An ASP application in integrative biology: identification of functional gene units

Philippe Bordron[1,2], Damien Eveillard[4], Alejandro Maass[2,3], Anne Siegel[6,5], and Sven Thiele[1,5,6]

[1] Inria-Ciric, Rosario Norte 555, Of. 703, Santiago, Chile
[2] Center of Mathematical Modeling and Center for Genome Regulqtion, Universidad de Chile, Av. Blanco Encalada 2120, Santiago, Chile
[3] Department of Mathematical Engineering, Universidad de Chile, Av. Blanco Encalada 2120, Santiago, Chile
[4] ComBi, lina, Université de Nantes, cnrs umr 6241, 2 rue de la Houssinière, 44300 Nantes, France
[5] Inria, Centre Rennes-Bretagne-Atlantique, Projet Dyliss, Campus de Beaulieu, 35042 Rennes cedex, France
[6] Cnrs, umr 6074 irisa, Campus de Beaulieu, 35042 Rennes, France

Abstract. Integrating heterogeneous knowledge is necessary to elucidate the regulations in biological systems. In particular, such an integration is widely used to identify functional units, that are sets of genes that can be triggered by the same external stimuli, as biological stresses, and that are linked to similar responses of the system. Although several models and algorithms shown great success for detecting functional units on well-known biological species, they fail in identifying them when applied to more exotic species, such as extremophiles, that are by nature unrefined. Indeed, approved methods on unrefined models suffer from an explosion in the number of solutions for functional units, that are merely combinatorial variations of the same set of genes. This paper overcomes this crucial limitation by introducing the concept of "genome segments". As a natural extension of recent studies, we rely on the declarative modeling power of answer set programming (ASP) to encode the identification of shortest genome segments (SGS). This study shows, via experimental evidences, that SGS is a new model of functional units with a predictive power that is comparable to existing methods. We also demonstrate that, contrary to existing methods, SGS are stable in (i) computational time and (ii) ability to predict functional units when one deteriorates the biological knowledge, which simulates cases that occur for exotic species.

1 Introduction

In biological systems one distinguishes several layers of information. One of them represents the set of (bio)chemical reactions that occur within the system. These reactions are called metabolic reactions and form chains of metabolic pathways (i.e. products of reactions are substrates of other reactions) of the whole metabolism. The metabolism is roughly controlled by genes. Indeed, a major

part of these reactions is catalyzed by enzymes that are encoded by genes. Thus, understanding the link between gene regulation and metabolism is of great interest in biological research. However, despite the interest and efforts, identifying these links remains a difficult task. Recent biological evidence shows that microbial genes present a specific organization. They are grouped together on the DNA strand when they are functionally related [11]. These groups of genes, so called functional units, became the main target of functional biology.

Following this topological assumption several bioinformatics approaches have been proposed to explain functional units [3,13]. All rely on genome scale models integrating genomic information (in particular, genes organization) with metabolic networks. Among these techniques in [2] is proved that shortest paths – namely, *wrr-paths* – in the so called *integrated model* correspond to functional gene units (i.e. called *metabolic operons* in the biological literature). Although shortest wrr-paths show its efficiency to recover functional units when computed in well-curated bacteria like *Escherichia coli*, one observes an explosion in the number of solutions when considering more exotic or less characterized bacteria. This explosion is somehow artifactual, since many wrr-paths in these species are merely combinatorial variations of the same set of genes and can be biologically merged into a single functional gene unit.

In this paper, we propose a modification of the concept of wrr-path in order to overcome the weakness due to the aforementioned combinatorial explosion in wrr-path computation in bad characterized bacteria. We present the concept of *genome segments* which is computable in reasonable time allowing the identification of meaningful functional units on this class of bacteria. Genome segments and wrr-paths are related, they are obtained within the same integrated framework but minimizing different metrics.

To compute functional gene units we formulate the shortest genome segment (SGS) problem by means of Answer Set Programming (ASP) [1] instead of a dedicated algorithm on a graph. This allows a flexible encoding that can be easily adjusted to test different metrics while still being computational efficient. ASP is a declarative problem solving paradigm from the field of logic programming and knowledge representation, that offers a rich modeling language [7] along with highly efficient inference engines [9] based on Boolean constraint solving technology. In the fields of integrative biology ASP applications include the reconstruction of metabolic networks [12], modeling the dynamics of regulatory networks [6], inferring functional dependencies from time-series data [5], and integrating gene expression with pathway information [10].

Finally, we solve the SGS problem on a concrete biological system by taking advantage of ASP's optimization techniques for finding minimal solutions [7,8]. Via further experimental validation, we pinpoint that SGS are a suitable model of functional units with an accurate predictive power. We also demonstrate that the identification of SGS is stable in both computational time and the ability to predict functional units when one deteriorates the biological knowledge, which simulates cases that occur in more exotic species.

2 An integrated model: identifying functional gene units

Metabolic compounds and reactions. The metabolism of a given bacterial system is defined by the set C of biological compounds and the set R of metabolic reactions that take place in the system. Each reaction r in R describes the transformation of compounds in C into others, also in C. Consequently, a metabolic compound can be consumed or produced by a metabolic reaction. The compound takes the role of a substrate or a product of the reaction. We define the maps $\mathit{consume} : R \to \mathcal{P}(C)$ and $\mathit{produce} : R \to \mathcal{P}(C)$ (where $\mathcal{P}(C)$ contains subsets of C) to describe the set of substrates and the set of products of a given reaction respectively. The fact that the products of one reaction can be used as substrates by other reactions allows to connect reactions into complex chemical pathways. All possible reactions are usually represented as a metabolic network, which is a graph representation of the metabolism. We use the *reaction graph* representation (R, E) where vertices in R correspond to the set of possible reactions and edges are $E = \{(x, y) \mid x, y \in R, \mathit{produce}(x) \cap \mathit{consume}(y) \neq \emptyset, x \neq y\}$.

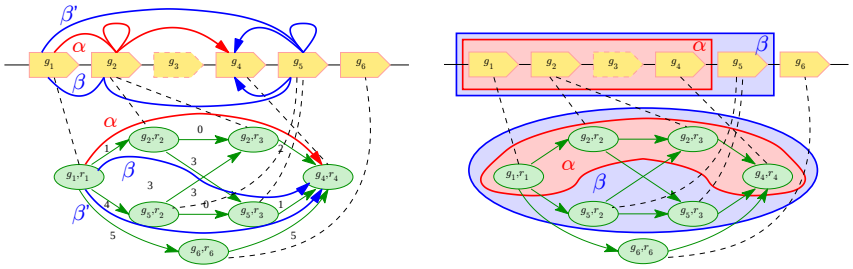
Genes, enzymes, translation and catalysis of reactions. Bacterial genomes are often constituted by only one circular chromosome. Formally, such a genome can be represented as an ordered sequence $\mathcal{G}$ of genes $g_1, \ldots, g_n$ where n is the number of genes in $\mathcal{G}$. The successor and predecessor of a gene $g \in \mathcal{G}$ is naturally defined when looking the genome as a circular word. We denote by G the set of distinct genes that appear in $\mathcal{G}$.

When genes in $\mathcal{G}$ are transcribed and then translated, some proteins catalyzing specific metabolic reactions are produced. The proteins having a catalytic function are called enzymes. Each reaction in R can be catalyzed by one or many enzymes. We define then the map $\mathit{catalyze} : G \to \mathcal{P}(R)$ to describe the set of reactions that a gene can catalyze via the associated enzyme.

The integrated gene-reaction graph. A model that put together all previously described heterogeneous biological knowledge into a weighted directed graph representation is the *integrated gene-reaction graph* [2]. For a genome $\mathcal{G}$, a set of reactions R and a metric between genes $w : G \times G \mapsto \mathbb{R}^+$, we define the integrated graph (V, A) where $V = \{(g, r) | g \in G, r \in R, r \in \mathit{catalyze}(g)\}$ is the set of vertices and $A = \{((g, r), (g', r')) | (g, r), (g', r') \in V, \mathit{produce}(r) \cap \mathit{consume}(r') \neq \emptyset, r \neq r'\}$ are the edges. Each edge $a = ((g, r), (g', r'))$ of A is weighted by $w(g, g')$. The sum of weights along a path is the path weight. Note that this graph uses the reaction graph as a support which is enriched with genomic information. Using this integrated model allows us to investigate gene regulatory behavior based on the topology of the underlying metabolic reaction network.

Identifying functional gene units. A concept that is used to identify functional gene units is called *without reaction repetition* path (wrr-path) in the integrated gene-reaction graph. Given two reactions r and r' in R. A path

$p = v_1 \dots v_l$ in (V, A) is a wrr-path from r to r' if $v_1 = (g_1, r)$, $v_l = (g_l, r')$ and for all two vertices $v_i = (g_i, r_i)$, $v_j = (g_j, r_j)$ in p with $1 \leq i < j \leq l$ it holds $r_i \neq r_j$. We observe that several wrr-paths can have the same weight. The set of genes that are involved in a wrr-paths can be interpreted as a hint of a functional gene unit. Figure 1(a) illustrates wrr-paths.



(a) wrr-paths (b) genome segments

Fig. 1. Difference between wrr-paths and genome segments. On both figures flat arrows represent genes on the genome, and the graph under the genome is the integrated gene-reaction graph. In (a), the shortest wrr-path from r_1 to r_4 is α, and the two second shortest paths are β or β'. In addition, we project in the obvious way those paths on the genome. In (b), the shortest genome segment from r_1 to r_4 is α, whereas the second best is the segment β. The projection of these segments in the integrated gene-reaction graph is depicted.

It has been shown in the well studied organism *Escherichia coli* that the concept of wrr-path is highly useful to determine functional metabolic-gene units [2]. Indeed, besides the fact that this concept was not designed to find operons, the resulting gene units matched up to 45% with known operons which is considered to be a top range technique for identifying operons [4]. Moreover, in [2], only the information of *Escherichia coli* is required, whereas dedicated predictors rely on organisms comparison or/and learning methods [4]. In the case of exotic organisms, the use of information about other organisms may be hurtful.

While computing shortest (in relation to the weight) wrr-paths in a well studied organism like *E. coli* is not problematic, it has certain shortcomings when applied to less refined models. We observed that in unrefined models the number of equally short wrr-paths increases. This is mostly due to the fact that usually gene annotations are less specific. One gene is then often mapped to a multitude of metabolic reactions and the direction of most reactions is unknown and is set to be reversible by default. Thus, the integrated gene-reaction graph of a less well defined model contains naturally more wrr-paths. Most of these paths involve the same genes and therefore belong to the same gene units. These equally short paths are merely combinatoric variants of few gene-reaction pairs. The huge number of alternative paths makes its computation unfeasible on poorly defined models.

3 Genome segments, an alternative approach to identify functional gene units

We present an alternative approach to identify functional gene units based on gene organization instead of wrr-paths. This allows us to avoid the huge combinatorial problem described above when considering integrated gene-reaction graphs.

Definition 1 (Genome segment). *Given a circular sequence of genes* $\mathcal{G} = g_1 \dots g_n$*, we define a genome segment of* $\mathcal{G}$ *as a sequence* $\sigma = g_i \dots g_j$*, where* $1 \leq i \leq j \leq n$*, or a sequence* $\sigma = g_i \dots g_n g_1 \dots g_j$*, where* $0 \leq j \leq i \leq n$*.*

Definition 2 (Induced subgraph). *Let* $\mathcal{G}$ *be a circular sequence of genes and* (R, E) *the reaction graph associated to such sequence. Given a genome segment* σ *of* $\mathcal{G}$*, the induced subgraph is* (R_σ, E_σ) *where* $R_\sigma = \{r \in R \,|\, \exists\ g \in \sigma,\ r \in catalyze(g)\}$ *and* $E_\sigma = \{(r, r') \in E \,|\, \exists\ g, g' \in \sigma,\ r \in catalyze(g),\ r' \in catalyze(g')\}$*.*

The induced subgraph reflects the regulatory influence of the genes in the segment, their corresponding enzymes products and the catalyzed reactions relation of these enzymes. Figure 1(b) illustrates genome segments and the induced subgraphs.

To each wrr-path in an integrated gene-reaction graph one associates the shortest genome segment containing the genes in the path. Thus, two wrr-paths involving the same set of genes are associated to the same genome segment but they may have different weights. In unrefined models, one observes that the number of wrr-paths is huge and that a genome segment can be associated to a big amount of wrr-paths.

Thus, to reduce complexity and in analogy to the wrr-paths problem, where one searches the shortest wrr-paths in terms of their weight, we define the Shortest Genome Segment (SGS) problem as follows.

Definition 3 (Shortest Genome Segment Problem). *The SGS problem receives the following data: a genome sequence* $\mathcal{G}$*, a reaction graph* (R, E)*, a map* $catalyze : G \to \mathcal{P}(R)$ *and two reactions* $r, r' \in R$*.*

A solution to a SGS problem is the shortest genome segment σ *of* $\mathcal{G}$ *such that the induced subgraph* (R_σ, E_σ) *contains a path from* r *to* r'*.*

A solution to the SGS problem points to a set of genes that take an active part in the metabolic regulation and form an active gene unit.

Definition 4 (Active gene units). *Let segment* σ *be a solution to the SGS problem with data* $(\mathcal{G}, (R, E), catalyze, r, r')$*. A gene* g *is part of an active gene unit* $AU(\sigma)$ *if and only if* $g \in \sigma$ *and there exists a path* p *in* (R_σ, E_σ) *from* r *to* r' *such that reactions catalyzed by* g*,* $catalyze(g)$*, intersect the path* p*.*

These active gene units can be biologically interpreted as functional gene units.

To explore the space of further suboptimal genome segments, we adjust the SGS problem by adding constraints on the minimal length of a segment.

Definition 5 (SGS problem with minimal length). *An SGS problem with minimal length needs the following input data: a genome sequence $\mathcal{G}$, a reaction graph (R, E), a map catalyze : $G \to \mathcal{P}(R)$, two reactions $r, r' \in R$, and a minimal length min.*

A solution to a SGS problem with minimal length is the shortest genome segment σ of $\mathcal{G}$ with length $l \geq min$, such that the induced subgraph (R_σ, E_σ) contains a path from r to r' and there exists no segment σ' of length $l' < min$ with $AU(\sigma) \subseteq AU(\sigma')$.

Observe that the original SGS problem is a special case of this problem where $min = 1$. It is clear that when min is too large this problem has no solution. On the other hand condition $AU(\sigma) \subseteq AU(\sigma')$ looks to avoid artificial extension of segments with no active genes.

In our application, the identification of functional gene units in exotic organisms, we are especially interested in computing optimal solutions as well as solutions that are close to the optimum. Therefore, we need to solve the following sub-tasks:

- **Problem 1**. Compute the minimal length l of a segment σ with $l \geq min$, such that the induced subgraph contains a path from r to r' and there exists no segment σ' of length $l' < l$ with $AU(\sigma) \subseteq AU(\sigma')$.
- **Problem 2**. Enumerate all segments σ of a given length l such that the induced subgraph contains a path from r to r' and there exists no segment σ' of length $l' < l$ with $AU(\sigma) \subseteq AU(\sigma')$.

In the following, we will show on a real world application that genome segments are a good alternative for the computation of functional gene units.

4 ASP encoding

We now present our ASP encoding of the SGS problem as defined in Section 3. Additionally the encoding will use an upper bound max representing our knowledge on the maximal length of a genome segment. In some cases can be the length of the genome but typically it is no longer than a few hundreds.

Therefore, an instance of the SGS problem consists of seven components, the sequence of genes $\mathcal{G}$, the reaction graph (R, E), the function $catalyze$ which maps genes to metabolic reactions, metabolic reactions s and e, which represent the start and end of the desired pathway, a lower bound min on the length of the desired genome segment, as well as an upper bound max on the length of the desired genome segment. For our ASP solution, we represent such a problem instance as a set of ground logic facts $\mathcal{F}(\mathcal{G}, (R, E), catalyze, s, e, min, max)$

defined as follows:

$$\begin{aligned}\mathcal{F}(\mathcal{G}, (R, E), catalyze, s, e, min, max) =& \{edge(u, v) \mid (u, v) \in E\} \\ & \cup \{gene(g) \mid g \in \mathcal{G}\} \\ & \cup \{cat(g, r) \mid g \in \mathcal{G}, r \in R, r \in catalyze(g)\} \\ & \cup \{start(s),\ end(e)\} \\ & \cup \{const\ min,\ max\}.\end{aligned} \tag{1}$$

Such a problem instance can then be combined with the logic program in Listing 1.1 to solve the SGS problem.

Listing 1.1. sgs.lp: ASP encoding of shortest genome segments.

```
1  sgene(G) :- start(R),cat(G,R).
2  egene(G) :- end(R),cat(G,R).
3
4  pse(F,L) :- gene(F;L), F<L,
5              (L-F)+1 <= max, (L-F)+1 >= min,
6              sgene(S), F > S-max, L < S+max,
7              S >=F, S <= L,
8              egene(E), F > E-max, L < E+max,
9              E >=F, E <= L.
10
11 1{ se(F,L) : pse(F,L) }1.
12
13 on_segment(G) :- se(F,L), gene(G), G>=F, G<=L.
14
15 aedge(X,Y) :- edge(X,Y), cat(G1,X), cat(G2,Y),
16               on_segment(G1;G2).
17
18 from_start(X) :- start(X), on_segment(G), cat(G,X).
19 from_start(Y) :- from_start(X), aedge(X,Y).
20 :- not from_start(X), end(X).
21
22 to_end(Y) :- end(Y), on_segment(G), cat(G,Y).
23 to_end(X) :- to_end(Y), aedge(X,Y), cat(G,X).
24
25 aunit(G) :- on_segment(G), cat(G,X), from_start(X),to_end(X).
26 :- se(F,L), not aunit(F).
27 :- se(F,L), not aunit(L).
28
29 length((L-F)+1) :- F<=L, se(F,L).
30
31 :- length(X), X < min.
32
33 #minimize [ length(L) = L ].
```

This logic program represents a simplified ASP formulation of the SGS problem. We remark that the rules in lines (4-9, 13 and 29) are only handling the case of

linear genomes. For the sake of simplicity we omit here the definitions for the circular genome case. The complete encoding is available in the source code[7].

Starting with the rules in lines (1 and 2) the logic program defines the set of genes that are associated to the *start* and *end* reactions via the *cat* predicate. As there can exist more than one gene catalyzing a reaction, there is a set of genes corresponding to them. Each of the genes denoted by the predicate *sgene* can catalyze the *start* reaction, while the predicate *egene* denotes genes which can catalyze the *end* reaction respectively. Note that start and end genes do not necessarily correspond to the beginning and end of a desired genome segment. These genes can occur everywhere and in any order in a genome segment.

The rule in lines (4-9) defines the search space of possible genome segments. A possible genome segment is denoted by the predicate *pse* with two arguments, the first gene F and the last gene L of the segment. The length of the segment is determined by the formula $L - F + 1$ and only segments with $min \leq$ length $\leq max$ are considered. Furthermore, a segment must contain at least one start gene and one end gene respectively. Therefore, it must hold that there exist a start gene S and an end gene E such that $F \leq S \leq L$ and $F \leq E \leq L$ respectively.

Among the possible segments exactly one segment can be chosen. This choice is expressed by the rule in line (11). The genes that lie on the chosen segment are defined by the rule in line (13). These genes induce a set of edges in the reaction graph, the *induced subgraph*, connecting reactions that are catalyzed by genes on the segment. The set of active edges is defined by the rule in lines (15 and 16). An edge (X, Y) is active if X and Y are both catalyzed by genes on the segment.

Given the set of active edges one can test whether there exists a path from the start reaction to the end reaction. The rules in lines (18 and 19) define what is reachable from the start reactions and the integrity constraint in line (20) discards solutions where the end reaction is not reachable. The rules in lines (22 and 23) define nodes that lie on a path of active edges to the *end* reaction.

The genes that catalyze reactions on a pathway from *start* to *end* reaction form the *active gene unit*. They are defined by the rules in line (25). The integrity constraints in lines (26 and 27) discard segments that merely extend shorter segments without extending the active gene unit. A segment is not considered a solution if the first or the last gene is not part of the active gene unit. These integrity constraints are especially important if we look for shortest segments which are bigger than a given size *min*. Without these constraints every solution that is smaller or equal than *min* could easily be extended to a solution of size $min + 1$ by simply prolonging a shorter solution.

Line (29) defines the length of the segment. The integrity constraint in line (36) discards segments whose lengths are shorter than the required minimum *min*.

So far the rules in lines (1 to 31) define all segments that catalyze a reaction pathway from *start* to *end* reaction. To solve the corresponding SGS optimiza-

tion problem, line (33) declares the objective function via an optimize statement. Preferred solutions are those that minimize the length of the segment.

5 Enumerating shortest genome segments

We provide an application that computes optimal and sub optimal solutions close to the optimum. More precisely, we enumerate all solutions σ of a SGS problem with minimal length from 1 to n until we have at least k distinct active gene units $AU(\sigma)$ or there exist no further solutions.

The intuition is that these active gene units correspond to functional gene units like metabolic operons or regulons. They allow us to investigate the relationship between metabolic pathways and gene localization.

To compute these segments we developed the Python program `shogen`[7]. It depends on the `PyASP`[8] library for calling the ASP solvers `gringo` and `clasp` and for passing them logic program encoding and problem instances.

`shogen` takes as input the genome, the metabolic reaction network and information of the catalytic function of the genes. Further, a list of queries start and end reactions for which we want to find functional gene units. In a pre-processing step `shogen` filters queries that do not have a path in the metabolic reaction network. For the remaining queries `shogen` computes the shortest genome segments and their active gene units.

The computation is performed in a multi-step process. In a first step, `clasp` is used to solve **Problem 1**, computing the minimum length of a segment that can catalyze the desired metabolic pathway. Once the optimal length is known, `clasp` is used with the option `--opt-all` to solve **Problem 2**, enumerating all solutions that satisfy this optimality criterium. These steps are repeated until at least k segments are computed or no more solutions can be found. The minimum length of a segment is increased whenever all solutions of a given length are computed. The maximum length is fixed and part of the logic problem instance.

The following pseudo-code describes the algorithm.

Algorithm 1: compute the shortest genome segments

Input: A SGS problem instance as facts `instance.lp` and a parameter k
$Segments \leftarrow \emptyset$;
$min \leftarrow 1$;
while $|Segments| < k$ **do**
  $opt \leftarrow$ `gringo instance.lp sgs.lp --const min=`min `| clasp`;
  $\Sigma_{min} \leftarrow$ `gringo instance.lp sgs.lp --const min=`min `| clasp --opt-all=`opt;
  $Segments \leftarrow Segments \cup \Sigma_{min}$;
  $min \leftarrow opt+1$;
end
return $Segments$

[7] `https://pypi.python.org/pypi/shogen`
[8] `http://pypi.python.org/pypi/pyasp`

6 Experiments and results

The functional gene units produced by using the notions of shortest genome segments and shortest wrr-paths were compared. We consider an operational criterium, that is, the computational time needed to obtain them, and also the biological relevance of the results. The benchmark was conduced on the widely studied and well known *Escherichia coli* bacteria. In order to simulate more exotic or less studied organisms, with unrefined models, we create a set of deteriorated models of *E. coli*.

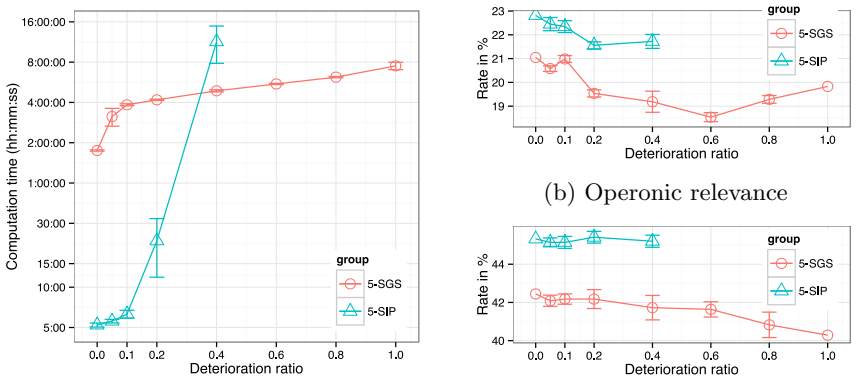
Simulating exotic organisms. Although modern genome sequencing techniques allow us to obtain genome data even for little genomes and exotic organisms, the main problem lies in the reconstruction of the metabolic networks for these organisms. While it is often possible to determine the reactions that occur in the metabolic network, a lot of complicated and costly experiments are needed to determine their direction. Thus, the direction of those reactions remains often unknown. *E. coli* is regarded as the best studied organism today and its metabolic network is the most refined existing one. Therefore, we us it as the reference to obtain deteriorated models containing less information about irreversible reactions. Given the set R_i of irreversible reactions known in *E. coli*, we created deteriorated models by taking subsets R_d of R_i and transforming them to reversible reactions. In this way we created models with different deterioration ratios $\frac{|R_d|}{|R_i|}$ ranging from 0 to 1. These deteriorated *E. coli* models aim to simulate the unrefined models of exotic species.

Benchmark and experimentations. The knowledge about *E. coli* was taken from the `Ecocyc` database (version 16.1). Its genome is composed of one circular chromosome of 4498 genes and its metabolism consists of 2070 distinct reactions separated into 816 reversible and 1254 irreversible ones. We generated four deteriorated models for each of the following deterioration ratios: 0.05, 0.1, 0.2, 0.4, 0.6 and 0.8. We also generated the model with a deterioration ratio of 1.

We confronted each model with a set of queries, a couple of start and end reactions, for which we computed the optimal solutions and the next four levels of suboptimal solutions for the shortest wrr-paths problem and the shortest genome segments problem. We call such solutions a 5-SIP and 5-SGS respectively. The selection of 5 is motivated by computations in [2].

The computations were done on a MacBook Pro 9,2 equipped with an Intel Core i7-3520M processor and 8 Gb of RAM. This computer was running under Mac OS X 10.7.5. The dedicated program `sipper`[9] for computing shortest wrr-paths uses the Java SE Runtime Environment build 1.7.0_13-b20, and the ASP solution for computing shortest genome segments uses the PyASP library including the grounder `gringo` in version 3.0.4 and the ASP solver `clasp` version 2.1.1. Both programs were configured to use only one core. Each computation was repeated twice with a timeout of 24 hours.

[9] `https://sipper.googlecode.com/`



(a) Computation time

(b) Operonic relevance

(c) Operonic justification

Fig. 2. Results of the computation of 5-SGS and 5-SIP on *E. coli*. The subfigure (a) indicates the computation times according to the deterioration ratio whereas the subfigures (b) and (c) present respectively the operonic relevance and the operonic justification of the sets of gene units.

Computation time. Figure 2(a) shows the computation times for the 5-SIP and 5-SGS. When no deterioration is applied, the computation of 5-SIP is faster than the one of 5-SGS. The computation times of both 5-SIP and 5-SGS increase with the deterioration ratio, but the time to compute 5-SIP increases exponentially while the one for 5-SGS increases in a more linear way. For a deterioration ratio over 0.4, the computation of 5-SIP takes more time than the computation of 5-SGS. Moreover, increasing the deterioration ratio even further, the computation of 5-SIP quickly reaches the 24 hours timeout limit.

Biological relevance. We evaluated the biological relevance of the predicted functional gene units by comparing them against the known metabolic operons in *E. coli*, which are functional units playing a role on both genome regulation and metabolic network studies. We used the set of 278 manually-curated metabolic operons reported in the `Ecocyc` database version 16.1. We computed the similarity between each predicted gene unit and each operon using the Jaccard measure. As biologists have interest in groups of genes, we do not consider gene units of one gene. A predicted gene unit is said to be similar to an operon when the Jaccard measure between them is greater than or equal to 0.6. We computed then (1) the *operonic relevance* and (2) the *operonic justification* for the set of 5-SGS and the set of 5-SIP. The operonic relevance of the predicted gene units is the proportion of units that are similar to at least one operon. The operonic justification for predicted gene units is the proportion of operons that is similar to at least one predicted gene unit.

Figures 2(b) and 2(c) show the results for operonic relevance and operonic justification, respectively. In both experiments, the deterioration of information leads to predicting gene units with more genes. Thus, we can observe a small

decrease of the number of identified operons and a decrease of the operonic justification. Observe also that for 5-SIP a bigger proportion of the predicted gene units is similar to operons and allows to explain a bigger part of the operon than 5-SGS.

7 Discussion

This study demonstrates the biological interest of using SGS as an alternative to shortest wrr-paths, by showing similar capabilities in the identification of confirmed functional gene units. Both concepts were compared using different levels of deteriorated biological knowledge. While using the concept of SGS produced only slightly lower prediction scores for metabolic operons than wrr-paths, it allowed to get meaningful predictions even on unrefined networks with a high deterioration ratio. Therefore, SGS can be considered as an efficient computable alternative to predict functional gene units even on unrefined models.

Beyond the biological evidences for the quality of predictions with SGS, the major outcome of this study remains the efficiency results. Indeed, although wrr-paths is an interesting concept to study organisms for which the biological knowledge is globally complete, this technique drastically fails when applied to incomplete ones, as observed when one studies exotic species. Although the dedicated algorithm to compute the shortest wrr-paths has a better performance on the well refined *E. coli* models, this study emphasizes that the computational time of shortest wrr-paths increases exponentially when the biological knowledge is degraded. This is mainly due to the fact that for one set of genes many equally short wrr-paths can exist, differing only in the order of the involved genes (i.e. a permutation of genes).

SGS does not regard these different permutations. Therefore, the computation times for SGS remain relatively stable, but they are comparatively higher than the dedicated algorithm on the well refined model. This overall high runtime can be explained by the fact the each query represents a separate instance of the ASP problem. Therefore, the times for the problem generation and initialization of data structures, which must be done only once in the dedicated algorithm, is roughly multiplied by the number of queries. A further improvement of the ASP solution could be reached if one can reuse data structures on all problem instances.

Our results confirmed the interest of integrated models and SGS for investigating functional units of exotic species, and the interest of using ASP for deciphering these biological units. From a methodological point of view, ASP allows us to quickly test biological assumptions. In particular, the expressiveness of ASP presents a clear advantage for exploring several hypothesis on biological systems. As a biological perspective, further studies will focus on the extension of the SGS framework to the identification of new functional units. We exploit the flexibility of declarative programming with ASP to create models using more constraint metabolic behavior [12], to explore different metrics based on transcriptomic correlation data instead of a genomic distance, and to identify

graph-based units such as CCC (Common Connected Component) [3] or regulons [13], describing co-regulated operons. Therefore, we rely on the versatility of the ASP language and the solving capabilities of ASP solvers to integrate large-scale heterogeneous biological knowledge into computational models.

Acknowledgments. This work was supported by ANR Biotempo (ANR-10-BLANC-0218), Basal-CMM, Fondap-CRG 15090007, INRIA-UChile Integrative-BioChile Associate Team and CIRIC INRIA-Chile.

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
2. P. Bordron, D. Eveillard, and I. Rusu. Integrated analysis of the gene neighbouring impact on bacterial metabolic networks. *IET Systems biology*, 5(4):261–268, 2011.
3. F. Boyer, A. Morgat, L. Labarre, J. Pothier, and A. Viari. Syntons, metabolons and interactons: an exact graph-theoretical approach for exploring neighbourhood between genomic and functional data. *Bioinformatics (Oxford, England)*, 21(23):4209–4215, 2005.
4. R. W. W. Brouwer, O. P. Kuipers, and S. A. F. T. van Hijum. The relative value of operon predictions. *Briefings in bioinformatics*, 9(5):367–375, 2008.
5. M. Durzinsky, W. Marwan, M. Ostrowski, T. Schaub, and A. Wagler. Automatic network reconstruction using ASP. *Theory and Practice of Logic Programming*, 11(4-5):749–766, 2011.
6. T. Fayruzov, M. D. Cock, C. Cornelis, and D. Vermeir. Modeling protein interaction networks with answer set programming. In *Bioinformatics and Biomedicine, 2009. BIBM '09. IEEE Int. Conf. on*, pages 99–104, 2009.
7. M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele. A user's guide to `gringo`, `clasp`, `clingo`, and `iclingo`. Available at `http://potassco.sourceforge.net`, 2010.
8. M. Gebser, R. Kaminski, and T. Schaub. Complex optimization in answer set programming. *Theory and Practice of Logic Programming*, 11(4-5):821–839, 2011.
9. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-driven answer set solving. In M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 386–392. AAAI Press/The MIT Press, 2007.
10. I. Papatheodorou, M. Ziehm, D. Wieser, N. Alic, L. Partridge, and J. M. Thornton. Using answer set programming to integrate rna expression with signalling pathway information to infer how mutations affect ageing. *PLoS ONE*, 7(12):e50881, 2012.
11. E. P. C. Rocha. The organization of the bacterial genome. *Annual review of genetics*, 42:211–233, 2008.
12. T. Schaub and S. Thiele. Metabolic network expansion with ASP. In P. Hill and D. Warren, editors, *Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09)*, volume 5649 of *Lecture Notes in Computer Science*, pages 312–326. Springer-Verlag, 2009.
13. H. H. Zhang, Y. Y. Yin, V. V. Olman, and Y. Y. Xu. Genomic arrangement of regulons in bacterial genomes. *PLoS ONE*, 7(1):e29496–e29496, 2011.