Finding Patterns in Biochemical Networks
A Constraint Programming Approach

Stéphane Zampelli, Yves Deville, Pierre Dupont

Department of Computing Science and Engineering
Université catholique de Louvain
B-1348 Louvain-la-Neuve - Belgium
{sz,yde,pdupont}@info.ucl.ac.be

Abstract

Graph pattern matching is a central problem in many application fields, and may be associated with problems in bioinformatics, such as finding patterns in biochemical networks. This problem can be view as a particular case of labelled subgraph isomorphism (SGI). In this paper, we focus on a constraint programming approach. Two new constraints are introduced to solve this problem. We consider labelled graphs, especially suited for representing biochemical networks, and we propose a constraint exploiting this information. Another constraint considers neighbors within k steps, generalizing the simple neighbor constraint. Experimental results show the potential benefit of the constraints when integrated in a backtracking-based constraint system.

1 Introduction

Bioinformatics is facing a new challenge in analyzing the functioning of biochemical networks. Many efforts focused on the decoding of complete genomes and the assignment of functions to genes and proteins. The huge amount of data about genes and proteins and the availability of complete genomes offer the possibility to study more globally the interactions between bio-entities cooperating in complex cellular processes. The focus can shift now from molecular functions of genes and proteins to cellular functions. Informations about cellular processes have already been digitalized and stored in databases, and models for such a knowledge have already been designed. The complexity and scale of molecular interactions in cellular processes is such that new computational tools to perform analysis are needed. This article is part of the interdisciplinary BioMaze project which aims to provide tools for analyzing biochemical networks in cellular processes. The BioMaze project covers the building of databases about cellular process knowledge, visualization tools, and computational tools to analyze cellular networks.

This paper presents an algorithm using constraint programming techniques, extending the work of Larrosa and Valiente [1], for finding patterns in biochemical networks. This algorithm finds each occurrence of a pattern network in a large cellular network. The pattern network may have general properties defined on its biochemical entities, ranging from general to specific properties about a single entity. This contribution is a first step towards a more general objective: the ability to find approximate pattern networks or to discover recurrent motifs in biochemical networks.

There are various applications of pattern finding tools for biochemical network analysis. One can compare biochemical pathways from different tissues, or at different stages of annotation. One can highlight common features and differences. One can predict missing elements for reconstruction. One can compile repertories of recurrent network motifs (topological patterns) at different resolution levels. A very concrete application aims at finding, among a library of biochemical networks of bacteria, the closest process producing a given protein involved in a human cellular process.

Basic biochemical network analysis were developed in projects such as aMaze [2], Kegg [3], BioCyc [4, 5], Um-BBD [6] , EMP [7], PathDB [8], but these analysis concern pathways instead of patterns. The concept of network motifs has already been studied for regulatory biochemical networks [9].

Objectives The primary objective of this work is the development of a pattern finding procedure in biochemical networks. Pattern finding can be viewed as a particular instance of a subgraph isomorphism problem. We propose an approach relying on Constraint Programming (CP) techniques [10] to tackle this problem. The second objective of this work is to demonstrate the feasibility and advantages of this approach.

Results In the CP framework, this paper extends the work of Larrosa and Valiente [1]. Two new global constraints are presented here. We discuss their design and implementation. Preliminary experiments on random graphs are described here. They illustrate the relative performances of the various constraints considered. They show the benefits of considering the new constraints for labelled graphs as in biochemical networks.

Outline Section 2 briefly describes the data models used to represent and analyze biochemical networks and the CP paradigm. Related works are also discussed. Section 3 focuses on constraints for pattern finding. Section 4 details our experimental results. Section 5 summarizes the contributions of this work and presents current research directions.

2 Background

In this section a model of biochemical networks is presented and the constraint programming paradigm is introduced. Related works are also discussed.

2.1 Biochemical Networks

Biochemical networks are networks of interactions between biochemical entities within the cell. Various models have been developed for the analysis of biochemical networks [11]. The object-oriented model developed in the aMaze project is especially suitable for our analysis purposes [2].

Bio-entities are composed of genes (segments of DNA involved in the production of polypeptide chains), polypeptides (proteins), complexes (composed of several proteins), and compounds (such as ATP, ADP, water,...). These bio-entities form the basic elements interacting with each other in cellular processes. An interaction may be a transformation or a control. A reaction is a transformation occurring between bio-entities within the cell. The expression of a gene into a protein is a typical transformation. An assembly is a transformation that forms a complex from polypeptides. A catalysis is a control where an enzyme controls a reaction. A regulation is a control that regulates the expression of a gene. These interactions between bio-entities form biochemical networks. These interactions and bio-entities can be conceptually considered as objects structured in a hierarchy (see Figure 1). This hierarchy is a data model to represent biochemical networks.

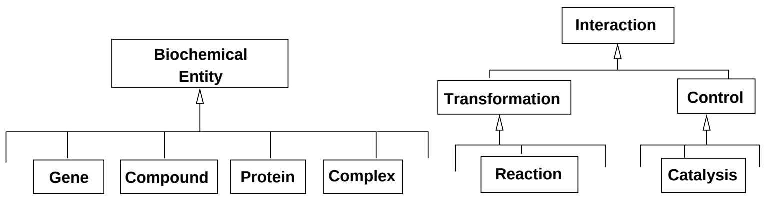

Figure 1: Object hierarchy of bio-entities and transformations.

Different kind of networks are usually distinguished. A metabolic network represents a series of reactions, controlled by enzymes, starting from specific substrates and leading to specific products. A regulatory network is a network focusing on the regulation of the enzyme activity or the stimulation of the enzyme expression. A signal transduction network focuses on the transport of information, generally from membrane to genes.

The hierarchical object-oriented model integrates three different types of networks. From this model one can derive oriented or non oriented graphs representing the interactions. Bio-entities and transformations are nodes of such a graph, and arcs represent relations between these concepts. Nodes and arcs can be labelled with types and attributes relevant to specific analysis. Figure 2 shows a graphical view of a biochemical network integrating the three kinds of networks described. In this example, there are two transformations, one reaction and one assembly. Two controls are represented, a catalysis and an inhibitor on the catalysis itself.

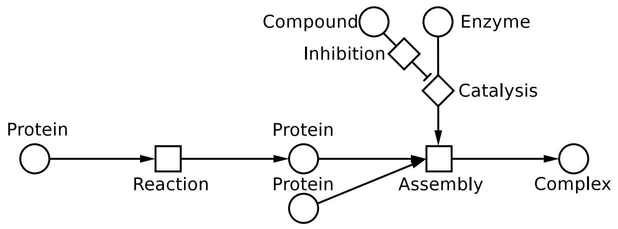

Figure 2: Graphical view of a biochemical network.

A pattern in a biochemical network can be defined as a specific subgraph of the general graph representing the whole network. Pattern finding can thus be viewed as a particular type of subgraph isomorphism.

Nodes with labels allow for complex biochemical network analysis as several properties can be used to constrain the matching between nodes. A node in the pattern graph can be labelled as a transformation or a reaction, a control or an inhibitor, a polypeptide with at least 16 amino acids, *etc.*

2.2 Constraint Programming

Many interesting problems can be modeled and solved using constraint programming [10]. In this paradigm, one chooses relevant variables from the problem, the domain of these variables (the possible values taken by the variables), and states constraints on acceptable solutions. A constraint defines a relation on (a subset of) variables. The solutions of the problem are found by search in a space of candidate solutions and constraint propagation. The search procedure browses through values of the variables, and backtracks if the domain of any variable is not consistent with the constraints. The constraint propagation uses the constraints in order to reduce the domains of the variables, keeping them consistent. A Constraint Satisfaction Problem (CSP) can be formally defined as follows.

Definition 1 *A finite* CSP *(Constraint Satisfaction Problem)* $P = (X, D, C)$ *is defined as a set of* n *variables* $X = \{x_1, ..., x_n\}$*, a set of finite domains* $D = \{D_1, ..., D_n\}$ *where* D_i

is the set of possible values for variable x_i and a set of constraints between variables $C = \{C_1, ..., C_m\}$.

A constraint *C_i is defined on a set of variables $\{x_{i_1}, ..., x_{i_p}\} \subseteq X$ as a subset of the Cartesian product $D_{i_1} \times ... \times D_{i_p}$. A* solution *is an assignment of values to all variables which satisfies all the constraints.*

Definition 2 *A variable $x_i \in X$ with domain D_i is* consistent *with a given n-ary constraint C_j iff there exists a tuple in $D_1 \times ... \times D_{i-1} \times \{a_i\} \times D_{i+1} \times ... \times D_n$ with $a_i \in D_i$ such that the constraint C_j is satisfied.*

A CSP $P = (X, D, C)$ is arc-consistent *iff $\forall\ x_i \in X$, $\forall a_i \in D_i$, $\forall\ C_S \in C$ constraining x_i, the value a_i is consistent with C_S.*

2.3 Related work

Related to our subject is the comparison of metabolic pathways performed in [12], and the analysis of transcriptional regulation networks of *Escherichia coli* in order to uncover its underlying structural design, by means of the discovery of network motifs [13, 9]. In this work, a network motif is a pattern of interconnections occurring in networks with a significantly higher frequency than what would be expected in random networks. This analysis relies on sophisticated algorithms for the generation of random networks, and has been applied to other networks in neurobiology, ecology and engineering.

Finding pattern in a graph is often referred to as the subgraph isomorphism problem. There exist many specific methods including backtracking, the clique approach, the relational approach and constraint satisfaction. This work extends the constraint approach proposed in [14]. A global constraint for graph isomorphism is proposed in [15], where an extension to subgraph isomorphism is also sketched.

3 Finding patterns in biochemical networks

In this section, the pattern finding problem is specified and modeled as a CSP problem. Constraints are described, based on the work of Larrosa and Valiente [1], and our new global constraints are presented.

3.1 Problem specification

Suppose we have a labelled *pattern* graph and a labelled *target* graph, with bio-entities and transformations associated to nodes in these graphs. The problem consists in finding one or all subgraphs of the target graph corresponding to the pattern graph. Pattern finding is thus a classical subgraph isomorphism (SGI) problem:

Problem specification 1 $\boxed{\textbf{\textit{SGI}}}$
A pattern graph is a graph $G_p = (N_p, A_p)$. The target graph is a graph G=(N,A). We have $|N_p| \leq |N|$. The problem is to find a function $f : N_p \rightarrow N$ such that:

A. f is injective

B. $\forall n_1, n_2 : (n_1, n_2) \in A_p \Rightarrow (f(n_1), f(n_2)) \in A$

The SGI problem must be extended in order to handle labelled nodes in the pattern and in the target graphs, as in biochemical networks.

Problem specification 2 $\boxed{\textbf{\textit{Labelled SGI}}}$
This problem extends the SGI problem with an additional label compatibility constraint:

C. $\forall n : n \in N_p \Rightarrow$ the label of n is compatible with the label of $f(n)$.

Compatibility between labels depends on the specific analysis to be performed.

We argue that constraint programming is particularly relevant for solving the labelled SGI problem in the context of biochemical analysis. The subgraph isomorphism problem is NP-complete. The CSP paradigm was initially developed to tackle hard computational problems in artificial intelligence. Provided relevant constraints are available, the search space can often be very efficiently reduced. As detailed in section 3.2, the subgraph isomorphism problem can be directly modeled in the CSP paradigm. Besides, several systems for analyzing biochemical networks rely on ad-hoc algorithms. These algorithms are often difficult to combine or extend to new analysis without requiring significant programming efforts. In our present problem, the addition of custom constraints on the labeling of nodes in the graphs is a direct extension of the proposed technique.

3.2 Subgraph Isomorphism as a CSP

The SGI problem has a direct translation into the CSP paradigm [16]. It is represented by the following CSP $P =< X, D, C >$, with $|X| = |N_p| = n$ (one variable per node in the pattern graph), $D_i = N$ (domains are the nodes in the target graph), and C is the set of constraints. We note $d = |A|$.

In this section, $i, j, ...$ will denote nodes in the pattern graph; $a, b, ...$ will denote nodes in the target graph; $x_i, x_j...$ will denote the variables in the pattern graph corresponding to its $i, j, ...$ nodes.

The set of constraints must now be described; conditions A, B, and C of the problem specifications have thus to be translated into constraints. The SGI problem in CSP was already studied in [17, 1]. We follow here Larrosa and Valiente's methodology, and extend it. We first consider the condition A (C1) and the condition B (C2) from the problem specification. Condition C on label compatibility will be held later.

3.3 C1 constraint

This first constraint states that the matching function f is injective, that is all variable values in the solution must be distinct. A naive model relies on n^2 local constraints:

$$\forall\, (x_i, x_j) \in X \times X : \quad x_i \neq x_j \quad \text{(C1)}$$

These constraints prune very inefficiently the search space since constraint propagation is performed only when one variable is set to a single value. Better pruning can be achieved by global constraints, such as the *AllDiff* constraint proposed by Régin [18]. This global constraint has a $O(n^2d^2)$ time complexity and $O(nd)$ space complexity.

3.4 C2 constraint

The C2 constraint states that the structure of G_p must respect the structure of G: $(i, j) \in A_p \Rightarrow (f(i), f(j)) \in A$. This is a necessary condition which may be expressed as follows. If a variable node of the pattern graph x_i is assigned to a node a in the target graph, a test must check if there is at least one value d for a neighbor x_j of x_i such that $(a, d) \in A$. If there is a neighbor of x_i not respecting this condition, a can be pruned from D_i.

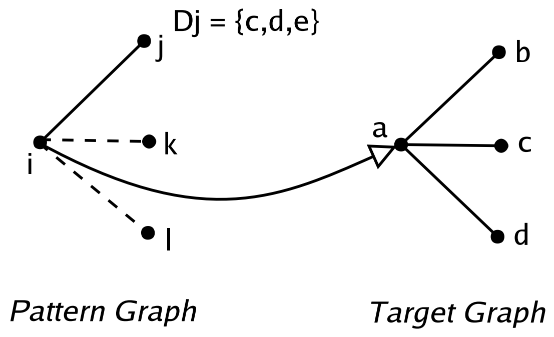

Figure 3: C2 constraint example.

Figure 3 illustrates this constraint. If the domain of a variable x_j with $j \in G_p$ has an empty intersection with the neighbors of a node $a \in G$, then value a can be pruned from all the neighbors j in G_p. In this example constraint C2 is verified for x_j since $\{c, d, e\} \cap \{b, c, d\} = \{c, d\}$.

For each $j \in neigh(G_p, i)$ and for each $a \in D_i$, the C2 constraint can be formulated as follows :

$$D_j \cap neigh(G, a) \neq \emptyset \quad \text{(C2)}$$

where $neigh(G, a)$ is the set of neighbors of a in the target graph G. An implementation of this constraint was proposed in [1]. It has a $O(nd^2)$ time complexity and $O(nd)$ space complexity. This implementation is more efficient than the classical arc consistency algorithm for C2 while providing the same pruning.

Constraints C1 and C2 are sufficient for solving the SGI problem. For solving the labelled SGI problem, condition (C) must be modeled by the addition of n compatibility constraints of the form $compatible(i, x_i)$. These constraints depend on the analysis and may require to access properties of the nodes.

3.5 A redundant constraint

C1 and C2 constraints are necessary constraints. Larossa and Valiente [1] define a redundant constraint on the possibility of matching $x_i \in G_p$ with $a \in G$. The C3 constraint states that, if we try to assign $x_i \in G_p$ to $a \in G$, the number of possible values for the neighborhood of x_i must be greater or equal to the number of variables around x_i:

$$| \cup_{j \in neigh(G_p,i)} D_j \cap neigh(G, a)| \geq |neigh(G_p, i)| \quad \text{(C3)}$$

The value a can be pruned from D_i if there are more neighbors than candidate values for these neighbors. The C2 constraint only considers candidates for one neighbor of x_i at a time while C3 considers all neighbors. Constraint C3 provides more pruning than C2. Assume for example that all variables have a single identical candidate that is a neighbor of a. None of them has an empty intersection with the neighbors of a, but there are not enough candidates that could be assigned.

Figure 4 illustrates the C3 constraint. Details about this constraint are given in [1]. This constraint has a $O(nd^2)$ time complexity and $O(nd)$ space complexity.

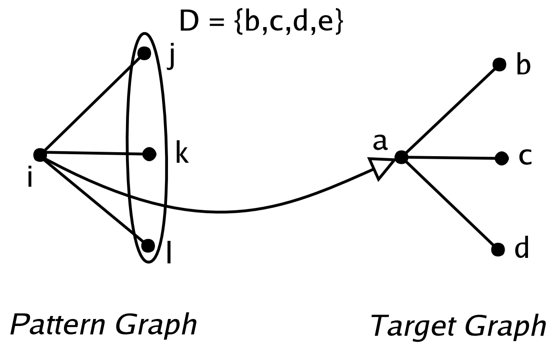

Figure 4: C3 constraint states that a can be pruned from D_i if there are more neighbors than candidate values. Here x_i may be matched to a, since $|D \cap neigh(G,a)| = 3 \geq |neigh(G_p,i)| = 3$, with $D = D_j \cup D_k \cup D_l$.

The redundant constraint C3 can be used for the SGI problem as well as for the labelled SGI problem.

3.6 A new redundant constraint for labelled SGI

We propose here a new global redundant constraint, called C4, generalizing constraints C2 and C3. The C2 constraint considers only one neighbor around a node $i \in G_p$. The C3 constraint considers all neighbors around a node $i \in G_p$. The general case states that each group of k variables around a node $i \in G_p$ must have at least k distinct candidates. C2 and C3 can be both satisfied while the C4 constraint would prune since there may be enough candidates for each individual variable and for all variables around $i \in G_p$, but not enough for a subset of the variables around i. Figure 5 illustrates this new constraint.

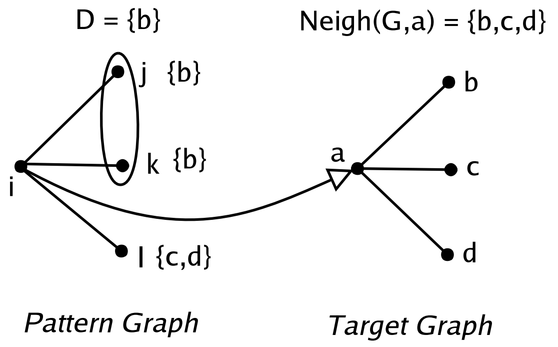

Figure 5: In this example, the C2 constraint is satisfied (each variable has at least one candidate), and C3 is also satisfied (there are three candidates for all three variables x_j, x_k, x_l), but C4 is not satisfied. The group formed by the variables x_j and x_k has only one candidate $\{b\}$. Here, $D = D_j \cup D_k$.

The formulation of the C3 constraint may be generalized to the new C4 constraint. A group g around a variable x_i is a subset of $neigh(G_p,i)$. Let $neigh(G_p,i,g)$ denote the list of neighbors of $i \in G_p$ belonging to the subset g. For a given group g around $x_i \in G_p$, the following inequality must be satisfied:

$$| \cup_{j \in neigh(G_p,i,g)} D_j \cap neigh(G,a)| \geq |neigh(G_p,i,g)| \quad \text{(C4)}$$

The data structures $R(i,b)$ and $CT(i,a)$, described in Valiente's implementation of the C3 constraint, can be extended to take the group g into account:

$$\begin{aligned} R(i,b,g) &= |\{j \in neigh(G_p,i,g) \mid b \in D_j\}| \\ CT(i,a,g) &= |\cup_{j\in neigh(G_p,i,g)} D_j \cap neigh(G,a)| \\ &= |\{b \in neigh(G,a) \mid R(i,b,g) > 0\}| \end{aligned}$$

$R(i,b,g)$ denotes the number of b in the domains D_j of the variables associated to the nodes in g. $CT(i,a,g)$ is the left member of inequality (C4).

While the additional pruning provided by C4 is interesting, the time complexity of this new constraint is an issue. Considering q subsets of variables around $x_i \in G_p$, the criterion must be checked for each of them. The space complexity of C4 is then $O(2qnd) = O(qnd)$ and its time complexity $O(qnd^2)$. The exponential number of such new constraints, if all subsets around x_i were considered, claims for a careful selection of the subsets which will be actually considered. In any case, q must be kept small, typically in $O(n)$, in order to limit the overall complexity of the C4 constraints to $O(n^2d^2)$.

In the typed case, a particular instance of labelled SGI, the subsets of variables will be chosen according to their types, creating groups of variables with *disjoint domains*. In the following, $Group(i)$ will denote the set of C4 groups around i, with $i \in G_p$.

Definition 3 *Groups around a node $i \in G_p$ form a* partition with disjoint domains *iff :*

- $\cup_{g\in Group(i)} neigh(G_p,i,g) = neigh(G_p,i)$ *and* $neigh(G_p,i,g) \neq \emptyset$.
- $\forall\, g_1, g_2 \in Group(i),\ j_1 \in neigh(G_p,i,g_1),\ j_2 \in neigh(G_p,i,g_2),\ g_1 \neq g_2\ :$ $D_{j_1} \cap D_{j_2} = \emptyset$

In that case, C3 is useless as its potential pruning will also be deduced by constraint C4. Moreover, its time complexity is $O(qnd_2)$, where q is the number of types, with $q << n$.

Theorem 1 *If C4 groups form a partition with disjoint domains, then C4 is stronger than C3.*
Proof.
Figure 5 shows an example where C4 provides more pruning than C3. We have to prove that whenever C3 prunes, C4 prunes also.

Suppose $C3$ prunes a out of D_i. By definition of C3,

$$|\cup_{j\in neigh(G_p,i)} D_j \cap neigh(G,a)| < |neigh(G_p,i)| \quad \text{(I)}$$

The two parts of the inequality (I) can be rewritten in term of C4 groups :

- $|neigh(G_p,i)| = \sum_{g\in Group(i)} |neigh(G_p,i,g)|$
- $|\cup_{j\in neigh(G_p,i)} D_j \cap neigh(G,a)| = \sum_{g\in Group(i)} |\cup_{j\in neigh(G_P,i,g)} D_j \cap neigh(G,a)|$

Those equalities are true because the C4 groups form a partition with disjoint domains. Inequality (I) can be rewritten :

$$\textstyle\sum_{g\in Group(i)} |\cup_{j\in neigh(G_p,i,g)} D_j \cap neigh(G,a)| < \sum_{g\in Group(i)} |neigh(G_p,i,g)|$$

and imposes that :

$$\exists\, l \in Group(i) \quad : \quad |\cup_{j\in neigh(G_p,i,l)} D_j \cap neigh(G,a)| < |neigh(G_p,i,l)|$$

and thus C4 prunes a out of D_i.

□

This theorem shows that, in the typed case, a more general version of C3 having the same time complexity can be used. Experiments will enforce this fact.

3.7 Generalizing C3 to path

Constraint C3 considers the direct neighbors of a node. This can be extended by considering the neighbors within k steps. If there is a path of length at most k from node a to node b in the graph G, then node b belongs to the set $neigh_k(G, a)$. In the following definition, we focus on the neighbors within 2 steps.

$$|\cup_{j \in neigh_2(G_p,i)} D_j \cap neigh_2(G, a)| \geq |neigh_2(G_p, i)| \quad \text{(C5)}$$

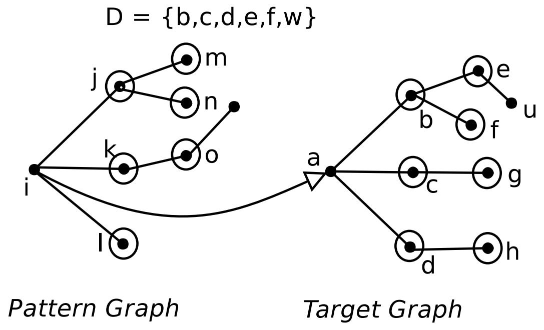

Figure 6: C5 constraint states that a can be pruned from D_i if there are more neighbors connected to i by a path of length k than candidate values connected to a by a path of length k. Here a must be pruned from D_i, since $k = 2$ and $|D \cap neigh_2(G, a)| = 5 < |neigh_2(G_p, i)| = 6$.

The new constraint C5 can be used for the SGI problem as well as for the labelled SGI problem.

4 Experiments

Experiments described here consist in finding one subgraph isomorphism of a random undirected graph with n nodes into a random undirected graph with d nodes ($n < d$), and with various average degree values. The graphs were generated with the freely available Stanford GraphBase software [19] as in [1]. This software has the advantage of being machine-independent.

The objective of our experiments is to assess the relative performances of the C2, C3 and C4 constraints. We report here results from sets of random target graphs and random pattern graphs with different average degrees. Specifically, we consider the matching of 15 nodes pattern graphs into 50 nodes target graphs (15-50 problems), and 20 nodes pattern graphs into 100 nodes target graphs (20-100 problems), which corresponds to difficult problems. For the 15-50 problems, 6 average degrees have been chosen for the target graph (from 2 to 42), and up to 13 average degrees for the target graph (from 2 to 14). A total of 62 pairs have been considered. A similar approach has been used for the 20-100 problems. For a given pair of average degrees, 5 problem instances were randomly generated. Experiments reported here represent 620 problem instances.

As we are mostly interested in the labelled SGI problem because of its biological relevance, types are randomly assigned to graph nodes. Three types were used, corresponding to the three main classes in biochemical networks (bioentities, transformation and control). In these experiments each type has the same probability to be assigned to a node. These types are used in the C4 constraint, where a constraint applies to a group of neighbors sharing the same type, provided the group has at least three nodes.

For each problem instance, three combinations of constraints were considered. Algorithm C2 uses only C1 and C2, the necessary constraints. Algorithm C3 uses C3 as well, and algorithm C4 uses constraints C1, C2, and C4. The algorithms tend to find a solution rapidly in most cases. In several cases however they may last for hours or days, illustrating the computational hardness of the problem. Following [1], we put a time limit of 5 minutes on any given run resulting in three possible outcomes. The run *succeeds* if at least one algorithm found a pattern match in the target graph. The run *fails* if at least one algorithm demonstrated that no such match exists. The result of a run is *unknown* if the time limit was exceeded for all tested algorithms. For the 15-50 problems, no *unknown* were observed.

The average running times are reported separately for the success and fail runs. When a run is a success but a given algorithm did not find a solution in the prescribed time, its reported time was set to the time limit (5 minutes). The same methodology was used for the fail runs. The experiments were performed on a Intel Xeon 2.66GHz processor.

Figure 8 details the observed results for the 15-50 problems. The left column corresponds to the *succeed* runs, and the right column to the *fail* runs. The top plots show the most effective algorithms. Each point indicates the fastest algorithm on average over the problem instances. The next figures show individual lines of the previous plots, giving the average running time for each algorithm for a particular average degree of the target graph.

We observe that for the *fail* runs, algorithm C4 outperforms C3, and also outperforms C2 in most cases. When C2 is better than C4, the overhead of C4 is small. For the *succeed* runs, algorithm C2 outperforms C3 and C4. However in the simple cases (when the average pattern degree is low) all the execution times are almost identical. For more difficult problems, C4 outperforms C3. The same observations can be made on the 20-100 problems (plots not shown here).

As pointed out in [1], "when solving non-trivial problems using a backtracking algorithm, most of the computational effort is spent proving unsolvability after making wrong assignments, in order to recover from the error". The authors show that, within a time limit, C3 proves the solvability or the unsolvability of more problems than C2. As our experiments show that C4 outperforms C3, one can argue that C4 would also outperform C2 and C3 in such tests on labelled graphs.

We also performed tests to verify Theorem 1. Larrosa and Valiente [1] proposed a benchmark composed of different types of graphs. We use here the same graphs, except that graphs are typed (using the above mentioned method). We use the same methodology reported in [1]. Searches are bounded to five minutes. A problem is said *unsolvable* if there is no subgraph isomorphism. Some problems, which exceed the time limit of 5 minutes, are said to be *not solved*.

	$C2$	$C3$	$C4$
Proved Unsolvable	957	959	967
Solution Found	219	218	223
Not Solved	49	48	35

	$Mean\ Running\ Time$
$C2$	2.58 *sec.*
$C3$	3.18 *sec.*
$C4$	3.21 *sec.*

Figure 7: Problems solved by C2, C3, C4 and the mean running time.

Figure 7 shows the results. The new C4 constraint outperforms C3 constraint by 27% in the typed case, asserting from an experimental point of view the theoretical facts explained in Theorem 1. Even if C4 has solved more problems than C3, its running time is very close from C3, verifying the theoretical claim about C4 time complexity in Section 3.6.

These results are encouraging for a useful integration of C4 into a constraint-based system.

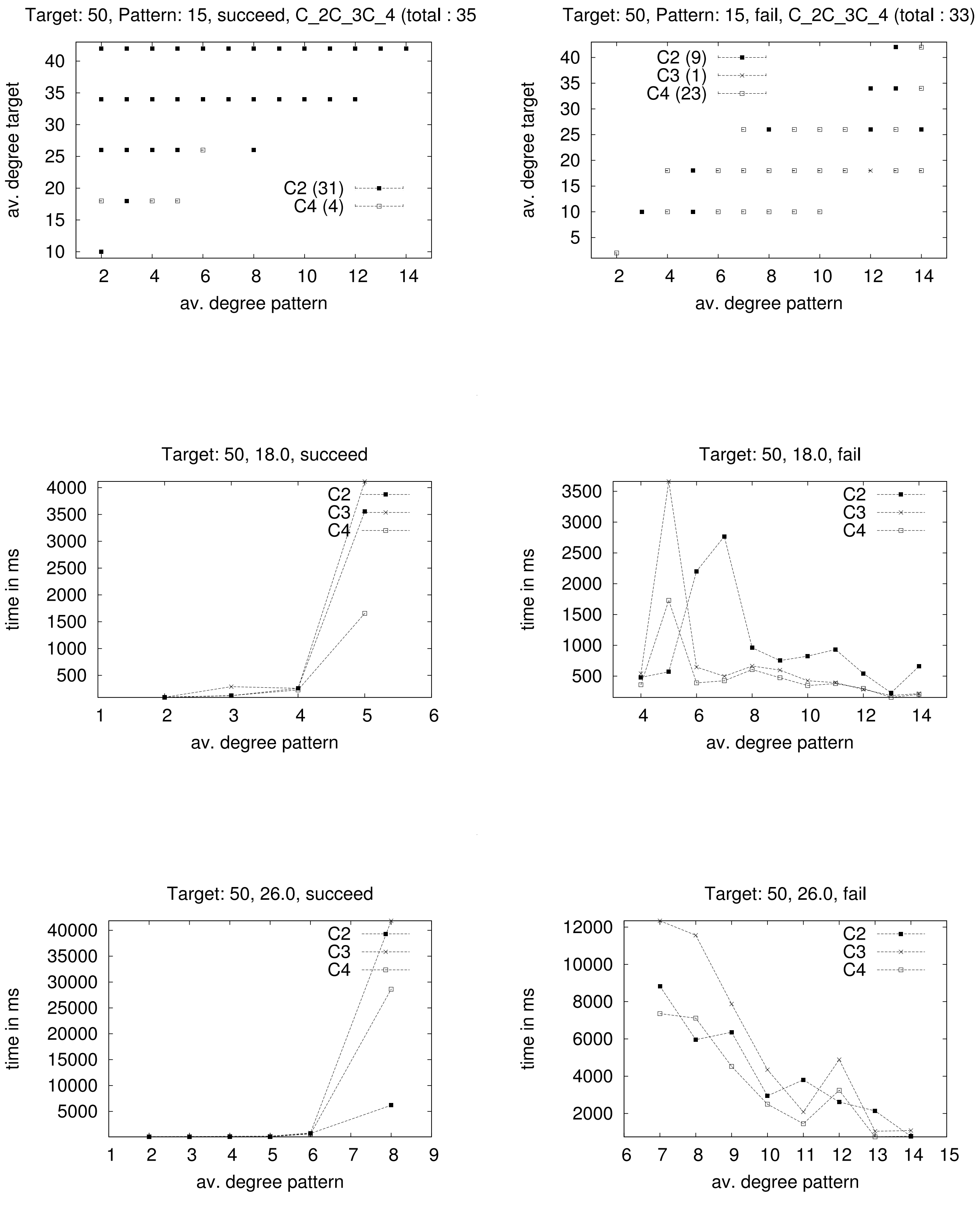

Figure 8: Plots of most effective algorithms for 15-50 problem (top), and average running time for selected target graph average degrees.

5 Conclusion and future works

Graph pattern matching is a central problem in many application fields, and may be associated with problems in bioinformatics, namely the analysis of biochemical networks. This problem can be view as a particular case of labelled subgraph isomorphism (SGI). In this paper, we focus on a CSP solution and we introduce two new constraints to solve this problem. We extend here the work of Larrosa and Valiente [1] in two ways. We consider labelled graphs, especially suited for biochemical networks, and we propose a constraint exploiting this information. Experimental results show the potential benefit of the constraints when integrated in a backtracking-based constraint system.

Future works include additional experiments, allowing a deeper comparison between various combination, especially for difficult problem instances. Our objective is to build a query language based on constraint programming techniques and allowing broad analysis of biochemical networks. An important extension will consider inexact matching, a type of subgraph isomorphism even more revelant to the analysis in biochemical networks.

References

[1] J. Larossa and G. Valiente. Constraint satisfaction algorithms for graph pattern matching. *Math. Struct. Comput. Sci.*, 12(4):403–422, 2000.

[2] J. van Helden, A. Naim, R. Mancuso, M. Eldridge, L. Wernisch, D. Gilbert, and S.J. Wodak. Representing and analyzing molecular and cellular function using the computer. *Biol. Chem.*, 381(9-10):921–35, 2000.

[3] K. Minoru, G. Susumu, K. Shuichi, and N. Akihiro. The KEGG databases at GenomeNet. *Nucleic Acids Research*, 30(1):42–46, 2002.

[4] P.D. Karp, M. Riley, M.H. Saier Jr, I.T. Paulsen, J. Collado-Vides, S.M. Paley, A. Pelligrini-Toole, C. Bonavides, and S. Gama-Castro. The EcoCyc database. *Nucleic Acids Research,*, 30(1):56–8, 2002.

[5] P.D. Karp, S. Paley, and P. Romero. The pathway tools software. *Bioinformatics*, 18(1):S225–S232, 2002.

[6] L.B.M. Ellis, B. Kyeng Hou, W. Kang, and L.P. Wackett. The University of Minesota Biocatalysis/Biodegradation Database : post-genomic data mining. *Nucleic Acids Research*, 31(1):262–265, 2003.

[7] EMP Project. Informations about EMP can be found at : http://www.empproject.com/.

[8] PathDB : a pathway database. http://www.ncgr.org/pathdb.

[9] S. Shen-Orr, R Milo, S Mangan, and U Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31:64–68, 2002.

[10] Kim Marriott and Peter J. Stuckey. *Programming with Constraints: an Introduction.* MIT Press, 1998.

[11] Y. Deville, D. Gilbert, J. van Helden, and S. Wodak. An Overview of Data Models for the Analysis of Biochemical Networks. *Briefings in Bioinformatics*, 4(3):246–259, 2003.

[12] C. V. Forst and K. Schulten. Evolution of metabolisms: a new method for the comparison of metabolic pathways. In *Proceedings of the third annual international conference on Computational molecular biology (RECOMB99)*, pages 174–181. ACM Press, 1999.

[13] R. Milo, S. Sen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[14] J. Larrosa and G. Valiente. Graph pattern matching using constraint satisfaction. In *Proc. Joint APPLIGRAPH/GETGRATS Worksh. Graph Transformation Systems, to appear in LNCS series*, pages 189–196, 2000.

[15] Sébastien Sorlin and Christine Solnon. A global constraint for graph isomorphism problems. In *CP-AI-OR 2004*. Springer Verlag, 2000.

[16] M. Rudolf. Utilizing Constraint Satisfaction Techniques for Efficient Graph Pattern Matching. In *6th International Workshop on Theory and Application of Graph Transformations 98, LNCS series vol.1764:238-251.*

[17] J.R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23, Issue 1:31–42, 1976.

[18] J.C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. 12th Conf. American Assoc. Artificial Intelligence*, volume 1, pages 362–367, 1994.

[19] Knuth D.E. *The Standford GraphBase: A Platform for Combinatorial Computing.* ACM Press, 1993.