Constrained metabolic network analysis: discovering pathways using CP(Graph)

Gregoire Dooms, Yves Deville, Pierre Dupont

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

1 Introduction

Biochemical networks – networks composed of the building blocks of the cell and their interactions are qualitative descriptions of the working of the cell. Such networks can be modeled as graphs. Metabolic networks are typical examples of such networks. They are composed of biochemical entities participating to reactions as substrates or products. Such a network can be modeled as a bipartite digraph which nodes are the biochemical entities and reactions and edges are the substrate or product link between an entity and a reaction.

Pathways are specific subsets of a metabolic network which were identified as functional processes of cells[1]. As these pathways are known to be working processes of the cell, they can be used to study the metabolic network. One type of metabolic network analysis consists in finding simple paths in the metabolic graph[2–6]. Here we focus on such analysis to discover pathways from a set of their reactions. A potential application is the explanation of DNA chip experiments using a CSP able to solve pathway discovery problems.

The study of the metabolic network is constantly evolving and most of the problems are solved with dedicated algorithms. This dedicated approach has the benefit of yielding very efficient programs to solve network analysis problems. This approach however has the drawback that it cannot be easily adapted to solve other problems or easily combined to solve combinations of various analyses.

In [7, 8], we proposed to use constraint programming to solve constrained path finding problems in metabolic networks. This declarative paradigm allows to easily adapt programs or combine different programs. In order to provide a high level modeling language and as the data and results are graphs, we defined a graph computation domain for constraint programming [9].

The two following sections are devoted to a short introduction to CP(Graph) and a description of its application to metabolic network analysis.

2 The CP(Graph) Framework

CP(Graph) [9] features graph variables, node and arc variables, and set variables for nodes and arcs. They are depicted in figure 2. That figure shows the notation

used for constants and variables in this paper. It also shows that a graph variable has an inherent constraints linking its arcs to its nodes.

Type	Representation	Constraint	Constants	Variables
Integer	$0, 1, 2, \ldots$		$i_0, i_1, \ldots$	$I_0, I_1, \ldots$
Node	$0, 1, 2, \ldots$		$n_0, n_1, \ldots$	$N_0, N_1, \ldots$
Arc	$(0, 1), (2, 4), \ldots$		$a_0, a_1, \ldots$	$A_0, A_1, \ldots$
Finite set	$\{0, 1, 2\}, \{3, 5\} \ldots$		$s_0, s_1, \ldots$	$S_0, S_1, \ldots$
Finite set of nodes	$\{0, 1, 2\}, \{3, 5\} \ldots$		$sn_0, sn_1, \ldots$	$SN_0, SN_1, \ldots$
Finite set of arcs	$\{(0, 3), (1, 2)\}, \ldots$		$sa_0, sa_1, \ldots$	$SA_0, SA_1, \ldots$
Graph	(SN, SA) SN a set of nodes SA a set of arcs	$SA \subseteq SN \times SN$	$g_0, g_1, \ldots$	$G_0, G_1, \ldots$
Weight functions	$\mathcal{N} \cup \mathcal{A} \rightarrow \mathbb{N}$		$w_0, w_1, \ldots$	–

Three kernel constraints suffice to express all MS-definable properties of graphs [10] as constraints:

$Arcs(G, SA)$ SA is the set of arcs of G.

$Nodes(G, SN)$ SN is the set of nodes of G.

$ArcNode(A, N_1, N_2)$ The arc variable A is an arc from node N_1 to node N_2. This relation does not take a graph variable into account as every arc and node has a unique identifier in the system. If A is determined, this constraint is a simple accessor to the tail and head of the arc A and similarly if both nodes are determined.

These constraints enable to express more complicated constraints such as $Reachables(G, N, SN)$ which states SN must be the set of nodes reachable from N in G (the transitive closure of the adjacency relation in G) or $Path(G, N_1, N_2)$ which holds if G is a path from node N_1 to node N_2. While these constraints can be expressed using kernel constraints, they are more efficient when implemented using dedicated global propagators (either for their consistency level or algorithmic complexity).

CP(Graph) enables to express constrained subgraph extraction problems such as the TSP or the equicut problem:

- Finding the TSP in graph g with weights w: minimize $Weight(G, w)$ s.t.

$$SubGraph(G, g) \wedge Cycle(G) \wedge Nodes(G) = Nodes(g)$$

- Graph partitioning problem: equicut of a graph g of even order: minimize $\#\left(Arcs(g) \setminus \left(Arcs(G_1) \cup Arcs(G_2)\right)\right)$ s.t.

$$SubGraph(G_1, g) \wedge SubGraph(G_2, g) \wedge Nodes(G_1) \cup Nodes(G_2) = Nodes(g) \wedge$$
$$\#Nodes(G_1) = \#Nodes(G_2) = \frac{1}{2}\#Nodes(g)$$

Related work: constraint programming was used to solve constrained path finding problems in [11] using a finite domain model (successor variables). Path variables were introduced in [12] to solve constrained path problems as part of a network design problem. A cost-based filtering technique for constrained shortest path problems was described in [13] and a global path constraint was presented in [14]. Graphs also play an important role in constraint programming in the design of propagators for global constraints: graph algorithms are used [15] and global constraints were modeled as networks of similar constraints [16, 17]. The problem addressed in this paper is similar to queries addressed in the model checking approach of BIOCHAM [18]. While queries about reachability can be handled by both systems, it seems that some queries such as optimization problems can be expressed using CP(Graph) but not with CTL logic.

As it is, CP(Graph) allows constrained subgraph extraction. However, we are also working on constrained approximate subgraph isomorphism [19] by extending CP(Graph) with map variables [20, 21].

3 Metabolic network analysis experiments

The general kind of analysis we wish to perform with CP(Graph) is pathway discovery by constrained subgraph extraction. One potential application of this type of analysis lies in assisted explanation of DNA chip experiments. In such experiments, the behavior of a sane cell and a mutant are compared in a given context (the substrate on which they are living or more generally their environment). This comparison is done at different times by extracting and amplifying the expressed RNA in the nucleus of the cells (this kills the cell). This RNA is then put on a DNA chip: an array of representative sequences of bases for a set of genes. The RNA binds to the chip in the locations which are specific to it. That array is then scanned to see the level of expression of each RNA strand. That RNA encodes for given enzymes which catalyze given reactions. Hence, the level of RNA can be translated into the information of which reactions were active in the cell at the time its RNA was extracted. Given this set of reactions, biologists would like to know which processes where active in the cell. If a CSP allows to recover known processes from sets of reactions, it could be adequate to discover the real processes given other sets of reactions. Hence, such a CSP could approximate the real processes at work in a cell from DNA chip results. These computational results could then be used to further guide other concrete experiments which are more expensive.

The current experiments focus on linear pathways by doing constrained shortest path finding. Future work comprise increasingly better characterizations of the CSP (ie. more constraints which increase the rate of correct recovery) and a formulation of a CSP for pathways which contains branchings or cycles.

3.1 Prototype of CP(Graph) Implemented in Oz/Mozart and Gecode

We implemented a prototype of CP(Graph) in the Oz/Mozart [22] constraint programming framework. A set of nodes and a set of arcs are used to implement each graph variable. We also implemented this prototype over the Gecode generic constraint development environment [23]. In these prototypes, we implemented, among others, the global path propagator of [14].

3.2 Constrained Shortest Path Finding

As about half of the known pathways are simple paths [24], one type of experiment consists in trying to find these pathways by using constrained path finding in a directed graph (knowing a few nodes of the path). In [25], experiments were done first with a dedicated shortest path finding algorithm. Then some nodes (the pool metabolites, molecules like ATP or H_2O which are ubiquitous and take part in many reactions) were removed from the graph and the results compared with the previous ones. Some pathways, such as glycolysis, however use some of these metabolites as intermediates. In order to decrease the likelihood of selecting these nodes while still allowing to select them, all nodes were assigned a weight proportional to their degree. As pool metabolites have a very high degree, they are much less likely to be selected in the shortest paths.

Our experiment consists in redoing the former experiment with an additional constraint of mutual exclusion for certain pairs of reactions. These pairs are reverse reactions (the reaction from substrates to products and the one from products to substrates). Most of the time, these reactions are observed in a single direction in each species. Hence we wish to exclude paths containing both in our experiment. Such additional constraints like mutual exclusion are not always easily integrated in dedicated algorithms [25]. In CP(Graph) it just consists in posting a few additional constraints. If $n_1, ..., n_m$ are the included reactions and $(r_{i1}, r_{i2}), 0 < i \leq t$ the mutually exclusive pairs of nodes, the program looks like: miminize $Weight(G, w)$ s.t.

$$SubGraph(G, g) \wedge Path(G, n_1, n_m) \wedge \forall 0 < i \leq m : n_i \in Nodes(G) \wedge \\ \forall i \in [0, t] : r_{i1} \notin Nodes(G) \vee r_{i2} \notin Nodes(G)$$

In our experimental setting we first extract a subgraph of the original metabolic bipartite digraph by incrementally growing a fringe starting by the included nodes. Then, given a subset of the reactions of a reference pathway, we try to find the shortest constrained path in that subgraph. The first process of extraction of a subgraph of interest is done for efficiency reasons as the original graph is too big to be handled by the CSP (it contains around 16.000 nodes). The results are presented in Table 1, it shows the increase of running time, memory usage and size of the search tree with respect to the size of the graph for the extraction of three illustrative linear pathways shown in [25]. All reactions are mandatory in the first experiment. The results of another experiment where one reaction

out of two successive reactions in the given pathway is included in the set of mandatory nodes, is presented in Table 2.

The running time increases greatly with the size of the graphs. The program can however be stated in a few lines and first results obtained the same day the experiment is designed. The limitation on the input graph size does not guarantee to get the optimal shortest path in the original graph. This should however not be a major problem as biologists are most of the time interested in a particular portion of the metabolic graph. The rapidity of expression and resolution of such a NP(Hard) [13] problem outweights this size limitation.

Future work comprise two main aspects. The first is being able to cope with bigger graphs. We could design more efficient heuristics for labelling. The use of a cost-based filtering method could prune the size of the graph given an upper bound of the cost. Such an upperbound is available as soon as a first solution is found. Another solution would be to use an a-priori upper bound of the cost which would need to be increased or removed if no solution is found. The second aspect of our future work consists in finding which additional constraints are needed to recover known pathways as it was shown in [25] that non-constrained shortest paths are not able to recover all of them.

We are currently working on a extention of this approach to discovering pathways containing branchings or cycles. A first formulation we wish to test is the following: find the smallest graph containing all the seeds such that there is a seed from which all other nodes are reachable.

Given sn_s a set of nodes (seeds), minimize $Weight(G, w)$ subject to:

$$N \in sn_s, sn_s \subseteq Nodes(G) \wedge Reachable(G, N, Nodes(G))$$

Glycolysis (m=8)					Heme (m=8)					Lysine (m=9)				
Size	t	Time	Nodes	Mem	Size	t	Time	Nodes	Mem	Size	t	Time	Nodes	Mem
50	12	0.2	20	2097	50	22	0.2	32	2097	50	18	0.2	38	2097
100	28	2.5	224	2097	100	36	0.3	22	2097	100	40	4.7	652	2097
150	48	41.7	1848	4194	150	62	1.0	28	2097	150	56	264.3	12524	15204
200	80	55.0	1172	5242	200	88	398.8	7988	18874	200	70	-	-	-
250	84	127.6	4496	8912	250	118	173.3	2126	9961	250	96	-	-	-
300	118	2174.4	16982	60817	300	146	1520.2	21756	72876	300	96	-	-	-

Table 1. Comparison of the running time [s], number of nodes in the search tree and memory usage [kb], for the 3 pathways and for increasing original graph sizes. m is the number of node inclusion constraints and t the number of mutual exclusion constraints.

4 Conclusion

The problem of discovering the processes at work in a cell given a set of reactions can be modeled as a constrained subgraph extraction problem. But the formal

Glycolysis (m=5)					Heme (m=5)					Lysine (m=5)				
Size	t	Time	Nodes	Mem	Size	t	Time	Nodes	Mem	Size	t	Time	Nodes	Mem
50	12	0.2	22	2097	50	22	0.3	44	2097	50	18	0.1	16	2097
100	28	2.5	230	2097	100	36	0.9	78	2097	100	40	13.3	1292	3145
150	48	79.3	5538	6815	150	62	7.3	144	3145	150	56	260.4	8642	14155
200	80	39.9	1198	5767	200	88	57.3	950	5242	200	70	4330.5	74550	192937
250	84	323.6	5428	14680	250	118	36.0	350	8388	250	96	-	-	-
300	118	10470.8	94988	296747	300	146	-	-	-	300	96	-	-	-

Table 2. Same experiment as in Table1, but with one reaction node included every two ($m = 5$ instead of 8 or 9).

expression of this problem is not yet clear. In order to refine it, we first evaluate the various problem formulations on recovering known pathways. We hope the best solution to the reduced problem will be suitable to solve the more general problem of discovering real pathways.

Such an approach is difficult to achieve using dedicated algorithms as new algorithms must be designed each time a new problem formulation is to be evaluated [25]. A declarative approach is more practical as it just requires the formulation of the problem in a declarative language. Constraint programming is a declarative framework which has been successfully used to solve hard problems. CP(Graph) is a constraint programming computation domain suitable to express constrained subgraph extraction problems. It provides a higher level interface to define such problems and should be easier to use by bio-informaticians than classical finite domain or finite set computation domains.

This first application of CP(Graph) on constrained shortest path problems in metabolic networks shows that it is appropriate to express and solve these metabolic network extraction problems. We shall continue this work by moving to non linear pathways and trying to cope with bigger graphs.

Our future work includes an extention to pathways which contain cycles and branchings, the handling of larger graphs, and the experimental characterization of the formalization of the problem of recovering known pathways from the metabolic graph using CP(Graph).

References

1. Minoru, K., Susumu, G., Shuichi, K., Akihiro, N.: The KEGG databases at GenomeNet. Nucleic Acids Research **30(1)** (2002) 42–46
2. Jeong, H., Tombor, B., Albert, R., Oltvai, Z., Barabasi, A.: The large-scale organization of metabolic networks. Nature **406** (2000) 651–654
3. Kim, B., Yoon, C., Han, S., Jeong, H.: Path finding in scale-free networks. Phys Rev E Stat Nonlin Soft Matter Phys **65** (2002) 27101–27104
4. R. Alves, R.A. Chaleil, M.S.: Evolution of enzymes in metabolism: a network perspective. Journal of Molecular Biology **320** (2002) 751–770
5. van Helden, J., Naim, A., Mancuso, R., Eldridge, M., Wernisch, L., Gilbert, D., Wodak, S.: Representing and analyzing molecular and cellular function using the computer. Journal of Biological Chemistry **381(9-10)** (2000) 921–35

6. van Helden, J., Wernisch, L., Gilbert, D., Wodak, S.: Graph-based analysis of metabolic networks. In: Bioinformatics and genome analysis. Springer-Verlag (2002) 245–274
7. Dooms, G., Deville, Y., Dupont, P.: Recherche de chemins contraints dans les réseaux biochimiques. In Mesnard, F., ed.: Programmation en logique avec contraintes, actes des JFPLC 2004, Hermes Science (June 2004) 109–128
8. Dooms, G., Deville, Y., Dupont, P.: Constrained path finding in biochemical networks. In: Proceedings of JOBIM 2004. (2004) JO–40
9. Grégoire Dooms, Yves Deville, Pierre Dupont: CP(Graph): Introducing a Graph Computation Domain for Constraint Programming. In: Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming. Number LNCS ...., Springer-Verlag (2005)
10. Courcelle, B.: The monadic second-order logic of graphs. i. recognizable sets of finite graphs. Inf. Comput. **85** (1990) 12–75
11. Pesant, G., Gendreau, M., Potvin, J., Rousseau, J.: An exact constraint logic programming algorithm for the travelling salesman with time windows. Transp. Science **32** (1996) 12–29
12. Lepape, C., Perron, L., Regin, J.C., Shaw, P.: A robust and parallel solving of a network design problem. In: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming. Volume LNCS 2470. (2002) 633–648
13. Sellmann, M.: Cost-based filtering for shorter path constraints. In: Proceedings of the 9th International Conference on Principles and Practise of Constraint Programming (CP). Volume LNCS 2833., Springer-Verlag (2003) 694–708
14. Cambazard, H., Bourreau, E.: Conception d'une contrainte globale de chemin. In: 10e Journ. nat. sur la résolution pratique de problèmes NP-complets (JNPC'04). (2004) 107–121
15. Régin, J.: A filtering algorithm for constraints of difference in CSPs. In: Proc. 12th Conf. American Assoc. Artificial Intelligence. Volume 1. (1994) 362–367
16. Beldiceanu, N.: Global constraints as graph properties on structured network of elementary constraints of the same type. Technical Report T2000/01, SICS (2000)
17. Beldiceanu, N.: Global constraint catalog. Technical Report T2005-08, SICS (2005)
18. Fages, F., Soliman, S., Chabrier-Rivier, N.: Modelling and querying interaction networks in the biochemical abstract machine biocham. Journal of Biological Physics and Chemistry **4** (2004) 64–73
19. Yves Deville, Grégoire Dooms, Stéphane Zampelli, Pierre Dupont: CP(Graph+Map): Constrained Approximate Subgraph Matching. In: INGI Research report 2005-07. (2005)
20. Gervet, C.: Interval propagation to reason about sets: Definition and implementation of a practical language. CONSTRAINTS Journal **1(3)** (1997) 191–244
21. Hnich, B.: Function Variables for Constraint Programming. PhD thesis, SICS, Sweden (2003)
22. Mozart Consortium: The mozart programming system version 1.2.5 (December 2002) `http://www.mozart-oz.org/`.
23. Gecode: Generic Constraint Development (2005) `http://www.gecode.org/`.
24. Lemer, C., Antezana, E., Couche, F., Fays, F., Santolaria, X., Janky, R., Deville, Y., Richelle, J., Wodak, S.J.: The aMAZE lightbench: a web interface to a relational database of cellular processes. Nucleic Acids Research **32** (2004) D443–D448
25. Croes, D.: Recherche de chemins dans le réseau métabolique et mesure de la distance métabolique entre enzymes. PhD thesis, ULB, Brussels (2005) (in preparation).