Ca' Foscari University of Venice
Department of Environmental Sciences, Informatics and Statistics

Master Degree in Computer Science
Second Cycle (D.M. 270/2004)

Final Thesis

Matching Hypergraphs with Game Theory

Graduand:
Giulia Sandi
Matriculation Number 865144

Supervisor:
Ch. Prof. Marcello Pelillo

Assistant supervisor:
Sebastiano Vascon, PhD

Academic Year 2017-2018

To Eva, Amelia and Febe:
don't be afraid to make a choice,
don't be frightened by change.
The future is how you decide it to be.

Contents

Abstract iii

1 Introduction 1

1.1 Thesis Goal . . . 3

1.2 Thesis Structure . . . 4

2 The Hypergraph Matching Problem 5

2.1 Basic Definitions on Hypergraphs . . . 5

2.2 Classification of Matching Problems . . . 7

2.3 The Association Hypergraph . . . 8

3 Hypergraph Matching as a Non-Cooperative Game 11

3.1 From Matching Graphs to Finding Cliques . . . 11

3.2 The Hypergraph Clustering Game . . . 12

3.2.1 A Continuous Formulation of the Maximal Clique Problem . . . 13

3.2.2 Notions from Game Theory . . . 15

3.2.3 Evolution towards an Equilibrium Point . . . 17

3.3 Finding Isomorphisms Using the Baum-Eagon Dynamics . . . 18

4 Experimental Set-up 23

4.1 Coding the Algorithm: Choosing the Language to Use . . . 26

4.2 Pruning the Association Hypergraph . . . 28

4.2.1 Pruning the Complete Isomorphism Problem . . . 29

4.2.2 Pruning the Sub-Graph Isomorphism Problem . . . 32

4.3 Looking for the Optimum . . . 33

5 Experimental Results **39**

5.1 Results for the Complete Isomorphism Problem 39

5.1.1 Correctness Evaluation 39

5.1.2 Running Time Evaluation 41

5.2 Results for the Sub-Graph Isomorphism Problem 48

5.2.1 Correctness Evaluation 48

5.2.2 Running Time Evaluation 61

6 Conclusions **69**

Acknowledgements **77**

Bibliography **79**

Abstract

The problem of finding isomorphisms, or matching partitions, in hypergraphs has gained increasing interest from the scientific community in the last years, particularly in the Computer Vision field. This is due to the advantages that arises from overcoming the limitations provided by pairwise relationship, thus encoding a bigger pool of information.

Association graph techniques represent a classical approach to tackle the graph matching problem and recently the idea has been generalized to the case of uniform hypergraphs. In this thesis, the potential of this approach, employed together with elements from the Evolutionary Game Theory, is explored. Indeed, the proposed framework uses a class of dynamical systems derived from the Baum-Eagon inequality in order to find the maximum (maximal) clique in the association hypergraph, that corresponds to the maximum (maximal) isomorphism between the hypergraphs to be matched.

The proposed approach has extensively been tested with experiments on a large synthetic dataset. In particular both the pure isomorphism and the subgraph isomorphism problems have been analysed. The obtained results reflect the different complexity classes these problems belong to, thus showing that despite its simplicity the Baum-Eagon dynamics does an excellent job at finding globally optimal solutions in the pure isomorphism case, while in the subgraph case the use of more complex dynamics might be more suitable.

Keywords

Hypergraph Matching, Sub-hypergraph Isomorphism, Complete Isomorphism, Baum Eagon Inequality, Association Hypergraph, Evolutionary Game Theory

Chapter 1

Introduction

The problem of matching hypergraphs is the problem of understanding whether, given two hypergraphs, there exists a subset of nodes, in common between the structures, that are connected by hyperarcs exactly in the same way. In particular, when we want to completely match all the nodes of two hypergraphs we can speak about finding an *isomorphism* between the structures. As Professor Douglas R. Hofstadter wrote in his book *Gödel, Escher, Bach: an Eternal Golden Braid* [16], winner of the Pulitzer Prize:

> The word *isomorphism* applies when two complex structures can be mapped onto each other, in such a way that to each part of one structure there is a corresponding part in the other structure, where "corresponding" means that the two part play similar roles in their respective structures.

In recent years hypergraphs (as opposed to graph) have been used to encode structural information in different fields such as computer vision, pattern recognition and machine learning, thanks to the advantages that arise from considering relationships among more than two elements, thus encoding an higher pool of information. Specifically, finding a correspondence between structures of this kind is of particular interest for solving problems such as, e.g., object recognition, feature tracking, shape matching and scene registration, where high-relations are naturally used.

Broadly speaking, given two images, the most significant features are extracted from them, together with their relations. The job consists in coupling these features, one for each image, in a way that the relations among features within the same image are preserved. If we represent the features of each image as the vertex set of an hypergraphs and describe the relations with the edge set, the transformation of the problem from coupling feature to matching hypergraphs is straightforward. However matching graphs, and therefore hypergraphs, is a task that has been proven to be NP-hard [12], so an approximation needs to be used.

When dealing with graph (instead of hypergraphs) matching problems, classical approaches generally measures the relations with a notion of similarity that incorporates only the geometrical structure of the problem, such for example the distance between feature points (see e.g. [14, 21, 32]). However pairwise relations are not able to completely describe the geometrical structure, and are not sufficient for successfully solving the features matching problem. Moreover, generally they do not consider any appearance information, even though this would greatly help in figuring the problem out. For this reason moving from graph matching to hypergraph matching is of great help, since using high-order relations gives the possibility to incorporate more complex geometric information and/or appearance information.

In recent years different studies started using high-order relations, thus trying to solve the hypergraph matching problem. In general all of them transformed this problem into an optimization one: maximizing the sum of the matching scores. In [20], the authors tried to solved the problem by reinterpreting the random walk concept on an association hyper-graph in a probabilistic manner. In [9] the authors formulate the problem as the maximization of a multilinear objective function, representing the total matching scores, that is the sum of the affinities between all permutations of the feature tuples. In [36] the problem has been solved using a discrete formulation that is optimized with a particular adaptive discrete gradient assignment method, that is theoretically guaranteed to converge to a fixed discrete point. Many other studies can be found on the topic, starting from the references in the aforementioned works.

Among the newest publications, [37] and [17] extend a successful formulation for matching graphs [25, 26], to hypergraphs: the classical approach of computing the *association hypergraph* from the two structures being matched is used, and then techniques from Evolutionary Game Theory are applied on the newly built hypergraph. In particular dynamics inspired from the Baum-Eagon inequality [2, 29, 27] are used. The authors of the aforementioned papers obtained good results on 3-graphs, but the developed approaches can easily be applied to uniform hypergraphs of larger cardinality.

1.1 Thesis Goal

Motivated by these recent works, this thesis aims to systematically explore the potential of this approach on two versions of the hypergraph matching problem: the complete isomorphism and the sub-graph isomorphism cases. In order to do so, two synthetic datasets have been created, one for each problem, made of uniform, un-weighted, un-directed hypergraphs of different cardinalities .

For the hypergraph isomorphism problem 10500 experiments have been performed on uniform hypergraphs of different order, cardinality and density.

For the sub-graph isomorphism problem 4500 experiments haven bee performed on uniform hypergraphs of cardinality 3, with different density values. The order of the biggest hypergraph has been fixed, while sub-graphs with different number of nodes are taken into consideration.

The reason for using synthetic datasets lays on the will to explore the behaviour of this approach in all the different possible scenarios, thus being able to analyse those structures that are quite rare in real problems as well as those structures that are often encountered when dealing with concrete situations.

The results obtained for the isomorphism problem are impressive: the proposed framework correctly identifies 100% of the isomorphisms, with all the different dynamics tested. Instead the results on sub-graph isomorphisms are not so good, returning in total only 33% of correct isomorphisms, even though the correctness rate is not uniformly distributed but strongly depends on the connectivity rates of

the hypergraphs being matched and on the order of the sub-graph to be found.

Given the outstanding results obtained on complete isomorphism, an extract of this thesis has been submitted in form of a paper [31], to the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition (S+SSPR 2018), being accepted as an oral presentation.

1.2 Thesis Structure

The outline of the thesis is as follows:

- Chapter 2 presents some basic definitions on hypergraphs, including the concept of association hypergraph that are fundamental to outline the problem. Moreover a characterization of the different types of matching between hypergraphs is given;
- Chapter 3 describes all the steps that are used to solve the problem, introducing some notions from Evolutionary Game Theory and the Baum-Eagon inequality with the related dynamics, also in their exponential form;
- Chapter 4 introduces the experimental set-up, the coding languages, the datasets, the optimizations and in general the methodology used to perform the experiments, and the way in which the results are evaluated;
- Chapter 5 presents the results obtained, in terms of precision and computational time, both for the hypergraph isomorphism and for the sub-graph isomorphism problems;
- Chapter 6 concludes this thesis giving also some hints on possible future extensions to this work.

Chapter 2

The Hypergraph Matching Problem

In this chapter the hypergraph matching problem is explained in details. In particular some useful notations and definitions are introduced, while the approach that has been used to solve the problem is analysed in chapter 3. This thesis deals with complete isomoprphism and sub-graph isomorphism on undirected, un-weighted, uniform hypergraphs, however more general types of matchings are summarized.

2.1 Basic Definitions on Hypergraphs

A formal definition of *un-weighted hypergraph*, in its most general form, is given by the following:

Definition 1. *An un-weighted hypergraph is defined by the triplet* $H = (V, E, K(H))$*, where:*

- $V = \{1, \ldots, n\}$ *represents the finite set of vertices or nodes;*
- $E \subseteq 2^V \setminus 0$ *represents the finite set of hyperedges or hyperarcs of various cardinalities (where* 2^V *denotes the powerset of* V*);*
- $K(H) = \{|F| : F \in E\}$ *represents the finite set of edge types of* H*, that are all the possible cardinalities of the edges in* E*.*

When the set of edge types of an hypergraph includes more than one element, i.e. when the hyperedges have different cardinalities, the hypergraph is said to be *non-uniform*. On the contrary, when $K(H)$ is made of just one element k, we speak about *uniform* hypergraphs, or k-graphs. In the special case in which $k = 2$ we trivially come back to the classic notion of graph.

In figure 2.1 different types of hypergraphs are shown, according to their size, order and edge cardinality. From left to right we have: H_1, a uniform hypergraph with cardinality $k = 2$ (namely a graph), order 4 and size 3; H_2, a uniform hypergraph with cardinality $k = 3$, order 5 and size 3; H_3, a non-uniform hypergraph of order 8, size 4 and edge cardinalities $K(H) = \{2, 3, 4\}$

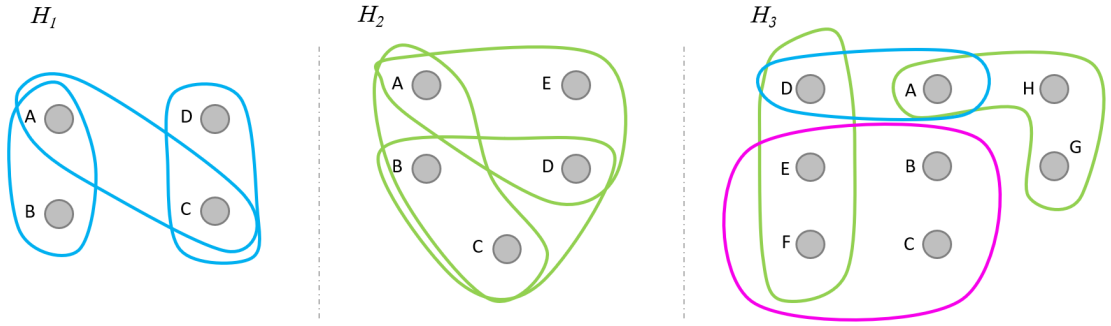

Figure 2.1: Different types of hypergraphs.

In this thesis we will focus on *uniform hypergraphs*, or k-graphs, whose edges have fixed cardinality $K(H) = \{k\}$, with $k \geq 2$.

The *order* of H is the number of its vertices, while its *size* is the number of edges.

Given n nodes $i_1, ..., i_n \in V$, they are said to be *adjacent* if $\{i_1, ..., i_n\} \in E$. The *degree* of a vertex $i \in V$, denoted by $\deg(i)$, is the number of vertices adjacent to it.

Given a k-graph $H = (V, E)$, a *clique* is defined as a subset of vertices C such that for all distinct $i_1, ..., i_k \in C$, they are mutually adjacent, that is $\{i_1, ..., i_k\} \in E$. A *maximal* clique is defined as a clique that is not contained in any larger clique, while a *maximum* clique is defined as the largest clique in the hypergraphgraph. The cardinality of the maximum clique is called the clique number $\omega(H)$.

In order to ease the lecture of this thesis, from now on the words hypergraph, k-graph, or graph will be used interchangeably for indicating uniform un-weighted

hypergraphs with hyperarcs of cardinality k, when no ambiguity can arise from the context. Analogously the words edge or arc will refer to hyperedges or hyperarcs.

2.2 Classification of Matching Problems

Definition 2. *Given two hypergraphs $H' = (V', E', k)$ and $H' = (V'', E'', k)$, an isomorphism between them is defined by any bijection $\phi : V' \to V''$ for which $\{i_1, ..., i_k\} \in E' \Leftrightarrow \{\phi(i_1), \ldots, \phi(i_k)\} \in E''$, $\forall i_1, ..., i_k \in V'$. If an isomorphism exists between two hypergraphs, then they are said to be isomorphic.*

Given this definition of isomorphism, the matching problem can be divided into three categories that belong to different class of complexity:

- *complete isomorphism problem*: understanding if the two hypergraphs are isomorphic;
- *sub-graph isomorphism problem*: understanding if the smaller of the two graphs is isomorphic to a subset of the larger graph;
- *maximum common subgraph problem*: finding a match between the largest subset of vertices of H' and H'', such that the subgraphs defined by these subsets of nodes are isomorphic.

Solving these three problems means therefore to decide if an isomorphism between at least sub-graphs exists, and, in case of a positive answer, find the isomorphism itself.
Usually, when speaking about the hypergraph matching problem, reference is made to the maximum common sub-graph problem, that is the most general of the three categories, and includes the graph and sub-graph isomorphism problems as special cases. Finding a maximal common subgraph, i.e. an isomorphism between sub-graphs that is not included in any larger sub-graph isomorphism, is a simple version of the hypergraph matching problem.

These different categories belong to different class of complexity [12]. In particular the isomorphism problem is the easiest one and is located in the low hierarchy

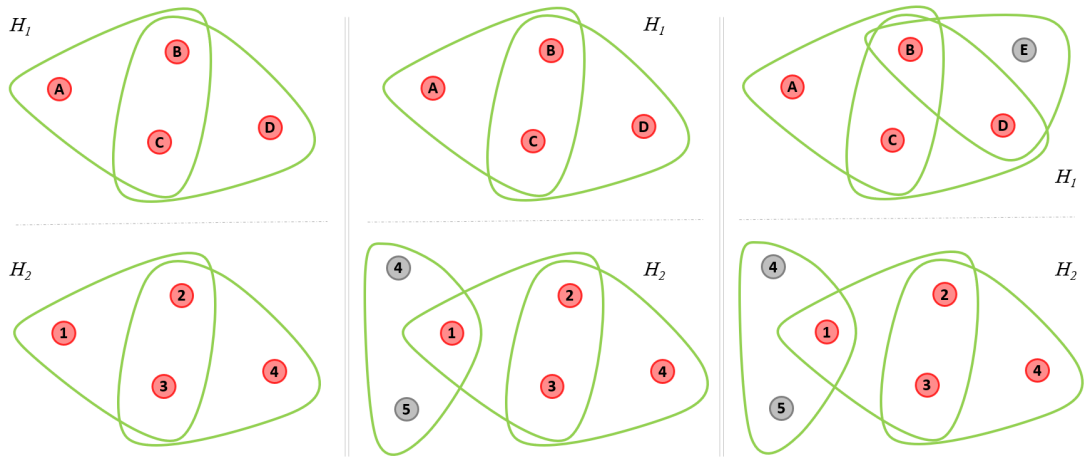

Figure 2.2: Different types of matching in 3-graphs. From left to right: complete isomorphism, subgraph isomorphism, maximum common subgraph. Red nodes are the matched ones, while grey nodes are not matched.

of NP [33], thus meaning that it is not NP-complete nor it has been shown to be in P. The sub-graph isomorphism is more complex, as is has be shown to be an NP-complete problem [7]. Finally the general matching problem is the most complex one, and it is known to be NP-complete. The complexity classes indicated before have been demonstrated to be valid on 2-graphs, however, given the analogous structure of the problem and the results obtained from the performed experiments, we can imagine that this classification can be similarly extended to general k-graphs.

In this thesis the complete and sub-graph isomorphism problems are investigated, with different results that reflects the complexity of the problems (see chapter 5), while a deep analysis on maximum common sub-graph problems is left aside for a future investigation.

2.3 The Association Hypergraph

The notion of *association graph*, a useful auxiliary graph structure designed for solving general graph matching problems, has been introduced in [1] and also in [19], and can be easily generalised to uniform hypergraphs.

Definition 3. *The association hypergraph derived from unweighted uniform hypergraphs* $H' = (V', E')$ *and* $H'' = (V'', E'')$ *is the undirected unweighted hypergraph*

$H = (V, E)$ *defined as:*

$$V = V' \times V''$$

and

$$\begin{aligned} E = \{\{(i_1, j_1), ..., (i_k, j_k)\} \in V' \times V'' : \\ i_1 \neq ... \neq i_k, j_1 \neq ... \neq j_k, \\ \{i_1, ..., i_k\} \in E' \Leftrightarrow \{j_1, ..., j_k\} \in E''\}. \end{aligned}$$

With the definitions we have given in this section we have now all the tools necessary to develop our approach for matching hyergraphs.

Chapter 3

Hypergraph Matching as a Non-Cooperative Game

Given the good results obtained by Pelillo in [25, 26] in finding isomorphism between graphs, the same approach has been extend to hypergraphs. In this section each step is analysed and described in details. In particular in section 3.1 the problem is moved from finding matches to finding cliques, then, in section 3.2 an overview on Game Theory is given, with some theoretical properties that are useful for solving the matching problem. A more detailed discussion of the topic can be found in the original works by Hofbauer and Sigmund [15] and Weibull [35]

3.1 From Matching Graphs to Finding Cliques

In the following result, which generalises to hypergraphs an analogous result obtained in [25, 26] for graphs, a one-to-one correspondence between the graph isomorphism problem and the maximum clique problem is demonstrated.

Theorem 1. *Let $H' = (V', E')$ and $H'' = (V'', E'')$ be two hypergraphs of order n and edge cardinality k, and let $H = (V, E)$ be the related association k-graph. Then H' and H'' are isomorphic if and only if $\omega(H) = n$. In this case, any maximum clique of H induces an isomorphism between H' and H'', and vice versa. In general, maximum and maximal common subgraph isomorphisms between H' and H'' are in one-to-one correspondence with maximum and maximal cliques in H, respectively.*

Sketch of proof. Suppose that the two k-graphs are isomorphic, and let ϕ be an isomorphism between them. Then the subset of vertices of H defined as $C_\phi = \{(i, \phi(i)) : \forall i \in V'\}$ is clearly a maximum clique of cardinality n. In reverse, let C be an n-vertex maximum clique of H, and for each $(i, h) \in C$ define $\phi(i) = h$. Then it is easy to see that ϕ is an isomorphism between H' and H'' because of the way the association k-graph is constructed. The proof for the general case is analogous.

With this result we can move our original problem of finding a matching between graphs to the problem of finding a maximum clique in the related association graph. Obviously this does not reduce the complexity of our problem since again, the clique decision problem is NP-complete [18], however it gives us the possibility to use one of the existing approaches that are known to work well in solving this second problem.

Unfortunately the cost of computing the association hypergraph has not to be underestimated, since it is not a trivial computation. In fact the number of nodes in the association graph, corresponding to all the possible associations between the nodes in the graphs to be matched, increases exponentially, thus creating computational problems both in terms of time and memory. For this reason some type of pruning has been taken into consideration, in order to resize this problem, as much as it is possible, and will be accurately defined in chapter 4.

3.2 The Hypergraph Clustering Game

Clustering means grouping together some elements of a set in such a way that:

- elements belonging to the same cluster are maximally similar, and
- elements belonging to different clusters are maximally dissimilar,

according to some predefined similarity measure.

This definition reflects the concepts of *internal coherency* and *external incoherency* that are at the basis of the idea of cluster. The problem of finding a clique in an un-weighted undirected hypergraph can be seen as the problem of clustering the

nodes in its vertex set, grouping together nodes that are connected to all the other nodes of the cluster. The clustering problem has been successfully solved in [29] using a game-theoretic approach, considering it as a multi-player non-cooperative "clustering game", where the nodes of the k-graph are the pure strategies that the k players can play. An interest aspect of this approach, that distinguish it from other clustering approaches, is that the clusters are "extracted" one at a time, starting from the most coherent cluster and continuing in decreasing order of cohesiveness. This is of particular attraction for our purposes, in fact we are interested in finding the maximum clique, therefore finding only the most coherent cluster is enough for us, as we have no need in grouping in some way the remaining elements.

In this section first a continuous formulation of the maximum clique problem is given, in order to define the pay-off function of the clustering game, then some notions on the clustering game are given, and finally the evolution of the of the optimization process from one initial point to an stable equilibrium point is described.

3.2.1 A Continuous Formulation of the Maximal Clique Problem

Consider the arbitrary undirected hypergraph of order n, $H = (V, E)$, and let S_n indicates the standard simplex of $\mathbb{R}^n$:

$$S_n = \left\{ \mathbf{x} \in \mathbb{R}^n : x_i \geq 0, \text{ for all } i = 1, ..., n, \text{ and } \sum_{i=1}^{n} x_i = 1 \right\} \tag{3.1}$$

Given a subset of vertices C of H, its characteristic vector is denoted by $\mathbf{x}^c$, and it represents the point in S_n determined by:

$$x_i^c = \begin{cases} 1/|C|, & \text{if } i \in C \\ 0, & \text{otherwise} \end{cases}$$

where the cardinality of C is indicated by $|C|$.

Now, consider the *Lagrangian* of H, which is the polynomial function defined as:

$$f(\mathbf{x}) = \sum_{e \in E} \prod_{i \in e} x_i \tag{3.2}$$

A point $\mathbf{x}^* \in S_n$ is said to be a global maximizer of f in S_n if $f(\mathbf{x}^*) \geq f(\mathbf{x})$, $\forall$ $\mathbf{x} \in S_n$. On the other hand it is said to be a local maximizer if $\exists$ an $\epsilon > 0$ such that $f(\mathbf{x}^*) \geq f(\mathbf{x})$ $\forall$ $\mathbf{x} \in S_n$ whose distance from $\mathbf{x}^*$ is smaller than ϵ. Moreover, $\mathbf{x}^*$ is said to be a strict local maximizer if $f(\mathbf{x}^*) = f(\mathbf{x})$ implies $\mathbf{x}^* = \mathbf{x}$.

In the case of graphs (namely, $k = 2$), the Motzkin-Straus theorem [22] establishes a noteworthy relation between global (local) maximizers of the Lagrangian in S_n and maximum (maximal) cliques of the graph itself. In particular, it asserts that in a graph G, a subset C of its vertices is a maximum clique if and only if a global maximizer of the Lagrangian of the graph G in the standard simplex S_n is in the form of the characteristic vector $\mathbf{x}^C$. This result interestingly gives the opportunity to move from a discrete to a continuous formulation of our problem. Even though this new formulation is extremely convenient it comes with a price: there are in fact spurious solutions, that may interfere with the process of finding the clique. A later formulation introduced in [5] solve this problem by adding a regularization factor to the main diagonal of the Lagrangian, thus eliminating the possibility of encountering spurious solutions, while at the same time, increasing the complexity of the polynomial to be optimized.

The formulation of $f(\mathbf{x})$ given in equation (3.2) is the same used in [17] and [37], and is motivated by the Motzkin-Straus theorem on graphs. Therefore I am going to focus on the this function in the experiments. However different formulations are possible, for example the ones that present the generalizations of the Motzking-Straus theorem, together with Bomze's formulation, to both uniform and non-uniform hypergraphs in [30] and [28] respectively.

Given the definitions and results just presented, the problem of finding an

isomorphisms can be reduced to solving the following:

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{x}) = \sum_{e \in E} \prod_{i \in e} x_i \\ \text{subject to} \quad & \mathbf{x} \in S_n \end{aligned} \tag{3.3}$$

A simple and effective way for optimizing this function is to use tools from evolutionary game theory, a biological discipline by J. Maynard Smith [34] that aims at modelling the evolution of animal behaviour, in contrast to human behaviour, with elements from the theory of games.

3.2.2 Notions from Game Theory

In classical game theory [11] any strategic game can be described by the following elements:

- the finite set of $k \geq 2$ players $P = 1, 2, \ldots, k$;
- the "pure strategy space" S_j for each player j, containing all the actions available to that player;
- the "pay-off function" π_j for each player, that for each pure strategies profile, defined as an ordered set of pure strategies played by the different players in a given moment, assigns to player j the related score.

The reader should note that we are dealing with a *doubly symmetric game*, since we are considering only undirected hypergraphs. For this reason the set of actions available to each player is always the same, and also the pay-off function is the same for all the players. Therefore we can speak about a single space containing all the n pure strategies available to all the players $S = 1, 2, \ldots, n$, and we can define a pure strategy profile as s $= (s_1, ..., s_k) \in S_n$. For the same reason we have a single pay-off function $\pi : S^k \to \mathbb{R}$.

Now let's take into consideration a vast population of individuals that we assume has the following characteristics:

- the population members are part of the same species, and contend a specific limited resource, for example territory or food;

- this conflict can be modelled as a symmetric k-player game, where the players are k-ples of individuals that are drawn randomly from the population;
- reproduction is asexual, so the genetic information is passed as it is from father to son, and the only possible modification is due to genetic mutations;
- players can not decide how to play, but act according to a inherited genetic behaviour or *pure strategy*;
- game's scores are given by the pay-off function according to the reproductive success or Darwinian fitness.

Given this population of individuals and the definition of game presented before, we are interested in understanding how the population evolves in time, supposing some evolutionary selection process changes the behaviour, or strategies, of the individuals themselves. Let $x_i(t)$ be the percentage of individuals playing the i^{th} strategy at time t, for all $i \in S$. The state of the system at time t is simply the vector $\mathbf{x}(t) = (x_1(t), ..., x_n(t))^T$. Since, obviously, at any time t, $\sum_i x_i(t) = 1$ and $x_i(t) \geq 0 \ \forall i$, the set of all possible states is given by S_n, as defined in equation (3.1). As we already said, in this biological context, the notion of pay-off can be simply measured in terms of reproductive success, that is the expected size of offspring for a particular player, or Darwinian fitness.
The definition of an utility function $u : S^k \to \mathbb{R}$ in the form:

$$u(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}) = \sum_{(s_1,...,s_k) \in S_n} \pi(s_1, ..., s_k) \prod_{i=1}^{n} x_i^{(i)} \tag{3.4}$$

is suitable for illustrating the notion of *average pay-off* of the population, that is the mean utility that the whole community gets according to the pure strategies played by the individuals. In particular, the pay-off obtained by a single population's member playing strategy j can be describe by $u(e^j, \mathbf{x}^{[k-1]})$, where e^j is the pure strategy with the j^{th} component equal to 1, and all the other components equal to 0, while $\mathbf{x}^{[k-1]}$ is an abbreviation for the series of $k-1$ identical strategies $(\mathbf{x}, ..., \mathbf{x})$. Using the same notation the average pay-off of the entire population can be represented by $u(\mathbf{x}^{[k]})$.

3.2.3 Evolution towards an Equilibrium Point

Starting from an initial state at time $t = 0$, the strategies' distribution among the population changes accordingly to the Darwinian process, in a way such that the fittest strategies spread, while the weakest ones disappear. From this viewpoint, the average pay-off is going to increase, up to reaching a stable point. In two papers of 1950 [24] and 1951 [23], John Nash defined an equilibrium point as:

> Any n-tuple of strategies, one for each player, may be regarded as a point in the product space obtained by multiplying the n strategy spaces of the players. One such n-tuple counters another if the strategy of each player in the countering n-tuple yields the highest obtainable expectation for its player against the $n - 1$ strategies of the other players in the countered n-tuple. A self-countering n-tuple is called an equilibrium point.

In other words, an equilibrium is a point in which all the players receive the same expected pay-off, thus no individual of the population have any incentive to change its own strategy. Formally a point $\mathbf{x} \in S_n$ is a *Nash equilibrium* if:

$$u(\mathbf{e}^j, \mathbf{x}^{[k-1]}) \leq u(\mathbf{x}^{[k]}), \quad \text{for all } j \in S. \tag{3.5}$$

However the notion of equilibrium is too weak, because it is not stable under small perturbations. In fact we would like that if a small number of "intruders", playing a different strategy, arrives in the population, the strategy that has spread around the entire population through time, will be stable enough to win against the strategy of the newcomers, thus overcoming the intruders and spreading again through all the population. For this reason J. M. Smith in [34] defined the concept of Evolutionarily Stable Strategy (ESS), that is specialization of Nash equilibrium. Even though in this work the definition is given for a 2-players game, a generalization to k-players games is provided in [6].

Suppose we have a population where a large fraction $(1 - \epsilon)$ of individuals plays strategy $\mathbf{p} \in S_n$, while a smaller fraction ϵ of mutants plays strategy $\mathbf{q} \in S_n$. The

resulting population is given by $\mathbf{w}_\epsilon = (1-\epsilon)\mathbf{p} + \epsilon\mathbf{q}$, and is equal to a single player playing a mixed strategy. Then the pay-off obtained by the initial individuals and the mutant ones are, respectively: $u(\mathbf{p}, \mathbf{w}_\epsilon^{[k-1]})$ and $u(\mathbf{q}, \mathbf{w}_\epsilon^{[k-1]})$. We can say that $\mathbf{p}$ is *evolutionarily stable* against $\mathbf{q}$ if:

$$u(\mathbf{p}, \mathbf{w}_\epsilon^{[k-1]}) > u(\mathbf{q}, \mathbf{w}_\epsilon^{[k-1]}) \tag{3.6}$$

This inequality must hold for every distribution of mutant individuals $\mathbf{q} \in S_n \setminus \mathbf{p}$, as long as the fraction ϵ is sufficiently small. Intuitively, if the biggest part of the population plays strategy $\mathbf{p}$, when the k players are randomly selected most of them will play $\mathbf{p}$, so if $\mathbf{p}$ performs better than $\mathbf{q}$, all individuals playing $\mathbf{q}$, will die thanks to natural selection. On the other hand, if $\mathbf{p}$ and $\mathbf{q}$ perform in the same way against $\mathbf{p}$, it is important to understand how the strategies perform against $\mathbf{q}$, and in particular, in order for $\mathbf{p}$ to be an ESS, $\mathbf{p}$ must perform better than $\mathbf{q}$ when playing against $\mathbf{q}$ itself. In this way, if all the players of a population play $\mathbf{p}$, they will win against a small group of invaders playing strategy $\mathbf{q}$. The stability of the strategy to small perturbations is the peculiarity of ESS, and it is also the main difference from the concept of Nash equilibrium, of which ESS are a refinement (proof can be found in [6]).

3.3 Finding Isomorphisms Using the Baum-Eagon Dynamics

In [29] Rota Buló and Pelillo show in details a connection between finding a cluster in an hypergraph and finding an Evolutionarily Stable Strategy of the related Hypergraph Clustering Game. In particular, given the function

$$f(\mathbf{x}) = u(\mathbf{x}^{[k]}) = \sum_{e \in E} \omega(e) \prod_{j \in e} x_j \tag{3.7}$$

they showed that strict local maximizers of f in S_n are in one-to-one correspondence with the ESS of the related game.

Looking at equation 3.2, we can easily see that it corresponds to the average pay-off of the entire population as defined in equation 3.7, when the $\omega(e)$ assumes only values in $\{0, 1\}$, that is when the hypergraph is unweighted, as it is in our case.

Therefore we can reduce the problem of finding an isomorphism between two hypergraphs to the following (linearly constrained) polynomial optimization problem:

$$\begin{aligned} \text{maximize} \quad & f(\mathbf{x}) = \sum_{e \in E} \prod_{i \in e} x_i \\ \text{subject to} \quad & \mathbf{x} \in S_n \end{aligned} \tag{3.8}$$

A simple and effective way of optimizing this function is to use a result introduced by Baum and Eagon [2] in the late 1960s. They presented a class of non-linear transformations in the standard simplex, and proved a central result, generalizing a previous one introduced by Blakley [4] on related characteristics for particular homogeneous quadratic transformations. The following theorem presents a result known as the Baum-Eagon inequality.

Theorem 2. (Baum-Eagon [2]) *Let $Q(\mathbf{x})$ be a homogeneous polynomial in the variables x_j with non-negative coefficients, and let $\mathbf{x} \in \Delta$. Define the mapping $\mathrm{z} = \mathcal{M}(\mathbf{x})$ from Δ to itself as follow:*

$$z_j = x_j \frac{\partial Q(\mathbf{x})}{\partial x_j} \Bigg/ \sum_{l=1}^{n} x_l \frac{\partial Q(\mathbf{x})}{\partial x_l}, \quad j = 1, ...n. \tag{3.9}$$

Then $Q(\mathcal{M}(\mathbf{x})) > Q(\mathbf{x})$ unless $\mathcal{M}(\mathbf{x}) = \mathbf{x}$.

A continuous mapping as the one defined in this theorem is known as a *growth transformation*. Interestingly, only first-order derivatives are used in the definition of the mapping $\mathcal{M}$, that is yet able to increase Q taking a finite number of steps, being in this way sharply in contrast with classical gradient methods that need to compute high-order derivatives in order to define the size of the infinitesimal steps to be taken. Moreover gradient descend need to perform some projection operator, causing some problems for points on the boundary. Instead, with theorem 2, only a computationally easy normalization on rows is needed.

For these reasons we can affirm that the Baum-Eagon inequality supplies a

powerful tool for maximizing polynomials functions in the standard simplex, and in fact they have been used as a main component for different statistical estimation techniques developed within the theory of probabilistic functions of Markov Chains [3], as well as for analysing the dynamical properties of relaxation labelling processes [27]. Looking at the problem in equation (3.8), we can easily see that f is indeed an uniform polynomial with non-negative coefficients that have to be maximized in the standard simplex, so theorem 2 can be applied to optimize it.

Paraphrasing the Baum-Eagon inequality we can formalize the following discrete time dynamic:

$$x_j(t+1) = x_j(t)\frac{\delta_j(\mathbf{x})}{\sum_{i=1}^{n} x_i\delta_i(\mathbf{x})}, \quad j = 1...n, \tag{3.10}$$

where for readability reasons we have defined $\delta_j(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial x_j}$.

Starting at time 0 with $\mathbf{x}(0)$ inside the standard simplex S_n, the dynamic in equation 3.10 iteratively updates the state vector until convergence. Using this dynamical system the average population fitness always increases, just like the average population pay-off does in the natural selection process (see the Fundamental Theorem of Natural Selection in [15], [8] and [35]). At the end of the process the vector state will be in the form of a characteristic vector, thus thresholding it with respect to a small amount close to zero will return only the pure strategies that are still played by the population, that correspond to the nodes of the association hypergraph that belong to the (maximum) clique.

As we will see in the results section, even though there is no theoretical guarantee that the discrete time dynamic just presented reaches the optimal maximizer of the function, the results of our experiments concerning isomorphism problems show that the basin of attraction of the global maximum are quite large, so that the dynamic in equation 3.10 always returned the maximum clique in the associations graph, and never incurred in local solutions. Different results are returned instead when the problem shifts to subgraph isomorphisms, since the basin of attraction of the global maximum probably becomes smaller.

Moreover, in order to obtain a faster convergence, an exponential version of the

dynamic can be defined as:

$$x_j(t+1) = x_j(t)\frac{e^{\kappa u(\mathbf{e}^j, \mathbf{x}(t)^{[k-1]})}}{e^{\kappa u(\mathbf{x}(t)^{[k]})}}, \quad j = 1...n. \tag{3.11}$$

Clearly, even though this exponential dynamic might decrease the time needed to find the clique, it introduces a new parameter κ that has to be tuned. This parameter has to be set in a way such that the optimization process is speeded up while guaranteeing the correctness of the results. In chapter 5 some remarks are made on how the value of κ influences the search for the global maximum.

Chapter 4

Experimental Set-up

The proposed approach has been systematically tested on random hypergraphs of different sizes, with different connectivities, in order to estimate its validity, and to understand if there are substantial differences in the results for hypergraphs that differs on these parameters. Both the pure isomorphism and the subgraph isomorphism problems are investigated, even though at different levels. In fact two different datasets have been created one for each type of problem.

In the case of pure isomorphism, hypergraphs of cardinality 3, 4 and 5 are taken into consideration. For each cardinality the dataset is made of graphs of different orders, in particular: for $k = 3$ we have $n = 25$, $n = 50$, $n = 75$ and $n = 100$, for $k = 4$ we have $n = 25$ and $n = 50$, while for for $k = 5$ we have only $n = 25$. For each possible pair of cardinality-order, 15 different connectivity rates in the interval [0.01, 0.99] are tested, and for each combination of parameters 100 different hypergraphs have been considered, for a total of 10500 experiments.

Due to the increasing complexity, the dataset for the sub-graph isomorphism problem is made only of 3-graphs of order $n = 25$. As for complete isomorphism case 15 different connectivities are tested. In this case since we are dealing with sub-graphs, 3 different orders of sub-graphs has been take into consideration, and in particular sub-graphs that are 10% ($n = 22$), 20% ($n = 20$) and 30% ($n = 17$) smaller. Also in this case, for each combination of parameters 100 different experiments have been set up, for a total of 4500.

The difference in the considered orders and cardinalities both within a single

type of problem and between the two different problems, is due only to complexity reasons, and in particular to the time and memory needed to perform larger experiments. In fact the proposed framework can be used for bigger experiments both in terms of cardinality and order, as long as there is enough time and memory available.

One hypergraph for each experiment (from now on H_1) has been randomly generated following the random graph model proposed by Gilbert [13]. Given the following parameters: cardinality k, order n, connectivity rate p; each k-graph is created with the following algorithm:

1. start from n isolated nodes;
2. select a k-tuple of nodes, and generate a random number between 0 and 1. If the random number is smaller than p than add an edge connecting the selected nodes, otherwise leave them disconnected;
3. repeat step 2. for each one of the possible $\binom{n}{k}$ edges.

Once H_1 has been created, the second graph H_2 is built starting from H_1 and randomly perturbing the nodes. In the case of complete isomorphism the k-graphs here generated are ready to be matched. In the case of sub-graph isomorphism some nodes have to be taken off H_2 in order to obtain a sub-graph. Given pc the percentage of nodes that we want H_2 to be smaller than H_1, $m = \lceil n \times pc \rceil$ nodes are randomly selected, and then removed from H_2 together with all the edges incident to them. In this way both the structures to be matched are ready also for testing the framework when solving sub-graph isomorphism problems.

In order to have the possibility to perform some statistical comparison between results on different problems, only the H_1 graphs related to the complete isomorphism problem have been randomly generated, the H_1 graphs related to sub-graph matching are the same of the ones generate for the hypergraph isomorphism.

The choice of using random graphs to test the framework has been made for a couple of different reasons. First, random graphs are not bound to any specific application, thus giving the possibility to test extensively all the variety of parameters combinations, even the ones that may be uncommon in some specific

application, but still of interest. Second, they provide an experimental system that is easy to replicate and can therefore be used to make comparisons with other algorithms.

All the experiments have been run on a workstation equipped with an Intel Core i7-6800K at 3.40GHz with 128GB of RAM

In this chapter and in chapter 5, all the graphs shoving mean values, both concerning running times or correctness results, will not indicate the variance values. This is because in all the considered results the variance is very small and the 95% confidence intervals are really close to the mean values, thus not giving any additional information. Moreover each plot consists of many lines, and inserting not useful additional marks only increases confusion without bringing any benefit. Image 4.1 gives an example of plot with mean values and 95% confidence intervals, thus exemplifying the absence of need of inserting the confidence intervals in all the plots.

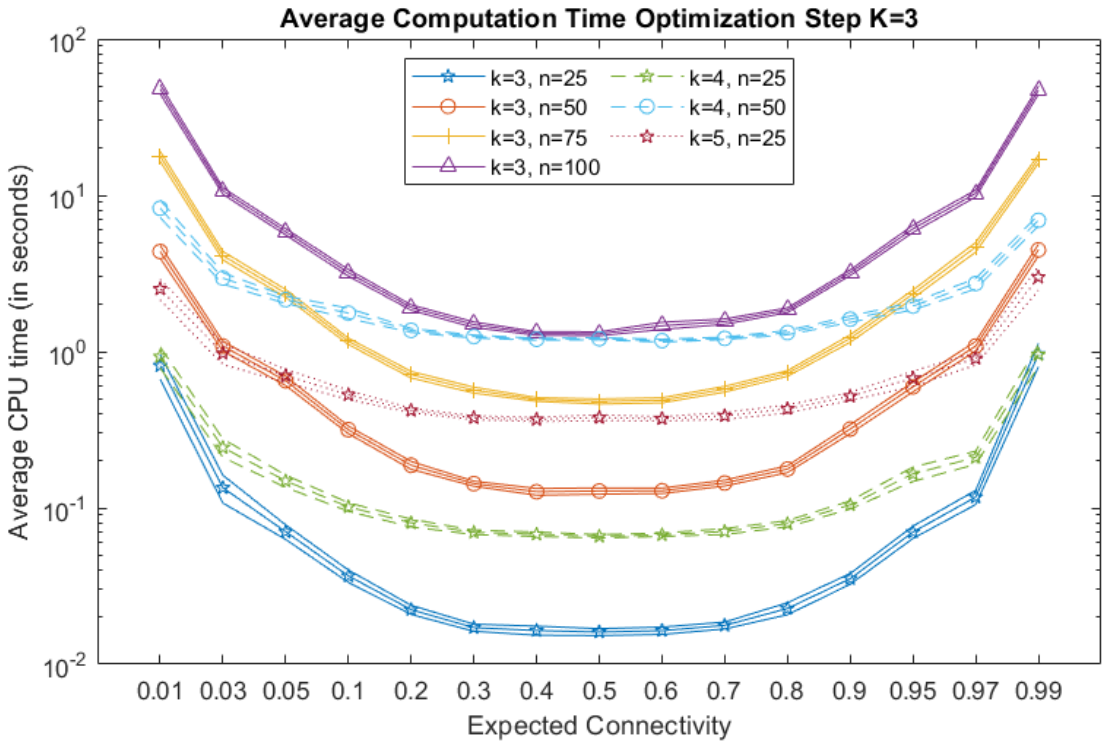

Figure 4.1: Copy of image 5.8, including also 95% confidence intervals. Given the large amount of lines and the closeness of the intervals to the mean value, it is preferable to remove the confident intervals in order to make the plot clearer, since they don't bring any additional information.

4.1 Coding the Algorithm: Choosing the Language to Use

Choosing the correct language to use has been a task that took a lot of time while developing the research project. At a first attempt MATLAB has been used, for its easiness in coding basic tasks on matrices, such as sums or multiplications, and also in providing all possible combinations of k elements taken from a large set, task that is necessary when computing the association hypergraph.

An hypergraph can be easily represented using matrices. In our case, since we are dealing with unweighted hypergraphs, two different representations are possible:

- using an NxM boolean matrix, where N is the total number of nodes, and M the total number of hyperarcs. Each point in the matrix is set to `TRUE` if the node in the row belongs to the arc in the column, otherwise is set to `FALSE`.
- using a kxM integer matrix, where k is the cardinality of the hypergraph and the M is the number of hyperarcs. Each column in the matrix contains the indexes of the nodes that are part of the hyperarc, so only positive integer values are used, ranging from 1 to N, with N the maximum number of nodes.

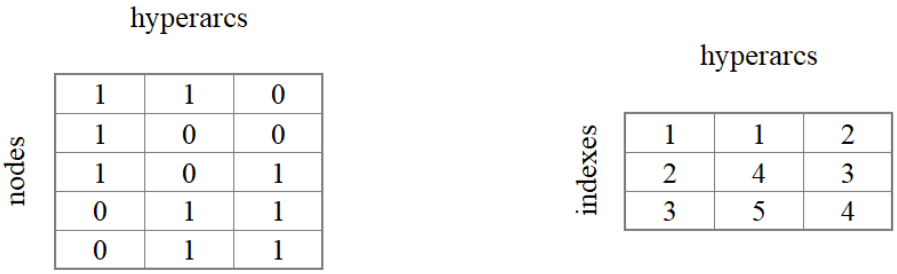

Figure 4.2: Different representations for hypergraph H_2 in figure 2.1. On the left the boolean matrix, on the right the indexes matrix.

Both these representations are very intuitive. However the first one can be easily used also with non-uniform hypergraphs, since the number of `TRUE` elements in each column gives the cardinality of the hyperarc, while the second one needs some adjustments, since the number of columns must be the largest cardinality of the hypergraph, and arcs with a smaller cardinality must include some elements set to

0. Even though it is not part of this thesis to investigate non-uniform hypergraphs, an extension of this methodology to hypergraphs is straightforward (see chapter 6 for future works), so implementing the algorithm in a way that can be easily extended to non-uniform hypergraphs without too much effort is to be considered a plus.

Two additional aspects that have to be taken carefully into consideration when choosing the implementation are memory and time. The boolean representation with a MATLAB implementation is definitely faster, since the algorithm involves only simple operations on matrices, but it becomes too much expensive in terms of memory as soon as the total number of nodes increases with respect to k, thing that happens very often when dealing with association hypergraphs even for original hypergraphs of 25 or 50 nodes. For this reason the hypergraphs have been implemented using indexes of nodes in the arcs, even though this required to add a `for` loop inside the code, thus slowing down the running time, in particular in the implementation of the Baum-Eagon dynamic.

A first set of experiments with random isomorphic 3-graphs made of 25 nodes and 15 different connectivities values has been carried out in a week, but as soon as experiments with larger structures were carried out it became clear that the developed implementation with MATLAB was too slow. For this reason a new coding in C++ has been written. However, in order not to loose all the high level functionalities provided by MATLAB for randomly creating the data and analysing the results, not all the code has been rewritten, but only the two main and very slow functions that create the association graph and use the Baum-Eagon dynamics in order to converge to a solutions. To do so, the MATLAB code has been connected to the C++ code using the C MEX API.

The results obtained are excellent, figure 4.3 shows the different running time for creating the association graph and optimizing the objective function when matching isomorphic 3-graphs with connectivities in $\{0.01, 0.03, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.97, 0.99\}$ and with 50 node. Both the implementations have been tested on the same 30 structures for each connectivity. As we can see the C++ code is more than one order of magnitude faster than the MATLAB one when

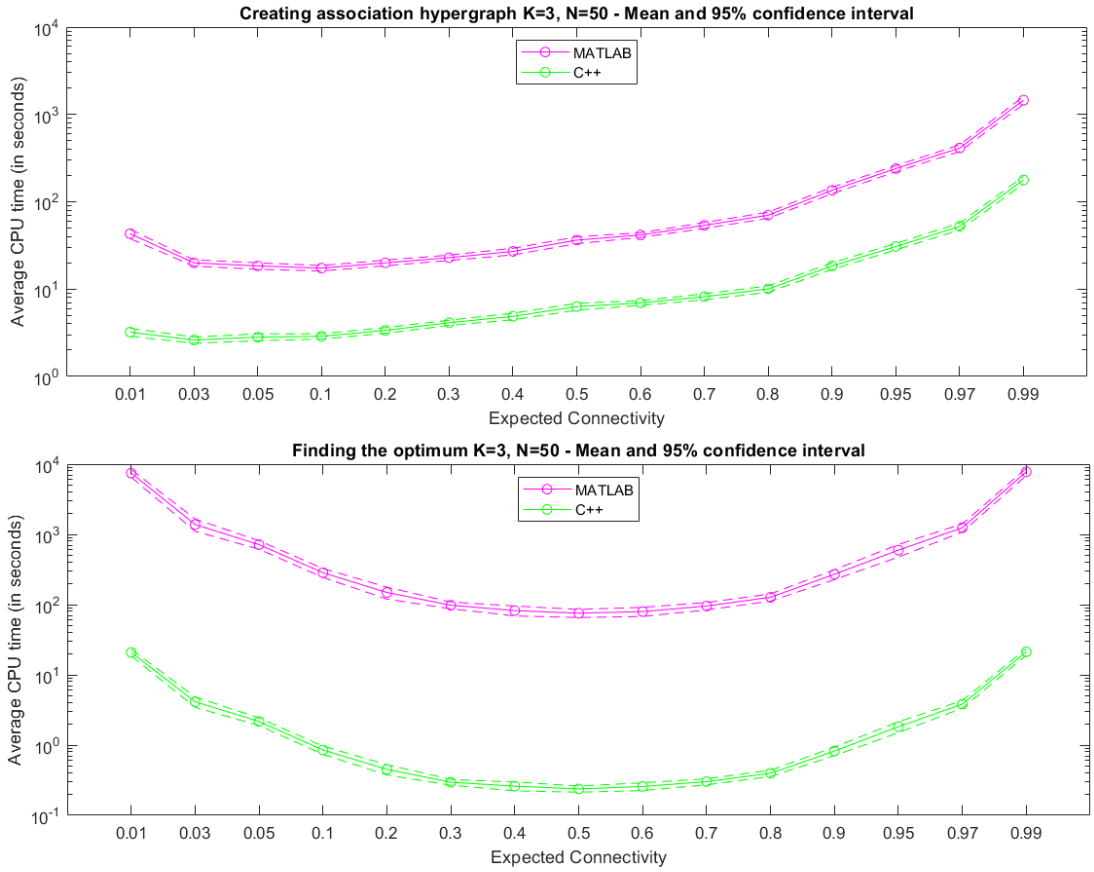

Figure 4.3: Average computational time, with 95% confidence intervals for creating the association hypergraph (top) and computing the optimum of the function (bottom) for isomorphic 3-graphs with 50 nodes. Times are in logarithmic scale.

creating the association hypergraph, and more than two order of magnitude faster when using the Baum-Eagon inequality for finding the optimum of the objective function.

For what concerns the correctness of the results, both the implementations have returned exactly the same results, as we could have expected. For these reasons, in the end, the C++ implementation has been used in all the experiments that have been carried out, even the smallest ones, in order to have comparable measurements.

4.2 Pruning the Association Hypergraph

One of the first problems to deal with is the size of the association hypergraph. Since the set of its vertices contains al the possible association of nodes, supposing we

are working with two small 3-graphs of order 25, the related association hypergraph becomes immediately of order 625. Moreover, since we have to test all the possible hyperedges, we have $\binom{625}{3} = 40495000$ checks to perform. As the size of the initial hypergraphs increases, the number of nodes and possible edges explodes. For this reason some sort of pruning, as far as it is possible without removing information, has to be taken into consideration.

First of all we can give some attention to those nodes that have degree equal to 0, i.e. the isolated nodes. Since they don't give any structure to the graph with respect to the other nodes, they can be matched with any un-connected node in the other structure. For this reason they can be simply removed from both the hypergraphs to be matched, and a simple check to see if their number is the same in the two graphs is enough. This simple idea permits to remove some nodes in the original structures, even though in general is not very useful since in our random experimental setting, only graphs with a very low connectivity (let's say 0.01 and sometimes 0.03), really have isolated nodes. However this type of pruning might be more useful in real world situations, where usually the graph involved are *scale free* models, where there might be more isolated nodes.

Another observation that has be done is the fact that, when dealing specifically with complete isomorphism and subgraph isomorphism problems, not all the association between nodes make sense.

4.2.1 Pruning the Complete Isomorphism Problem

In the case of looking for complete isomorphism we can easily understand that not all the pair of nodes have sense. In particular, since we are trying to understand if the two graphs are identical, there is no use in pairing nodes with different degree. For example, if we have to understand if H_1 is isomorphic to H_2, and in case finding the related isomorphism, coupling a node in H_1 of degree 3, with a node in H_2 of degree 6 is not useful, since in case of isomorphism this will never be a correct match.

Using this simple adjustment, the vertex set of the association hypergraph is

constructed as follow:

$$V = \{(i, j) \in V' \times V'' : deg(i) = deg(j)\}$$

and the edge set has been defined as in definition 3. When the two graphs are isomorphic, theorem 1 continues to hold, since the isomorphisms preserves the degree property of vertices. However, this simple heuristic greatly decrease the order of the association graph, and therefore its size, notably easing the optimization task.

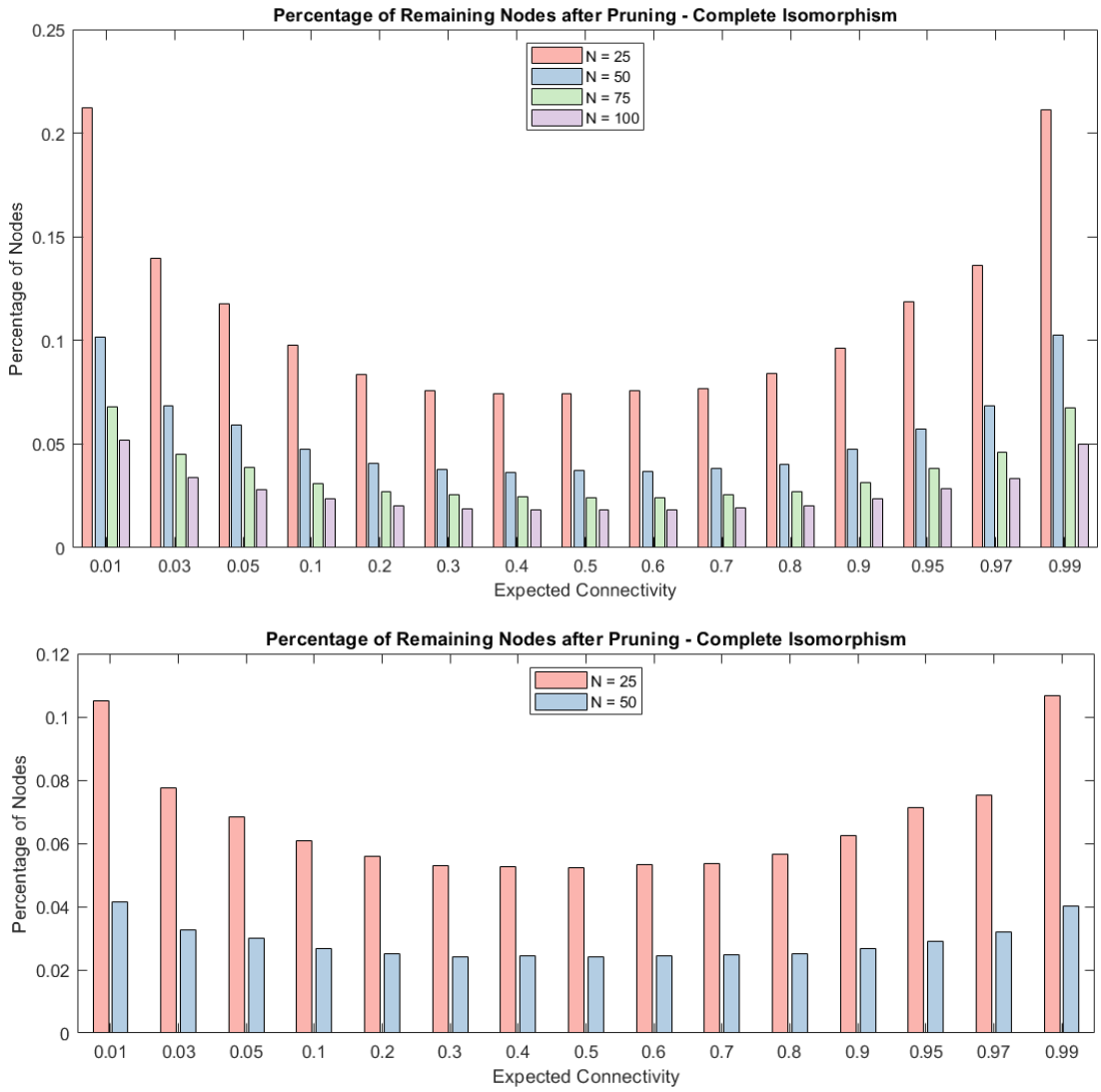

Figure 4.4: Percentage of nodes still valid after pruning the association hypergraph in the case of complete isomorphism problems with arcs of cardinality $k = 3$ (top) and $k = 4$ (bottom) .

As we can see in figure 4.4, after pruning, the association hypergraph contains only a small fraction of nodes. In particular with hypergraphs with hyperarcs of grade $k = 3$ and $n = 25$, in the best case, that is when the connectivity rate is 0.5, only about the 7% of all the possible associations are created, while in the worst case, considering the extreme connectivity rates, only around the 21% of the associations are taken into consideration. As the order of the graphs to be matched increases, the percentage of remaining nodes after pruning decreases, reaching even values around 2% in the best case with $n = 100$. It is worth remembering that if the order of the association hypergraph decreases to the 20%, its size does not decrease in the same way, but it becomes exponentially smaller. Let's take the same example as before and consider the association hypergraph between two 3-graphs with $n = 25$. If only 20% of the possible 625 combinations remains valid, we have an hypergraph of order $\simeq 125$. All the possible hyperedges to be tested are therefore $\binom{125}{3} = 317750$, that are way less than the 20% of the more than 40 millions edges that should have been tested without pruning. And even better results are obtained while k increases.

In this way, not only the time and memory needed are decreased, but also the task of the dynamic derived from the Baum-Eagon inequality is eased too, since there are less elements among which looking for the maximum clique, and the polynomial to be optimized is simpler, thanks to the smaller number of arcs involved.

Order of H_1 and H_2	Connectivity Rate				
	0.01	0.2	0.5	0.8	0.99
n=25	21,23%	8,34%	7,45%	8,39%	21,10%
n=50	10,18%	4,07%	3,70%	4,00%	10,24%
n=100	5,17%	2,02%	1,83%	2,00%	5,00%

Figure 4.5: Percentage of nodes still valid after pruning 3-graphs of different order. As the order of the graphs to be matched doubles, the order of the association hypergraf halves.

An interesting thing to note empirically from the resulted data is that, when

the number of nodes in H_1 and H_2 doubles, the percentage of nodes remaining after the pruning more or less halves (see figure 4.5 for exact numbers). Further investigations might be interesting to understand if this is just a coincidence or if there is some theoretical explanation.

4.2.2 Pruning the Sub-Graph Isomorphism Problem

When dealing with sub-graph isomorphism problems, a similar reasoning can be made, even though the degree of the coupled nodes must not be equal. Suppose we have two 3-graphs, H_1 and H_2, where H_2 has a smaller number of nodes. Intuitively, if we want to understand if H_2 is a sub-graph of H_1, when coupling the nodes in the association hypergraph, we are interested only in combining nodes of H_2 with those nodes of H_1 that have a larger number of incident arcs. Clearly, if a node has degree 6, and the other has degree 3, it will never be possible for the first to be in a sub-graph of a graph containing the second.
With this new adjustment, the vertex set of the association hypergraph in case of sub-graph isomorphism problems, is constructed as follow:

$$V = \{(i, j) \in V' \times V'' : deg(i) \geq deg(j)\}$$

where V'' is the vertex set of the hypergraph of smaller order, and the edge set has been defined as in definition 3. When the smaller graph is isomorphic to a subgraph of the bigger graph, theorem 1 continues to hold, since the isomorphisms preserves the degree property of vertices. Even though with this heuristic we have some improvements in building the association hypergraph when we have very low connectivity rates, as soon as H_1 and H_2 become a bit denser, there is no gain in pruning (see figure 4.6).

In fact, when dealing with random graphs built in the way the dataset under examination is, the degree of the nodes follows, more or less, a uniform distribution. For this reason even having a sub-graph that is just 10% smaller than the main graph is enough to decrease considerably the number of arcs incidents to all the nodes. Figure 4.7 represents the distribution of the degrees of the nodes for two

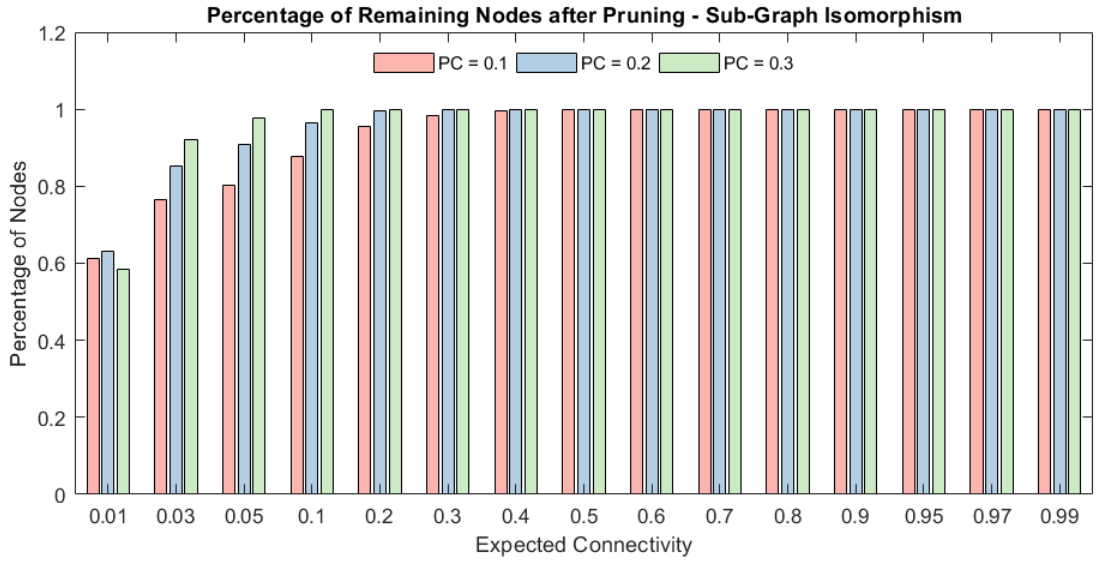

Figure 4.6: Percentage of nodes still valid after pruning the association hypergraph in the case of sub-graph isomorphism problems with arcs of cardinality $k = 3$ and order of the sub-graph 10% (pink), 20% (blue) and 30% (green) smaller than the bigger graph.

experiments randomly chosen on 3-graphs, one with connectivity rate 0.1 and the other with connectivity rate 0.5, where H_1 has order $n = 25$ and H_2 is 10%, 20% and 30% smaller.

Two different observations can be made: the smaller is the sub-graph with respect to the main graph, the smaller is the overlap between the maximum degree of nodes in H_2 and the minimum degree in H_1. Similarly, the more connected are the hypergraphs, the smaller is the probability of having an overlap.

Having no overlap means that the pruning has no effect. In fact, if the maximum degree in the sub-graph is smaller than the minimum degree in the main graph, when coupling the nodes in H_2 only with the nodes in H_1 that have larger degree, in practice all the possible couples are created, thus having no profit at all in pruning, and explaining in this way the results obtained in figure 4.6.

4.3 Looking for the Optimum

Once the association graph is built, the optimization process can start. Each experiment has been performed with the linear Baum-Eagon dynamic (see equation

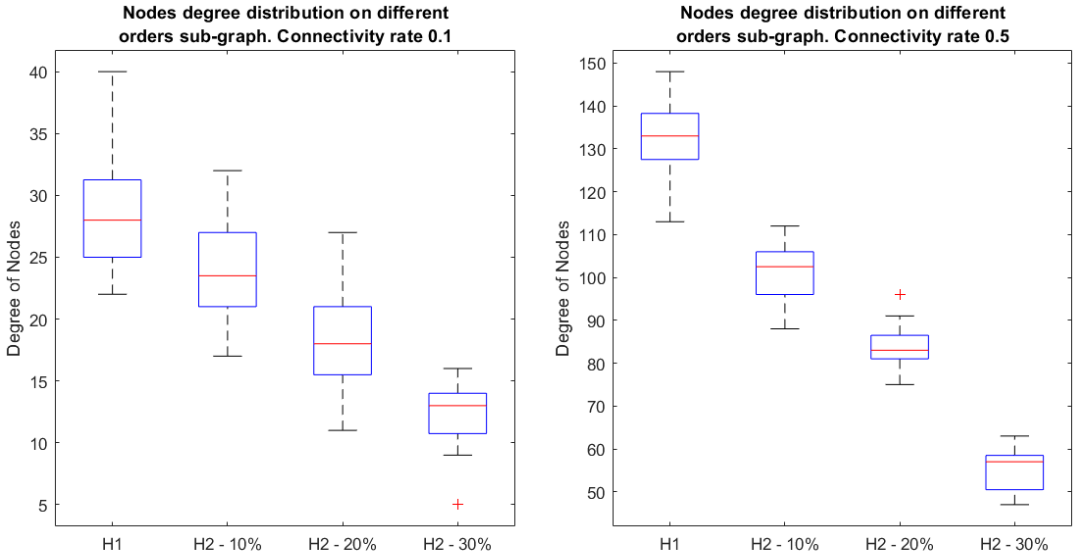

Figure 4.7: Boxplot of the distribution of the node's degree on 3-graphs with a connectivity rate of 0.1 (left) and 0.5 (right), where the bigger graph H_1 has order $n = 25$, while the sub-graphs are 10%, 20% and 30% smaller. The smaller the subgraph and/or the more connected the graphs, the further the node's distribution.

(3.10)) and with its exponential version (see equation (3.11)) with the κ parameter ranging in $\{10, 25, 50\}$.

The algorithm was started in the barycentre of the simplex and stopped when either the distance of two subsequent points was smaller than a given threshold, set to 1^{-5}, or when a maximum number of time-steps, equal to 600, has been processed. When the algorithm stops, we check if a clique has been found: in the negative case, the final point is randomly perturbed and the algorithm is started again, not using any more the barycentre of the simplex as initial point, but starting instead from the randomly perturbed point.

The resulting vector $\mathbf{x}$ is perturbed by adding a small value randomly generated from a Normal distribution with mean $\mu = 0$ and variance $\sigma = 1^{-3}$, and then it is normalized in order to remain inside the standard simplex. Perturbing is needed in order to avoid *saddle points*, that are solutions in which the algorithm is not able to return a clique because there are some symmetries in the graphs to be matched, thus resulting in more than one possible isomorphism, and the algorithm doesn't have any way of choosing one isomorphism with respect to the other. This

problem arises because the algorithm starts the search for the barycentre of the simplex. A way for avoiding this problem would have been not to start from the exact barycentre but from a closer point. However in this way we would drive the optimization process into one direction even when it is not necessary, taking the risk not to find the optimum because the search has started in the wrong way.

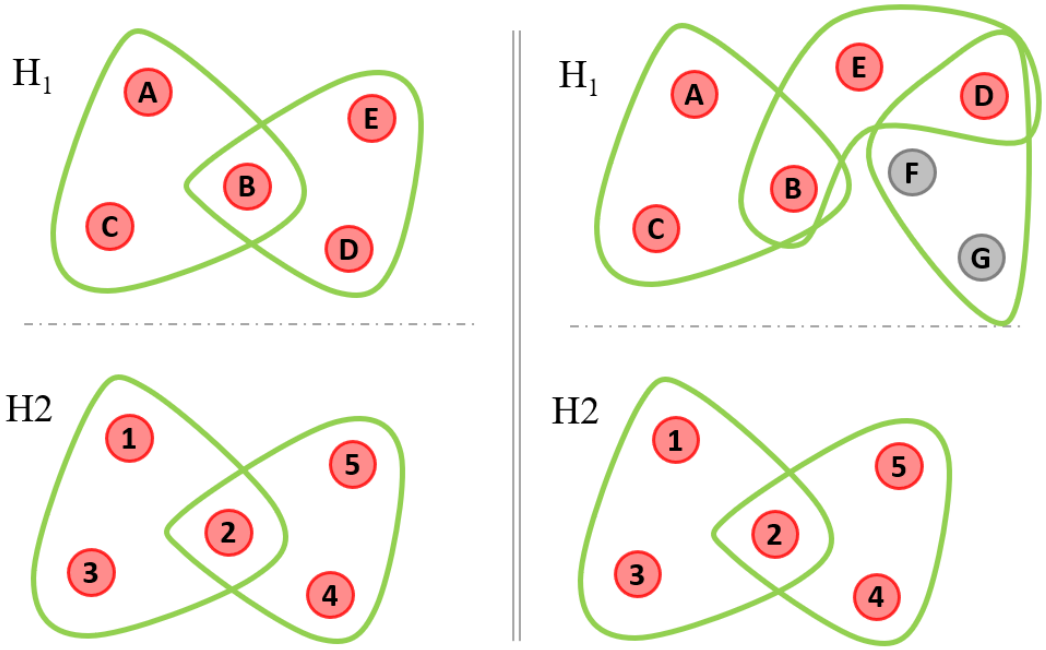

Figure 4.8: A pair of isomorphic and symmetric 3-graphs (left) and a symmetric sub-graph isomorphism (right).

Figure 4.8 shows this symmetry problem with two examples, one for the complete isomorphism and the other for the sub-graph isomorphism. In the first case there are a couple of isomorphic and symmetric 3-graphs, and the algorithm will need at least one perturbation in order to find one of the isomorphisms that arise from these two structure. In fact the only two nodes that can be coupled without making any choice are (B, 2). For what concerns the other nodes we can have 8 other different options, for example {(A, 1), (C, 3), (E, 5), (D, 4)}, or {(A, 1), (C, 3), (E, 4), (D, 5)}, but also {(A, 5), (C, 4), (E, 1), (D, 3)}. A similar reasoning can be done for the sub-graph problem, with the aggravation that also the central node 2 in H_2 can have multiple correct coupling. In fact both (2, B) and (2, D) can lead to correct isomorphisms.

Figure 4.9 shows all the steps done by the algorithm to solve the symmetry

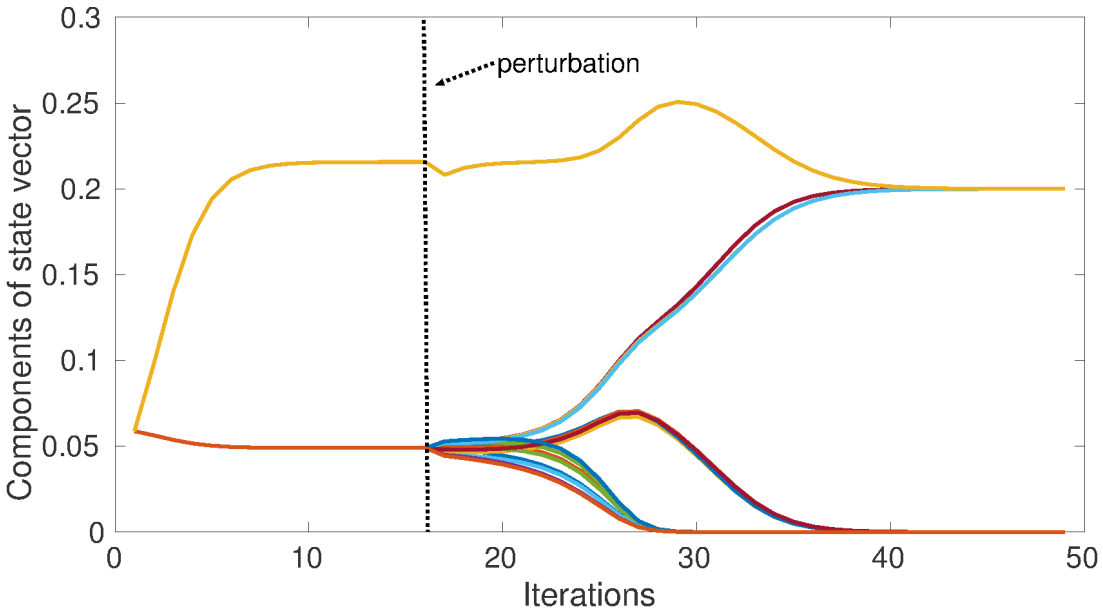

Figure 4.9: Evolution through time of the components of the state vector $\mathbf{x}(t)$ from the hypergraphs in figure 4.8 using the Baum-Eagon inequality. A perturbation can be seen at iteration 17 in order to escape a saddle point. After the perturbation the algorithm clearly makes a decision about what associations have to be chosen and what others have to be discarded.

isomorphism problem of figure 4.8. A perturbation is clearly visible at step 17. After the perturbation the symmetry is broken and the algorithm is able to "choose" some correct pairings, thus reaching the global optimum and finding the maximum clique. The perturbation is repeated for at most 4 times, then the algorithm is stopped independently from the fact that it has returned a clique or not.

Once the optimization is finished we have to understand if the returned state vector represents a correct isomorphism or not. In order to automatically check the results, the vector is thresholded against a small value close to 0 (the value 1^{-5} has been used), selecting in this way only the couples that represent the pure strategies still played by the population, i.e. the couples that should represent the isomorphism. Once retrieved these couples two things are taken into consideration:

- verify if the returned couples represent a clique in the association hypergraph;
- in case they are a clique, verify if the clique number is equal to the number of connected nodes in H_2.

In fact, from theorem 1, there is the guarantee of finding an isomorphism only if

the returned couples represent a maximum clique. Since H_2 has at most the same order of H_1, there cannot be an isomorphism concerning a number of nodes that is larger than the size of the vertex set of the smallest between the two graphs.

On top of the correctness of the results, also the time needed to run the algorithm is taken into consideration, since if the running time is too long, the framework is of no use.

In the next chapter the obtained results are shown both for experiments concerning the hypergraph isomorphism problem and for experiments concerning the sub-graph isomorphism problem.

Chapter 5

Experimental Results

In this section the results obtained by all the executed experiments are analysed, both in terms of correctness and running time. A result is considered correct only if a correct isomorphism is found, that is if a maximum clique is returned. If even just a node is missing, the experiment is considered failed. For what concerns time, the time needed to optimize the function is the main reference, since it changes according to the type of dynamic used. However a comparison between the performances in terms of running times spent for creating the association graph and finding the optimum with the best dynamic available will be considered too.

5.1 Results for the Complete Isomorphism Problem

5.1.1 Correctness Evaluation

In terms of correctness, the proposed framework, when used for solving the complete isomorphism problem, has returned impressive results: as we can see in figures 5.1 - 5.7, all the isomorphisms have been properly found, with the algorithm returning 100% of the nodes exactly coupled, for all the 10500 experiments, independently of the dynamic that has been used and independently of the cardinality k, the number of nodes n and the connectivity rate pc.

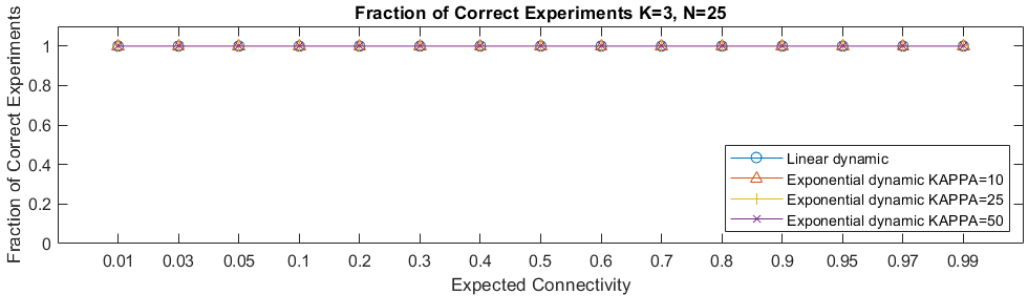

Figure 5.1: Correctness results for complete isomorphism on 3-graphs of 25 nodes, including all tested dynamics.

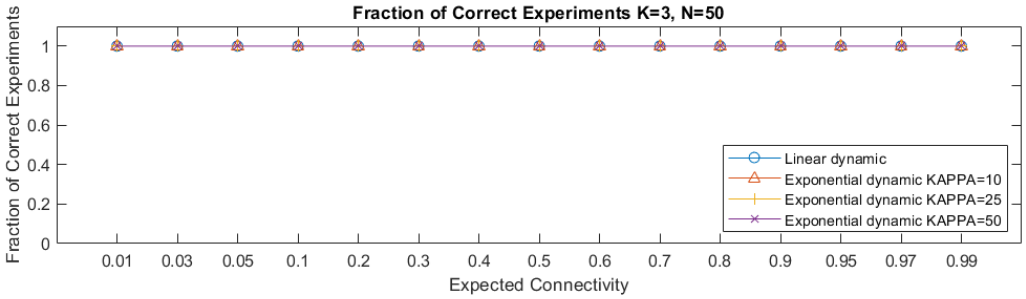

Figure 5.2: Correctness results for complete isomorphism on 3-graphs of 50 nodes, including all tested dynamics.

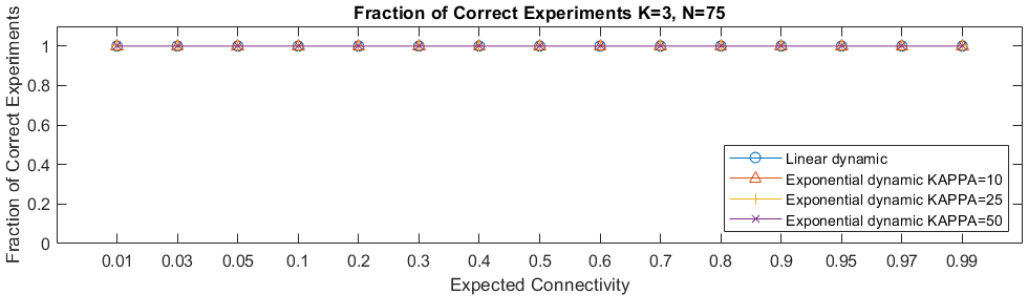

Figure 5.3: Correctness results for complete isomorphism on 3-graphs of 75 nodes, including all tested dynamics.

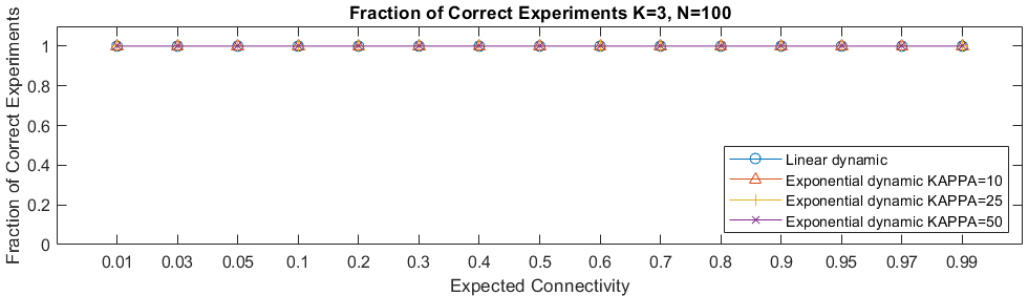

Figure 5.4: Correctness results for complete isomorphism on 3-graphs of 100 nodes, including all tested dynamics.

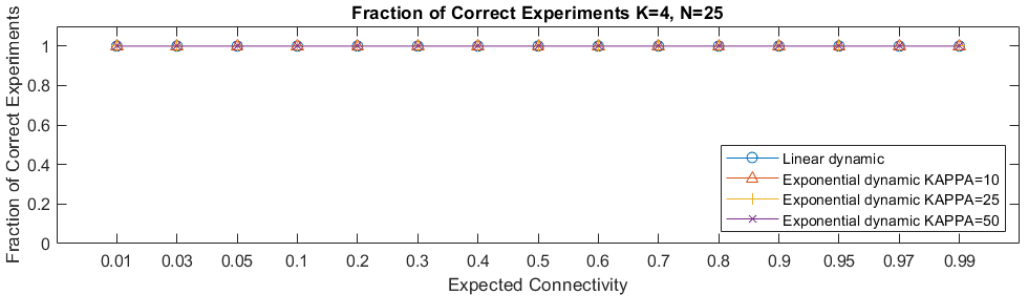

Figure 5.5: Correctness results for complete isomorphism on 4-graphs of 25 nodes, including all tested dynamics.

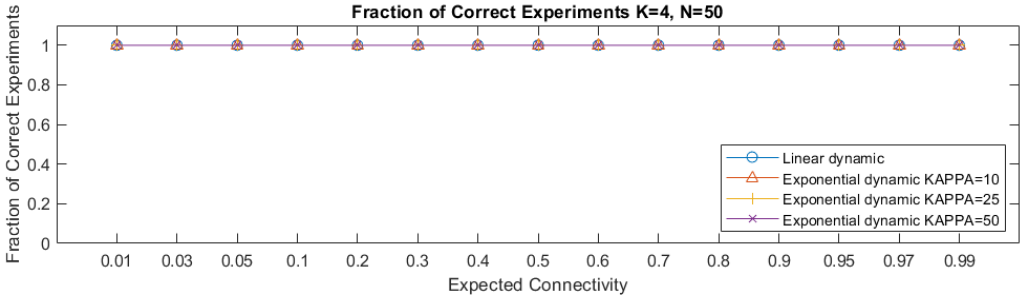

Figure 5.6: Correctness results for complete isomorphism on 4-graphs of 50 nodes, including all tested dynamics.

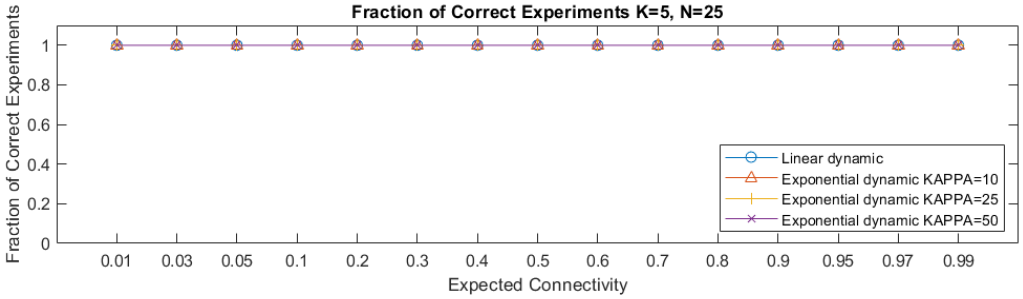

Figure 5.7: Correctness results for complete isomorphism on 5-graphs of 25 nodes, including all tested dynamics.

5.1.2 Running Time Evaluation

For what concern the execution time, we have to distinguish between the time needed to build the association graph and the time needed to find the clique. In the first case, the running time is independent of the dynamic used, since the association hypergraph is build only once from the structures being matched, and then the dynamics are applied on it. In the second case the running time strongly depends on the dynamic used, and one of the goals of this analysis is to understand

if there is, and in case, which is the best dynamic.

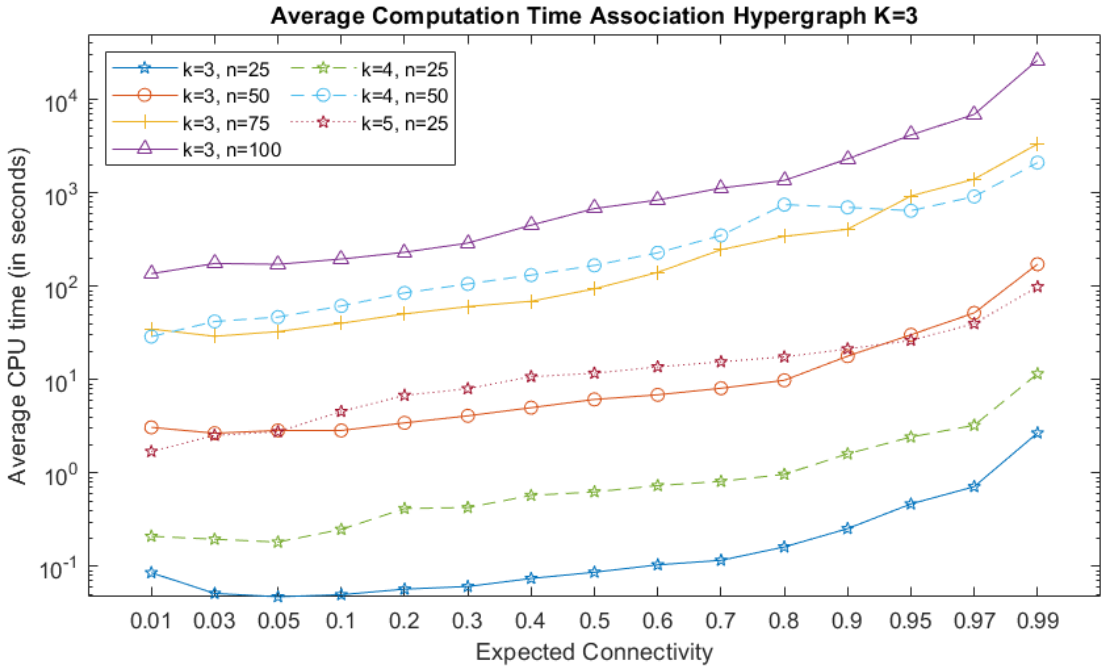

Figure 5.8: Running time needed for creating the association graph in the complete isomorphism problem. Times are in logarithmic scale.

Figure 5.8 shows the running time for creating the association graph in all the different experiments performed. The type of line identifies the cardinality of the hypergraphs under examination: straight lines represent 3-graphs, slashed lines represent 4-graphs while the dotted line represents 5-graphs. The marker identifies the order of the graphs: stars $\star$ represent $n = 3$, circles $\bigcirc$ represent $n = 50$, the plus $+$ represents $n = 75$ while the triangle $\triangle$ represents $n = 100$. Looking at the figure we can immediately see that the time needed to elaborate very sparse k-graphs is around 2 orders of magnitude less than the time needed to elaborate very dense graphs characterized by the same order and cardinality, this independently of the parameters of the structures to be matched. This is not due to the number of hyperarcs that needs to be checked nor to the number of nodes of the resulting association graph. In fact as we can see in figure 5.9, once fixed the parameters k and n, the mean number of nodes and of edges is symmetric with respect to the mean connectivity $p = 0.5$. For this reason neither the size nor the order of the hypergraphs can affect the running time as much as it is shown in

figure 5.8. Therefore the running time behaviour with respect to the connectivity

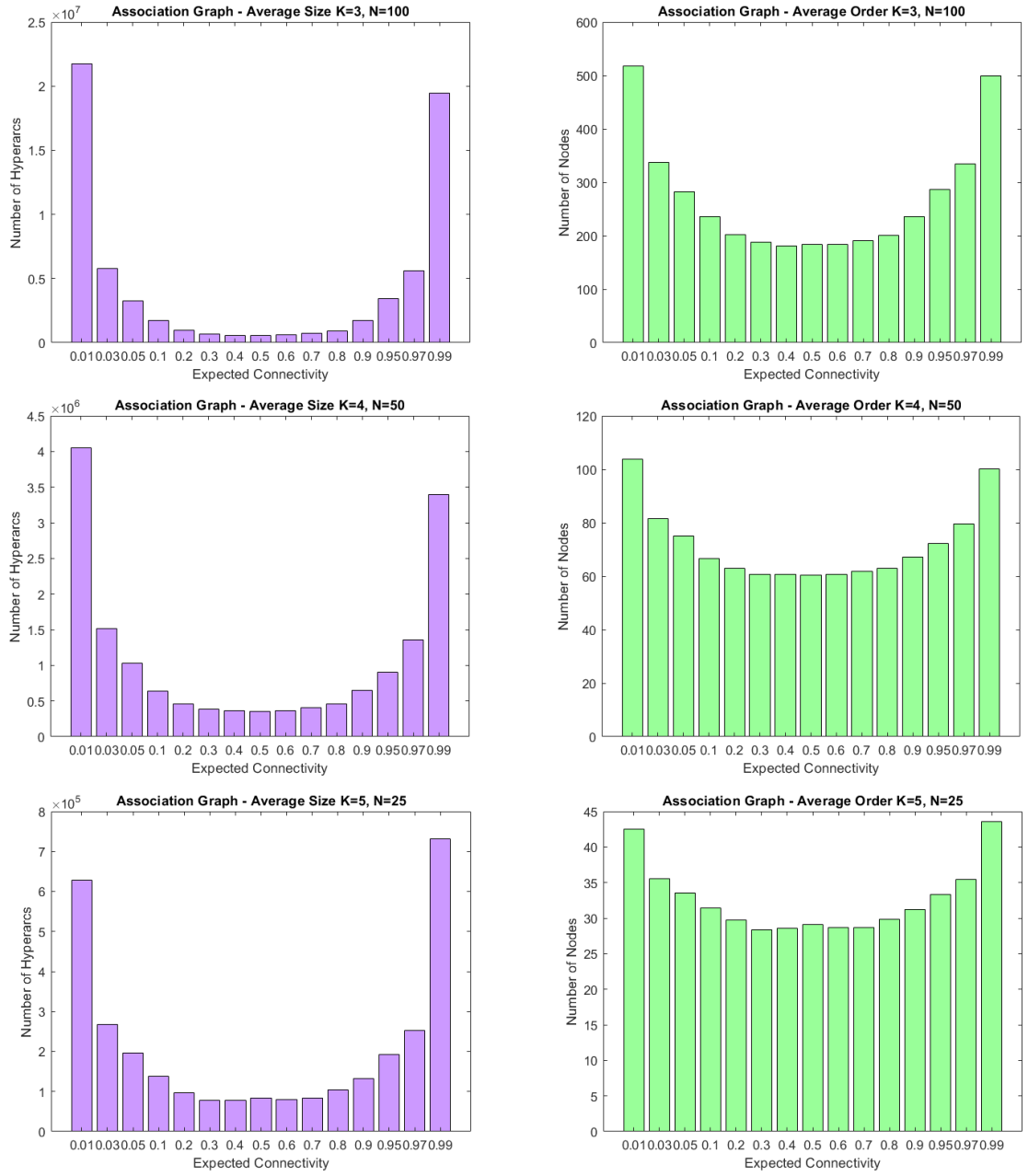

Figure 5.9: Average size and order of association hypergraphs build from graphs of different orders and cardinalities.

rate can be explained only with the size of the structures to be matched. In fact, the denser a graph the more arcs it has. When creating the association hypergraph the algorithm has to look for all the possible edges in the vertex sets of H_1 and H_2, the bigger they are, the more time the research takes. If we look only at the graphs with the same cardinality, that is the lines of the same type, unsurprisingly

we can see that increasing the number of nodes slow down the execution time. For example, if we consider 3-graphs, specified in figure 5.8 by the full line, we can see that doubling the number of nodes in the original graphs requires an execution time that is two order of magnitude bigger. The same behaviour can be observed by looking at 4-graph (slashed lines). On the other hand, if we consider original graphs of the same order, plotted with the same marker, clearly a higher cardinality implies longer running times. For example if we take into consideration graphs with 50 nodes, the ones represented with a circle ◯, there is more than an order of magnitude between 3-graphs and 4-graphs.

The mean CPU time needed to run the optimization is shown in figure 5.10. As we can see all the dynamics involved have the same behaviour, being extremely slow when dealing with very sparse or very dense graphs. Such a behaviour is not unexpected after having seeing figure 5.9, from which it is clear that the association hypergraphs in the extreme connectivity rates are way bigger and denser, thus giving birth to more complex polynomials to be optimized. With no surprise, we see that dealing with smaller hypergraphs results in shorter execution times, nevertheless the behaviour of the curves according to all the other parameters is exactly the same in all the plots. An observation has to be done in order to understand which is the faster dynamic. We can clearly see that the exponential dynamic always outperforms the standard Baum-Eagon inequality, thus resulting to be really attractive, from a computational point of view. However it introduces a parameter to be tuned, the κ in equation 3.11. Three different values have been tested in $\{10, 25, 50, 100\}$. $\kappa = 100$ is resulted to be too high, since the point in the search space oscillates too much, and the algorithm was not able to converge, thus giving rise to plots like the one in figure 5.11. The other values of κ have been used on all the experiments. In figure 5.10 we can see that, as expected, in nearly all the experiments the linear dynamic is the slowest, then there is the exponential dynamic with $\kappa = 10$, then $\kappa = 25$, and finally the fastest is $\kappa = 50$. However there are two combinations of parameters for which the situation is different: for the combination $k = 3$ and $n = 25$ the exponential dynamic with $\kappa = 50$ is nearly as slow as the linear dynamic, while for the combination $k = 5$ and $n = 25$ the linear

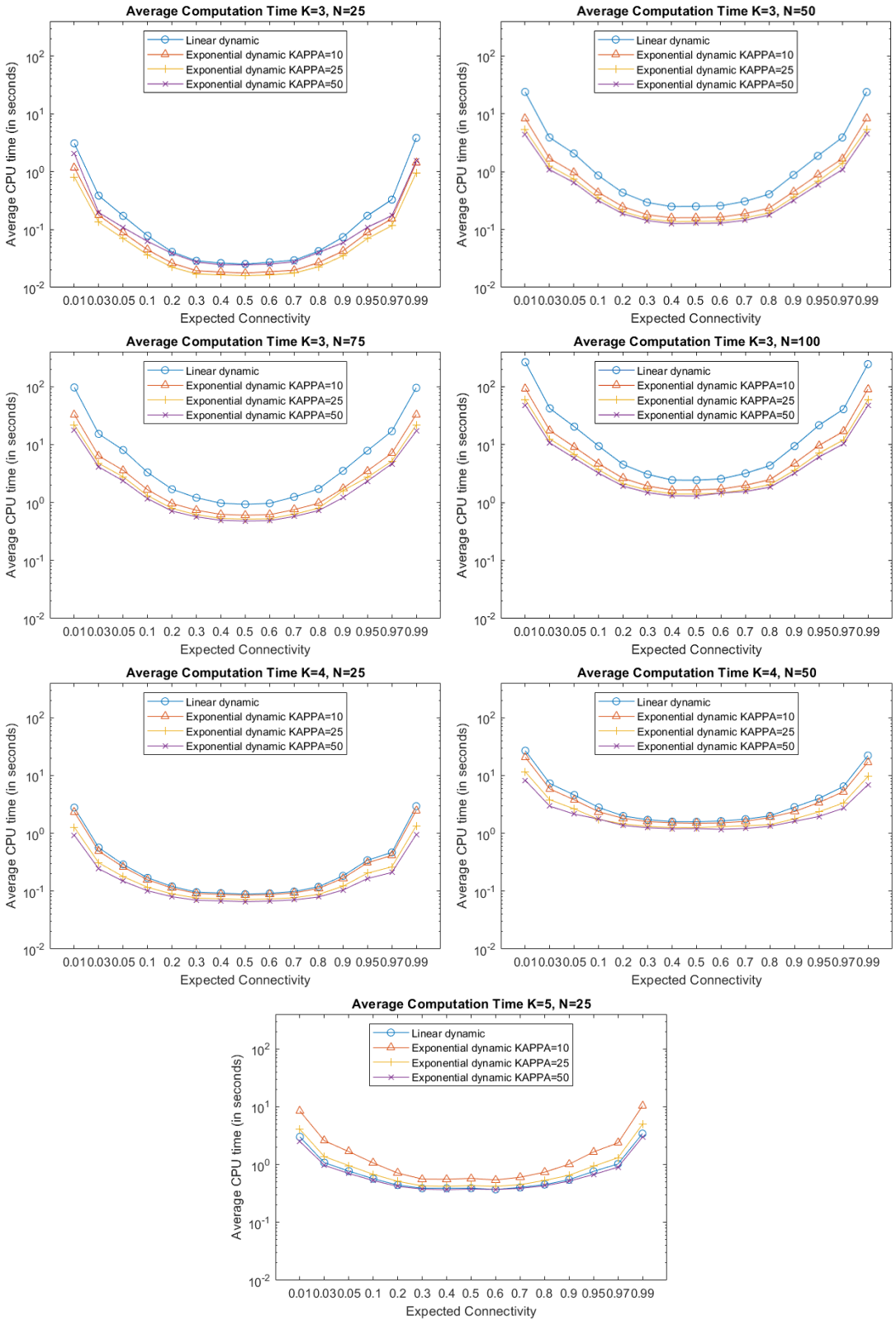

Figure 5.10: Mean CPU time needed to run the optimization algorithm for finding isomorphism on hypergraphs with different cardinality and order, using both linear and exponential dynamics. Y-axes are in logarithmic scale.

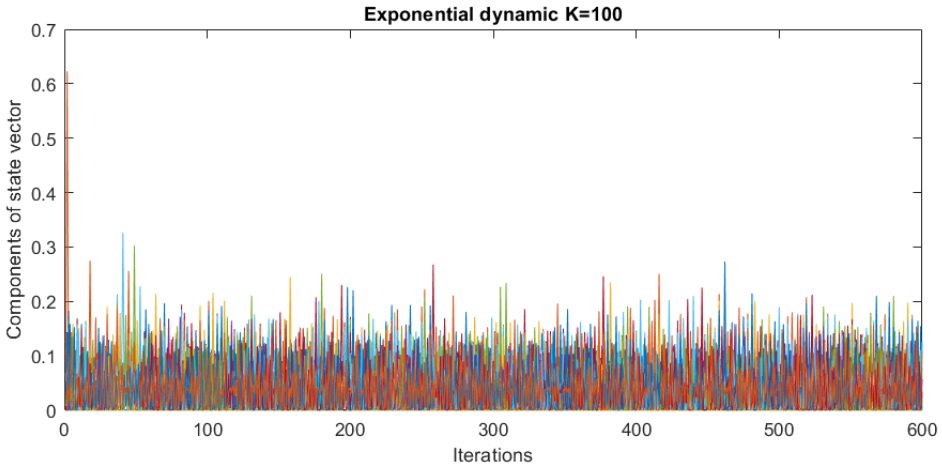

Figure 5.11: Evolution of the vector state through time using exponential dynamic with $\kappa = 100$, for an association 3-graph made from graphs of order 25 and connectivity 0.01.

dynamic is slower only than the exponential dynamic with $\kappa = 50$.

An example of the behaviour of the optimization process for 3-graphs of order 25 is shown in figure 5.12, and it clearly explains the low speed of the dynamic that is the fastest in all the other parameters' configurations. As we can see the outline of the different dynamics is the same, with a perturbation to break a symmetry situation, however the number of iterations needed to converge is really different: the linear Baum-Eagon dynamic needs about 700 iterations, the exponential dynamic with $\kappa = 10$ needs less than 200 iterations, the exponential dynamic with $\kappa = 25$ takes only 80 iterations to converge, while due to the great oscillations the exponential dynamic with $\kappa = 50$ takes more than 600 oscillations, thus taking a lot of time to run. The reason why this happens only with this configuration of parameters and not on the others lays in the fact that the structures to be matched are very small, the smallest under examination, and therefore the related associations hypergraphs are very small too. Probably the value of $\kappa = 50$ is too high for such a small structure, while it works well for bigger systems. This demonstrates that the value of κ is strictly dependent on the problem under examination, and a single value cannot be used on all the experiments. Thus, in a real setting with a specific problem, the parameters of that problem have to be analysed and evaluated, and the vaue of κ has to be tuned according to those specific values.

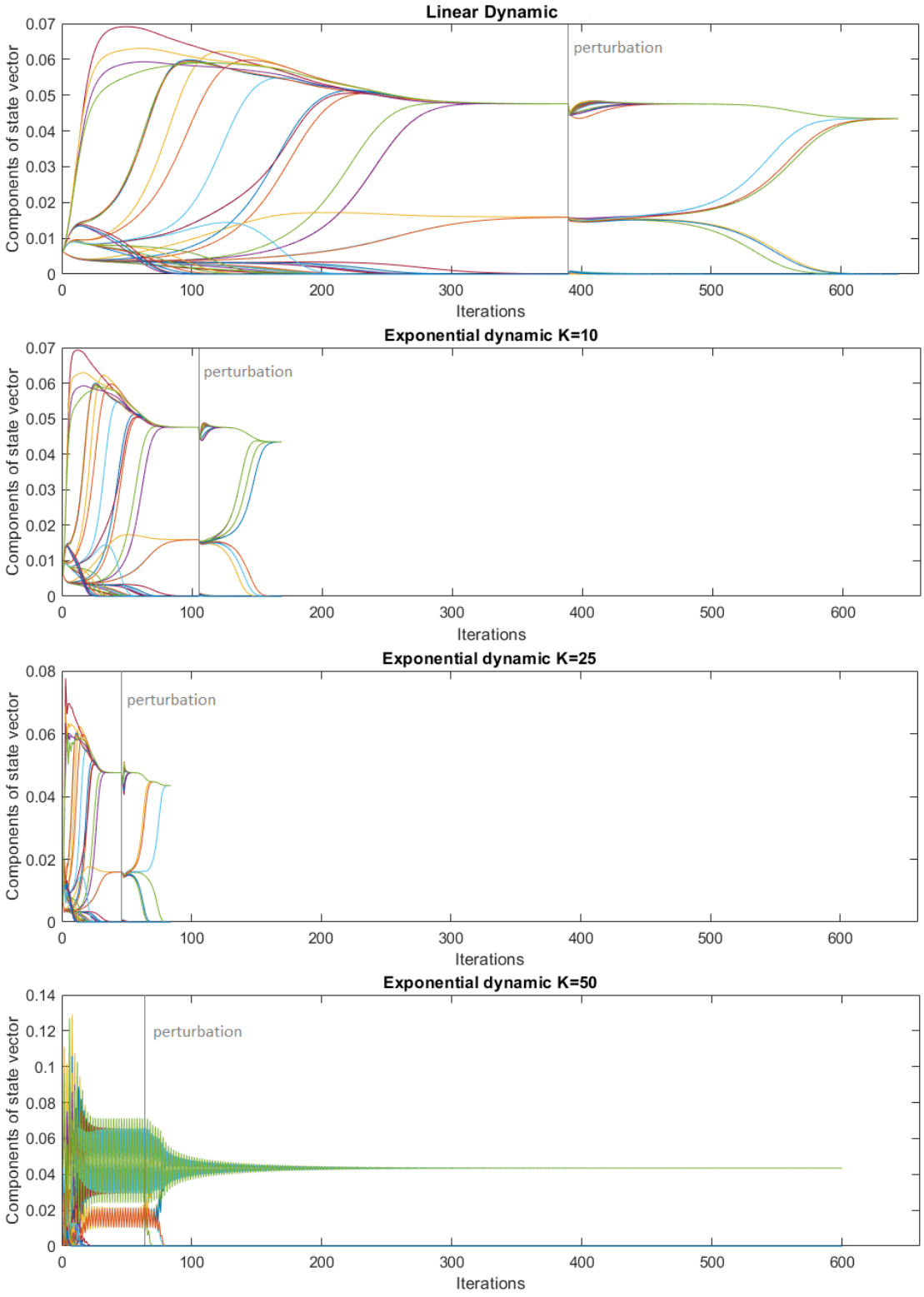

Figure 5.12: Different behaviour of the dynamical systems under examination, with various dynamics for the same 3-graph made from graphs of order 25 and connectivity 0.01.

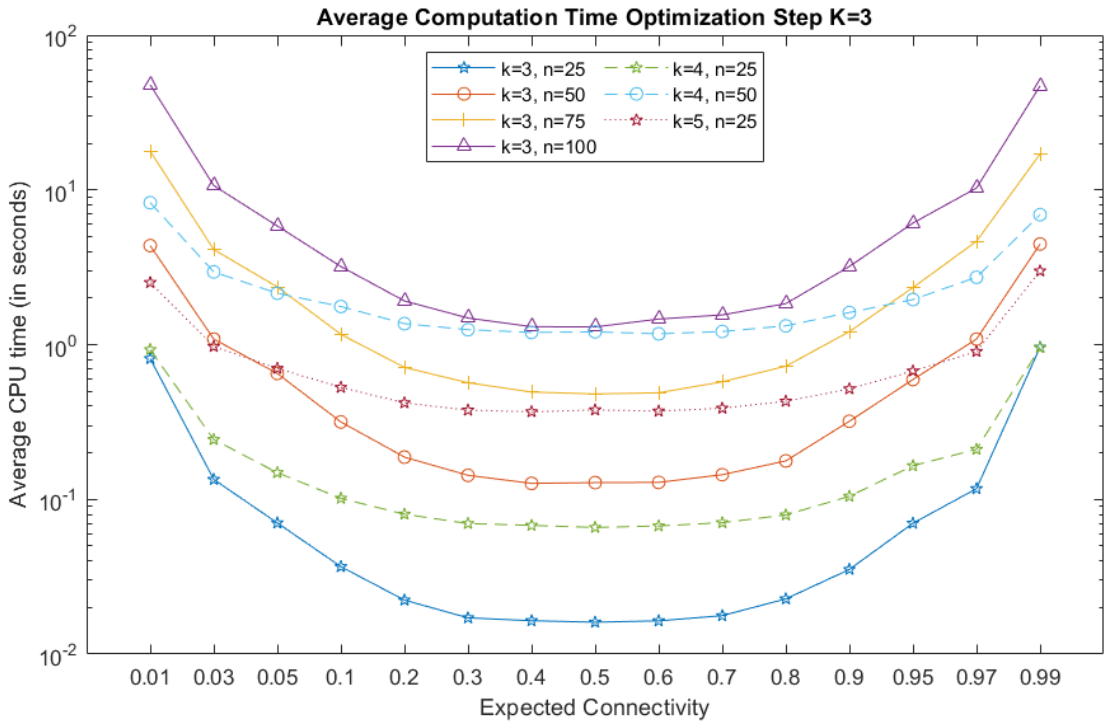

Figure 5.13: Comparison of the mean CPU time for the best dynamics.

Finally, figure 5.13 shows a comparison of all the execution time for the optimization step, considering only the fastest dynamic for each combination of parameters. With no surprise, the same behaviour of the execution times for building the association hypergraph can be observed: keeping the cardinality k fixed, execution takes longer for hypergraphs of bigger order; vice versa, keeping the order n fixed, the higher the cardinality the longer the running time.

5.2 Results for the Sub-Graph Isomorphism Problem

5.2.1 Correctness Evaluation

As stated in chapter 2 the sub-graph isomorphism problem is more complex than the complete isomorphism one, and this distinction is reflected in the results obtained. In fact in terms of correctness the algorithm does not return as good results as the ones obtained in section 5.1.1. As we can see in figures 5.14, 5.15 and

5.16 not one combination of parameters returned 100% correct experiments. In

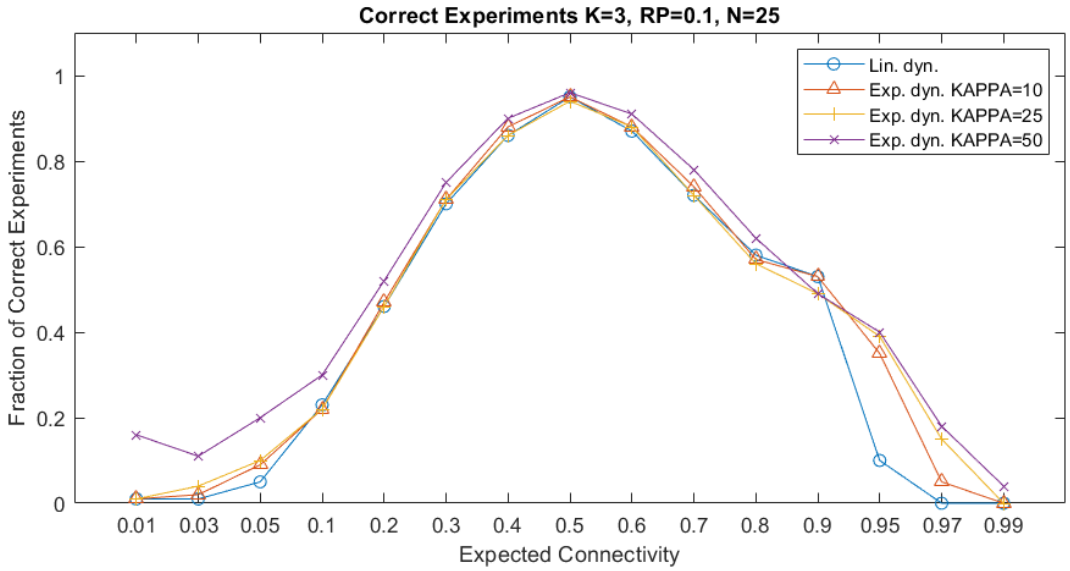

Figure 5.14: Correctness results for sub-graph isomorphism on 3-graphs of 25 and 22 nodes (10% of removed nodes), including all tested dynamics.

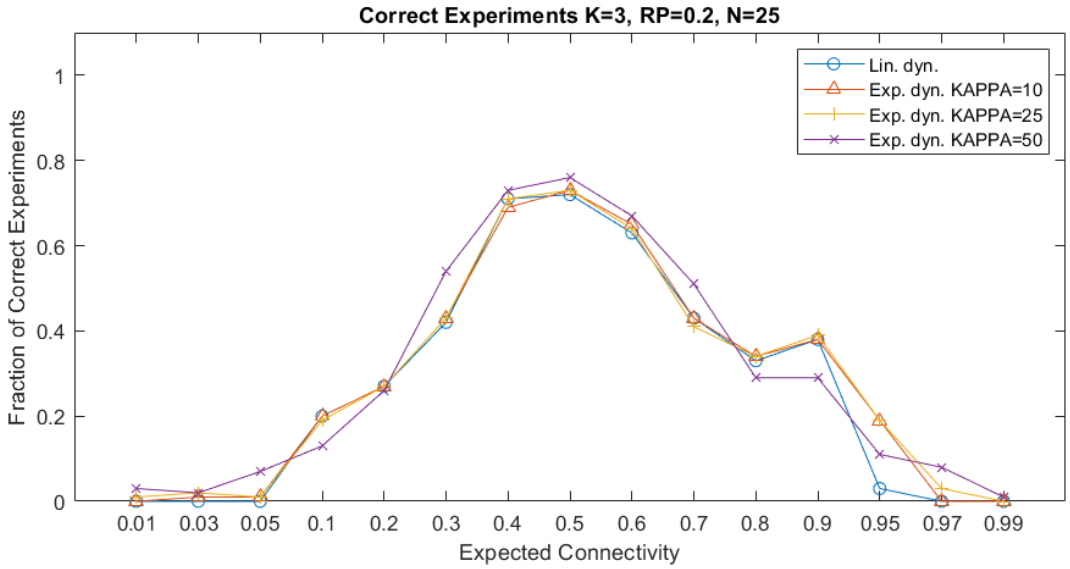

Figure 5.15: Correctness results for sub-graph isomorphism on 3-graphs of 25 and 20 nodes (20% of removed nodes), including all tested dynamics.

fact there are differences in the percentage of correct results according both to the connectivity rate and to the percentage of removed nodes. In particular the best performances are returned with connectivity rate $p = 0.5$, as the graphs' density becomes more extreme, the results get worse. With a similar analysis we can see

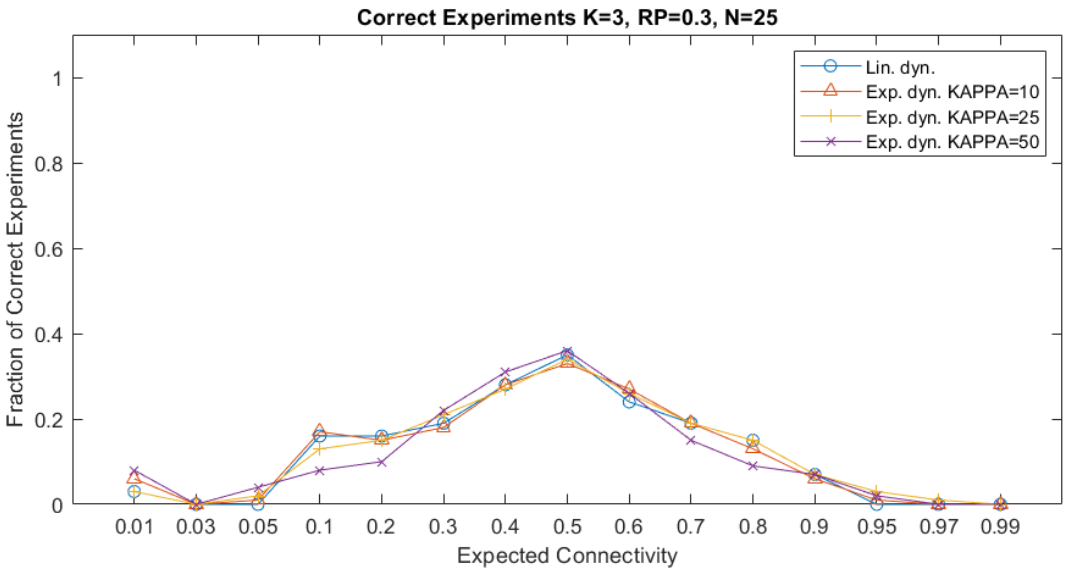

Figure 5.16: Correctness results for sub-graph isomorphism on 3-graphs of 25 and 17 nodes (30% of removed nodes), including all tested dynamics.

that the largest the sub-graph, the higher is the probability of obtaining a correct result.

In fact if we fix a dynamic and look at the behaviour of the different types of graphs (figures 5.17) we can see that independently on the dynamic used and on the connectivity rate, the experiments where H_2 has only 10% of nodes less than H_1, return always the best results, and as the order of H_2 decreases the percentage of correctness decreases too. This fact could be intuitive if we think that the larger the sub-graph the more similar it is to the main graph, but it is counter-intuitive if we think that we have to find a smaller isomorphism. Of all the 4500 experiments tested, only a small part has returned a correct result. Depending on the dynamic used the number of total valid results are between 1200 and 1400, as shown in figure 5.18. An interesting thing to note is that there is not a single dynamic that outperforms the others. In fact, while the exponential dyanmic with $\kappa = 10$ works well when H_2 is 30% smaller than H_1, it is one of the dynamics that perform worse when H_2 is just 10% smaller than H_1. Moreover, even if we fix the number of nodes of H_2, still there is not a dynamic that can be preferred in all the cases. For example if we look at figure 5.15, the exponential dynamic with $\kappa = 50$, is the one

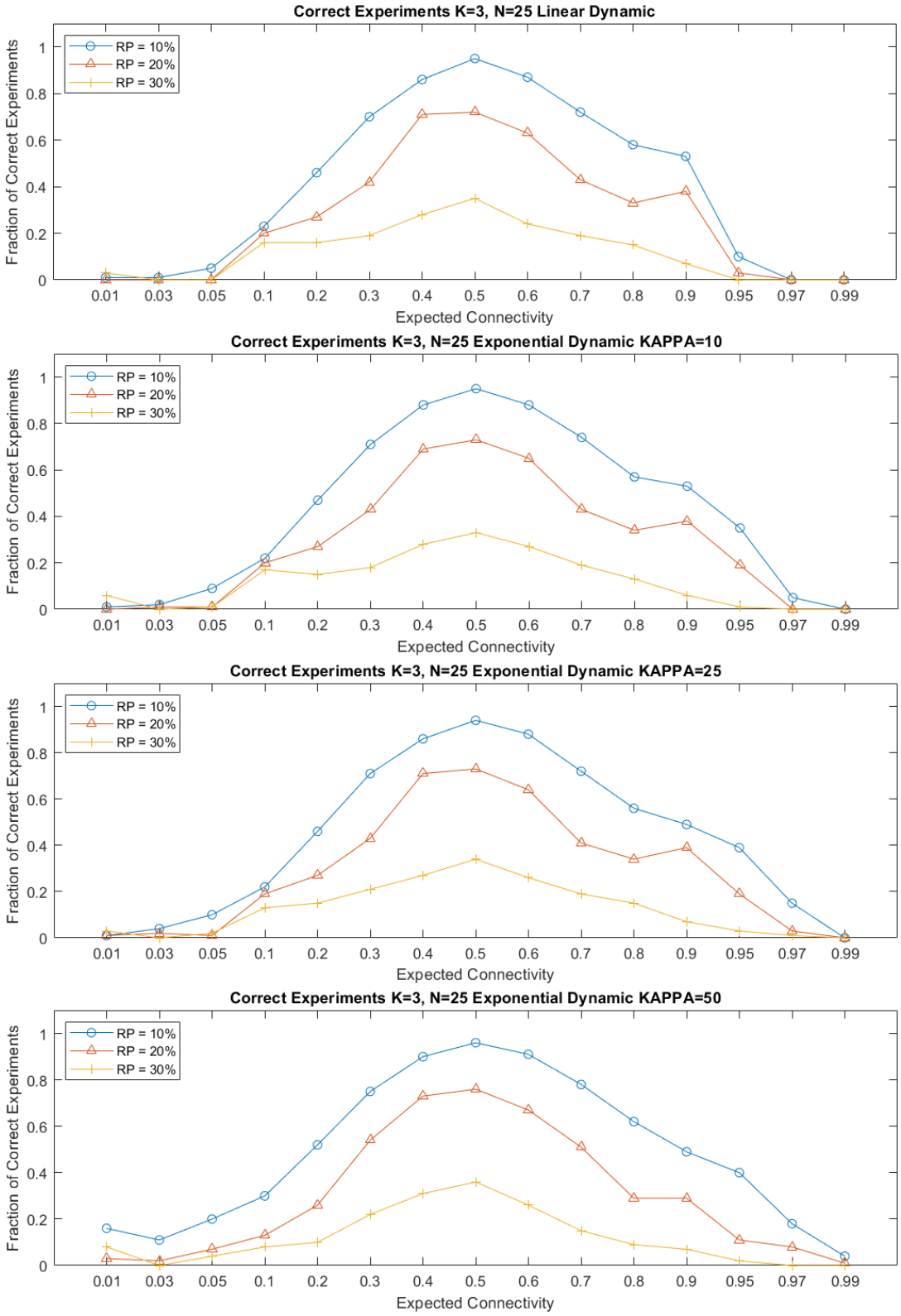

Figure 5.17: Percentage of correctn results for sub-graph isomorphism with different dynamics: linear (first from top), exponential with $\kappa = 10$ (second), exponential with $\kappa = 25$ (third), exponential with $\kappa = 50$ (bottom).

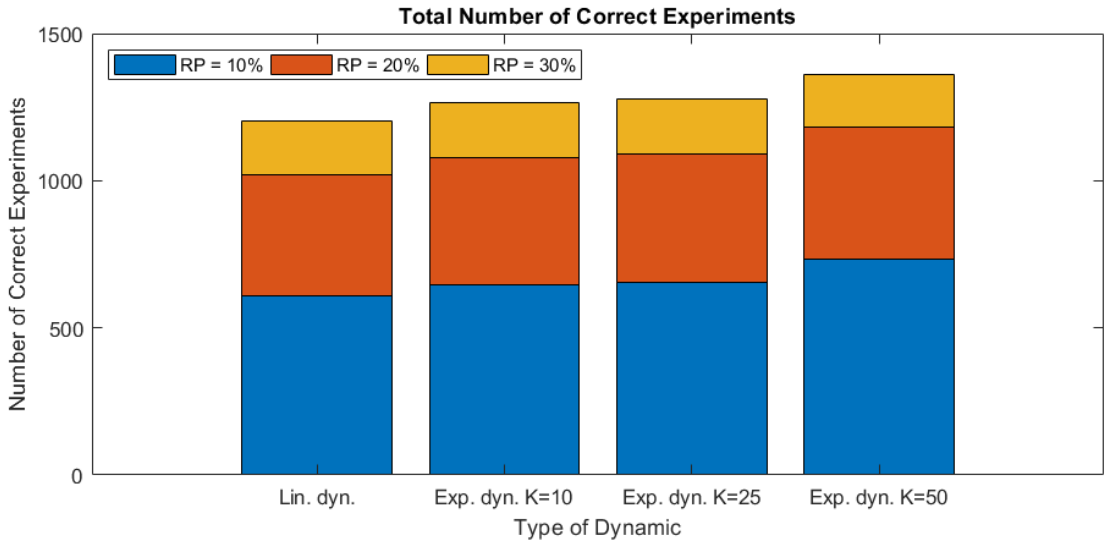

Figure 5.18: Total number of correct experiments according to the specific dynamic used.

that returns the best results when the connectivity rate is 0.1, while it is the one that returns the worst results when the connectivity rate is 0.3. For this reason we have no way to choose a specific dynamic as the best working one, not even for a specific combination of parameters.

Another interesting thing to investigate is if, giving a specific set of parameters, the correct experiments returned by the worst performing dynamic are correctly returned by the other better performing dynamics too. In order to do so we have to fix all the parameters, and check the correct results returned by the different dynamics.

Let's take for example the following parameters:

- cardinality $k = 3$,
- order $n = 25$,
- connectivity rate $p = 0.2$
- percentage of removed nodes $rp = 0.1$.

The best performing dynamic is the exponential with $\kappa = 50$, that returned 52 correct sub-isomorphisms, then there is the exponential with $\kappa = 10$, that returned 47 correct sub-isomorphisms, and finally the linear dynamic and the exponential

with $\kappa = 25$, returned both 46 correct sub-isomorphisms. However if we count all the experiments that were successfully solved by at least one dynamic we find 60 different experiments. This means that not all the sub-isomorphisms correctly found by the worst performing dynamic are found by the better performing dynamics too. In figure 5.19 all the 60 experiments that returned a correct isomorphism are

Correct Experiments. Parameters: k=3, n=25, rp=0.1, p=0.2

	Experiment	2	3	5	7	8	10	11	12	17	19	20	21	23	28	29	31	35	36	37	38
Dynamic	Linear			X		X	X	X	X	X	X	X	X		X	X		X		X	X
	Exp K=10			X		X	X	X	X	X	X	X	X		X	X		X		X	X
	Exp K=25			X		X	X	X	X	X	X	X	X		X	X		X		X	X
	Exp K=50	X	X	X	X	X	X	X	X		X	X		X	X	X	X	X	X	X	X

	Experiment	40	41	43	44	45	48	49	50	53	54	56	58	60	61	62	63	66	67	71	72
Dynamic	Linear	X	X	X	X	X	X	X	X		X		X	X	X		X	X	X	X	X
	Exp K=10	X	X	X	X	X	X	X	X			X	X	X	X		X	X	X	X	X
	Exp K=25	X	X	X	X	X	X	X	X			X	X	X	X		X	X	X	X	X
	Exp K=50	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X		X	X

	Experiment	74	76	79	80	81	82	83	84	85	86	87	88	89	90	92	93	94	95	99	100
Dynamic	Linear		X		X	X	X		X	X	X	X		X		X	X	X	X	X	X
	Exp K=10	X	X		X	X	X		X	X	X	X	X			X	X	X	X	X	X
	Exp K=25		X		X	X	X		X	X	X	X	X			X	X	X	X	X	X
	Exp K=50			X	X	X	X	X	X	X	X	X	X	X	X	X			X	X	X

Figure 5.19: List of experiments that returned a correct isomorphism in at least one of the tested dynamics.

listed, together with the different dynamics. An **X** is inserted when the dynamic was successful on that specific experiment. As we can see most of the experiments were correctly solved by all the dynamics, with no surprise some other experiments (the yellow ones) were correctly solved only by the best performing dynamic. However there are some experiments that were solved by all the dynamics except the exponential with $\kappa = 50$ (the blue ones), and some others (the green ones) returned the maximum isomorphism only on some worse performing dynamics. These two sets of experiments are something that we could not have expected, and it makes it impossible to identify a single dynamic to use with all the experiments. A similar behaviour can be observed with every other combination of parameters (see figure 5.20 for another example with different parameters, where in particular there are a lot of experiments (in blue) that returned a correct answer only with the exponential dynamic with $\kappa = 50$, that is the worst performing one). The evolution

Correct Experiments. Parameters: k=3, n=25, rp=0.2, p=0.8

	Experiment	2	3	5	6	7	9	11	17	21	22	23	24	25	26	27
Dynamic	Linear	X		X	X		X	X	X		X	X	X	X	X	
	Exp K=10	X	X	X	X		X	X		X	X	X	X	X	X	
	Exp K=25	X	X	X	X		X	X		X	X	X	X	X	X	
	Exp K=50			X	X	X	X	X				X			X	X

	Experiment	29	34	37	38	42	43	46	47	48	53	58	63	69	70	75
Dynamic	Linear	X	X	X	X		X		X	X	X			X	X	X
	Exp K=10	X	X	X	X		X		X	X	X			X	X	X
	Exp K=25	X	X	X	X		X	X	X	X	X			X	X	X
	Exp K=50	X	X	X	X	X	X	X		X	X	X	X	X		

	Experiment	76	77	82	83	85	86	87	91	92	93	94	95	97
Dynamic	Linear	X	X	X	X	X		X	X		X	X	X	X
	Exp K=10	X	X	X	X	X		X	X		X	X	X	X
	Exp K=25		X	X	X	X		X	X		X	X	X	X
	Exp K=50		X	X	X		X	X		X	X	X	X	

Figure 5.20: Another example of a list of experiments, with different parameters, that returned a correct isomorphism in at least one of the tested dynamics.

through time of a couple of experiments with this strange performance has been examined in order to better understand this phenomenon. Figure 5.21 shows the vector state at different time steps for experiment number 17 of the parameters analysed in figure 5.19. While the linear dynamic and the exponential dynamics with $\kappa = 10$ and $\kappa = 25$ are able to converge, the exponential with $\kappa = 50$, oscillates to much and is not able to stop in a point. This behaviour is distinctive of all the experiments coloured in yellow. Note that in this case the oscillations of $\kappa = 50$ are analogous to the ones observed with the exponential dynamic with $\kappa = 100$, and they are the reason why that dynamic has not be taken into consideration for the experiments. However while with $\kappa = 100$, nearly all the experiments failed, the exponential with $\kappa = 50$ is often the best dynamic in terms of correctness.

On the opposite side of the same coin there are the results returned in experiments like the ones coloured in yellow. With these experiments only, the exponential dynamic with $\kappa = 50$ returns the correct result, while all the other dynamics don't. In figure 5.22 we can see the different dynamics on experiment number 23. Even though the exponential with $\kappa = 50$ doesn't stabilize, hence not finding an ESS, the oscillations are between very close equilibrium points, and when the algorithm

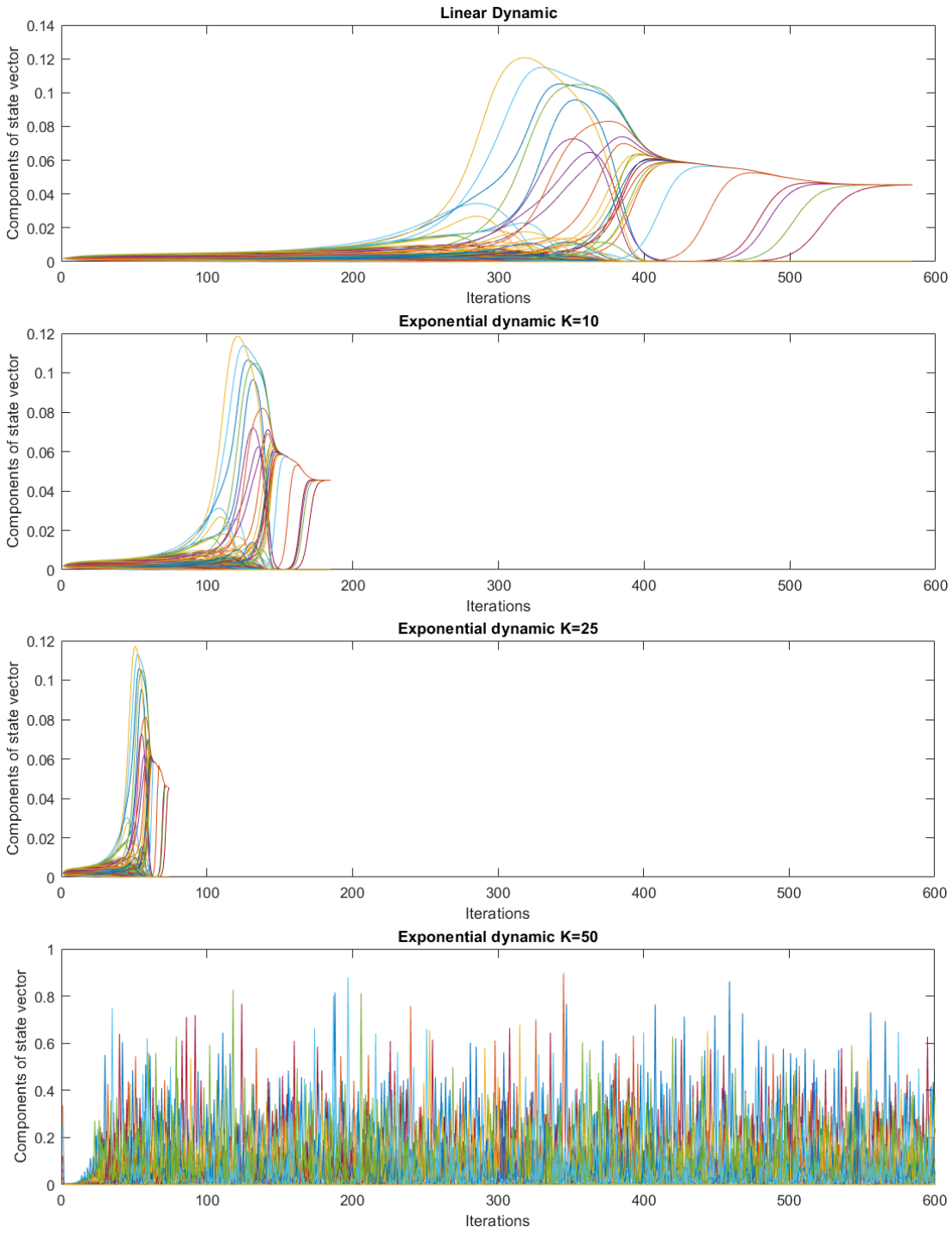

Figure 5.21: Evolution through time of the vector state for experiment number 17 of figure 5.19, with all the different dynamics tested.

reaches the maximum number of iterations, the current equilibrium point represent indeed a maximum clique, thus a correct isomorphism. On the other hand, all the other dynamics are not able to converge to a clique even after three perturbations, thus not being capable of returning a correct result. Finding a reason to the different

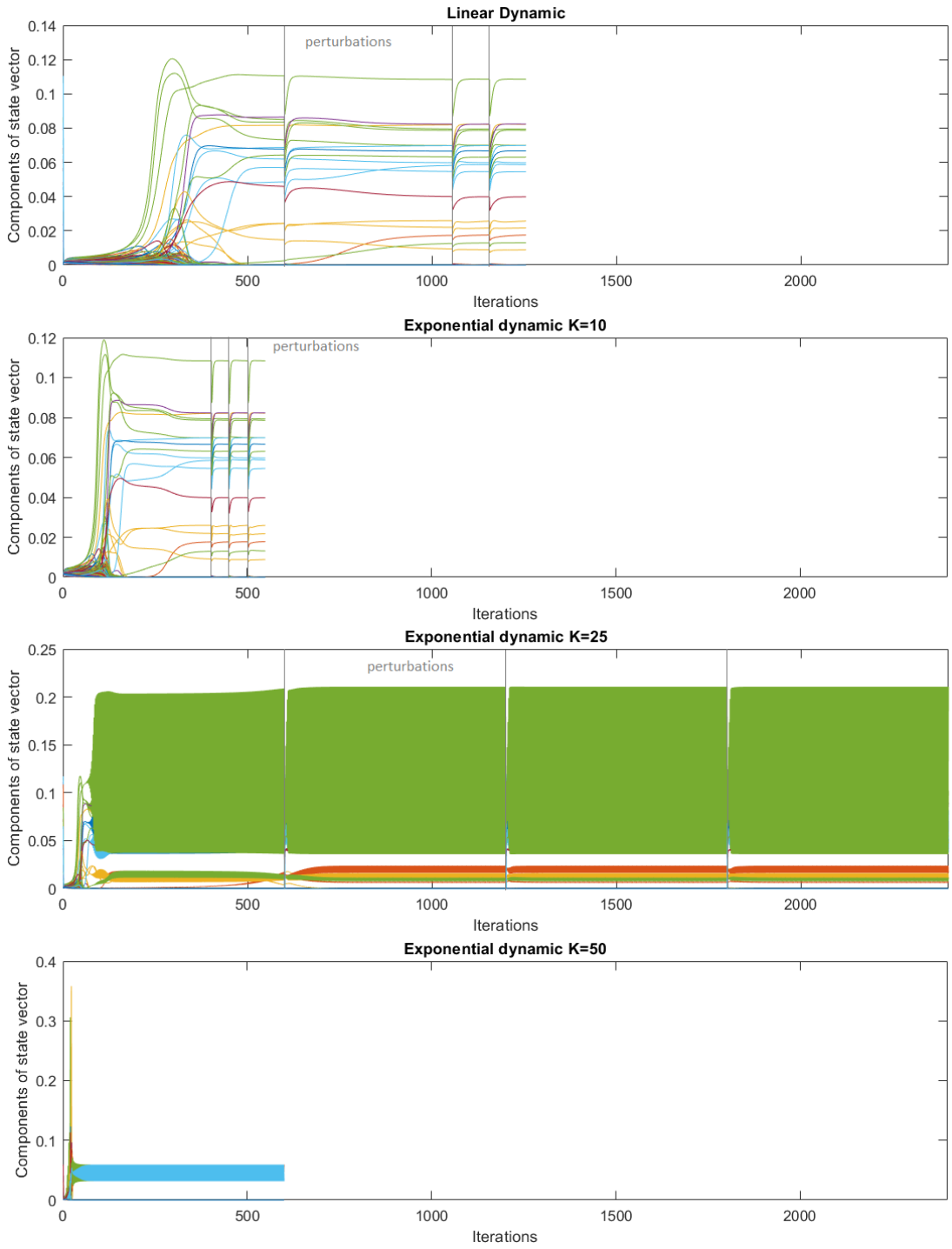

Figure 5.22: Evolution through time of the vector state for experiment number 23 of figure 5.19, with all the different dynamics tested.

performances of the dynamics in this case is not easy, however it seems there is no connection with the structure of the graph to be optimized and the results with different dynamics, since the size and order of the association hypergraphs both in the set of experiments that works well with the exponential with $\kappa = 50$ (the yellow

ones) and in the set of experiments that return correct results only with the other dynamics (the blue ones) are more or less the same. In fact the 3-graphs in the experiments belonging to the yellow set have a number of nodes that varies between 512 and 544, and a number of arcs between 12044317 and 14210692, while the order in the blue set is between 515 and 545, and the size is between 11623540 and 13780820, thus not highlighting any significant discrepancy between the structures. Finally figure 5.23 shows the evolution of the vector state for an experiment (number 1) with the same parameters set as the one of experiments in figure 5.19, that does not return the correct isomorphism with any dynamic. As we can see all the dynamics do not converge to a clique, even after multiple perturbations. This behaviour is the same observed by the non converging dynamics of experiment number 23 (figure 5.22), and also in this case there is nothing particular in the structure of the association graph (516 nodes, 12125688 arcs), thus again suggesting that the structure is not the (only) reason for the absence of convergence.

From this perspective it is interesting to have an overview of the type of results given by the experiments which did not returned the correct isomorphisms. Figure 5.24 gives an idea of the types of results provided by the experiments according to the different dynamics. As we have already stated around 30% of the returned results are correct. However we can see that the wrong results can be split in two categories:

- a very small set that contains all the experiments that returned a maximal clique, instead of a maximum one. In these cases the optimization process has not been able to reach the global optimum, falling in the basin of attraction of local solutions. The size of this set is really small if compared with the total number of experiments performed, involving only the 1.5% of the total tests;

- a very large set, containing around 70% of the total experiments, that includes all those trials that did not returned a clique at all, neither a maximal one, and returned results as the ones in 5.23.

A reason that might be behind this inability to converge to a clique is that

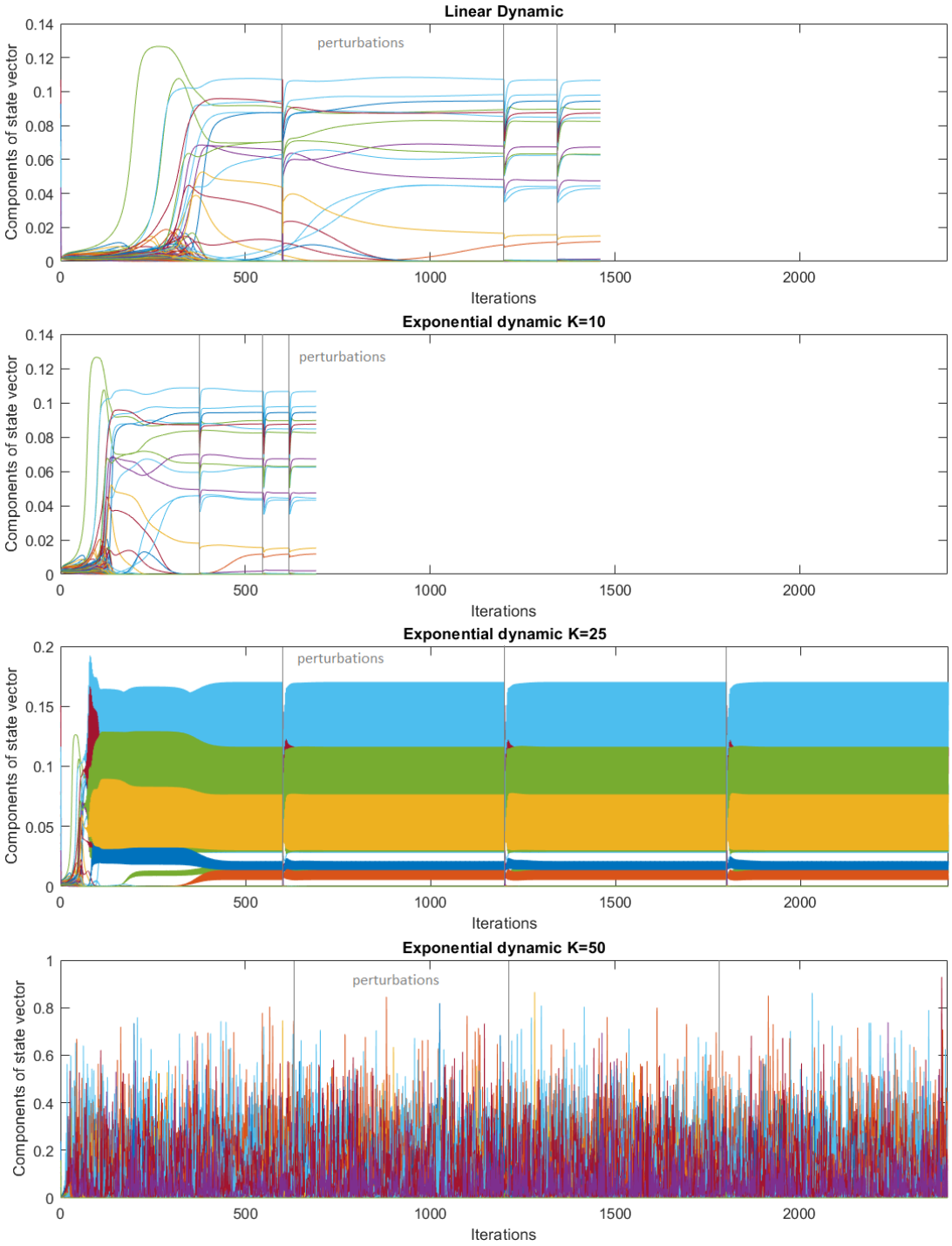

Figure 5.23: Evolution through time of the vector state for experiment number 1 of figure 5.19, with all the different dynamics tested.

the solution of problem 3.8 does not have any theoretical guarantee of finding a clique. In fact, Frankl and Rödl in [10] showed that maximizing the Lagrangian of an hypergraph does not necessary produce a clique, but a *2-cover*, i.e. an hypergraph such that all the pairs of nodes are connected by at least one edge.

	Total n° of experiments	Correct results	Local optimum	Non corvenging to clique
Linear Dynamic	4500	1201	74	3225
Exp. Dynamic K = 10	4500	1264	79	3157
Exp. Dynamic K = 25	4500	1276	102	3122
Exp. Dynamic K = 50	4500	1360	13	3127
Total	18000	5101	268	12631

Figure 5.24: Recap of the type of results returned in all the experiments by the different dynamics.

While in 2-graphs the notion of 2-cover coincide with the notion of clique, this in not true for generic k-graphs with $k \geq 2$. This problem has been overcome in [30] where a continuous formulation of maximal cliques in k-graphs is given as a minimization of the Lagrangian of the complement of the hypergraph plus a parametrized regularization term, however this formulation has not been used in this thesis because of its complexity and the need for tuning an additional parameter. Moreover the results on complete isomorphism using equation 3.2 are perfect. Anyway, an application of the formulation proposed by Rota Bulò and Pelillo to the sub-graph isomorphism problem is certainly a possible direction that can be examined in future works.

After all the considerations done, we can not chose a single definitive dynamic to be used on all the experiments, even on experiments with the same parameters. In most of the combination of parameters, the exponential dynamic with $\kappa = 50$ is the one that returns the biggest number of correct isomorphisms. However some isomorphisms that are not found by this dynamic are found by the others. For this reason, in solving real problems, if enough time is available, the best option would be to use more than one dynamic and then select the correct results on all of them. Figures 5.25 and 5.26 shows the experiments that have returned a correct isomorphism in at least one dynamic. As we can see in the first image, the outline of the curves in all the different fractions of removed nodes is the same: graphs with a medium connection rate achieve better results, while in the extreme connectivity rates the results are really bad. Moreover, even considering all the different dynamics, the larger the sub-graph, the higher the number of correct

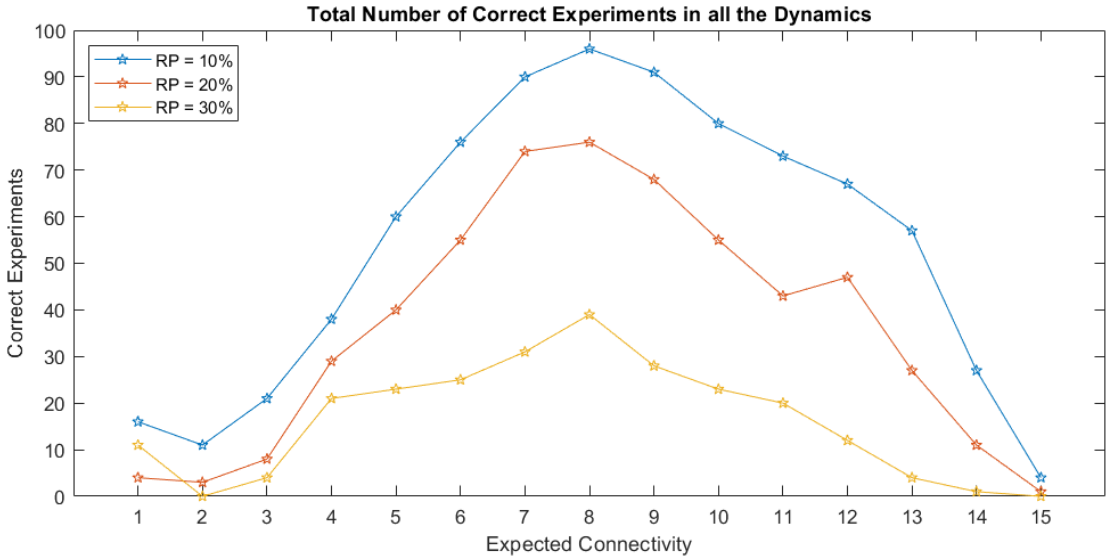

Figure 5.25: Number of correct experiments in at least one dynamic divided by number of removed nodes in H_2.

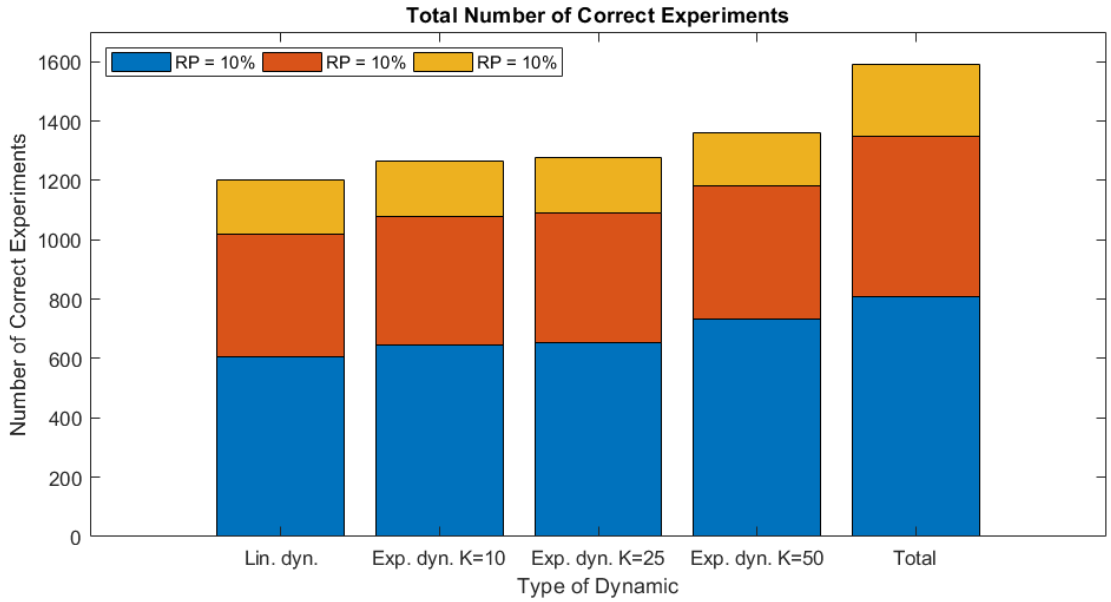

Figure 5.26: Total number of correct experiments with each different dynamic, and with all the dynamics.

results. However there is a detail that we have to take into consideration, that is the fact that the shapes of the curves are not exactly symmetric from the mean density $p = 0.5$. In fact the right side of the plot returns slightly better performances in terms of correctness with respect to the left side, even though the success rate is decreasing while the connectivity rate become more extreme. This event can

maybe be related to the fact that the association graphs created from dense graphs are in general larger both in terms of size and order (see figure 5.28). The second image instead shows the number of total correct results. As we could have expected considering all the dynamics the number of isomorphisms found is larger than with any single dynamic. However note that at most 1600 experiments results correct; if we consider that the number of tests performed for the sub-graph problem is 4500, we have that only one third of the optimizations process was successful.

5.2.2 Running Time Evaluation

For what concerns the execution time, we have to distinguish also in this case between the time needed to build the association graph and the time needed to find the clique. Exactly as in the complete isomorphism problem, in the first case the running time is independent of the dynamic used, while in the second case the running time strongly depends not only on the dynamic used, but, in general, also on the fact that the experiment returned a correct result or not. If an experiment converges, the step needed are usually bounded to a maximum of 600, while, if it doesn't converges it may take up to 2400 iterations considering at most three perturbations.

Figure 5.27 shows the running time for creating the association graph in all the different experiments performed. Not dissimilarly from the complete isomorphism problem we can see that the time needed depends on the size and order of the 3-graphs to be matched. The denser the original structures the more time the research for the hyperedges takes, up to two orders of magnitude from expected connectivity $p = 0.01$ to expected connectivity $p = 0.99$, independently of the number of nodes removed in H_2. Moreover the smaller the order of H_2, the smaller the number of possible associations between nodes, thus shrinking the total number of arcs to be checked when building the association hypergraph and consequently the running time. In contrast to what happened in the complete isomorphism problem, as we have seen in figure 4.7 the denser the graphs to be matched, the smaller the overlap between the grade of the nodes, thus resulting in less pruning, that means more pairs of nodes, that implies a bigger order in the association graph

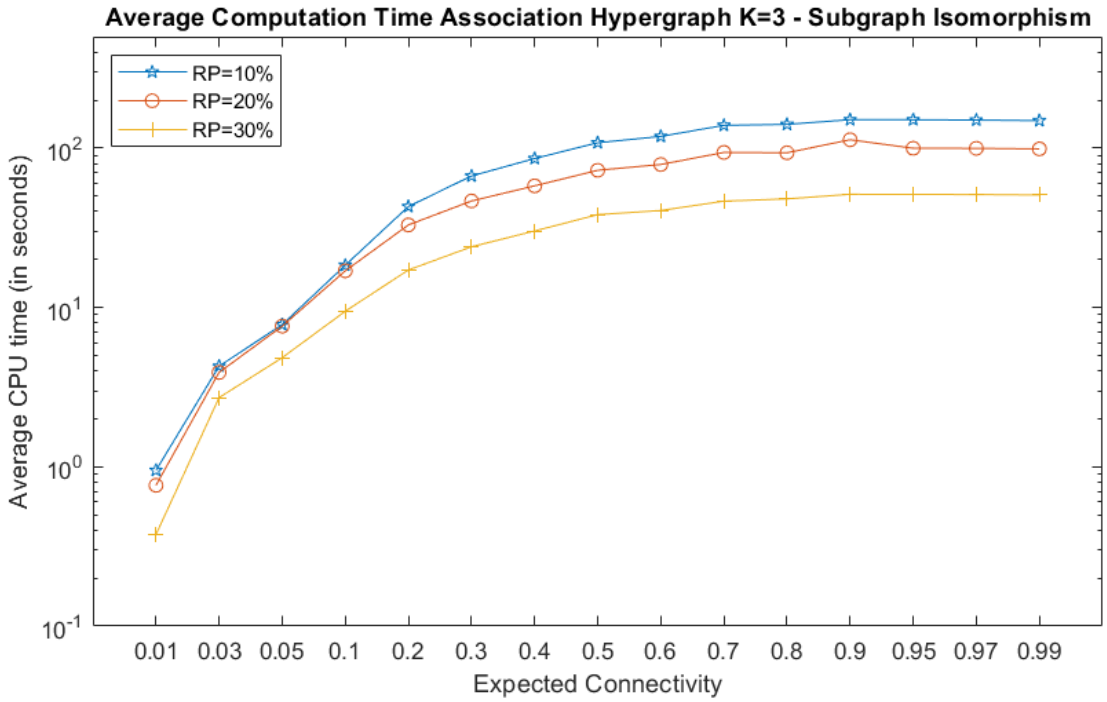

Figure 5.27: Running time needed for creating the association graph in the sub-graph isomorphism problem. Times are in logarithmic scale.

and therefore a larger number of possible arcs to be checked. This behaviour can be easily seen in figure 5.28. Looking at the order of the association graph on the right we see that the denser the graphs the smaller the pruning thus resulting in an higher number of association nodes; on the other hand, the higher the number of removed nodes in H_2, the higher the maximum number of pairings possible. On the left the average size depends both on the density rate and on the number of nodes. Therefore in this type of problems we can deduce that the CPU time for creating the association graph is due to a mix of time needed to check a large number of hyperedges, that increases as the number of total nodes, and of time needed to check each hyperarc in the edge set of the original k-graphs, set that becomes bigger with the density of the graph.

The mean CPU time needed to run the optimization, independently of the correctness of the results, is shown in figure 5.29. The first thing worth noticing is that all the dynamics, except the exponential one with $\kappa = 25$, have the same behaviour. It is strange that the linear and exponential with $\kappa = 50$ are the slowest dynamics, needing more or less the same time to execute in all the different

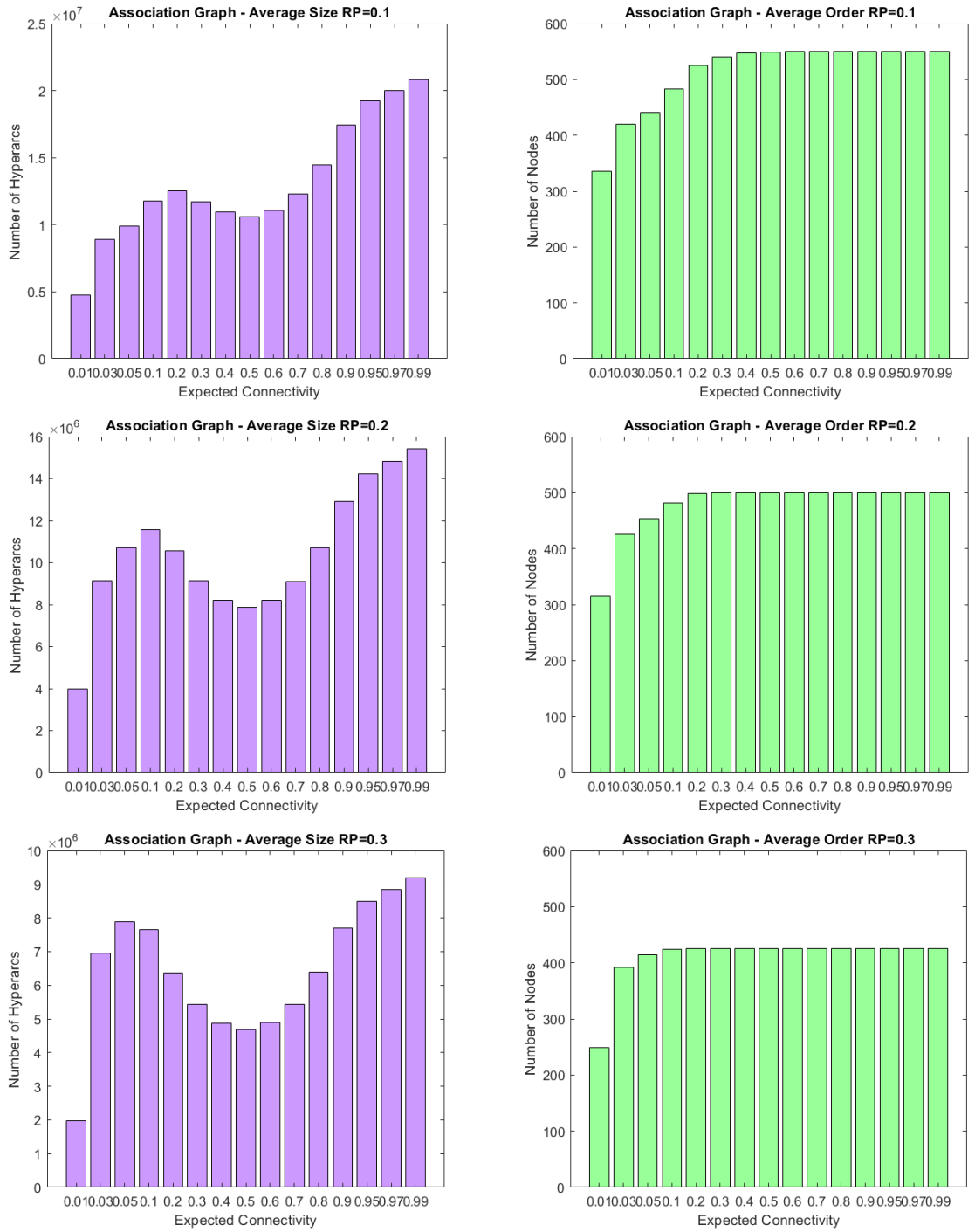

Figure 5.28: Average size and order of association hypergraphs build from graphs of different orders and cardinalities.

connectivity rates and independently on the percentage of nodes removed in H_2. The exponential dynamic with $\kappa = 10$, has the same curvature, but the execution time takes always fewer time, even though less than half an order of magnitude. The exponential dynamic with $\kappa = 25$ draws a different shape, especially in the extreme

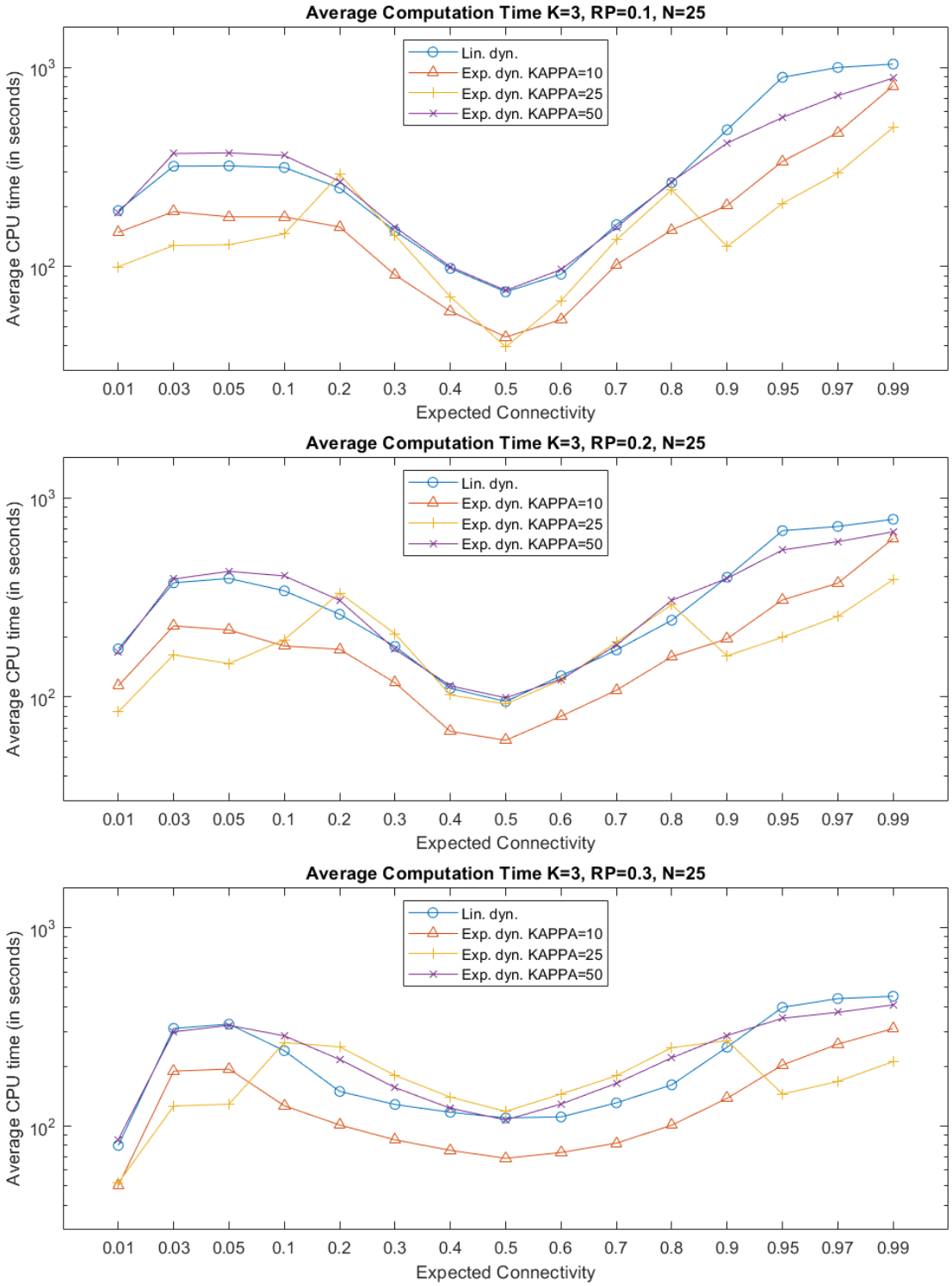

Figure 5.29: Mean CPU time needed to run the optimization algorithm for finding sub-graph isomorphism on hypergraphs with different percentage of node removal, using both linear and exponential dynamics. Y-axes are in logarithmic scale.

connectivity rates, sensibly speeding up the execution time when the density of the graphs to be matches is in [0.01, 0.1] and in [0.9, 0.99]. This acceleration is not related to the correctness of the results, since the performance of the dynamic with respect to the performances of the other dynamics in this sense is not distant enough to justify such a rush.

If we plot the the 95% confidence intervals (see figure 5.30), we cannot see any particular increase in the variance that can explain this behaviour, since in the extreme rates both the confidence intervals are close to the mean.

Another interesting thing to see is the shape drawn by the execution times, that is really similar to the outline of the average size of the association graph plotted in figure 5.28. The connectivity rate $p = 0.01$ is the one with smallest size, and also the one that takes less to optimize. Increasing the density, results in a larger number of arcs in the association hypergraphs as well as longer execution times, up to connectivity $p = 0.1$. Then both the size and the time needed decrease again up to $p = 0.5$, when they increase again up to the most dense values. This parallelism exists in all the experiments with different fraction of removed nodes even though the depression in the central density value is sharper where we have the maximum number of correct isomorphism (10% of removed nodes), because of the shorter time needed, in general, when convergence is achieved.

Finally, figures 5.13 - 5.34 show all the execution time for the optimization step, comparing the time needed for the successful experiments against the unsuccessful ones. With no surprise, if we consider the same rate of node removal for each dynamic, the running time for the correct tests is always shorter than the time needed for the wrong ones. Moreover we can notice that, in general, the curves for the wrong experiments are way smother that the curves for the correct experiments. In fact, except for the strange behaviours of the exponential dynamic with $\kappa = 25$ that we already noticed, the dashed lines slowly constantly increase, while the full lines are those that mainly grab the valley-like shape of the size, thus meaning that the number of arcs is actually a discriminant only for the correct resulting experiments.

When solving the sub-graph isomorphism problem, more than for the complete

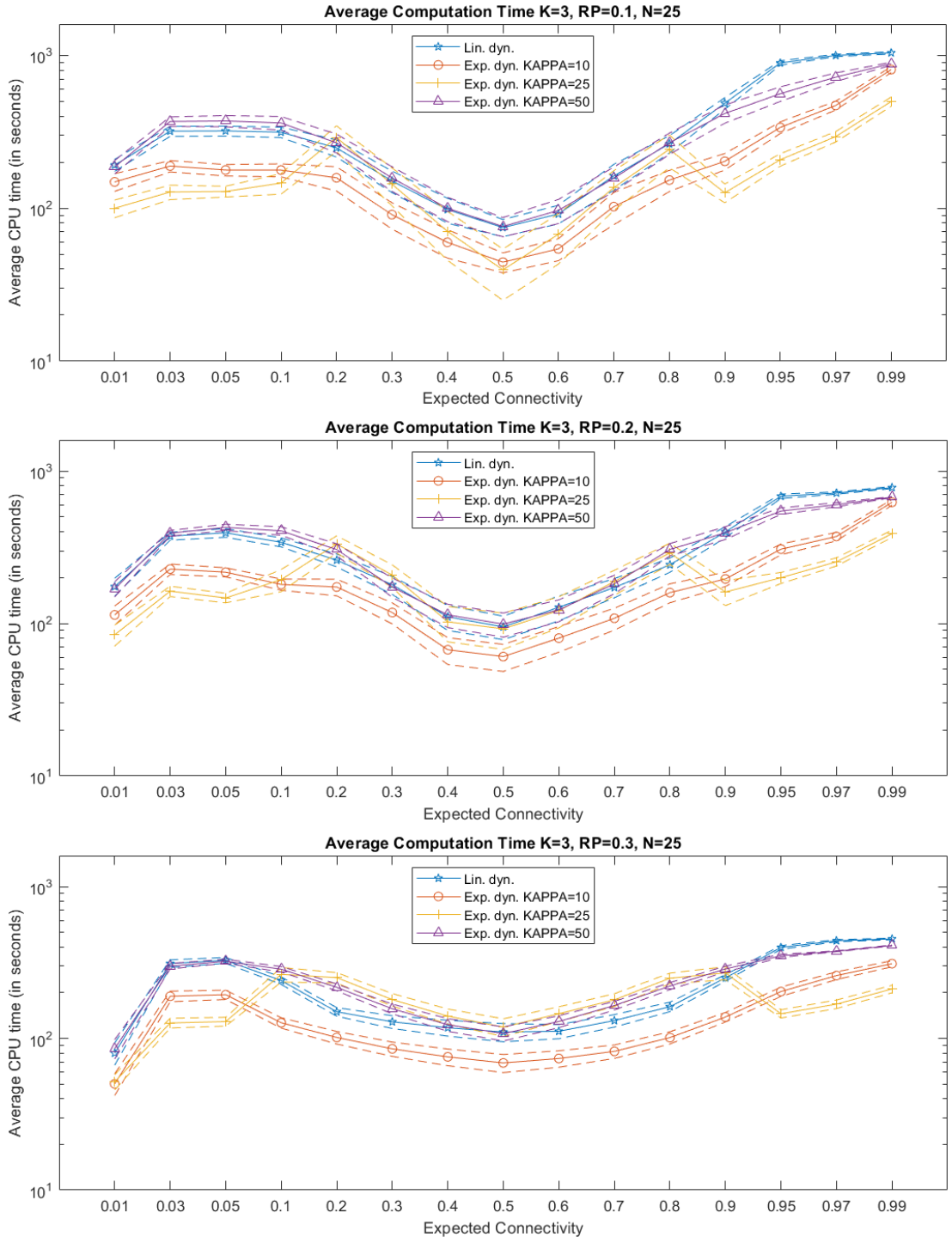

Figure 5.30: Mean and 95% confidence intervals of CPU time needed to run the optimization algorithm for finding sub-graph isomorphism on hypergraphs with different percentage of node removal, using both linear and exponential dynamics. Y-axes are in logarithmic scale.

isomorphism problem, choosing the correct dynamic is a complex task. In this kind of problem in fact, it is not just a matter of time, but also a matter of correctness.

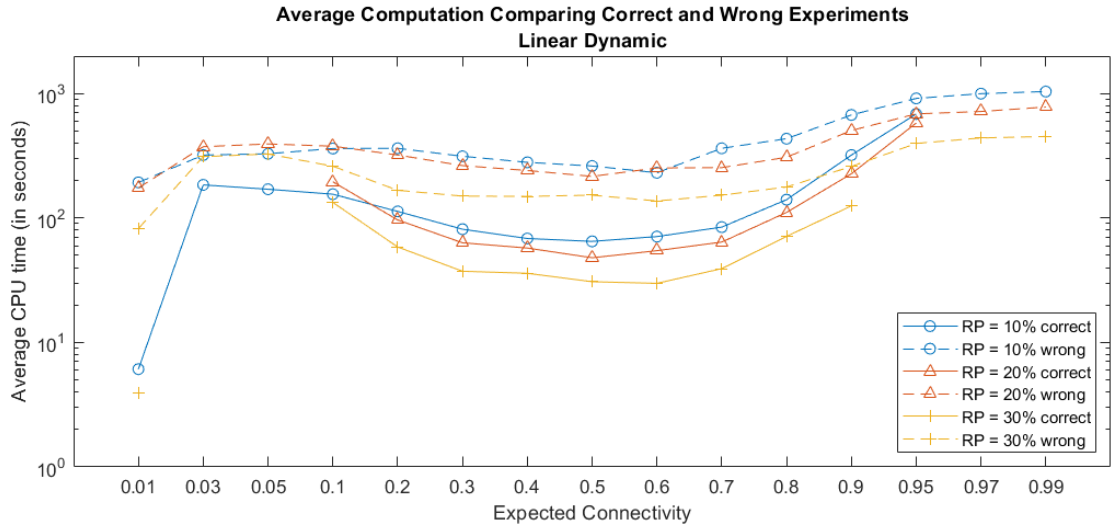

Figure 5.31: Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using linear dynamic. Times are in logarithmic scale.

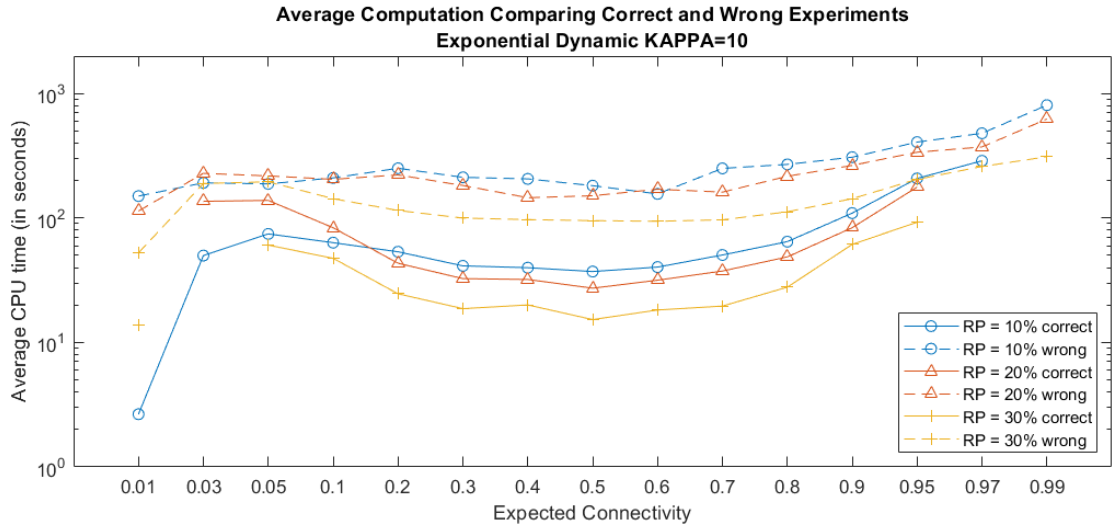

Figure 5.32: Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using exponential dynamic with $\kappa = 10$. Times are in logarithmic scale.

For this reason it is even more important to test the framework with some real examples before using it on real problems, in order to be able to choose the best dynamic, and if it is possible, consider the possibility to use more than one dynamic in order to have a higher probability of finding the correct results.

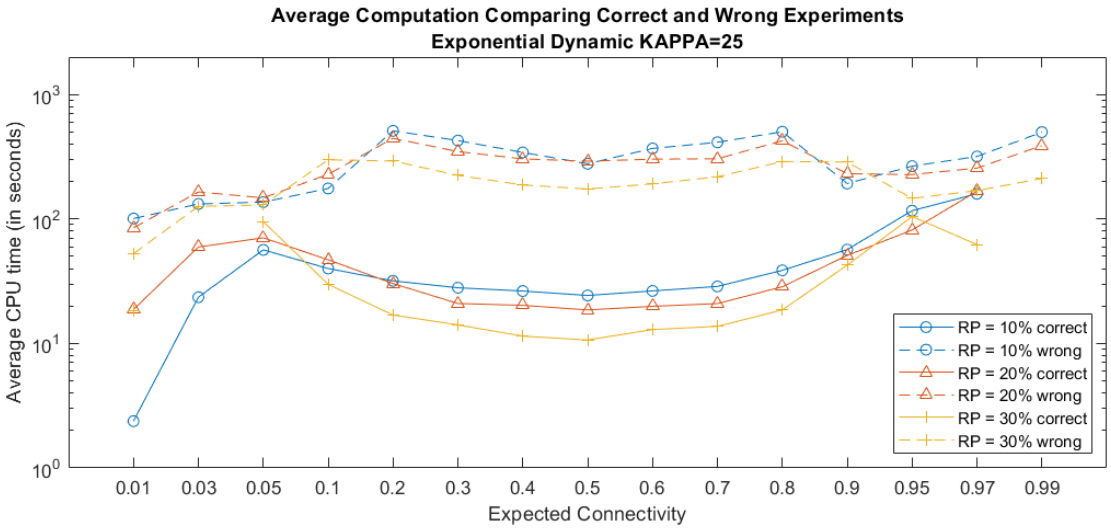

Figure 5.33: Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using exponential dynamic with $\kappa = 25$. Times are in logarithmic scale.

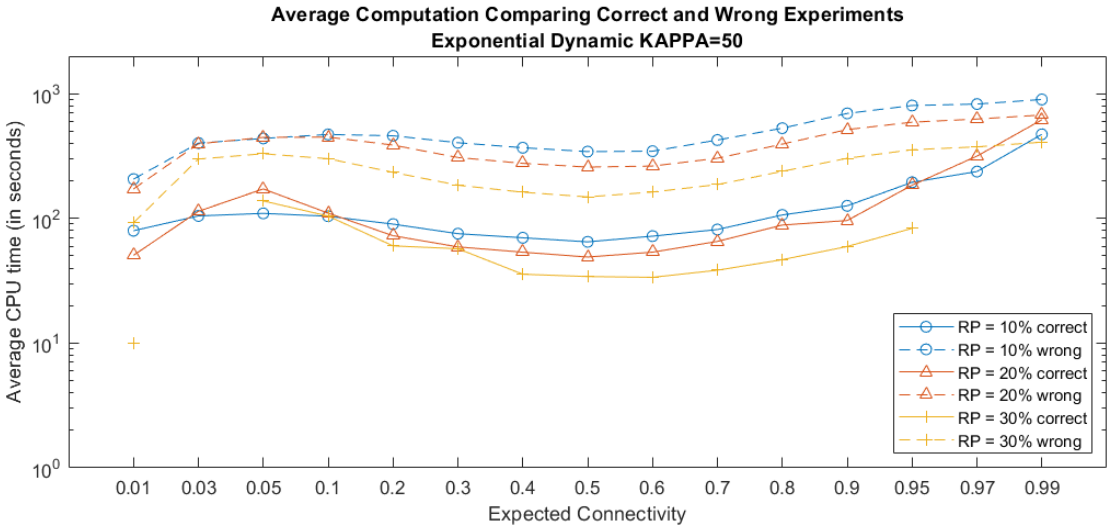

Figure 5.34: Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using exponential dynamic with $\kappa = 50$. Times are in logarithmic scale.

Chapter 6

Conclusions

In this thesis, the potential of a framework based on association graphs for solving hypergraph isomorphism problems has been explored. Dynamics derived from the Baum-Eagon inequality, together with notions from Evolutionary Game Theory, have been introduced to optimize the objective function, thus finding the maximum clique in the association graph that has been proven to be in one-to-one correspondence with the isomorphism. Impressive results have been obtained in terms of precision for the hypergraph isomorphism problem, as in 10500 experiments run on random generated hypergraphs of different orders and connectivities 100% of correct isomorphisms have been obtained, independently on the dynamic used, thus showing the great ability of the simple Baum-Eagon inequality to escape local minima in this kind of problems, and confirming earlier results on graphs [25, 26]. Unfortunately the results obtained in the 4500 experiments on sub-graph isomorphism are not so positive, since only 1600 experiments, 1 out of 3, have returned the correct isomorphism. In particular the structures with medium density returned the best results, while for the ones with extreme connectivity rates the framework returned very poor results, having some densities with 0, or nearly to 0, correct isomorphisms found. Moreover an analysis of the results for the sub-graph isomorphism problem has not being able to identify which is the best dynamic to use, since, keeping fixed the main parameters, there seems to be no concrete distinction between those structures on which a correct isomorphism has been found by a particular dynamic and those that were successful with a different dynamic.

From a computational point of view, the exponentially increasing size of the association graph might become an issue, even though the use of a performing implementation of the algorithm and the use, only when looking for hypergraph isomorphisms, of an exponential dynamic might ease this problem.

Given the collected results we can conclude that the proposed framework works very well for solving the complete isomorphism problem on k-graphs of different order and cardinality, independently on the dynamic used. However some further investigations are needed in the sub-graph isomorphism problem, in particular to avoid that the algorithm returns 2-covers instead of cliques, or runs into local optimum solutions. This result might be obtained modifying the objective function with the one proposed in [30], that gives the theoretical guarantee of returning a (maximal) clique. Moreover some different dynamics can be tested, that might ease the search for the maximum clique.

A further development of the proposed framework is its extension to non-uniform hypergraphs. In fact a few experiments, concerning the isomorphism problem, have been performed on very small structures with edged of different cardinalities, obtaining good results. Even though the set of experiment is way too small for declaring that the proposed algorithm works well also on non-uniform hypergraphs, it gives a hint that this is something worth a deeper and extensive investigation.

List of Figures

2.1 Different types of hypergraphs. 6

2.2 Different types of matching in 3-graphs. From left to right: complete isomorphism, subgraph isomorphism, maximum common subgraph. Red nodes are the matched ones, while grey nodes are not matched. 8

4.1 Copy of image 5.8, including also 95% confidence intervals. Given the large amount of lines and the closeness of the intervals to the mean value, it is preferable to remove the confident intervals in order to make the plot clearer, since they don't bring any additional information. 25

4.2 Different representations for hypergraph H_2 in figure 2.1. On the left the boolean matrix, on the right the indexes matrix. 26

4.3 Average computational time, with 95% confidence intervals for creating the association hypergraph (top) and computing the optimum of the function (bottom) for isomorphic 3-graphs with 50 nodes. Times are in logarithmic scale. 28

4.4 Percentage of nodes still valid after pruning the association hypergraph in the case of complete isomorphism problems with arcs of cardinality $k = 3$ (top) and $k = 4$ (bottom) 30

4.5 Percentage of nodes still valid after pruning 3-graphs of different order. As the order of the graphs to be matched doubles, the order of the association hypergraf halves. 31

4.6 Percentage of nodes still valid after pruning the association hypergraph in the case of sub-graph isomorphism problems with arcs of cardinality $k = 3$ and order of the sub-graph 10% (pink), 20% (blue) and 30% (green) smaller than the bigger graph. 33
4.7 Boxplot of the distribution of the node's degree on 3-graphs with a connectivity rate of 0.1 (left) and 0.5 (right), where the bigger graph H_1 has order $n = 25$, while the sub-graphs are 10%, 20% and 30% smaller. The smaller the subgraph and/or the more connected the graphs, the further the node's distribution. 34
4.8 A pair of isomorphic and symmetric 3-graphs (left) and a symmetric sub-graph isomorphism (right). 35
4.9 Evolution through time of the components of the state vector $\mathbf{x}(t)$ from the hypergraphs in figure 4.8 using the Baum-Eagon inequality. A perturbation can be seen at iteration 17 in order to escape a saddle point. After the perturbation the algorithm clearly makes a decision about what associations have to be chosen and what others have to be discarded. 36

5.1 Correctness results for complete isomorphism on 3-graphs of 25 nodes, including all tested dynamics. 40
5.2 Correctness results for complete isomorphism on 3-graphs of 50 nodes, including all tested dynamics. 40
5.3 Correctness results for complete isomorphism on 3-graphs of 75 nodes, including all tested dynamics. 40
5.4 Correctness results for complete isomorphism on 3-graphs of 100 nodes, including all tested dynamics. 40
5.5 Correctness results for complete isomorphism on 4-graphs of 25 nodes, including all tested dynamics. 41
5.6 Correctness results for complete isomorphism on 4-graphs of 50 nodes, including all tested dynamics. 41
5.7 Correctness results for complete isomorphism on 5-graphs of 25 nodes, including all tested dynamics. 41

5.8 Running time needed for creating the association graph in the complete isomorphism problem. Times are in logarithmic scale. 42
5.9 Average size and order of association hypergraphs build from graphs of different orders and cardinalities. 43
5.10 Mean CPU time needed to run the optimization algorithm for finding isomorphism on hypergraphs with different cardinality and order, using both linear and exponential dynamics. Y-axes are in logarithmic scale. 45
5.11 Evolution of the vector state through time using exponential dynamic with $\kappa = 100$, for an association 3-graph made from graphs of order 25 and connectivity 0.01. 46
5.12 Different behaviour of the dynamical systems under examination, with various dynamics for the same 3-graph made from graphs of order 25 and connectivity 0.01. 47
5.13 Comparison of the mean CPU time for the best dynamics. 48
5.14 Correctness results for sub-graph isomorphism on 3-graphs of 25 and 22 nodes (10% of removed nodes), including all tested dynamics. . . 49
5.15 Correctness results for sub-graph isomorphism on 3-graphs of 25 and 20 nodes (20% of removed nodes), including all tested dynamics. . . 49
5.16 Correctness results for sub-graph isomorphism on 3-graphs of 25 and 17 nodes (30% of removed nodes), including all tested dynamics. . . 50
5.17 Percentage of correctn results for sub-graph isomorphism with different dynamics: linear (first from top), exponential with $\kappa = 10$ (second), exponential with $\kappa = 25$ (third), exponential with $\kappa = 50$ (bottom). 51
5.18 Total number of correct experiments according to the specific dynamic used. 52
5.19 List of experiments that returned a correct isomorphism in at least one of the tested dynamics. 53

5.20 Another example of a list of experiments, with different parameters, that returned a correct isomorphism in at least one of the tested dynamics. 54
5.21 Evolution through time of the vector state for experiment number 17 of figure 5.19, with all the different dynamics tested. 55
5.22 Evolution through time of the vector state for experiment number 23 of figure 5.19, with all the different dynamics tested. 56
5.23 Evolution through time of the vector state for experiment number 1 of figure 5.19, with all the different dynamics tested. 58
5.24 Recap of the type of results returned in all the experiments by the different dynamics. 59
5.25 Number of correct experiments in at least one dynamic divided by number of removed nodes in H_2. 60
5.26 Total number of correct experiments with each different dynamic, and with all the dynamics. 60
5.27 Running time needed for creating the association graph in the sub-graph isomorphism problem. Times are in logarithmic scale. 62
5.28 Average size and order of association hypergraphs build from graphs of different orders and cardinalities. 63
5.29 Mean CPU time needed to run the optimization algorithm for finding sub-graph isomorphism on hypergraphs with different percentage of node removal, using both linear and exponential dynamics. Y-axes are in logarithmic scale. 64
5.30 Mean and 95% confidence intervals of CPU time needed to run the optimization algorithm for finding sub-graph isomorphism on hypergraphs with different percentage of node removal, using both linear and exponential dynamics. Y-axes are in logarithmic scale. . 66
5.31 Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using linear dynamic. Times are in logarithmic scale. 67

5.32 Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using exponential dynamic with $\kappa = 10$. Times are in logarithmic scale. 67
5.33 Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using exponential dynamic with $\kappa = 25$. Times are in logarithmic scale. 68
5.34 Comparison of the mean CPU time for the correct and wrong experiments with the different rate of node removals in H_2 using exponential dynamic with $\kappa = 50$. Times are in logarithmic scale. 68

Acknowledgements

If somebody had told me, a little more than two years ago, that today I would have been here, at the end of my Master Degree adventure, I would have never believed them. But here I am, and there are at least a couple of people that I want to thank for helping me through this journey.

Professor Marcello Pelillo, who made me love the subjects I've been working on in the last months. Thank you for the time spent in assisting me through this thesis, and for giving me your trust.

Sebastiano, who always encouraged me, listened to me, discussed with me about all the doubts passing in my mind, and employed days, and even a few nights, helping me with these experiments. You probably are the only person I know who is more optimistic than me, and you know I've needed your positivity in more than one circumstance.

My friends Ylenia and Marco, oh yes, friends, you read it right, you are not just colleagues. You are the most unexpected thing that happened to me in these last two years. You accepted me without any prejudice, even though outside the campus we live totally different lives. You made me become a smarter person, exchanging ideas, opinions, points of view. You gave me the strength to study even the most boring classes, and you supported me in those endless exams sessions. Thank you for the lunches, the ski days, the department's party, the dinner out ... but most of all, thank you for the poultry shears ;-P .

My mother Irma, for her invaluable help, every single day in the last two years (not to mention the previous 33 years of my life). Without you I would have never had the time to get here today.

My three little girls, who constantly remember me that life is amazing, no matter all the problems, the exams, the deadlines. Thanks to you I know that there is life outside Via Torino.

Finally, last but not least, a huge thank you goes to Gabriele, who always supported me and stood by me. I know sometimes I'm awful, I pour on you all my frustration, and anxiety, and nervousness, but you are still here...even though it's been months since the last time I cooked you something for dinner, and I make you eat a bowl of salad every single evening.

Bibliography

[1] H. G. Barrow and R. M. Burstall. Subgraph isomorphism, matching relational structures and maximal cliques. *Inf. Process. Lett.*, 4(4):83–84, 1976.

[2] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

[3] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[4] G. R. Blakley. Homogeneous nonnegative symmetric quadratic transformations. *Bulletin of the American Mathematical Society*, 70(5):712–715, 1964.

[5] I. M. Bomze. Evolution towards the maximum clique. *Journal of Global Optimization*, 10(2):143–164, 1997.

[6] M. Broom, C. Cannings, and G. T. Vickers. Multi-player matrix games. *Bulletin of mathematical biology*, 59(5):931–952, 1997.

[7] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[8] J. F. Crow, M. Kimura, et al. An introduction to population genetics theory. *An introduction to population genetics theory.*, 1970.

[9] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2383–2395, 2011.

[10] P. Frankl and V. Rödl. Hypergraphs do not jump. *Combinatorica*, 4:149–159, 1984.

[11] D. Fudenberg, J. Tirole, J.A. TIROLE, and MIT Press. *Game Theory*. Mit Press. MIT Press, 1991. ISBN: 9780262061414.

[12] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[13] E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[14] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on pattern analysis and machine intelligence*, 18(4):377–388, 1996.

[15] J. Hofbauer and K. Sigmund. *Evolutionary games and population dynamics*. Cambridge university press, 1998.

[16] D.R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Winner of the Pulitzer prize. Vintage Books, 1979. ISBN: 9780394756820.

[17] J. Hou and M. Pelillo. A game-theoretic hyper-graph matching algorithm. In *24th International Conference on Pattern Recognition (ICPR)*, 2018.

[18] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[19] D. Kozen. A clique problem equivalent to graph isomorphism. *ACM SIGACT News*, 10(2):50–52, 1978.

[20] J. Lee, M. Cho, and K. M. Lee. Hyper-graph matching via reweighted random walks. In *CVPR 2011*, pages 1633–1640, June 2011.

[21] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1482–1489. IEEE, 2005.

[22] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17:533–540, 1965.

[23] J. Nash. Non-cooperative games. *Annals of Mathematics*:286–295, 1951.

[24] J. F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[25] M. Pelillo. A unifying framework for relational structure matching. In *Proc. 14th International Conference on Pattern Recognition, (ICPR)*, pages 1316–1319, 1998.

[26] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural computation*, 11(8):1933–1955, 1999.

[27] M. Pelillo. The dynamics of nonlinear relaxation labeling processes. *Journal of Mathematical Imaging and Vision*, 7(4):309–323, 1997.

[28] Y. Peng, H. Peng, Q. Tang, and C. Zhao. An extension of the motzkin-straus theorem to non-uniform hypergraphs and its applications. *Discrete Applied Mathematics*, 200:170–175, 2016.

[29] S. Rota Bulò and M. Pelillo. A game-theoretic approach to hypergraph clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1312–1327, June 2013. ISSN: 0162-8828.

[30] S. Rota Bulò and M. Pelillo. A generalization of the motzkin–straus theorem to hypergraphs. *Optimization Letters*, 3(2):287–295, 2009.

[31] G. Sandi, S. Vascon, and M. Pelillo. On association graph techniques for hypergraph matching. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):530–535, 1997.

[32] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):530–535, 1997.

[33] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988.

[34] J.M. Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982. ISBN: 9780521288842.

[35] J. W. Weibull. *Evolutionary game theory*. MIT press, 1997.

[36] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. M. Chu. Discrete hypergraph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1520–1528, 2015.

[37] H. Zhang and P. Ren. Game theoretic hypergraph matching for multi-source image correspondences. *Pattern Recogn. Lett.*, 87:87–95, 2017.