Sritharan, and Y. Tang, Finding a maximum induced matching in weakly chordal graphs, Discrete Math. 266 (2003) 133-142.

[48] J.-M. Chang, Induced matchings in asteroidal triple-free graphs, Discrete Appl. Math. 132 (2003) 67-78.

[49] K. Cameron, Induced matchings, Discrete Appl. Math. 24 (1989) 97-102.

[50] K. Cameron, Induced matchings in intersection graphs, Discrete Math. 278 (2004) 1-9.

[51] K. Cameron and T. Walker, The graphs with maximum induced matching and maximum matching the same size, Discrete Math. 299 (2005) 49-55.

[52] K.K. Dabrowski, M. Demange, and V.V. Lozin, New results on maximum induced matchings in bipartite graphs and beyond, Theor. Comput. Sci. 478 (2013) 33-40.

[53] W. Duckworth, D.F. Manlove, and M. Zito, On the approximability of the maximum induced matching problem, J. Discrete Algorithms 3 (2005) 79-91.

[54] M.C. Golumbic and M. Lewenstein, New results on induced matchings, Discrete Appl. Math. 101 (2000) 157-165.

[55] Z. Gotthilf and M. Lewenstein, Tighter approximations for maximum induced matchings in regular graphs, Lecture Notes in Comput. Sci. 3879 (2006) 270-281.

[56] D. Kobler and U. Rotics, Finding maximum induced matchings in subclasses of claw-free and P_5-free graphs, and in graphs with matching and induced matching of equal maximum size, Algorithmica 37 (2003) 327-346.

[57] L. Lovász and M.D. Plummer, Matching Theory, vol. 29, Annals of Discrete Mathematics, North-Holland, Amsterdam, 1986.

[58] V.V. Lozin, On maximum induced matchings in bipartite graphs, Inf. Process. Lett. 81 (2002) 7-11.

[59] H. Moser and S. Sikdar, The parameterized complexity of the induced matching problem, Discrete Appl. Math. 157 (2009) 715-727.

[60] H. Moser and D.M. Thilikos, Parameterized complexity of finding regular induced subgraphs, J. Discrete Algorithms 7 (2009) 181-190.

[61] L.J. Stockmeyer and V.V. Vazirani, NP-completeness of some generalizations of the maximum matching problem, Inf. Process. Lett. 15 (1982) 14-19.