From Genomic to Functional models

Macha Nikolski

Numéro d'ordre : 448

UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE MATHÉMATIQUES ET INFORMATIQUE

Laboratoire Bordelais de Recherche en Informatique

From Genomic to Functional models

mémoire présenté par

Macha Nikolski

en vue d'obtention du diplôme de

L'HABILITATION À DIRIGER DES RECHERCHES

Thème : Bioinformatique

Soutenance le 6 octobre 2009

Rapporteurs	Thierry Colin	Professeur à l'Université de Bordeaux, France
	Betül Kirdar	Professeur à l'Université de Bogazici, Turquie
	Jens Stoye	Professeur à l'Université de Bielefeld, Allemagne
Examinateurs	Michel Aigle	Professeur à l'Université de Lyon, France
	Serge Dulucq	Professeur à l'Université de Bordeaux, France
	Guy Melançon	Professeur à l'Université de Bordeaux, France
	Pascal Weil	Directeur de recherche au CNRS, France

Nous aimons la chaleur glaciale des nombres
Et la beauté des dons divins,
Nous comprenons l'esprit gaulois, comme le sombre
Génie orageux des Germains.

Les Scythes par Alexandre Blok

Contents

This document encapsulates the research that I have pursued since 2003 and recaptures the main results during this period that were obtained by myself and the students that I have supervised. It has benefited from strong collaboration with fellow researchers, as can be seen in the complete list of publications (see page 105). It is organized in four chapters having distinct objectives, and thus varying in form and style.

The first chapter – bearing a quite original title *Introduction* – has in mind an ingenuous reader who is not necessarily familiar with the general research domain of bioinformatics and systems biology. There is absolutely no claim to provide an introduction to these fields, but rather to set some common reference points that determine the boundaries of my own work.

The second chapter treats a classical but essential topic in bioinformatics, that of uncovering the common points between related genomes. Precisely, here we study the question of building high-quality protein families in a computationally efficient way. We then address the question of detecting gene fusion and fission, a case where alignment-based computation of families will fail.

The third chapter concerns a slightly less classical approach for studying evolution by means of rearrangement analysis. Here we first revisit the theory of computation of rearrangement analysis, second propose a new method of analysis of ancestral architectures that reconciles both breakpoint and rearrangement analyses, and third introduce optimization algorithms that can address this problem in practice.

In the last chapter I describe the work in construction of dynamic models of living organisms that exhibits my enduring interest in systems biology. I do not consider the boundary between bioinformatics and systems biology by any means a rigid one: both disciplines essentially being concerned with building models of living organisms. We propose here a novel high-level modeling framework integrating discrete and continuous multi-scale dynamics within the same semantics domain. I conclude with experimental results using a software implementation of this framework.

A *curriculum vitae* and a publication list can be found at the end on page 105.

Chapter 1

Introduction

1.1 Did you say 'models'?

> Everyone knows that computational and information technology has spread like wildfire throughout academic and intellectual life. But the spread of computational ideas has been just as impressive.
>
> Biologists not only model life forms on computers; they treat the gene, and even whole organisms, as information systems. Philosophy, artificial intelligence, and cognitive science don't just construct computational models of mind; they take cognition to be computation, at the deepest levels.
>
> Physicists don't just talk about the information carried by a subatomic particle; they propose to unify the foundations of quantum mechanics with notions of information. Similarly for linguists, artists, anthropologists, critics, etc. Throughout the university, people are using computational and information notions – such as information, digitality, algorithm, formal, symbol, virtual machine, abstraction, implementation, etc. – as fundamental concepts in terms of which to formulate their theoretical claims.
>
> Brian Cantwell Smith on "The Wildfire Spread of Computational Ideas", 2003

Erwin Schrödinger pioneered the search for a modern definition of life with his 1943 book "What Is Life?". The main question that is addressed in this work is how life defies the laws of physics and creates order even on the molecular level, while general physics laws on the contrary tend to entropy. Crick was inspired by Schrödinger's book for his work on the structure of DNA. Crick's own work can be seen as a search for understanding of how a small set of repeating elements can give rise to a large number of combinatorial products, a mathematical relationship that Schrödinger illustrated using the Morse Code.

In turn, Erwin Chargaff postulated his famous rules, and in particular on the variability of DNA composition from one species to another. Chargaff's rules enabled the grand discovery by Watson and Crick. Combining these rules with Schrödinger's code script made it possible for them to conclude their work by stating lapidarily: "It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material."

To this day there is no definition of what life is, as Radu Popa states; indeed, he has found at least three hundred definitions when researching for his book "Between Probability and Necessity: Searching for the Definition and Origin of Life" [154]. Nevertheless, at least some consensus is emerging as to what are necessary features for any living entity: a container, a way to encode and replicate information, and a way to capture and use energy. Moreover, each one of these features

relies on the two others. DNA can only exist within a cell membrane, and it depends on metabolism to get energy for replication. In its turn DNA produces proteins that are building blocks of membranes. The energy and the genes are stored within the cell, so that the metabolism becomes possible. In its turn metabolism provides the raw ingredients for genes and membranes.

This example of the definition of what is a living system illustrates the fact that the biological sciences fundamentally differ from the other physical sciences by a lack of unifying, universally applicable theories. This results in such features of biological knowledge, as the predominance of mechanistic and functional explanations over other forms of explanation. At the risk of stating the obvious, I would like to emphasize that in biology and in its *in silico* sister sciences such as bioinformatics and systems biology, the effort of modeling is necessarily skewed since in these subjects we do not fully know what is a model.

Modeling strategies themselves are extremely varied in both the form that they take, which may be diagrammatic, computational, or mechanistic; and the way in which they bridge between theories and data. When this modeling is done *in silico* the models are often said to be *formal*. With respect to all the previous observations we can summarize the challenges related to this *formal modeling approach* as follows.

- We do not possess complete knowledge of any biological phenomenon.
- Each of the possible modeling approaches represents the phenomenon from a particular point of view, is necessarily partial, and has the potential to bring answers to questions only in the frame of this point of view.
- Even worse, here "formal" is a poorly defined term. Depending on the context it can mean precise, pictorial, abstract, syntactic, digital, etc.

Nevertheless, a common trend towards an increased focus on the formal description of relationships between biological entities, has clearly become predominant in the last decade. These descriptions concern such different objects as evolutionary relationships providing evidence for protein function, interactions encoding cellular processes, kinetic relationships in the form of rate equations used to simulate dynamic behaviors of cellular processes, and many others.

Modeling is a process of abstraction and representation of the phenomenon of interest from a certain point of view, pertinent to the modeler and susceptible to answer the questions that interest him. The living world is too complex to be modeled in sufficiently accurate detail. Thus we call for techniques (for example from statistics and machine learning) to detect *signals* in the data and separate them from noise. Here "signal" is used in a very general sense and can mean anything of biological interest – from a sequence alignment exhibiting the evolutionary relationship of two proteins, to the co-expression of two genes under certain experimental conditions.

The methods of analysis are inexact, and so are the results. This is fundamentally different from theoretical computer science, where the term "optimum" has a precise definition and you are either required to compute the optimum solution or, at least calculate the distance from the given solution to the optimum if the absolute optimal solution is not reached. However, cost functions in computational biology themselves are inexact. Indeed, notions such as evolutionary distance or free energy are much too complex to be reflected adequately by easy to define cost functions.

Validating models in biology is also tricky, the main loophole being whether the model can use the knowledge of the intended outcome, either on purpose or inadvertently. The ultimate test of any computational biology methods and models is blind prediction without previous knowledge of the outcome.

Faced with these challenges and inexactitudes, should we stop? The answer is an emphatic "no". Perversely, one can argue that the sheer scale of problems we are faced with, is a guarantee of

employment for the years to come. More constructively, the current "mess" is an extremely fertile ground on which novel techniques and methods for computational and systems biology are about to blossom. In the rest of this chapter I will go over the particular computational models I have worked with over the past years, ranging from comparative genomics to modeling of dynamic behaviour. The same topics will be reconsidered in detail in the chapters that follow.

1.2 Comparative genomics

Fundamentally, the biological sciences address questions of **function** and of **history** of living systems. By function we mean how components of these systems act, and act together, to realize biological processes; by history, we mean how and by what means these components and their functions arose through evolution. The goal of comparative genomics is to understand the history and function of genomes through the comparison of related species. While this goal is inherently biological, the techniques brought into play are inherently informatic and comprise a domain of scientific study in their own right.

Genome annotation is the process of associating biological knowledge to sequences. This involves identification of the genes through analysis of the sequence, clustering the genes and other elements into phylogenetic and functional groups, and integrating heterogeneous data sources into efficient software tools for exploration, analysis, and visualization. References [175] and [178] provide an overview of our work.

Sequence analysis using probabilistic models, notably hidden Markov models, are used for syntactic analysis of macromolecular sequences, applying rules derived from models of how the cell's transcriptional machinery recognizes and interprets the DNA sequence to predict whether a given sequence codes for protein, is intronic, participates in gene regulation, etc. Sequence analysis is also used for alignment, looking for homologous regions in different sequences [3], and can also look for motifs, common subsequences whose similarity within and between species provides clues about common mechanisms [188]. Such approaches are essentially statistical and the key challenges revolve around the gap between knowing the primary structure of the sequence itself, and understanding the derivative structures such as 3D and 4D conformation that confer true function in physiological conditions. Sequence analysis is largely studied elsewhere and provides tools that we use and adapt.

Combinatory analysis, including algorithms for permutations and other word problems, and graph algorithms are widely used for biological data. Our own work involves combinatorial methods for reconstructing ancestral genomes inspired by [84, 153, 193], but including biologically-inspired constraints such as centromere position and a cost model adapted to our models of yeast genome evolution. Ancestral reconstruction requires three basic steps: identification of common markers in contemporary genomes, construction of comparative genome maps, and reconciliation of these maps using parsimony criteria to construct ancestral maps. Common markers can be identified by cytogenic methods [167] (such as chromosomal painting) or through genome sequencing and sequence analysis to find conserved segments of chromosomal homology (an approach pioneered by Pevzner and others, intuitively similar to the identification of syntenic blocks e.g. [203, 153, 105]). Mathematically, comparative maps are constructed by representing each genome as a signed permutation of the common markers. Computational reconciliation of comparative maps is formulated as the *multiple genome rearrangement problem* [163, 84]: given a set of N contemporary genomes and a distance d, find a tree T with the N genomes as leaf nodes and assign permutations (plausible ancestral architectures) to internal nodes such that the sum of the pairwise rearrangement distances along the branches is minimized. When $N = 3$ this is called the *median genome problem*. Methods using

the *breakpoint distance* [133] and the *rearrangement distance* [84] were developed respectively by Sankoff and Blanchette [162], and Bourque and Pevzner [23]. In either case the problem has been shown to be NP-complete ([28, 152] for the breakpoint distance, [33, 34] for the rearrangement distance), leading to the need for approaches that treat this as an optimization problem in the space of genome rearrangements. A key advantage of my approach is that it gives the means to explore rearrangement scenarios that are suboptimal with regard to the mathematical formulation, but possibly more reasonable with regard to biological constraints.

Through structured comparisons of contemporary genomes, it is possible to project a network defined on the genome of a reference species to the genome of a new species using a variety of mathematical techniques. Metabolic reconstruction typically starts with a database of biochemical pathways, where the metabolic network is decomposed into a a set of reactions linking enzyme and substrate nodes of the graph. An annotated genome is then used to determine which reactions may be present in the given species, first by assigning homologous genes identified by gene conservation to enzyme nodes, then by gap filling to find unassigned genes that may play missing roles in the reconstructed reactions. Gap filling typically uses the connectivity of the upstream and downstream reactions to search for likely candidate genes, then filters them using a machine learning approach or manual curation [103, 76, 5]. The resulting reactions can be represented as a stoichiometric matrix, where each column is a substrate that may appear in a reaction, and each row describes the quantitative relationship between substrates defined by a reaction. The set of equations defined by this matrix can then be submitted to a number of analyses, including *elementary modes* [169] and *extreme pathways* [168] (see [148] for review), and *flux balance analysis* [204, 21].

A central element of our approach is the use of homology relations between genes supported by consideration of a large number of genomes and a wide range of evolutionary distances, which permits the extrapolation of approximate networks from those of a reference organism (in this case *Saccharomyces cerevisiae*). Resulting networks are then post-processed using genome coherency rules and gap-filling, and analyzed using graph theoretic techniques[97].

Consensus clustering Broadly speaking, *data-mining* methods seek to find meaningful patterns in volumes of data, ideally patterns that are both previously unknown and useful for some application. We can contrast this with *data integration*, where the goal is to link related information in a semantically coherent way.

Clustering is a widely used data-mining technique whose goal is to learn a set of classes or categories for the given data, without an predetermined idea of what those classes will be. Its utility for applications in computational biology stems from widespread use of "guilt by association" reasoning: phenomena that appear under the same conditions in a experiment often take part in a common, unknown mechanisms of biological interest. Many varieties of clustering algorithms for biological data have consequently been developed, and in large numbers (see [12] for review), which leads to an important practical problem: how to decide which algorithms, or which learning parameters, to use for a given application?

We have addressed one part of this problem through the development of techniques for clustering ensembles, where the goal is to combine the strengths of a chosen set of different (presumably complementary) clustering techniques. This can be formulated as a search for a median partition Π that minimizes $S = \sum_{i=1}^{k} d(\Pi^i, \Pi)$, given k partitions $\Pi^1, \ldots, \Pi^k$ and a distance function d. The first mathematical treatment goes back to Régnier [157], and [14] shows that the general problem is NP-complete. If the partition Π of the dataset D, $|D| = n$ to discover is not necessarily one of the original partitions $\Pi_1, ..., \Pi_k$, then the size of the potential search space corresponds to the *Bell numbers* [17]. Heuristic approaches have been developed for this inherently intractable problem: exact methods using cutting planes [77], co-association methods [63], voting approach [197], information-

theoretic approach [126], hypergraph partitioning methods [190] and using mixture models [196].

The solution we have developed [143] is tailored to the specific problem of consensus clustering for protein families, where in our application $n = 50\,000$ but singleton families (containing only one protein) are allowed. The approach uses a compact bipartite graph encoding of the confusion matrices of pairwise comparisons between two input partitions Π^1 and Π^2, where nodes are clusters in one or the other inputs, and edges indicate that the two clusters have an element in common. Choice of a consensus among the k partitions can be made by choosing within the connected components of the confusion matrix, in such a way as to cover all the initial elements. Such a choice can be formulated as an instance of minimum exact cover (MDC), also NP-complete [68]. Since we allow singleton families we can further relax the problem to minimum *inexact* cover. In [143] we define an efficient heuristic running in low-order polynomial time that uses a Condorcet election procedure to choose an inexact cover that minimizes inter-partition distance while maximizing cluster similarity. Through collaboration with the UniProt consortium (Georgetown) we have evaluated our results through comparison with the PIR-SF curated classification of proteins; very high agreement was obtained for families conserved across phyla, and the other differences could be reconciled with a refinement into *lineage specific* families. In additional to refining these classification methods, it is necessary to develop new ones that take into account relations between genes that are not 'tree-like', arising for example from gene fusion and fission events. We studied these events using an approach based on multiple alignments in [52]

1.3 Modeling of biological processes

Hierarchical models. A recurring challenge for *in silico* modeling of cell behavior is that hand-tuned, accurate models tend to be so focused in scope that it is difficult to repurpose them. Hierarchical modeling [4] is one way of combining specific models into networks. Effective use of hierarchical models requires both formal definition of the semantics of such composition, and efficient simulation tools for exploring the large space of complex behaviors. The major difficulty in applying hierarchical modeling to biological systems is the definition of criteria for demarcating components in order to guarantee a certain level of autonomy. Even though a multitude of methods for decomposition of networks into modules has been suggested, the specification of these functional units is a challenge. As of today, they are most often defined from textbook-driven decomposition into components realizing a certain function.

Biological processes take place at different tim -scales, for example, transforming metabolites is fast, while synthesising enzymes is slow. If modelled by ODEs, this multi-scale problem in time is known as stiffness [27]. It is the fast time-scaled reactions that are stable, but it is the slow reactions that determine the trajectory of the system. If done naively, a simulation will have to use extremely short time steps (which implies long simulation times) due to the fast time scales. Hierarchy can be seen as a way to overcome this issue. Indeed, simulation of hierarchical systems can be done efficiently by having dedicated solvers for processes at different granularities. Whenever a multi-component system is modelled, a precise semantics for module composition has to be provided in order for the model to hold as a whole.

Stochastic models. Recognizing conditions when stochastic effects in cellular processes are important is a challenge. While the deterministic representations such as ODEs may be appropriate for some systems, it has been recognized that stochastic kinetics can produce significant variations in gene expression, especially considering the typically low concentrations of reactants [125, 6]. However, simulating stochastic kinetics comes at a great computational cost since the exact procedure

simulates every reaction [72], and even taking into the account recent improvements [71, 31, 32] real-sized systems can not be treated by this approach. A related observation is that in a cell quantities of different molecules vary greatly, and while it is reasonable to model processes involving small quantities of molecules by stochastic processes, when large quantities are involved, ODE models provide a solution that can be efficiently simulated. Pragmatically speaking, the question is then, how to switch for a given process from one kind of model to another, when the quantities of involved molecules reach certain thresholds.

Two possible directions for solving the inefficiency of the simulation of stochastic systems can be considered. First, design more efficient numerical algorithms for simulating large stochastic systems. Second, use formal methods model analysis techniques that circumvent simulation altogether.

The behavior of a stochastic system is generally studied from a transient or a steady-state point of view [198, 122, 10]. *Transient analysis* aims at providing measures like the state distribution given a timed horizon, the mean time to reach a subset of states characterized by some property, etc. *Steady-state analysis* aims at deciding whether one or more steady state behaviors exist, what are the probabilities to reach a particular steady-state behavior, what are the associated steady state distributions, etc. In the last decade, these two approaches have been unified and extended by the introduction of quantitative temporal logics equipped with model-checking algorithms. Besides expressing the above properties, such logics also specify path properties that are of particular interest in the framework of this project. For instance, one can determine the probability that event "fermentation stop" occurs in some interval after certain previous events related to the environment [9, 47].

Among the techniques developed to obtain quantitative measures, simulation [161] presents advantages (no special requirement on the stochastic process, reduced space complexity, etc.) and drawbacks (time complexity, inability to efficiently manage the steady-state behaviors, etc.). Thus it is critical to design alternative methods as a complementary way for the analysis of dynamic systems [70, 117, 118, 81, 80]. However such design has to deal with the following verification challenges, that can be seen as combinatorial explosion problems arising from the complexity of the qualitative behavior of the system, from the multiple time-scales of the events, and from the fact that the distributions associated with the events may require enlarging the state description.

Qualitative behavior. Accounting for unknown gene functions and/or interactions, is hard as was shown by [206]. The authors speculated that genetic regulatory networks have evolved their connectivity to become robust to alterations in kinetic parameters. This means that the qualitative behavior is essential in order to understand the system as a whole, while the accurate values of the kinetic parameters are of secondary importance. Indeed a consensus on this point is forming in the research community [165], which suggests that the time is ripe for moving from simulation only towards more symbolic model analysis methods.

Behavioral analysis of models has been extensively studied in the context of formal verification of hardware and software systems. In this context, models are usually non-deterministic automata equipped with variables ranging over finite or infinite domains. Formal verification does not face theoretical limitations (i.e. it is decidable) for finite-state models. However this approach is in practice limited by the so-called *state explosion problem*: the size of the model's state space, which must be explored by the verification tool, is (at least) exponential in the number of variables. The introduction of symbolic techniques [29] was a major breakthrough in this field, and today models with several hundreds of Boolean variables can be verified.

Approximation techniques have been developed in order to extend the reach of formal verification to larger models that are not amenable to standard symbolic verification. These models might be finite-state (but with too many states) or infinite-state, such as automata with (unbounded) integer variables. The two main approaches are *abstract interpretation*, which reduces reachability problems

to the least fixpoint of some operator on the state space, and simplifies the computation by lifting this operator to some abstract, simpler domain; and *abstraction refinement*, which, for a given property to verify, abstracts away from details that are not relevant to that property.

BioRica modeling platform Instead of modeling individual processes individually *de novo*, we claim that a sustainable effort in building efficient behavioral models must proceed incrementally. In order to leverage existing models when building new ones, it is necessary to provide reliable means for connecting them. Connections must be on the basis of biologically pertinent elements (such as metabolites or signalling), must respect a formal semantics so that the meaning of the combined model is sensible, and must avoid when possible an explosion in complexity. The goal of the *BioRica* platform is provide **modeling middleware** that meets these requirements in a pragmatic way.

A central point of model merging is the semantics of the models, both in the biological and computational senses. When biologists combine models by hand, they are aware of the meanings of the variables and parameters for individual models. A computer program of course needs this information in an explicit form via controlled vocabularies and biological ontologies. But more importantly, these models have to be compiled into an unique operational semantics in order for the whole model to have a meaning. Defining and simulating such models represent a significant challenge. Most biological models are described using ODEs and, in practice, combining uncoupled ODEs is not obvious. Co-existence of different kinds of ODEs can lead to multi-scale models, that are computationally inefficient. Composition of models that use discrete formalisms can lead to semantic incompatibility.

BioRica [79] is a high-level modeling framework integrating discrete and continuous multi-scale dynamics within the same semantics domain, while offering an easy to use and computationally efficient numerical simulator. It is based on a generic formalism that captures a range of discrete and continuous formalisms and admits a precise operational semantics. BioRica models have a corresponding compositional semantics in terms of an extension of Generalized Markov Decision Processes. This semantics allowed us to prove that BioRica models admit an operational semantics in terms of continuous stochastic processes, and that this operational semantics is correctly simulated by the discrete event stepper used during numerical simulation.

BioRica programs are hierarchical and are built by connecting and composing simple units called *nodes*. Nodes are basically a set of variables and a set of events, that can modify the content of variables. The dynamics of a node is described by a set of events. It is also possible to describe more complex behavior by associating to any event timing and stochastic information. Systems are built when connecting and composing nodes, following the object-oriented paradigm. Three different kinds of composition are possible in BioRica: *parallel composition*, *synchronized composition* and *flow connections*.

Biological process dynamics as found in the literature are mostly described using sets of Ordinary Differential Equations. A BioRica node containing ODEs relies on the separation of discrete events from the timing functions that trigger these events. The latter rely on classical numerical integration. These functions solve internally the ODE system and output the time delay before the next discrete event. This allows for direct composition of BioRica nodes with ODE systems.

In a collaboration with M. Cvijovic and E. Klipp, we demonstrated the advantage of the BioRica hierarchical approach[42]. In this study, a continuous single-cell ODE model of inheritance of oxidatively damaged proteins was hierarchically combined with two levels of discrete controllers, maintaining mother-daughter cell relations and cell-population relations. Large-scale simulation up to cell senescence revealed lineages with a *cell rejuvenation* effect, experimentally verified by M. Cvijovic in the lab of T. Nyström (article in press).

While combination of discrete and continuous ODE models is a significant step forward, fu-

ture work must address combinations with other kinds of continuous process such as metabolic flux analysis.

Chapter 2

Algorithms for comparative genomics

2.1 Protein families

Reliable identification of protein families is key to phylogenetic analysis, functional annotation and the exploration of protein function diversity in a given phylogenetic branch. As more and more complete genomes are sequenced, there is a need for powerful and reliable algorithms facilitating protein families construction. In fact, the diversity and complementarity of existing clustering algorithms forms a challenge when choosing an effective method

We have formulated the problem of protein family construction as an instance of *consensus clustering*, for which we designed a novel algorithm that is computationally efficient in practice and produces high quality results. Our algorithm uses an election method to construct consensus families from competing clustering computations.

Our consensus clustering algorithm is tailored to serve the specific needs of comparative genomics projects. First, it provides a robust means to incorporate results from different and complementary clustering methods, thus avoiding the need for an *a priori* choice that may introduce computational bias in the results. Second, it is suited to large-scale projects due to the practical efficiency. And third, it produces high quality results where families tend to represent groupings by biological function.

2.1.1 Background

Overview

When confronted with computation of protein families, one must decide which algorithm to use, evaluate the quality of the result, and decide which computed families–if any–correspond to real protein families. We argue that it is the comparison of different results that produces the set of the most plausible families.

Algorithmic means for establishing protein families on a large scale are based on a notion of similarity. Often the only available similarity is sequence similarity; some authors go so far as to define the notion of a protein family itself *via* the similarity of sequences [48]. Use of structural similarities [89] is currently limited by the relatively small number of structures available in PDB [19], but by studying sequence variation among members of such families [110] one can guide the use of sequence similarity information.

To perform similarity-based detection of families one first selects the pairwise similarities to be used and then applies an algorithm that uses these similarities to group proteins into families. Algorithms for detection of protein families are unsupervised (see for example [201]) or supervised

(see for example [78, 16]) learning procedures. Pairwise similarities can be provided by sequence alignments (BLAST [3], Smith-Waterman [183]) or by more sophisticated approaches using domain architecture databases [15, 171, 132]. Other approaches use domain order as a fingerprint for the protein [69].

Methods that quantify similarity by using some attribute of the best BLAST hit and use single-linkage clustering are not always successful. Such straightforward approaches may group together dissimilar multidomain proteins that share a common domain [194], and can be fooled by promiscuous domains [49, 121]. Several graph-based methods have been proposed to overcome some of these limitations of single-linkage clustering [124, 57, 56].

Reliability of computed clusters has been assessed by e.g. Zhang and Zhao, who study the reliability of hierarchical clusters from gene expression data [216] and propose a resampling algorithm for building a consensus tree; and Van Dongen in [202] who introduces criteria for evaluating clustering results using a novel inter-cluster distance. It must be noted that the quality of a protein family computation cannot be truly assessed in the absence of an objective external criteria based on biological knowledge.

Here we describe a method that integrates results of multiple classifications in a single scheme, using an election algorithm. First, we formulate the problem as an instance of *consensus clustering*. We propose a heuristic that efficiently realizes the computation in practice. Finally, we compare our method to others, and illustrate the quality of results in the case of protein families computed for Génolevures Hemiascomycetous yeasts.

Agreement between clusterings

Definition 1 *A* clustering Π *is a partition of a* data set $D = \{d_i\}$ *into* **disjoint subsets** $\Pi_1, ..., \Pi_k$ *called* clusters*, such that* $D = \bigcup_1^k \Pi_i$.

In what follows we will use the terms clustering and partition, and cluster and subset, interchangeably.

Let the number of elements in D and in cluster Π_i be n and n_i respectively. Let Π' be a second partition of D, $\Pi' = \{\Pi'_1, ..., \Pi'_{k'}\}$, with cluster sizes n'_i.

Most criteria for comparing clusterings are based on the *confusion matrix* [111] which measures the size of the intersection between the clusterings.

$$M_{ii'} = |\Pi_i \cap \Pi'_{i'}|$$

Traditional measures of the proportion of agreements between two clusterings are the *Rand index* [156] and the *Jaccard index* [98]. As the expected value of the Rand index does not take a constant value[1], the *adjusted Rand index* [92] it preferred; it assumes the generalized hypergeometric distribution as the model of randomness, and provides an index bounded above by 1 with constant expected value 0.

A well-known measure on the space of partitions is the *equivalence mismatch coefficient* emc ([129] p. 241), which is precisely the Hamming distance for binary vectors if the set of all pairs $\langle d_i, d_j \rangle$ is enumerated, and a partition is represented by a characteristic vector defined on this enumeration. One may interpret emc in terms of edge numbers of complete graphs, where it represents the number of node moves needed to convert one partition into another.

Another category of criteria for comparing clusterings is based on set cardinality. The *projection number* π of Π onto Π', defined as $\pi_\Pi(\Pi') = \sum_i \max_{i'} M_{ii'}$, is an asymmetric index of the degree

[1] When the cluster size is small, the Rand index tends towards 1 as the number of clusters increases

of refinement of one partition versus another. The same index normalized by n is introduced in [18], and a symmetric version is defined in [126]. Laresen and Aone [115] use the following asymmetric criterion:

$$L(\Pi, \Pi') = \frac{1}{k} \sum_i \max_{i'} \frac{2M_{ii'}}{n_i + n_{i'}} .$$

Based on the projection number van Dongen introduces a distance [201] between two partitions Π and Π':

$$d(\Pi, \Pi') = 2n - \pi_{\Pi}(\Pi') - \pi_{\Pi'}(\Pi) . \tag{2.1}$$

This gives the shortest distance between Π and Π' in a certain undirected weighted graph constructed on the set of all partitions of D, where two partitions are connected by an edge if one can be obtained from the other by joining two of its sets. The weight of the edge is the size of the smallest of two sets. In this construction $d(\Pi, \Pi')$ is the length of the shortest (weighted) path between Π and Π'.

Consensus clustering

Combining the strengths of different clustering algorithms is the focus of research on clustering ensembles. This problem can be seen as finding the median partition with respect to given partitions.
Consensus clustering (CC): Given k partitions, $\Pi^1, ..., \Pi^k$, find a consensus partition Π that minimizes $S = \sum_{i=1}^{k} d(\Pi^i, \Pi)$ where d is a distance.

The first mathematical treatment goes back to Régnier [157], where the problem was transformed into an integer problem and a branch and bound solution was proposed. The problem is proven to be NP-complete [14]. Wakabayashi gives an in-depth analysis of the median partition problem in [208] and concludes that approximating relations with a transitive one is NP-complete in general.

CC is NP-complete in general, yet it is not known whether it is NP-complete for any particular k. The case of $k = 1$ is trivial. The case of $k = 2$ is also simple since any of the partitions solves the problem optimally. Nothing is known for $k \geq 3$.

If the partition Π of the dataset D, $|D| = n$ to discover is not necessarily one of the original partitions $\Pi_1, ..., \Pi_k$, then the size of the potential search space corresponds to the *Bell numbers* [17] or equivalently to the sum of Stirling numbers of second kind for all $m, 1 \leq m \leq n$ (see [109] on [189]).

In light of its hardness, exactly solving large instances of **CC** is intractable. Even so, exact methods have been implemented. Grötschel and Wakabayashi [77] give a cutting planes solution, which works well for n in the hundreds.

A variety of approximations have been applied to this problem. For example Filkov [61] proposes a simple approximation that works by dramatically reducing the search-space.
Factor-2 approximation: Given an instance of **CC**, select a partition Π among partitions $\Pi_1, ..., \Pi_k$ that minimizes $S = \sum_{i=1}^{k} d(\Pi^i, \Pi)$.

The time complexity is $O(k^2 n)$ since it takes $O(n)$ to compute the distance between any two partitions, and there are $O(k^2)$ pairs. This algorithm is a factor-2 approximation to **CC**.

Many heuristics based on local search have been studied in application to **CC**, such as simulated annealing that explores the search space by one-element moves, and a greedy algorithm [61].

The problem of finding a consensus clustering can be approached from a graph-based, combinatorial or statistical perspective. Several methods exist for discovering a consensus clustering solution from many multiple partitions: co-association methods [63], voting approach [197], information-theoretic approach [126], hypergraph partitioning methods [190] and using mixture models [196].

2.1.2 Results

Comparing protein families

Given a set of m proteomes $P_1, ..., P_m$, we are interested in computing the protein families. A proteome $P = \{p_i\}$ is defined as a set of proteins. The universe of proteins $\mathcal{P}$ is defined over the proteomes under study as $\mathcal{P} = \bigcup P_i$. In what follows we suppose that a protein p can appear only in one proteome, and that only once[2].

As described earlier any method for protein family computation first selects the support data, then computes the families. We suppose that the filtering algorithm[3], α, defines a subset $\mathbb{P}^{\alpha} = \bigcup P_i^{\alpha} \subseteq \mathcal{P}$, the set of proteins over which a family computation will be performed.

Distance and similarity

Definition 2 *A* family computation F *over* $\mathbb{P}^{\alpha}$ *defines a partitioning of* $\mathbb{P}^{\alpha}$ *into disjoint subsets* $F = \{f_1, ..., f_k\}$ *such that* $\mathbb{P}^{\alpha} = \bigcup f_i$. *The subsets* f_i *are called* families.

Given two family computations $F^1 = \{f_1^1, ..., f_{k_1}^1\}$ over $\mathbb{P}^{\alpha_1}$ and $F^2 = \{f_1^2, ..., f_{k_2}^2\}$ over $\mathbb{P}^{\alpha_2}$, the *confusion matrix* can be used to measure the similarity:

$$M_{ij} = |f_i^1 \cap f_j^2|, f_i^1 \in F^1, f_j^2 \in F^2.$$

Note that the fact that relative complements $\mathbb{P}^{\alpha_1} \setminus \mathbb{P}^{\alpha_2}$ and $\mathbb{P}^{\alpha_2} \setminus \mathbb{P}^{\alpha_1}$ may not be empty is not a problem since the only implication is that certain M_{ij} values will be 0.

The confusion matrix can be equivalently represented by a bipartite graph in a straightforward manner. Let $\mathsf{FRel} = \langle V, E \rangle$, where the vertex set $V = V^1 \cup V^2$ is labeled by family names in F^1 and F^2 respectively, and the E is the set of edges weighted by the values in M_{ij} where $M_{ij} > 0$. That is, for two families $f_i^1 \in F^1$ and $f_j^2 \in F^2$ with an element in common, two vertices with labels f_i^1 and f_j^2 are created as well as an edge $e = \langle f_i^1, f_j^2 \rangle$ such that $w(e) = M_{ij}$.

Given a graph FRel, its strongly connected components $\{c_1, ..., c_m\}$ represent relationships between families in F^1 and F^2.

Definition 3 *The* similarity score *of a strongly connected component* $c = \langle V_c^1 \cup V_c^2, E_c \rangle$, *where* $V_c^1 \subset V^1$, $V_c^2 \subset V^2$ *and* $E_c \subset E$ *is*

$$s(c) = \max_{e \in E_c} w(e). \tag{2.2}$$

The *similarity score* for FRel is defined as

$$s(\mathsf{FRel}) = \frac{1}{n} \sum_{1 \leq i \leq m} s(c_i). \tag{2.3}$$

This measure is symmetric and provides information on what is preserved between two family computations.

[2]Since we are only interested in families, we do not introduce any information about the genes coding for these proteins (chromosomes, relative positions, etc.), in this way, a simple set-based approach is sufficient

[3]Even if the filtering criteria are not the subject of this study, it must be noted that they have a great impact in practical applications since they affect the distance/similarity values

Making a choice The basic case we have just considered consists of two family computations F^1 and F^2. Connected components $\{c_1, ..., c_m\}$ define all the local distances between families in two original partitions. Let us suppose that the comparison $\mathsf{FRel}^{1,2}$ has exactly k non-singletons (indices $[1..k]$): $\{c_1, ..., c_k, ...c_m\}$.

Having made the comparison $\mathsf{FRel}^{1,2}$ how do we choose the families that will form the solution? We propose to consider solutions locally, connected component by connected component, using exclusively families in F^1 or in F^2. That is for a non-singleton component $c = \langle V_c^1 \cup V_c^2, E_c \rangle$ we are only allowed to pick either V_c^1 or V_c^2. The rationale behind this choice strategy is that we are not allowed to invent classifications *ab initio*, but only to choose between the existing.

Definition 4 *A* family choice F^c *is defined by* $\{V_1^{c_1}, ..., V_k^{c_k}\} \cup \{V_{k+1}, ..., V_m\}$*, where* c *is a vector of size* k *with elements taking values in* $[1, 2]$.

A family choice F^c defines a partition of a certain subset of the protein set $\mathbb{P}$, namely the partition containing proteins of the families in F^c. The number of family choices is equal to the number of different vectors c of size k, that is 2^k.

Lemma 1 *For any two family choices* $F^{c'}, F^{c''}$ *their cumulative distances to the original partitions* F^1 *and* F^2 *are the same, that is* $d(F^{c'}, F^1) + d(F^{c'}, F^2) = d(F^{c''}, F^1) + d(F^{c''}, F^2)$.

Proof: For any family choice F^c the cumulative distance is $d(F^c, F^1) + d(F^c, F^2) = \sum_1^m d(c_i)$ where m is the number of connected components in the family relationship graph $\mathsf{FRel}^{1,2}$.

Definition 5 *A* reference set *is the set of connected components* C *used as the choice basis for the final partition.*

Lemma 1 indicates that all the consensus choice partitions are equivalent wrt. the reference set. This implies that the concrete choice is dependent on the needs of the application (see for a discussion section 2.1.3).

Consensus for families

Let us consider n family computations $\{F^1, ..., F^n\}$ over their respective support sets of proteins $\mathbb{P}^{\alpha_1}, ..., \mathbb{P}^{\alpha_n}$. We denote by $\mathbb{F} = \bigcup F^i$ the set of all families.

Rules of the game Having made all the $n(n-1)/2$ possible comparisons $\mathsf{FRel}^{i,j}$, how do we exploit this result for families computation?

As **CC** is NP-complete, the key to the approximation algorithm is to reduce the search-space. We propose to do so by only considering the families belonging to $\mathbb{F}$ in the manner analogous to the discussion in section 2.1.2.

Then any *acceptable solution* F is composed only of the families from $\mathbb{F}$ that are disjoint, that is $\forall f^i, f^j \subset F, f^i \cap f^j = \emptyset$. In this manner an acceptable solution defines a partition of some subset of $\mathbb{P}$. The set of all acceptable solutions $\mathcal{F}$ is a subset of the powerset of $\mathbb{F}$, $\mathcal{F} \subset 2^{\mathbb{F}}$. The inclusion is strict since only empty intersections are accepted. Thus we have $|\mathcal{F}| < |2^{\mathbb{F}}| = 2^{|\mathbb{F}|}$.

While the size of $\mathcal{F}$ is dramatically smaller than the whole search-space (whose size is equal to the n-th Bell number), this reduction of search-space is not sufficient and the corresponding exact algorithm remains NP-complete.

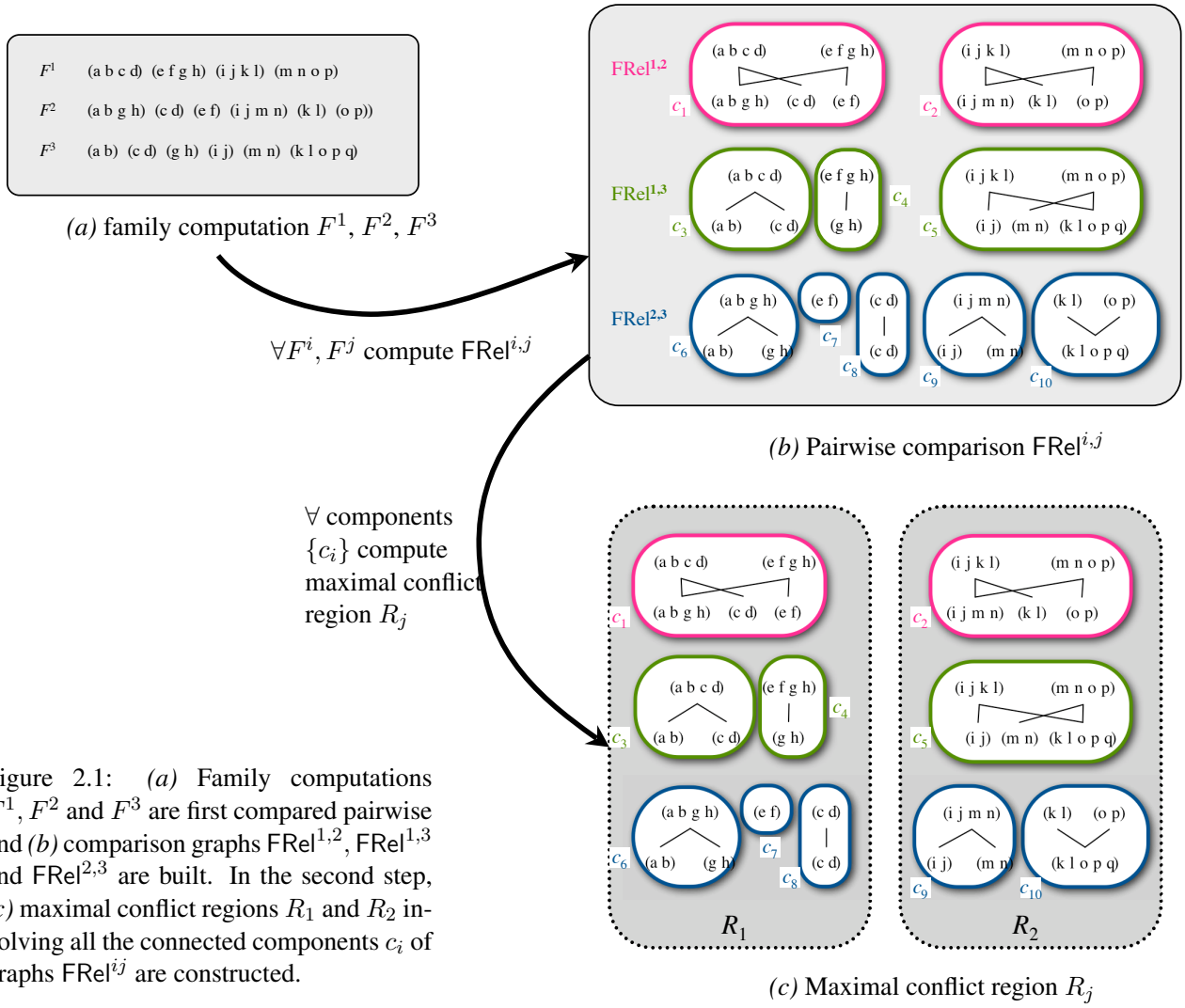


Figure 2.1: *(a)* Family computations F^1, F^2 and F^3 are first compared pairwise and *(b)* comparison graphs $\mathsf{FRel}^{1,2}$, $\mathsf{FRel}^{1,3}$ and $\mathsf{FRel}^{2,3}$ are built. In the second step, *(c)* maximal conflict regions R_1 and R_2 involving all the connected components c_i of graphs FRel^{ij} are constructed.

Definitions

Definition 6 *A protein $p \in \mathbb{P}$ belongs to at most n distinct families in $\mathbb{F}$ defined by function φ:*

$$\varphi : \mathbb{P} \rightarrow 2^{\mathbb{F}}, \quad \varphi(p) = F_p = \{f \in F^i \mid p \in f\},\ |F_p| \leq n.$$

Let us now consider all pairwise comparisons $\mathsf{FRel}^{i,j}$. A comparison graph $\mathsf{FRel}^{i,j}$ can be decomposed into its $m^{i,j}$ connected components as defined in section 2.1.2, $C^{i,j} = \{c_k\}, k \leq m^{i,j}$. The set of connected components over all of the comparison graphs is defined as

$$\mathbb{C} = \bigcup_{1}^{n(n-1)/2} C^{i,j} = \{c_k\},\ k \leq \sum m^{i,j}.$$

Definition 7 *A family f can appear in at most $n(n-1)/2$ distinct connected components in $\mathbb{C}$ defined by function κ.*

$$\kappa : \mathbb{F} \rightarrow 2^{\mathbb{C}}, \quad \kappa(f) = \{c \mid f \in c\}.$$

We call the set C_p the set of p-components.

Then, for a protein p we can define by composition the set of *p-components* as the set of all connected components where p is a member. We will denote this function $\sigma = \varphi \circ \kappa$.

$$\sigma : \mathbb{P} \rightarrow 2^{\mathbb{C}}, \quad \sigma(p) = \{c \mid \exists\, f \in c \,\text{s.t.}\, p \in f\}$$

Note that for C_p, the inverse image given by σ induces a set of proteins $P = \{p_i\} = \sigma^{-1}(C_p)$ such that all $p_i \in P$ belong to some family f that itself belongs to a component $c \in C_p$. We will call this set of proteins a *support set* for a given p-component.

We say that two sets of p-components C_p and C_q *have an intersection*, noted by $C_p \cap C_q \neq \emptyset$, if their corresponding support sets have an intersection, that is $\sigma^{-1}(C_p) \cap \sigma^{-1}(C_q) \neq \emptyset$.

Definition 8 *For a given $\mathsf{FRel}^{i,j}$, we say that a set $R \subset 2^{\mathbb{C}}$ is a* conflict region *if and only if $\forall\ c_k, c_l \in R$, $c_k \cap c_l \neq \emptyset$. We say that a conflict region R is* maximal *if and only if $\forall\, c_m \in \mathbb{C}$ such that $c_m \notin R$ and $\forall\, c_k \in R$, $c_m \cap c_k = \emptyset$.*

Consensus Let $\mathbb{R}$ be the set of all maximal conflict regions. A conflict region R has a support set of proteins $P_R \in \mathbb{P}$, $P_R = \bigcup_{c \in P_R} \sigma^{-1}(c)$. P_R has n subsets (n being the number of family computations) defined by $P_R^i = \{\varphi^{-1}(f) \mid \exists\, c \in R \,\text{s.t.}\, f \in c \,\text{and}\, f \in F^i\}$, each corresponding to the proteins belonging to families of a particular family computation F^i.

Consensus clustering over the reduced search space defined in section 2.1.2 can be reformulated, using the definitions of section 2.1.2, in terms of minimum set cover. The idea is to cover exactly one of the sets among P_R and $\{P_R^i\}$ with the support sets of the connected components constituting a given conflict region. A cost function w on $\mathbb{C}$ can be given by the van Dongen distance (equation 2.1) or our similarity score (equation 2.2), for example.

Minimum exact cover (MDC): Let R be a maximal conflict region. Let P_R and $\{P_R^i\}$ be the support set of proteins of R and the family computation-specific sets of proteins, respectively. We denote by $P_c = \sigma^{-1}(c)$ the support set of proteins for a connected component that is included in the region R. Find a subset of connected components $C \subseteq R$ such that: (i) every protein from $\bigcup P_c, c \in C$ appears in one and only one of P_c[4], (ii) there exists P among P_R and $\{P_R^i\}$ entirely covered by the support sets in C, that is $P = \bigcup P_c, c \in C$, and (iii) C minimizes $S = \sum_c w(c), c \in C$.

The consensus family computation can then be defined as a MDC solution for all $R \in \mathbb{R}$. This problem is a variant of minimum cover problem with an additional condition for the sets to be disjoint, and is known to be NP-complete [68].

[4]That is for any $c_i, c_j \in C$, their intersection is empty.

Efficient heuristic Two further relaxations make the problem tractable. First, we do not require to have all the $n(n-1)/2$ comparisons. Second, we do not require to have an *exact* cover. It is the second criterion that allows our heuristic to run in polynomial time. The obvious consequence of this relaxation is that not all of the proteins in $\mathbb{P}$ will be found in the result. This is not an issue in the context of protein families since there is no reason to suppose that all proteins from $\mathbb{P}$ belong to families of cardinality higher than 1.

Let us consider n family computations and $m \leq n(n-1)/2$ family comparisons $\mathsf{FRel}^{i,j}$. Each element $c_i \in \mathbb{C}$ has a distance (say, van Dongen distance) value d_i and a similarity value s_i. Starting from the maximal conflict regions in $\mathbb{R}$ we build a conflict graph.

Definition 9 *For family comparisons* $\mathsf{FRel}^{i,j}$ *we define a* conflict graph $G = \langle V, E \rangle$ *as follows.*

- *The vertex set* V *is labeled by elements from* $\mathbb{C}$*, and we have* $|V| = |\mathbb{C}|$*.*
- *The edge set* E *is defined by all non-empty intersections between connected components* $c_i, c_j \in \mathbb{C}$*, that is* $e = \langle c_i, c_j \rangle \in E$ *if and only if* $c_i \cap c_j \neq \emptyset$*.*

We distinguish two disjoint classes of edges $E = E_p \cup E_f$*, those that are only based on proteins* E_p *and those that correspond to families* E_f*. The latter means that there is a non-empty intersection on the level of families, that is* $\kappa^{-1}(c_i) \cap kappa^{-1}(c_j) \neq \emptyset$*, while the former is the complement* $E \setminus E_f$*.*

By construction the set of connected components of G is exactly the set of conflict regions $\mathbb{R}$.

The consensus computation works by resolution of conflicts for all connected components of a conflict graph G and is given in the algorithm 1. The conflict resolution is done by a voting procedure *vote*.

Complexity The complexity for computing strongly connected components is known to be $O(V+E)$ for a graph $G = \langle V, E \rangle$ [192]. The Condorcet election complexity is $O(p^2)$ for p voters (that is the number of proteins in a given conflict region). Let us denote by m the largest set out of n family computations: $m = \max_i |F^i|$. The largest possible conflict region will involve all proteins in $\mathbb{P}$. The worst-case complexity of algorithm 1 is then $n^2 O(m^2) + O(|\mathbb{P}|^2)$.

2.1.3 Application and Discussion

Algorithm 1 has been used to calculate protein families for *Saccharomyces cerevisiae* and the 8 fully sequenced species of Hemiascomycetous yeasts in the context of the Génolevures project [51, 178]. The following is a summary of the key steps in this application.

Data, alignments and filtering The data consists of 5 proteomes of *S. cerevisiae*, *Candida glabrata*, *Kluyveromyces lactis*, *Debaryomyces hansenii* and *Yarrowia lipolytica*. These proteomes taken together comprise the total set of proteins $|\mathcal{P}| = 49145$. We produced complementary alignments using Blast and Smith-Waterman filtered according to scoring criteria[5] and an approximation to *homeomorphy* (see [214]). These criteria produce four datasets of pairwise similarity: $\mathbb{P}^{\mathrm{SH}}$ corresponding to Smith-Waterman homeomorphic alignments, $\mathbb{P}^{\mathrm{SNH}}$ to Smith-Waterman alignments without the restriction of homeomorphy, $\mathbb{P}^{\mathrm{BH}}$ corresponding to Blast homeomorphic alignments and $\mathbb{P}^{\mathrm{BNH}}$ to Blast non-homeomorphic alignments.

Clustering TribeMCL software [56] was used for clustering. Three different inflation coefficients (1.2, 2.4 and 4.0) have been applied to each dataset, thus resulting in 12 different clustering results.

Parameter tuning Four parameters were added to the algorithm 1 in order to make it efficient in practice. The first three parameters help to break too large conflict regions. The last one realizes the choice policy.

[5] Based on statistical analyses not detailed here

Algorithm 1 Family computation algorithm by weak consensus

Require: $G = \langle V, E_c \cup E_f \rangle$ the conflict graph, a voting procedure $vote$
  Let L be an empty list of components
  for all $\mathcal{C}$ connected component of G **do**
    # First, determine D_i and S_i scores for all components of this conflict region
    for all c_i vertex in $\mathcal{C}$ **do**
      Let $D_i = 0, S_i = 0, n = 0$
      for all $\langle c_i, c_j \rangle$ such that $\langle c_i, c_j \rangle \in E_f$ **do**
        $D_i \mathrel{+}= d_j$, $S_i \mathrel{+}= s_j$, $n{+}{+}$
      end for
      $D_i = d_i + D_i/n$
      $S_i = s_i + S_i/n$
    end for
    # Second, determine votes for all proteins of this conflict region
    for all p protein of $\sigma^{-1}(C)$ **do**
      Let $\mathcal{V}_p$ be the set of votes for p that is equal to $\sigma(p)$ ordered by $D_i \nearrow$ and by $S_i \searrow$
    end for
    Let C_w be the result in order of $vote(\mathcal{V}_{p_1}, ..., \mathcal{V}_{p_k})$, where $k = |\sigma^{-1}(C)|$
    # Third, compute the most consensual cover
    Let $P = \emptyset$ be the set of covered proteins
    for all $c \in C_w$ in winning order **do**
      if $\sigma^{-1}(c) \cap P = \emptyset$ **then**
        $P = P \cup \sigma^{-1}(c)$
        push L, c
      end if
    end for
  end for
  return L

1. $S_{\max}$ is the maximal allowed size of the support protein set for conflict regions.

2. $f_{\max}$ is the maximal number of families allowed in a conflict region.

3. p is the percentile above which families are removed from R; it is determined based on the distribution of family sizes in a given conflict region R.

4. A choice strategy (c.f. section 2.1.2). Given a connected component $c = \langle V_c^1 \cup V_c^2, E_c \rangle$, either systematically choose V_c^i such that $|V_c^i| = \max(|V_c^1|, |V_c^2|)$ or systematically choose V_c^i such that $|V_c^i| = \min(|V_c^1|, |V_c^2|)$.

The voting procedure *vote* has been implemented as *Condorcet election* [45].

Consensus families Various parameter ranges were tested resulting in 256 family computations. The final result contains 7927 protein families (including singletons). Evaluation of the result was performed by comparison to manually-curated literature standards, to reciprocal best hit partitions (RBH) [158], and to Biofacet [73] single-linkage clustering. Results using our consensus method were qualitatively and quantitatively more satisfying, and have been adopted by the Consortium.

The first example is provided by families GL3C0253 and GL3R1049, respectively homologs to RPL24 ribosomal-like protein 24, and homologs to ribosomal protein L30 of the large (60S) ribosomal subunit. Instead of producing two groups of homologs, RBH clustering produces three groups corresponding to three species groups.

A second example illustrates the too-large families that are typically produced by blind MCL or RBH clustering, in this case an impossible family over 500 members. Our consensus algorithm splits this group into 30 families which are in fact different sets of protein kinases, annotated as "BCK1 ser/thr protein kinase" (GL3C0408, 18 members), "PKC1 protein Kinase C" (GL3C2603, 9 members), "cytoplasmic serine/threonine protein kinase" (GL3R0210, 21 members), "protein kinase involved in morphogenesis and septin checkpoints" (GL3C0403, 18 members), "PSK1 and PSK2 proteins, PAS domain containing S/T protein kinases" (GL3R0626, 11 members), etc.

These families were used in a variety of specific studies by the Génolevures Consortium and were recently expanded as new yeast genomes sequenced [40].

2.2 Gene fusions and fissions

Gene fusion and fission events are key mechanisms in the evolution of gene architecture, whose effects are visible in protein architecture when they occur in coding sequences. Until now, the detection of fusion and fission events has been performed at the level of protein sequences with a *post facto* removal of supernumerary links due to paralogy, and often did not look for events defined only in single genomes. We propose a method for the detection of these events, defined on groups of paralogs to compensate for the gene redundancy of eukaryotic genomes, and apply it to the proteomes of 12 fungal species. We collected an inventory of 1680 elementary fusion and fission events. In half of the cases, both composite and element genes are found in the same species. Per-species counts of events correlate with the species genome size, suggesting a random mechanism of occurrence. Some biological functions of the genes involved in fusion and fission events are slightly over- or under-represented. As already noted in previous studies, the genes involved in an event tend to belong to the same functional category. We inferred the position of each event in the evolution tree of the 12 fungal species. The event localization counts for all the segments of the tree provide a metric that depicts the "recombinational" phylogeny among fungi. A possible interpretation of this metric as distance in adaptation space is proposed.

As the number of complete genome sequences increases, comparative genomics unveils the mechanisms of gene and genome evolution. Duplication, sequence divergence, and recombination are the major mechanisms at work in gene evolution [53]. Recombinational events such as translocation, inversion or segmental duplication can create accidental fusion of DNA sequences associated with different genes, or conversely the fission of a gene into several parts. Potentially, these events can create new genes from already existing parts, or reciprocally shuffle genes into sub-parts across a genome. These rare events participate in the evolutionary history of the species, and must be taken into account in genome rearrangement models. Methods to inventory gene fusion and fission events on a large biological scale can provide insights about the multimodular architecture of proteins [55, 215, 149], as well as a metric between genomes independently of the mutation rate [55, 184, this work]. Computational detection of fusion and fission events uses sequences from several species, usually proteome sequences. This implies that the detection is only performed in the coding regions, a reasonable approximation as non-coding regions evolve faster. After a recombinational event, gene fusion can occur and is situated either in coding or non-coding sequences. In non-coding sequences, gene fusion can give rise to the misregulation of the expression of a gene now under the control of the *cis*-regulatory sequence of another gene. For instance, the cells in the majority of human prostate

cancers bear a gene fusion where the regulatory sequence of the TMPRSS2 gene controls the coding sequence of a transcription factor, either ERG or ETV1, resulting in over-expression of this factor and hence anarchic growth [195]. In coding sequences, gene fusion results in the assembly of a new gene, thereby allowing the emergence of new functions by the accretion of peptide modules into multidomain proteins. As an example, the Tre2 (USP6) oncogene emerged from the fusion of the USP32 and TBC1D3 genes in the hominoid lineage of primates, and it has been proposed that this has contributed to hominoid speciation [151]. Gene fission splits a gene into several parts and can be produced by either recombinational events or single base events, such as frameshift and nonsense mutations. The outcome can be the misregulation of the expression of a gene when a *cis*-regulatory sequence is concerned. Due to the fast evolution of non-coding sequences, the detection of fission events involving such sequences will be out of reach when comparing the genomes of distant species. Loss of continuity in the coding sequences, produced by any of the above events, can give rise to a less complex protein by domain depletion, as, for instance, in the *monkey king* family of genes in *Drosophila* species [209]. Gene fission events can also produce pseudogenes [43].

In completely sequenced prokaryotic genomes, fusions occur more frequently than fissions, and there is no striking bias in the functions of the genes that have undergone these events [184]. The same conclusions hold true in the three kingdoms of the tree of life, by considering the structural domains of the proteins [113]. In mammalian genomes, the close evolutionary distances make it possible to detect fusion and fission events in coding and non-coding sequences; the events between coding sequences involve genes whose protein products have a significant propensity to interact [217]. Fusion events between proteomes have been used to predict protein-protein interactions [54, 121] with some degree of success, in particular metabolic enzymes for which stable protein-protein interactions in one species could be advantageously substituted by the products of fusion events in other species. Altogether, such large-scale comparisons of proteomes revealed that about 4% of the proteins are the products of fused genes and 9% are encoded by genes which are fused in other genomes [55]. These methods work at the level of individual genes, which is an appropriate approach in prokaryotes as the number of duplicated genes is low.

We present here a large scale computation method for detecting gene fusion and fission events in eukaryote genomes, even when they have a noticeable amount of internal gene redundancy. Contrary to the methods published so far, we directly worked at the level of groups of paralogs. We applied the method to the proteomes of a coherent phylogenetic group of species over a large evolutionary range. We chose to focus our study on 12 species covering the phylum of fungi in which a number of complete or near complete genomes are currently available, especially in the group of hemiascomycetes (yeasts). Nonetheless we also chose other ascomycete species as well as basidiomycete and zygomycete species (table 2.2.1). As the evolutionary distances between genomes are large, even inside the group of hemiascomycetes [51], the divergence of non-coding sequences is too high [39] to search for fusion events in them. Since our study is restricted to coding sequences, we employed complete proteomes to track fusion and fission events. In whole we detected 1103 fusion/fission events; the number of events in which a species is involved is correlated with the genome size of the species.

We chose to focus on genome evolution rather than individual domain structure of fusion proteins. Thus we computed the localization of each event in the evolution tree of the 12 fungal species, on the parsimonious assumption that a fusion or fission event happens once during evolutionary history [113]. The weighted counts of events localized in each segment of the phylogenetic tree provided a metric between species, independently of the mutation rate of the genes. From this perspective, it is apparent that some species have undergone massive genome shuffling.

The events relative to the hemiascomycetes is available in the Genolevures database [178] (`http://cbi.labri.fr/Genolevures/`).

Phylum	Sub-phylum	Species	Database	Reference
Ascomycota	Hemiascomycota	*Saccharomyces cerevisiae*	SGD	[11]
		Candida glabrata	Génolevures	[178]
		Kluyveromyces lactis	Génolevures	[178]
		Eremothecium gossypii	AGD	[87]
		Candida albicans	CandidaDB	[46]
		Debaryomyces hansenii	Génolevures	[178]
		Yarrowia lipolytica	Génolevures	[178]
	Euascomycota	*Neurospora crassa*	Broad Institute	[67]
		Aspergillus nidulans	Broad Institute	[66]
	Archeascomycota	*Schizosaccharomyces pombe*	Wellcome Trust Sanger Institute	[213]
Basidiomycota		*Cryptococcus neoformans*	Stanford Genome Technology Center	[119]
Zygomycota		*Rhizopus oryzae*	Broad Institute	Rhizopus seq. project(2005)

Table 2.1: The species are listed in order from the *Saccharomyces cerevisiae* reference and according to the phylogenetic tree computed by [62].

2.2.1 Materials and Methods

Proteomes The detection of fusion and fission events was performed on the proteomes of species belonging to the group of fungi, with some emphasis in hemiascomycetes as several complete genomes are available. Only complete, or near complete, genomes can provide sets of protein sequence data exhaustive enough to allow precise counts of events. Thus we restricted our study to genomes which were highly covered by sequences (table 2.2.1). When the sequence of a single protein is split into several entries in the proteome file, we deduced that these were sequences of exons and merged these entries to avoid false positive artifacts. A small number of sequences were omitted as they were too short (10 amino-acids or less) to be treated. In some proteomes, a part of the detected events may nonetheless be spurious, due to the quality of sequences and the accuracy of the gene models used to predict introns and coding sequences.

Algorithm As stated above, the algorithm works at the level of groups of paralogous proteins and extracts simultaneously fusion and fission events in several proteomes :

(**i**) For each proteome, we built a set of paralogous groups (hereafter named *P-groups*), based on sequence similarities between proteins. The set of P-groups is thus a partition of the protein set. Note that a P-group may consist only of one protein. Each P-group has a unique name made with an acronym of the species and a number; the acronym is built from the first two letters of the genus and the first two letters of the species, e.g. ASNI-1004 is a P-group of *Aspergillus nidulans*.

(**ii**) We then compared all proteomes using an all-against-all comparison of protein sequences. We filtered out the alignment results and converted each valid similarity relation between two proteins to a relation between two P-groups. Note that there are relations between P-groups belonging to different proteomes as well as relations between P-groups of the same proteome.

(**iii**) The detection of a fusion/fission event requires knowing the extent of the similarity regions between the relevant P-groups. We thus converted each P-group into a multiple alignment, which was in turn converted into a Hidden Markov Model (HMM); in the case of a P-group containing a single sequence, the multiple alignment step was skipped. As HMM-HMM comparisons are very computationally intensive, we restricted these comparisons to the relation between P-groups determined in step (ii), and extracted the coordinates of the aligned regions.

(**iv**) We define an *Event* as an n-ary relation between P-groups, at least one composite P-group (hereafter named *C-group*) and at least two element P-groups (named *E-groups*), which fulfill three constraints: the E-groups belong to the same proteome, they align on the C-group, the alignment

regions have no or reduced overlap on the C-group. Obviously, there could be more than one C-group in an event, as well as more than two E-groups. In [54] the term *component* was used for elements.

We considered all the P-groups and their relations, computed in steps (i) and (ii), as a directed graph where P-groups are nodes and each alignment relation is a pair of edges in opposite orientations. Using the above definition of an event, we recursively deleted edges of the graph according to the constraints; when two E-groups had overlapping alignment regions on a C-group, these two E-groups were merged with regard to their relation with the C-group. The events are extracted from the resulting graph as connected components (the term *component* here is used as defined in graph theory).

(**v**) At this stage, a parsimonious interpretation of the events from a phylogenetic point of view, led us to distinguish five types: *Fusion events*, where a single C-group is linked to E-groups issued from at least two species (figure 2.2, A); *Fission events*, where several C-groups are linked to a set of E-groups coming from a single species (figure 2.2, B); *Multiple events*, where several C-groups are all linked to several sets of E-groups (figure 2.2, C); *Undecidable events*, where a single C-group is linked to one set of E-groups coming from a single species, this case can neither be interpreted as a fusion event nor a fission one (not shown); *Complex events*, where several C-groups are linked to different sets of E-groups (figure 2.2, D).

(**vi**) Considering that complex events come from ubiquitous protein domains that are found in several protein architectures, we split these events into events of the four other types, at the expense of doubling some nodes in the graph.

The outcome of this method is an inventory of *elementary* events.

NOTES: The algorithm can find events inside a single proteome. Tandem duplication of the same domain within one protein is ignored by the algorithm. The P-groups that are neither C-groups nor E-groups are called *O-groups*, for *Other*. The software packages used and relevant parameters are provided in the original paper [52].

2.2.2 Fusion / fission metric

For each event, we mapped the species which contained E-groups or C-groups (figure 2.3) onto the phylogenetic tree underlying the 12 species [62]. Under the parsimonious assumption that any event occurred once during evolution, the event should be localised on the tree in one of the edges between the species containing E-groups and the species containing C-groups. Thus, we extrapolated the status of the internal, *i.e.* ancestral, nodes of the tree as either E-group containing node or C-group containing node: (**i**) all internal nodes belonging to a shortest path between two E-group containing species, are extrapolated as E-group containing nodes, *i.e.* the nodes 1, 2, 4, 5, 6, 7, 9 in figure 2.3 example; (**ii**) likewise, the status C-group containing node applies to internal nodes between C-group containing nodes, *i.e.* none in the example; (**iii**) the event is inferred to be localised on the shortest path between E-group containing nodes/species and C-group containing nodes/species, *i.e.* either on the edge [node7-node8] or on the edge [node8-*A. nidulans*] in the example; (**iv**) if a given species without status is connected to this last path and if it contains P-groups equivalent to some of the E-groups which defined the event, then the species is assimilated to an E-group containing species and the path can be shortened, *i.e. N. crassa* in the example shortens the path, leaving the [node8-*A. nidulans*] edge as the remaining path; (**v**) each of the n remaining edges receives a score of $1/n$, *i.e.* the [node8-*A. nidulans*] edge receives a score of 1 in the example.

In 4 of the 15 cases of multiple events, the mapping onto the tree brought about internal nodes in which the probable ancestral content in C-groups or E-groups could not be inferred, leading us to suppose that a particular fusion or fission occurred more than once over time. Here, we ordered

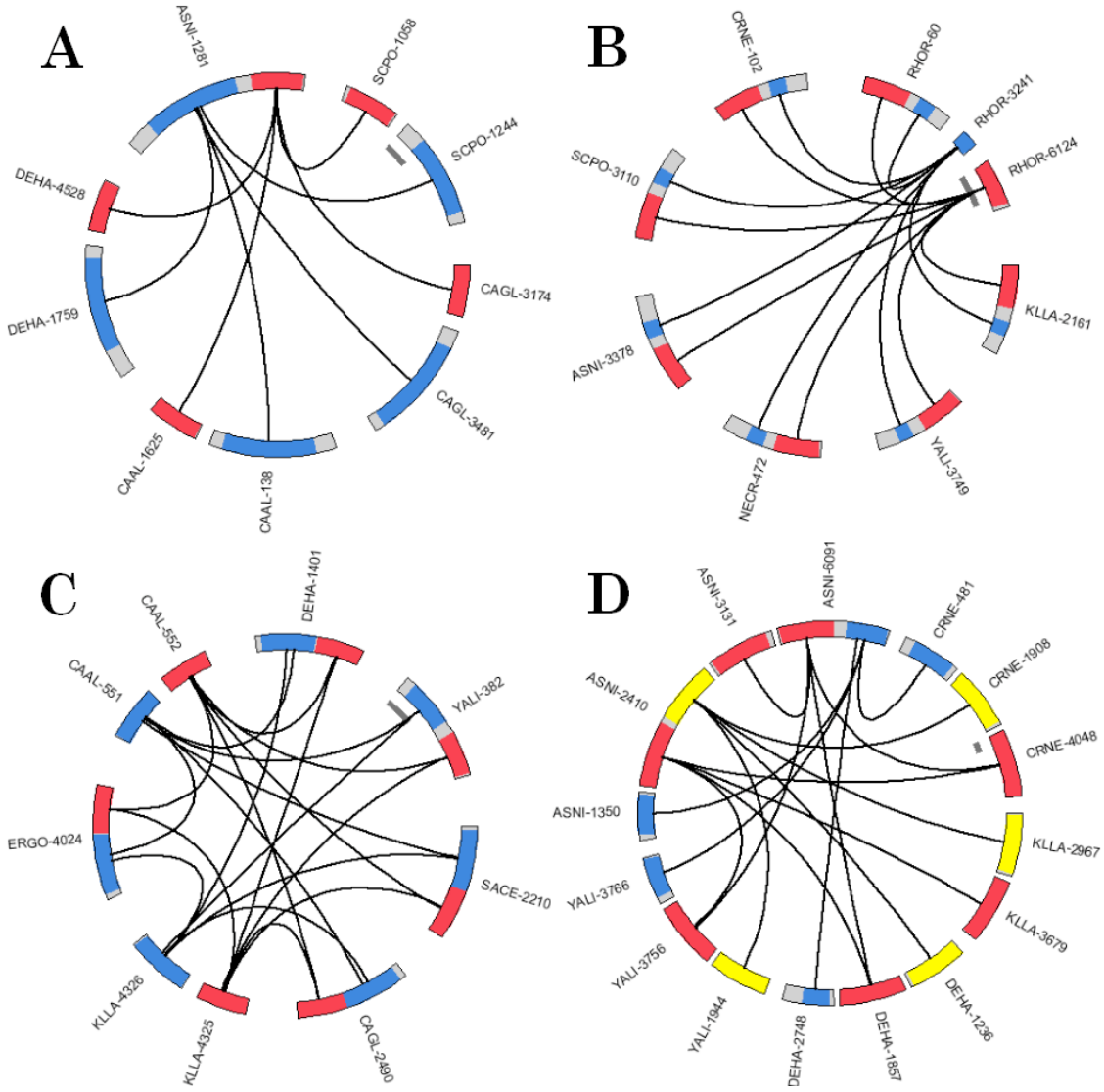


Figure 2.2: **A**: Fusion event, **B**: Fission event, **C**: Multiple event, **D**: Complex event, see Algorithm section for definitions. P-groups are drawn to scale and oriented clockwise. Colored areas represent alignment domains, white areas are non-aligned regions. Arcs symbolize relations of similarity between C-groups and E-groups.

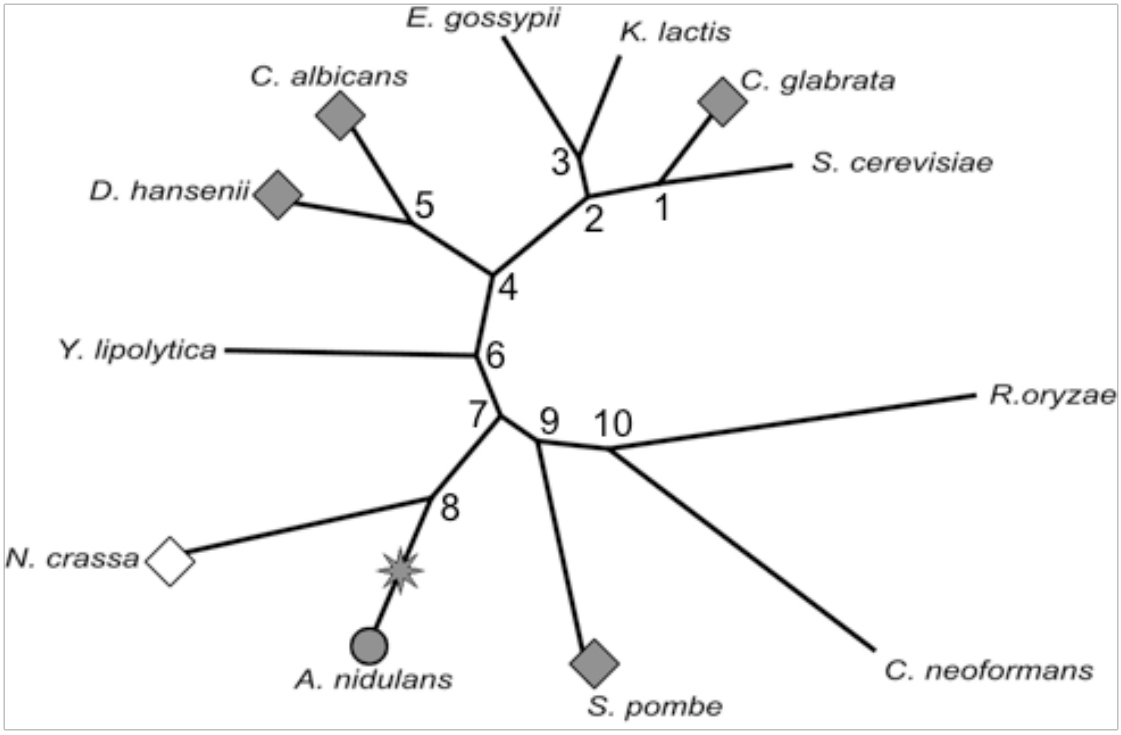


Figure 2.3: Event localization. This event is also represented in figure 2.2, A. *Grey circle*: species having a C-group. *Grey diamond*: species having E-groups as listed in the event. *Open diamond*: species having a P-group similar to one of the E-groups. *Grey star*: parsimonious localization of the event in the phylogeny.

the involved species in decreasing order of the number of uncertain internal nodes that were resolved when the species was removed. We then used each of these species in this order as the starting point of a shortest path, see preceding paragraph, and removed species until no uncertain internal nodes remained and all of the species were treated. At the end, we identified the minimal number of events necessary to take into account all the species which defined the multiple event and attributed scores to the relevant segments.

2.2.3 Results

We identified gene fusion/fission events in a coherent phylogenetic group of fungi, where completely sequenced and annotated genomes are available, especially in the hemiascomycete yeasts. This method only identified events which occurred inside protein coding genes, but, given the evolutionary distances between species [50], trying to detect events in intergenic regions would have certainly have been worthless.

We expected gene redundancy since we worked with eukaryotic genomes. If duplicated genes were involved in a fusion / fission event, this event could accordingly be counted several times. To counter this redundancy, we built a set of paralogous groups (*P-groups*) for each proteome. The clustering of several protein sequences inside a P-group was based on sequence similarity and the length of the alignment, to ensure that the proteins shared the same architecture. The set of P-groups is thus a partition of the protein set in a given species (see Dataset S1, online Supplementary Material for [52]). Our method is designed to detect events at the level of groups of paralogs (P-groups) and in several proteomes simultaneously (see Material and Methods). The method also finds events which contain E-groups and C-groups belonging to the same species. We detected 1103 events, 176 of them being complex events were subsequently split, giving altogether 1680 elementary events (table 2.2.3). These events only involve 12% of the P-groups over all the species, either as E-groups or C-groups. The *Euascomycota* and *Zygomycota* species happen to be the species the most involved in events; these species are those with the larger proteomes and hence the larger genomes. Indeed, we found a correlation between the genome size of a species and the number of events where it appears (figure 2.4), a relation also found in a large genome survey [102].

We estimated the value of the fusion over fission ratio to be 1.28 from the number of events classified either as fusion events or as fission events (see Material and Methods), although undecidable events (995 events, 2.2.3) could not be included in this calculation. This ratio is slightly in favor for fusion events which is in accordance with earlier studies [184, 113].

We then assessed the robustness of the events by removing all the P-groups of one species at a time and then by checking how many events remained (table 2.2.3, column *Exc.*). The number of events exclusive to one species ranged between 31 to 800, suggesting that the set of events is not saturated and that it will increase upon the addition of new species. These numbers, along with the manual curation of the events, indicated that *A. nidulans* and *R. oryzae* genomes were likely to have undergone a large-scale reshuffling.

Our method allowed us to retrieve well-known fusion examples, such as the event involving *S. cerevisiae TRP1*, *TRP3* genes [30] and their homologs in other species (event GFE-1104, see Dataset S2, online Supplementary Material for [52]), in which the corresponding polypeptides are separated entities in *Hemiascomycota* and fused in a single protein in *Euascomycota*, *Archeascomycota*, *Basidiomycota* and *Zygomycota*. Another well known example is the one which includes *S. cerevisiae URA2* gene [185]. This very ancient event is thought to have happened before the branching of the fungus phylum, but is still visible as every species kept E-groups and C-groups (event GFE-0970).

Instead of focusing on the individual domain structure of fusion proteins, we chose to consider each event from an evolutionary perspective of genome rearrangement. We thus needed to distin-

Species	Genome	Prot.	P	C	E	Inv.	Fus.	Fis.	Mul.	Und.	Exc.	Loc.	Rec.
S. cerevisiae	12.1	6710	5431	172	312	323	124	151	8	40	54	106	23
C. glabrata	12.3	5210	4377	152	251	299	124	138	8	29	36	88	2
K. lactis	10.7	5331	4601	150	330	334	127	161	9	37	64	102	15
E. gossypii	9.2	4725	4180	146	257	289	124	133	7	25	31	84	3
C. albicans	15.0	6165	5152	125	599	428	161	158	11	98	181	144	33
D. hansenii	12.2	6277	5114	194	379	405	178	150	11	66	72	123	12
Y. lipolytica	20.5	6431	5187	162	401	382	192	149	7	34	45	83	9
N. crassa	43.0	10427	9321	213	592	510	245	130	12	123	175	87	21
A. nidulans	31.0	9536	7404	514	671	664	298	149	10	207	459	95	171
S. pombe	14.0	4990	4078	152	304	318	155	112	7	44	61	68	4
C. neoformans	19.1	6578	5502	173	351	354	158	118	10	68	92	84	14
R. oryzae	46.1	17461	10349	682	2046	1062	244	184	12	622	800	220	554
Total	245.2	89841	70696	2835	5683	1680	376	294	15	995	1665	365	847

Table 2.2: Event statistics. Prot.: proteins; P: P-groups; C: C-groups; E: E-groups. Inv.: events where a species is involved; Fus.: fusion events; Fis.: fission events; Mul.: multiple events; Und.: undecideable events; Exc.: events which no longer exist if a species is removed from the dataset; Loc.: events where there are adjacent proteins between E-groups; Rec.: events with contain at least C-groups and E-groups of the same species. Genome sizes are given in Mbases. An event can concern several species, therefore the numbers of events on the Total line are not the sums of the counts per species. All E-groups are counted, even if they can be subsequently merged in events (see Material and Methods).

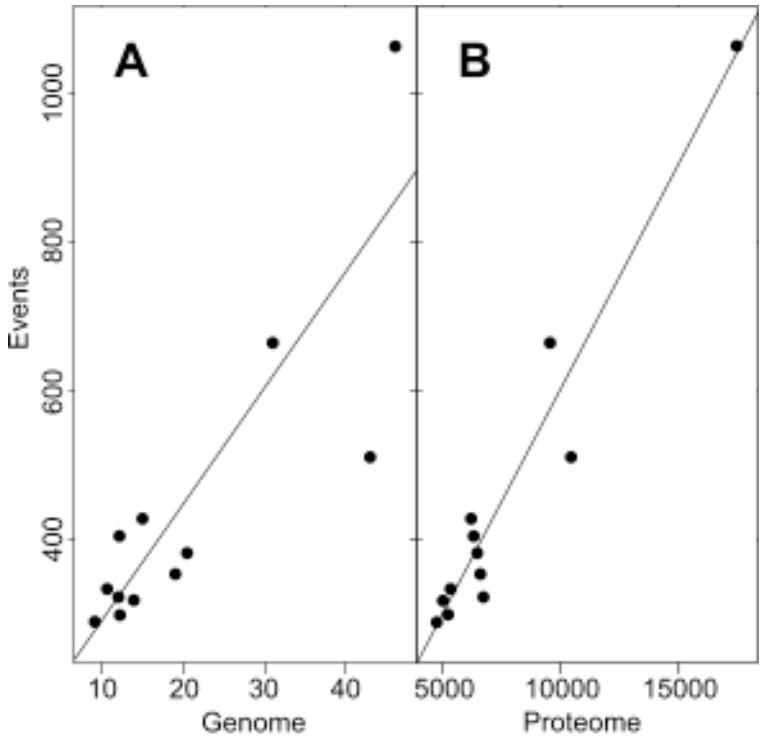


Figure 2.4: Density of events. Scatterplots of the number of fusion/fission events against (**A**) genome size in megabases, and (**B**) proteome size in number of proteins. Straight lines indicate the coefficients determined by bootstrap estimates of robust linear models.

guish two types of event. (**i**) The 365 events where at least one pair of E-groups correspond to adjacent genes on a chromosome, are likely to derive from nonsense or frameshift mutations which transform one coding sequence into two coding sequences or more. We did not take these events into consideration as they *a priori* do not involve genome rearrangement (table 2.2.3, column *Loc.*). (**ii**) The 1315 other events, which contained nonadjacent E-group members, have likely occurred through a recombination event and were therefore the basis of our computation. We then, computed the position of each of these latter events in the evolution tree of the 12 fungal species, derived from the study of [62], with the parsimonious assumption that a fusion or fission event might happen once during evolutionary history [113]. This tree is based on the comparison of the protein sequences translated from families of orthologous genes, and thus was called, in the framework of our study, the "mutation tree" (figure 2.5, A). Keeping the same topology, we computed the weighted counts of events positioned in each segment of the tree (see Material and Methods), and we changed the length of each tree segment accordingly to make a "recombination tree" (figure 2.5, B). The use of the event localization weighted counts as a metric dramatically changed the aspect of the tree, making it obvious that some species (*N. crassa*, *A. nidulans* and *R. oryzae*) underwent massive genome shuffling.

2.2.4 Discussion

Until now, the detection of fusion and fission events has been performed at the level of protein sequences with a *post facto* removal of supernumerary links due to paralogy. Also, earlier reports often did not look for events only defined in a single genome. We designed a large scale computation method to detect gene fusion and fission events in eukaryotes genomes taking into account their internal gene redundancy and thus operated at the level of groups of paralogs in the proteomes, named P-groups. The method basically consisted in building a graph of similarity relations between the protein sequences of several species and then pruning this graph according to rules specific to the definition of gene fusion/fission events. The method works simultaneously between every species as well as within species.

We obtained a set of 1680 elementary fusion and fission events in the coding sequences of 12 fungal species. The number of detected events for a species is related to its genome and proteome size, as it appears to be the case in any species of the tree of life, with few exceptions typically associated with parasitic or infectious lifestyle [184, 102]. The numbers of gene fusion/fission events confirm that these events are relatively rare [55, 184], albeit these numbers are provisional and underestimated as they are not saturated. Thus, the roster of detected events will very likely increase upon the addition of new species into the study.

The fusion/fission ratio of 1.28 was less large than in comparable studies [184, 113], but was still in favor of the fusions. From a phylogeny point of view, we can expect such a tendency, as its beneficial effect would be to permit either the gathering of several biochemical functions into a single polypeptide molecule, thereby reducing the regulation burden of the cell, or the creation of new functions in a scenario which congregates gene duplication, gene fusion and sequence mutation. In the evolution from prokaryotes through lesser eukaryotes and up to higher eukaryotes, a witness of this fusion rate propensity is the observation that proteins have more different domains per protein, along with a larger repertoire of domain combinations [112, 205].

As some of the genomes we used are thoroughly annotated, we could search for biases in the biological functions of the genes involved in fusion and fission events (see the original manuscript [52] for details). Only a few were found. Similar findings were reported in other studies [215, 102, 199, 91] although these functions do not appear to be the same in each report. This variation is not surprising since the different works were done on different sets of species, covering one or several

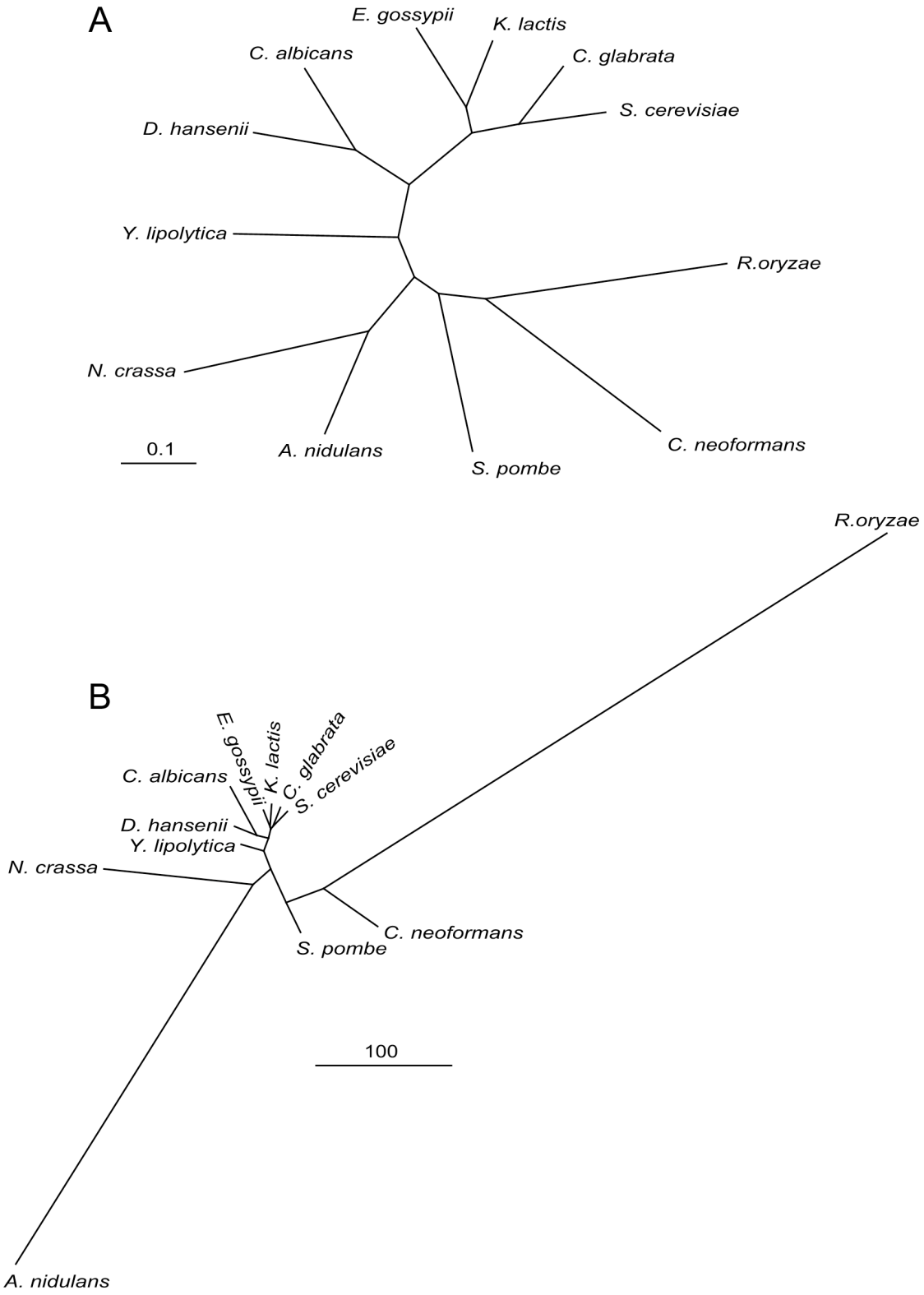


Figure 2.5: Fungal distance trees. A: Maximum likelihood tree based on accumulated mutations in 153 universally distributed fungal genes (excerpt from [62]). B: Rearrangement tree based on the topology of the mutation tree with a modification of branch lengths according to parsimonious localisations of fusion and fission events; the scale bar corresponds to weighted occurences of events.

kingdoms. In addition, the sets had unequal sizes and, as stated above, the number of events depends on the number of species. Nevertheless, the genes involved in an event in the fungal phylum tend to belong to the same functional category, a feature already found in other contexts [55, 215, 102].

Each event which does not involve adjacent genes on a chromosome, can be interpreted as a landmark of a recombinational event giving rise to gene fusion or fission. We positioned each of such events in the evolution tree of the 12 fungal species on the parsimonious assumption that each happens once during evolutionary history [113]. We only found 7 cases where two independent fissions were necessary for the event to be compatible with the phylogenetic tree (data not shown). For each segment of the tree, the weighted counts of positions provided a metric between the 12 fungal species. This metric is independent of the gene mutation rate, and hence of the "mutation" phylogenetic tree. Rather, the metric depends on another aspect of genome evolution, recombination and gene shuffling. Under this perspective, some species underwent massive genome shuffling, compared to species with more stable chromosome architecture. Other metrics have been proposed to account for a recombinational distance between species, such as a metric based on synteny breakpoints [25]. However this last metric can only be applied on relatively narrow evolutionary distances where synteny exists, such as the vertebrates phylum. In contrast, the fungi phylum encompasses larger distances, for instance even in the Hemiascomycota sub-phylum, synteny blocks shared by *Saccharomyces cerevisiae* and *Yarrowia lipolytica* are too few and far between [51]. The metric we propose deals with traces of recombination events which can persist even if a genome has been totally shuffled.

Several mechanisms of genome recombination could be put forward to explain the appearance of gene fusion and fission. Translocation or inversion can potentially fuse or split genes at their boundaries [35, 44]. Segmental duplication can potentially fuse or split gene at their boundaries, as well as put next to each other exon containing sequences of different origin [41]. Horizontal gene transfer in bacteria can account for 3% of the fused or split genes [113]. Horizontal gene transfer is a minor mechanism in fungi [82], but cannot be ruled out as a contributor for fusion/fission events. Partial copies of genes could be inserted in ectopic sites through retrotransposons, potentially creating chimerical genes at the insertion points [166]. Other plausible mechanisms would be transcription mediated gene fusion [22] or retroposition of trans-spliced genes [2]. Whatever the recombination mechanism, it is genetically easier to make a gene fusion than a gene fission [113], because in gene fusion one partner could bring its promoter and the other its terminator, whereas in gene fission, one of the offspring has to come under the control of a new promoter in order to be expressed.

During evolutionary time, genomes underwent recombinational events, some of which gave rise to gene fusion or fission, hence new genes and new proteins. Gene fusion and fission can abruptly change the length and composition of a gene, as opposed to point mutations which can alter gene content at a more continuous pace. Evolutionary pressure caused some of the genes produced by fusion or fission to be maintained and propagated until present time. Such genes could thus be considered as participating to the overall fitness and adaptation of a species. If we speculate that a species could be considered as a point in an "adaptation space," and ecological niches as regions of this space, we could propose the metric we defined as an indirect, or approximate, measure of distance between species in this space. The fact that there is no striking bias in the biological functions of the genes involved in gene fusion or fission, suggests that the recombinational events are basically random. This hypothesis has already been put forward, considering versatility and domain abundance in proteins [205]. Under this consideration, we could also propose that the metric we defined, does not need to be normalized for biological functions, as there is little bias.

The events relative to the hemiascomycetes are available in the Genolevures database [178] (`http://cbi.labri.fr/Genolevures/`).

Chapter 3

Uncovering evolutionary events by rearrangement analysis

3.1 Introduction

The large scale study of molecular evolution through the comparison of contemporary genomes is frustrated by the impossibility of knowing with certainty the architecture of the common ancestral genomes. Constructing plausible hypotheses about the structural characteristics of these ancestral architectures is a computational task whose results may provide deep insight both into the past histories of particular genomes and the general mechanisms of their formation. This task has two important difficulties: how can we guarantee that the solution is biologically plausible? how can we find these solutions in an efficient manner?

Analysis of genome rearrangements provides a measure for the evolutionary distance between species. Two closely related problems are considered in the study of genome rearrangements. The first problem is to find, by parsimony criteria and for a defined set of rearrangement operations, the exact number of such operations needed to rewrite one genome into another. The second problem is to compute a most parsimonious rearrangement scenario.

In the considered model (see section 3.2.1), two genomes defined on the same set of gene markers without duplications, are represented by signed permutations. Thus, the analysis of genomes leads to a combinatorial problem of transforming one signed permutation into another. The underlying theory is proposed by Hannenhalli and Pevzner [84, 172] for unichromosomal genomes based on reversals only; their main results consist in an exact formula for reversal distance, and the first polynomial time algorithm for computing a parsimonious reversal-based scenario between two signed permutations.

This theory was further adapted by the same authors to the multichromosomal case, taking into consideration a larger set of rearrangement operations: translocations, fusions and fissions as well as reversals. In [85], Hannenhalli and Pevzner devise a method that mimics all multichromosomal rearrangements by reversals operating on an unique permutation.

Ancestral reconstruction methods require three basic steps: identification of common markers in the contemporary genomes, construction of comparative maps of the genomes, and reconciliation of these maps using a criterion of maximum parsimony to reconstruct ancestral maps. Common markers can be identified using cytogenetic methods such as *chromosomal painting* [210] or through complete genome sequencing and subsequent search for chromosomal homology [203]. Mathematically, comparative maps are constructed by representing each genome as a signed permutation of the common markers. The goal of this encoding is not to align one genome against the other, but rather to compare the order of gene markers. Two main approaches to compare marker orders exist:

counting the differences between two genomes in terms of *breakpoints* [133, 162], and counting the minimal number of edit operations that transform one genome into another [84]. Both of these approaches define a *distance function* on the space of signed permutations. In this work we will follow the Hannenhalli and Pevzner approach [84] where the allowed edit operations are *fusion*, *fission*, *reciprocal translocation* and *reversal*. It was originally established that this *rearrangement distance* can be computed in polynomial time [84, 172]. Later work has improved these results establishing a linear-time algorithm for the rearrangement distance computation [8]. Minimizing this distance is a fundamental step in computing plausible ancestors.

Computational reconciliation is most often formulated as the *multiple genome rearrangement problem* [163, 83]: given a set of N contemporary genomes and a distance d, find a tree T with the N genomes as leaf nodes and assign permutations (plausible ancestral architectures) to internal nodes such that $D(T) = \sum_{(\pi,\gamma)\in T} d(\pi, \gamma)$ is minimized. When $N = 3$ this is called the *median genome problem*. Sankoff and Blanchette [162] developed a method based on the breakpoint distance in [133], while Bourque and Pevzner used the rearrangement distance [84] to find an ancestral genome [23]. In both cases the median genome problem was proved to be NP-hard (see [28, 152] for breakpoint distance and [33, 34] for reversal distance). Although these two distance measures are closely related (see [164]), the corresponding methods are devised without considering the impact of the other distance on results.

A wider debate exists between the proponents of the *in silico* approach cited above and the proponents of the cytogenetic approach. Exemplified by Froenicke *et al.* [65], the latter group argues essentially that under-sampling in the *in silico* approach combined with the tendency of closely related genomes to attract the median M, leads to non-unique results that diverge from those found using cytogenetic methods. Bourque *et al.* in their response [24] argue that under-sampling will disappear with time and that the distinction between strong and weak adjacencies identified in the *in silico* method permits reliable comparison between the different approaches. Rocchi *et al.* in their perspective [160] suggest that a combination of the two approaches should lead to more realistic ancestral architectures, but furthermore that it is necessary to better model biological considerations, especially centromere repositioning and segmental duplication.

A considerable drawback to formulating the problem as the search for a single complete assembly that minimizes the sum of genome distances, is that the set of mathematically equivalent solutions is quite large: in [25] more than 3000 solutions are found for the human-murid ancestor, and indeed a statistical study of the variance between minimal solutions by [58] suggests that reporting an unique median architecture is misleading. A more realistic approach is to consider what common structural features of ancestral genomes might be found. Partial reconciliation of comparative maps identifies permutations of markers as above but does not necessarily provide a total order between segments. In [120], for example, contiguous ancestral regions (CARs) are found by assigning to each node of a given phylogenetic tree a set of adjacencies that represent a consensus between those found in contemporary genomes, computed using a method analogous to Fitch parsimony and relying on knowledge of the phylogenetic tree.

3.2 Rearrangement scenarios revisited

Hannenhalli and Pevzner have devised a method that mimics all multichromosomal rearrangements by reversals operating on an unique permutation [85]. This is achieved by a conversion to the unichromosomal model, which requires an *optimal capping* to cleverly delineate chromosomes of a given genome, as well as an *optimal concatenate* in order to assemble them into a single permutation. The computed parsimonious scenario relies on the structure of this permutation.

However, both the formula for rearrangement distance and the algorithm for computing a parsimonious sequence of operations given by Hannenhalli and Pevzner [85] present errors. Tesler in [193] partially corrected the rearrangement distance formula. In the same paper, the algorithm that leads to optimal concatenates was completed by a proper bonding step. Ozery-Flato and Shamir in turn redefined some notions and suggest further corrections essentially for the rearrangement distance formula [147]. Nevertheless, the algorithm that is supposed to construct an optimal capping, fails.

Various definitions and their relationships presenting incoherences between papers by different authors, we first propose a single and coherent classification of interleaving graph components based on relevant literature. This classification permits a better understanding of what is wrong in the existing algorithm for determining optimal capping. We present cases for which Ozery-Flato and Shamir's algorithm fails and provide a counterexample for each case. Finally, we introduce a correct algorithm for optimal capping with a proof of its correction. This whole work was published in [99].

3.2.1 Preliminaries

Let $\Pi = \{\pi^1, ..., \pi^{N_\Pi}\}$ be a *multichromosomal genome* defined as a set of N_Π *chromosomes*. The i^{th} chromosome $\pi^i = \pi^i_1 ... \pi^i_{n_i}$ is a sequence of n_i gene markers where each gene marker is represented by a signed ordinal. The sign indicates the direction of a given marker.

Let Π and Γ be two multichromosomal genomes defined over the same set of gene markers N_g. We define *adjacencies* and *breakpoints* in the way analogous to Nadeau and Taylor [133].

Definition 10 *Two consecutive elements π_i and π_{i+1} of a chromosome π are said to be* adjacent *in Π. If two elements π_i and π_{i+1} are adjacent in Π but neither $\pi_i.\pi_{i+1}$ nor $-\pi_{i+1}.\pi_i$ are present in Γ, then the pair $\pi_i.\pi_{i+1}$ forms a* breakpoint *in Π.*

The notions of adjacencies and breakpoints are transferred to the *breakpoint graph* defined in [84]. The breakpoint graph $G(\pi, \gamma)$ is an edge-colored graph built from unsigned representations of two signed permutations. A signed permutation $\pi = \pi_1 .. \pi_n$ over n elements is transformed into an unsigned representation $u(\pi) = \pi_0 .. \pi_{2n+1}$ over $2n+2$ elements. Each positive element $+x$ from π is replaced by two vertices $2x-1$ and $2x$ while each negative element $-x$ is replaced by two vertices $2x$ and $2x-1$. Vertices $\pi_0 = 0$ and $\pi_{2n+1} = 2n+1$ are added for taking into account adjacencies with the first and the last elements. Edges represent adjacencies either in Π (edges $\{\pi_{2i}, \pi_{2i+1}\}$, drawn with solid lines), or in Γ (edges $\{\gamma_{2i}, \gamma_{2i+1}\}$, drawn with dashed lines) for $i = 0, ..n$ (see for an example figure 3.2).

Two steps are needed to encode a multichromosomal genome as an unique permutation: *capping* and *concatenation*. A *capping* of Π consists in adding two ordinals called *caps* to the extremities of each chromosome. Let $\{c_0, c_1, .., c_n\}$ with $n = 2N_\Pi - 1$ be the set of distinct caps different from the N_g gene markers in Π. We denote by $\hat{\Pi} = \{\hat{\pi}^1, ..., \hat{\pi}^{N_\Pi}\}$ a capping of Π where the i^{th} chromosome is $\hat{\pi}^i = c_{2(i-1)}\ \pi^i_1 ... \pi^i_{n_i}\ c_{2(i-1)+1}$. A *concatenation* $\hat{\pi}$ of $\hat{\Pi}$ is a signed permutation $\hat{\pi}$ obtained by choosing an orientation and order for each chromosome.

The breakpoint graph for multichromosomal genomes is built from permutations $\hat{\pi}$ and $\hat{\gamma}$. The distance value computed on $G(\hat{\pi}, \hat{\gamma})$ depends on the chosen capping and concatenation. Let $G(\Pi, \Gamma)$ be the graph obtained by removing all edges that involve concatenation and capping from $G(\hat{\pi}, \hat{\gamma})$. Then we can distinguish three types of vertices: isolated vertices called *tails*, cap vertices of degree 1 called Π-*caps*, and other vertices of degree 1 called Γ-*tails*.

The graph $G(\Pi, \Gamma)$ can be decomposed into cycles and paths. If a path starts and finishes with Π-caps (two Γ-tails, or one Π-cap and one Γ-tail, respectively) then it is a $\Pi\Pi$-*path* ($\Gamma\Gamma$-*path* or $\Pi\Gamma$-*path*, respectively).

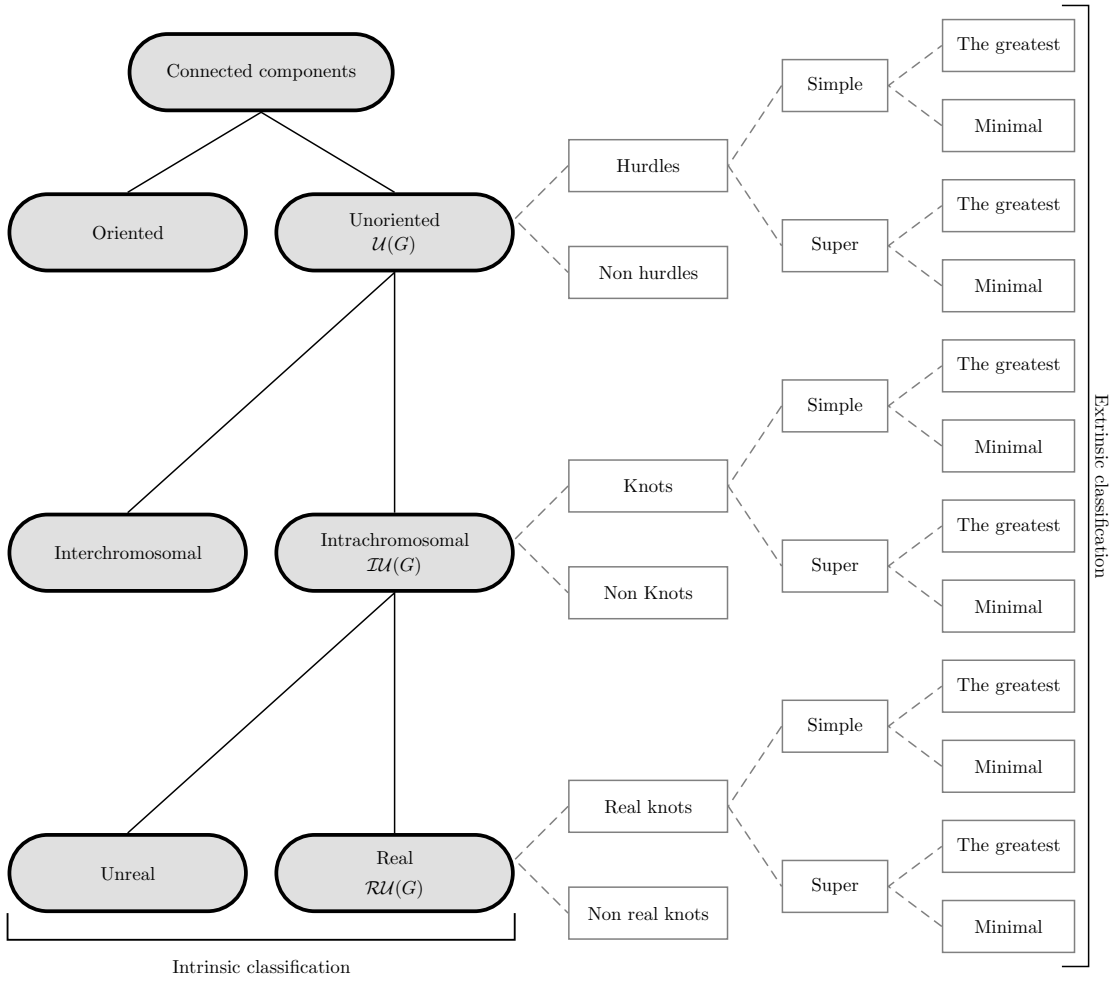


Figure 3.1: Double classification of connected components. The children nodes form a partition of the component set represented by their parent node. Intrinsic classification is read from top to bottom while extrinsic classification is read from left to right.

An *interleaving graph* $I(G)$ is a graph where each vertex represents a non trivial path or cycle (with more than 2 edges) of the breakpoint graph $G = G(\Pi, \Gamma)$ [84]. Two vertices are linked by an edge if the *spans* of corresponding cycles or paths overlap, but none of their intervals contain the other.

Definition 11 *The* span *of a cycle or a path C of $G(\Pi, \Gamma)$ is an interval $[i, j]$ such that $\forall \hat{\pi}_k \in C$ we have $i \leq k \leq j$ and $\exists\, \hat{\pi}_i, \hat{\pi}_j \in C$.*

We propose a coherent and unambiguous classification for the connected components of an interleaving graph that is the result of a synthesis of previously cited references. In fact, the components can be classified in two different and complementary ways, as shown in figure 3.1.

We call *intrinsic classification* the way to discriminate between components based on the properties of their edges. It is represented by the vertical hierarchy of filled nodes in figure 3.1. A

dashed edge (representing an adjacency in Γ) $\{\hat{\pi}_i, \hat{\pi}_j\}$ in $G(\Pi, \Gamma)$ is *oriented* if $|j - i|$ is even, otherwise it is *unoriented*. The same edge is *intrachromosomal* if the vertices $\hat{\pi}_i$ and $\hat{\pi}_j$ belong to the same chromosome, and *interchromosomal* otherwise. A connected component K of $I(G)$ is *oriented* (*interchromosomal*, respectively) if any cycle or path belonging to K has at least one oriented (interchromosomal, respectively) dashed edge, otherwise K is *unoriented* (*intrachromosomal*, respectively). Within unoriented and intrachromosomal components, we distinguish *real* components from *unreal* components.

Definition 12 *A connected component K of $I(G)$ is* real *if K is intrachromosomal, unoriented and if it has no Π-cap or Γ-tail in its span.*

Let $\mathcal{U}(G)$ be the set of unoriented components of $I(G)$, $\mathcal{IU}(G)$ the set of unoriented and intrachromosomal ones and $\mathcal{RU}(G)$ the set of real components.

We call *extrinsic classification* the way to describe a component by its relationship with surrounding components. It is represented horizontally by dashed lines in figure 3.1. $\mathcal{U}(G)$ is partitioned into *non hurdles* and *hurdles*, where a hurdle is a component of $\mathcal{U}(G)$ that does not *separate* two other components in the same set. The notion of separation defines in the same way the partitions of $\mathcal{IU}(G)$ and $\mathcal{RU}(G)$: *knots* and *non knots* for the former, and *real knots* and *non-real knots* for the latter. A hurdle is *super* if it *protects* a non hurdle, otherwise it is *simple*. A hurdle can be *the greatest* one if its span contains all the spans of the others hurdles, otherwise it is a *minimal hurdle*. These notions are defined similarly for knots and real knots. We can also extend the notion of *fortress* in $\mathcal{U}(G)$ in *fortress of knots* in $\mathcal{IU}(G)$ and a *fortress of real knots* in $\mathcal{RU}(G)$.

Within the set of unreal components we can distinguish those called *semi-real knots* that are characterized by their potential of becoming real knots.

Definition 13 *A* semi-real knot *is a component in $\mathcal{IU}(G)\backslash\mathcal{RU}(G)$ that does not contain a $\Gamma\Gamma$-path in its span and that becomes a minimal real knot or the greatest simple real knot after closing its $\Pi\Gamma$-paths.*

We say that a graph G is a *weak fortress of real knots* if (a) G has an odd number of real knots, (b) there exists the greatest real knot in G, (c) all real knots are super except the greatest one and (d) the number of semi-real knots in G is > 0. Note that a weak fortress of real knots becomes a fortress of real knots by closing the $\Pi\Gamma$-paths in a semi-real knot.

A *simple component* is defined as a component with at least one $\Pi\Gamma$-path and which is not a semi-real knot. Example 1 gives the intrinsic and extrinsic classifications for the breakpoint graph of the figure 3.2.

Example 1 *Figure 3.2 presents a breakpoint graph $G(\Pi, \Gamma)$. The intrinsic classification is as follows: K_1 is intrachromosomal oriented, $\mathcal{U} = \{K_2, K_3, K_4, K_5\}$, $\mathcal{IU} = \{K_2, K_3, K_4\}$ and $\mathcal{RU} = \{K_2, K_3\}$. The extrinsic classification is: K_3 is a super hurdle while K_4 and K_5 are simple hurdles, and K_3 and K_5 are super knots. However, K_2 and K_3 are real knots (K_2 is the greatest one), while K_5 is a minimal semi-real knot and K_1 is a simple component.*

Denote by $\bar{G}(\Pi, \Gamma)$ the graph obtained by closing all the $\Pi\Gamma$-paths in simple components of $G(\Pi, \Gamma)$. Ozery-Flato and Shamir [147] give an exact formula for distance between two multichromosomal genomes Π and Γ as shown in theorem 1.

Theorem 1 (Ozery-Flato [147]) $d(\Pi, \Gamma) = b - c + p_{\Gamma\Gamma} + r + \lceil \frac{s' - gr' + fr'}{2} \rceil$.

The parameters of the formula are defined as:

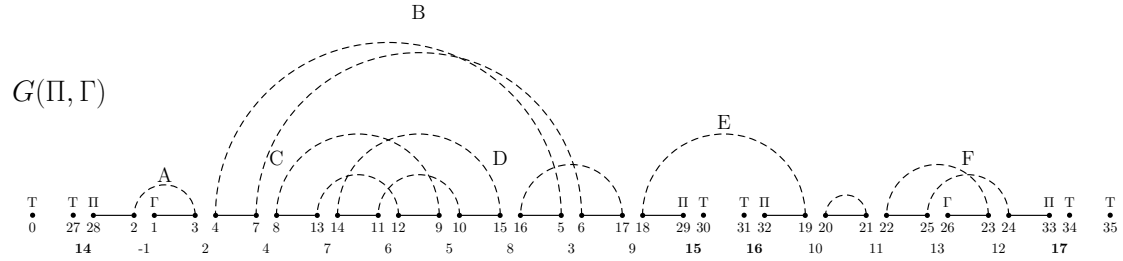


Figure 3.2: Breakpoint graph $G(\Pi, \Gamma)$ for $\Pi = \{1\ 2\ 4\ 7\ 6\ 5\ 8\ 3\ 9, 10\ 11\ 13\ 12\}$ and $\Gamma = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\}$. Tails vertices are marked by T, Π-caps by Π and Γ-tails by Γ. Non trivial cycles and paths are denoted by letters from A to F. The interleaving graph $I(G)$ corresponding to $G(\Pi, \Gamma)$ is composed of 5 connected components: $K_1 = \{A\}$, $K_2 = \{B\}$, $K_3 = \{C, D\}$, $K_4 = \{E\}$ and $K_5 = \{F\}$.

- b is the number of solid edges in $G(\Pi, \Gamma)$ ($b = N_g + max(N_\Pi, N_\Gamma)$),
- c is the number of cycles and paths,
- $p_{\Gamma\Gamma}$ is the number of $\Gamma\Gamma$-paths,
- r is the number of real knots,
- s' is the number of semi-real knots in $G(\Pi, \Gamma)$,
- gr' is equal to 1 if $\bar{G}$ has the greatest real-knot and $s' > 0$, and is 0 otherwise,
- fr' is equal to 1 if either (i) $\bar{G}$ is a fortress of real knots and the greatest semi-real knot does not exist in $\bar{G}$, or (ii) $\bar{G}$ is a weak fortress of real knots.

Computing the distance between two multichromosomal genomes is independent of capping and concatenation. However, computing a parsimonious scenario consists in finding a sequence of reversals mimicking multichromosomal rearrangements that satisfy the minimal distance. Thus, *optimal capping* and *optimal concatenation* lead to a parsimonious scenario.

3.2.2 Optimal capping

Optimal capping $\hat{\Pi}$ and $\hat{\Gamma}$ is finding positions and signs for caps in the genome Γ such that $d(\hat{\Pi}, \hat{\Gamma}) = d(\Pi, \Gamma)$ (see lemma 3 in [85]). This is done for any arbitrary capping in Π. In the breakpoint graph, it consists in adding $2N_\Gamma$ edges linking a Π-cap to a Γ-tail and $N_\Pi - N_\Gamma$ edges between two Π-caps if $N_\Pi > N_\Gamma$.

Ozery-Flato and Shamir [147] give an algorithm for construction of an optimal capping. However, their algorithm is incorrect. The case for which the algorithm fails is described as follows: (i) s' is even and $s' > 2$, (ii) G is a fortress of real knots and (iii) G has the greatest semi-real knot. If G is a fortress of real knots and there exists the greatest semi-real knot then $fr' = 0$. Moreover, the greatest semi-real knot and the greatest real knot can not exist simultaneously, so $gr' = 0$. Hence, the genomic distance is $d = b - c + p_{\Gamma\Gamma} + r + \lceil \frac{s'}{2} \rceil = b - c + p_{\Gamma\Gamma} + r + \frac{s'}{2}$ since s' is even. The step 5 of the optimal capping algorithm in [147] joins any two semi-real knots. Suppose that the greatest semi-real knot is joined by an interchromosomal or oriented edge to another semi-real knot. The obtained graph is still a fortress of real knots, but the greatest semi-real knot does not exist anymore,

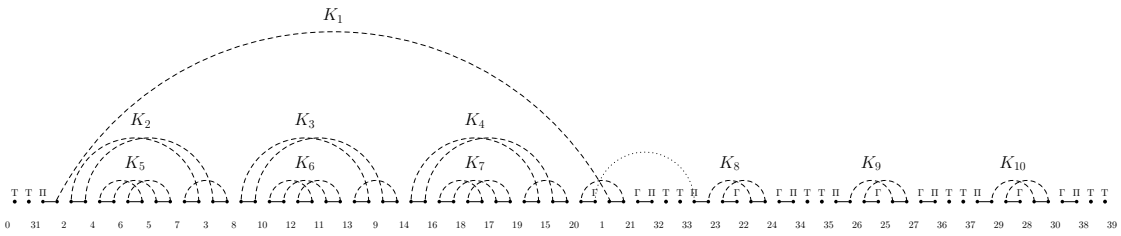


Figure 3.3: Counterexample of Ozery-Flato and Shamir's algorithm [147] for building an optimal capping. Breakpoint graph $G(\Pi, \Gamma)$ with $\Pi = \{2\ 4\ 6\ 5\ 7\ 3\ 8\ 10\ 12\ 11\ 13\ 9\ 14\ 16\ 18\ 17\ 19\ 15\ 20\ 1\ 21,\ 23\ 22\ 24,\ 26\ 25\ 27,\ 29\ 28\ 30\}$ and $\Gamma = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21,\ 22\ 23\ 24,\ 25\ 26\ 27,\ 28\ 29\ 30\}$. The connected components K_5, K_6 and K_7 are super real knots that protect respectively K_2, K_3 and K_4. G has the greatest semi-real knot K_1 and three minimal semi-real knots K_8, K_9 et K_{10}.

so $fr' = 1$. Thus, we get $d = b - (c-1) + p_{\Gamma\Gamma} + r + \lceil \frac{(s'-2)+1}{2} \rceil = b - c + 1 + p_{\Gamma\Gamma} + r + \frac{s'}{2}$. See figure 3.3 and example 2 for a counterexample.

Example 2 *The breakpoint graph $G = G(\Pi, \Gamma)$ in figure 3.3 is a fortress of real knots with $fr' = 0$. The distance is $d(\Pi, \Gamma) = 34 - 14 + 0 + 3 + \lceil \frac{4-0+0}{2} \rceil = 25$. Step 5 of Ozery-Flato and Shamir's algorithm allows joining the greatest semi-real knot K_1 to K_8 by a interchromosomal edge (dashed line), which results in a new graph G'. G' is still a fortress of real knots, but $fr' = 1$. So $d = 34 - 13 + 0 + 3 + \lceil \frac{2-0+1}{2} \rceil = 26$, which does not respect the minimal distance.*

In what follows we propose a new algorithm for optimal capping and the proof of its correction (theorem 2). The proof is based on two technical lemmas from [85] (lemmas 4 and 5).

Theorem 2 *Let $d = d(\Pi, \Gamma)$ be the distance between two multichromosomal genomes Π and Γ. Algorithm 2 constructs an optimal capping $\hat{\Gamma}$ for any arbitrary capping $\hat{\Pi}$, such that $d(\hat{\Pi}, \hat{\Gamma}) = d$.*

For proof, see the original manuscript [99].

3.3 Ancestral architectures and super-blocks

The study of evolutionary mechanisms is made more and more accurate by the increase in the number of fully sequenced genomes. One of the main problems is to reconstruct plausible ancestral genome architectures based on the comparison of contemporary genomes. Current methods have largely focused on finding complete architectures for ancestral genomes, and, due to the computational difficulty of the problem, stop after a small number of equivalent minimal solutions have been found. Recent results suggest, however, that the set of minimum complete architectures is very large and heterogeneous. In fact these solutions are collections of conserved blocks, freely rearranged. In this work, we identify these conserved *super-blocks*, using a new method of analysis of ancestral architectures that reconciles both breakpoint and rearrangement analyses, as well as respects biological constraints. The resulting algorithms permit the first reliable reconstruction of plausible ancestral architectures for several non-WGD yeasts simultaneously, a problem hitherto intractable due to the extensive map reshuffling of these species.

In this study, we propose an approach for identifying common ancestral features for the general, N-genome instance, that builds a bridge between breakpoint and rearrangement methods and

Algorithm 2 Correct_Optimal_Capping

Construct the graph $G = G(\Pi, \Gamma)$
while there is a $\Gamma\Gamma$-path in G **do**
  Find an interchromosomal or oriented edge joining this $\Gamma\Gamma$-path with a $\Pi\Pi$-path (lemma 4 [85]) and add it to G
end while
Close all remaining $\Pi\Pi$-paths in G
Close all $\Pi\Gamma$-paths in simple components in G
if s' is even and $s' \geq 2$ and G is a fortress of real knots **then**
  if G has the greatest semi-real knot **then**
    Close all $\Pi\Gamma$-paths in the greatest semi-real knot
  else
    Close all $\Pi\Gamma$-paths in any one semi-real knot
  end if
end if
while G has more than one semi-real knot **do**
  Find an interchromosomal or oriented edge joining $\Pi\Gamma$-paths in any two semi-real knot (lemma 5 [85]) and add it to G
end while
Close all remaining $\Pi\Gamma$-paths in G
Find a capping $\hat{\Gamma}$ defined by the graph $G(\hat{\Pi}, \hat{\Gamma})$

additionally permits the use of biological constraints. The main contribution is the computation of *super-blocks*, sequences of markers chosen in function of the frequency of the corresponding adjacencies without any use of phylogeny. Here we follow the hypothesis that adjacencies having support in two or more contemporary genomes constitute the semantic basis of an ancestral architecture [162]. Super-blocks can of course be joined to produce final assemblies. Algorithmically, it is an optimization problem in terms of rearrangement distance of the sequence of fusions of super-blocks. The solution space of genome median is thus reduced, and only architectures respecting the adjacency semantics are returned. Although the mathematical model does not make possible the consideration of segmental duplication, centromere positions are introduced and constrain the final assemblies by allowing only one active centromere in each chromosome of the ancestral architecture.

We show that in theory our method allows for solutions that are either minimal or reasonably close to the minimal in mathematical model. Even though the addition of biological constraints can lead to non-optimal mathematical solutions, our method decreases the number of mathematically equivalent solutions by using biological constraints as a filter on the solution space.

This section is organized as follows. Section 3.3.1 gives the necessary preliminaries. In section 3.3.2, we introduce the notion of dependency for the adjacencies and show the relationship between adjacencies and distances. Section 3.3.3 gives the methodology to construct super-blocks from adjacencies, and the strategy for building final assemblies by an optimal sequence of fusions[1]. In section 3.3.4 we apply our method to a set of *non-WGD*[2] *Hemiascomycete* genomes in the *Kluyveromyces* and related clades provided by the Génolevures Consortium, with divergence similar to that of mammals [50]. For this phylogenetic branch, our method shows a high convergence in the structure of different versions of super-blocks (16 in all), reinforcing the intuition that super-blocks encode the

[1] All supplementary materials (figures and proofs) can be found at `http://www.genolevures.org/supplementary/Jean2008/super`

[2] Whole-Genome Duplication, an unique polyploidization event proposed in the ancestral *Saccharomyces* lineage; non-WGD yeasts from the other branches of the phylogenetic tree are not affected by this catastrophic event.

(a) π_i is positive (b) π_i is negative

Figure 3.4: Vertices of the breakpoint graph corresponding to the element π_i.

semantics of the ancestral genome. We can thus perform a reconstruction, despite extensive map reshuffling. Additional evaluation wrt. existing method as well as more detailed theoretical developments can be found in the original paper [101].

3.3.1 Preliminaries

Let $\Pi = \{\pi^1, .., \pi^{N_\Pi}\}$ be a *multichromosomal genome* defined as a set of N_Π chromosomes. A *chromosome* $\pi = \pi_1, ..., \pi_n$ is a sequence of n gene markers represented by signed ordinals. The sign indicates the direction of a given marker.

We define adjacencies and breakpoints in the way analogous to Nadeau and Taylor [133] (see definition 10).

The number of breakpoints b between two genomes is a distance such that for 2 multichromosomal genomes Π and Γ with $N_\Pi < N_\Gamma$, the number of breakpoints is $b = \{(\pi_i, \pi_{i+1}) | \pi_i.\pi_{i+1}$ is a breakpoint in $\Pi\} + (N_\Gamma - N_\Pi)$ or $b = \{(\gamma_i, \gamma_{i+1}) | \gamma_i.\gamma_{i+1}$ is a breakpoint in $\Gamma\}$.

Let $G_1, ..., G_N$ be N multichromosomal genomes defined over the same set of distinct gene markers $\mathcal{G}$. We denote by $u(g.h)$ the frequency of the adjacency $g.h$ in the N genomes. We denote by $\mathcal{A}$ the set of all adjacencies in $G_1, .., G_N$.

Following Hannenhalli and Pevzner [84], we will use the non-signed representation of a signed genome in terms of *breakpoint graph*. The signed permutation for a chromosome π over n elements is transformed into a permutation over $2n + 2$ elements. Two vertices $2\pi_i - 1$ and $2\pi_i$ are created for each element π_i, and the sign of π_i determines the order of these vertices in the new permutation as shown on figure 3.4. Two additional vertices 0 et $2n + 1$ delimit the extremities of the new permutation.

The notions of adjacencies and breakpoints are transferred to the breakpoint graph quite naturally. Choice of adding vertices at the extremities of each chromosome being arbitrary, we denote by 0 any telomere without taking into the account its chromosome. Hence, for a chromosome $\pi = \pi_1 ... \pi_n$ we introduce two supplementary adjacencies denoted by $0.\pi_1$ and $\pi_n.0$. In what follows, we will systematically use greek letters to denote elements of a signed permutation and latin letters to denote elements of a non-signed permutation: we will note by $(g_i\, h_i).(g_j\, h_j)$ the adjacency corresponding to $\pi_i.\pi_j$ except for adjacencies with telomeres that will be noted $(0).(g_1\, h_1)$ and $(g_n\, h_n).(0)$. For any adjacency $a = \pi_i.\pi_j = (g_i\, h_i).(g_j\, h_j)$ its reversal $-a$ is defined by $-\pi_j.\ -\pi_i$ in the signed permutation, and by $(h_j\, g_j).(h_i\, g_i)$ in the non-signed permutation.

Example 3 *Let us consider four genomes* $G_1 = \{1\ 2\ 3\ 4,\ 5\ 6\}$, $G_2 = \{1\ 2\ 34, -5, -6\}$, $G_3 = \{2\ 1\ 3\ 4, -6\ 5\}$ *and* $G_4 = \{3\ 1\ 4\ 2\ -5,\ 6\}$. *Their adjacencies can then be partitioned according to frequency of occurrence in* G_i *as shown in table 3.1.*

frequency	adjacency
4	6.0
3	3.4, 0.5, 4.0
2	0.1, 1.2, 0.6, 5.0,2.3
1	5.6, 0.2, 2.1, 1.3, 4.2, 3.1, 1.4, 0.3, -5.6, $2. - 5$

Table 3.1: Adjacencies for genomes G_1, G_2, G_3 and G_4 sorted by frequency.

3.3.2 Dependent adjacencies

The construction of *super-blocks* is based on the study of adjacencies. This study consists in defining the frequency of adjacencies in the genomes and the adjacency relationships.

The intuition behind our approach is that an adjacency of higher frequency should be preferentially present in a median genome. Mathematically, we are looking for an ancestral architecture that represents a compromise between the rearrangement distance and the number of breakpoints under the parsimony criterion.

In what follows, the considered rearrangement distance is expressed in terms of reversals, fusions, fissions and translocations and is computed according to Hannenhalli and Pevzner's theory [84].

Pairwise adjacency relationships

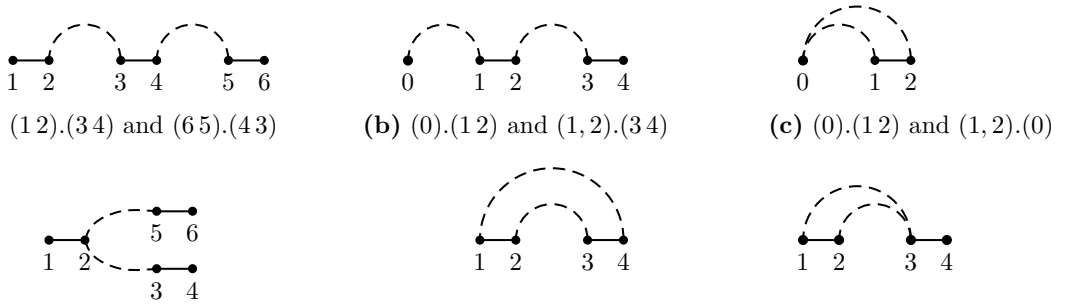
Let A be a subset of the set of all adjacencies $\mathcal{A}$ for genomes $G_1, ..., G_N$. We build the *adjacency graph* $G = (V, E)$ for A in the following way. For any adjacency $(g_i\, h_i).(g_j\, h_j)$, we create four vertices (g_i, h_i, g_j and h_j) and three edges. Two of the edges represent elements of the original permutation: $e_1 = (g_i, h_i)$ and $e_2 = (g_j, h_j)$. One of the edges represents the adjacency itself: $e_3 = (h_i, g_j)$.

Two adjacencies are *dependent* if their elements are related, either by completing or by contradicting each other. Let a and b be two adjacencies $a = (g_1^a\, h_1^a).(g_2^a\, h_2^a)$ and $b = (g_1^b\, h_1^b).(g_2^b\, h_2^b)$, and $G = (V, E)$ the adjacency graph for $\{a, b\}$.

Definition 14 *We say that* a *and* b complement each other *if either (i)* $\exists\, v_1, v_2 \in V$ *such that* $d(v_1) = d(v_2) = 1$ *and* $\forall v \neq v_i, i \in [1, 2]$ *we have* $v \neq 0$ *and* $d(v) = 2$*, or (ii)* $\exists v \in V$ *such that* $v = 0$ *and* $\forall v \in V$ *we have* $d(v) = 2$*. We say that* a *and* b contradict each other *if either (i)* $\exists\, v \in V$ *such that* $d(v) > 2$*, or (ii)* $\forall v \in V$ *we have* $v \neq 0$ *and* $d(v) = 2$*.*

For example, adjacencies $(1\,2).(3\,4)$ and $(6\,5).(4\,3)$ complement each other. Indeed, we can form the sequence 1 2 3 4 5 6. On the contrary, $(1\,2).(3\,4)$ and $(6\,5).(2\,1)$ are in contradiction as well as $(1\,2).(3\,4)$ and $(2\,1).(3\,4)$. As can be seen on figure 3.5, the two contradictions are slightly different. Indeed, the latter involves the presence of a cycle (*cycle contradiction*), while the former does not (*vertex contradiction*).

When adjacencies complement each other there is no problem to put them together in order to form a coherent chromosome. However, when two adjacencies a and b are in contradiction, we need to choose one or the other. The intuition given in the beginning of this section is to prefer adjacencies with higher frequencies. However, it is possible to have a median genome in terms of rearrangement distances with an adjacency of lower frequency that is in contradiction with an adjacency of higher frequency as illustrated in the example 4. Notice that the adjacency 3.2 that is present in M_1 has frequency 2, while the adjacency 2.3 present in M_2 is of frequency 1. Because of a better global number of common adjacencies (13 breakpoints against 14 for M_2), M_1 appears as the best median



(a) $(1\,2).(3\,4)$ and $(6\,5).(4\,3)$ **(b)** $(0).(1\,2)$ and $(1,2).(3\,4)$ **(c)** $(0).(1\,2)$ and $(1,2).(0)$

(d) $(1\,2).(3\,4)$ and $(6\,5).(2\,1)$ **(e)** $(1\,2).(3\,4)$ and $(3\,4).(2\,1)$ **(f)** $(1\,2).(3\,4)$ and $(2\,1).(3\,4)$

Figure 3.5: Adjacency graphs showing (a), (b) and (c) two adjacencies that complement each other, (d), (e) and (f) two adjacencies that contradict each other. Element edges are represented by solid lines; adjacency edges are represented by dashed lines.

genome in terms of rearrangement distances and breakpoint number but M_2 is also a good candidate for ancestral gene order in terms of rearrangement distances.

Example 4 *Consider three genomes* $G_1 = \{1\,2\,3\,4\,5\,6\,7\}$, $G_2 = \{1\,3\,2\,4\,5, 6\,7\}$ *and* $G_3 = \{1\,4\,3\,2\,5\,6,\ 7\}$. *Their pairwise rearrangement distances are:* $d(G_1, G_2) = 4, d(G_1, G_3) = 4$ *and* $d(G_2, G_3) = 5$. *Two optimal (median) solutions* M_1 *and* M_2 *are possible for these genomes:* $M_1 = \{1\,$***-2 -3***$\,4\,5,\ 6\,7\}$ *and* $M_2 = \{1\,$***-3 -2***$\,4\,5,\ 6\,7\}$. *The rearrangement distances from* M_1 *and* M_2 *to* G_1, G_2 *and* G_3 *are shown below.*

	G_1	G_2	G_3
M_1	3	1	4
M_2	2	2	4

Notice that the adjacency 3.2 *that is present in* M_1 *has frequency* 2*, while the adjacency* 2.3 *present in* M_2 *is of frequency* 1.

Adjacencies and distances

Example 4 is in apparent contradiction with the intuition that the adjacencies of higher frequencies should be preferred. In this section, we analyze in more detail in which cases it is appropriate to follow this intuition.

Bounds for rearrangement distances and breakpoints If two genomes Π and Γ are not equal, then $d(\Pi, \Gamma)$ is at least 1. If $d(\Pi, \Gamma) = 1$, then there are exactly two breakpoints in Π (say a and c), and two in Γ (say b and d). See figure 3.6 for illustration. We say then that Π and Γ are *identical up to* a, c (symmetrically b, d).
In [101] we establish two following results. Let $\mathcal{A}$ be the adjacency set of genomes $G_1, ..., G_N$, and let C be the set of all pairs of contradictory adjacencies from $\mathcal{A}$.

Theorem 3 *For any pair of adjacencies* $\{a, b\} \in C$ *and two genomes* M_a *and* M_b *identical up to 2 adjacencies with* $a \in M_a$ *and* $b \in M_b$*, it holds that*

$$\left| \sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \right| \leq N.$$

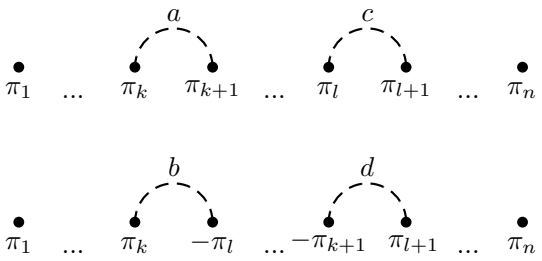


Figure 3.6: Π and Γ are identical up to a, c (or b, d). This implies (a) the existence of 2 breakpoints $a = \pi_k.\pi_{k+1}$ and $c = \pi_l.\pi_{l+1}$ in Π, and (b) of 2 breakpoints $b = \pi_k. - \pi_l$ and $d = -\pi_{k+1}.\pi_{l+1}$ in Γ.

Theorem 3 can be transposed to breakpoints as shown in theorem 4. Let us denote N_a the number of genomes with adjacency a, N_b the number of genomes with adjacency b and N_o the number of genomes without adjacencies a or b.

Theorem 4 *For any pair of adjacencies $\{a, b\} \in C$ such that $u(a) > u(b)$ and two genomes M_a and M_b identical up to 2 adjacencies with $a \in M_a$ and $b \in M_b$, it holds that*

$$N_a - 2N_b - N_o \leq \sum_{i}^{N} b(M_b, G_i) - \sum_{i}^{N} b(M_a, G_i) \leq 2N_a - N_b + N_o.$$

Moreover, in the same work [101] we show that while theorems 3 and 4 provide general theoretical bounds, the worst cases are rarely met, especially in practice.

3.3.3 From adjacencies to final assemblies

Section 3.3.2 implies that we can choose adjacencies for higher frequencies because they lead to a reasonable compromise between the breakpoint and the rearrangement distance approaches. Based on the adjacencies we propose to build *super-block assemblies* of median genomes.

The construction of super-blocks is done in two steps. First, we build a partition P of adjacencies where each part is composed of inter-dependent adjacencies. P is partially ordered by adjacency frequency of the parts' elements. Second, P is inspected in decreasing order of its parts, and the super-block sets are constructed by favoring adjacencies with higher frequency.

Finally, to find adjacencies unresolved before, the last part of our method looks for a sequence of fusions of super-blocks that minimizes the rearrangement distances.

Groups of dependent adjacencies

We have seen previously that there exist different relationships between adjacencies. They can complement each other and, in this case, we can assemble them together in order to form a coherent chain of elements. When two adjacencies are in contradiction, then either there are different possibilities to complement the same element (*vertex contradiction*), or these two adjacencies have the same elements up to their order and/or their orientation (*cycle contradiction*).

Let $\mathcal{P}(A)$ be a partition of $\mathcal{A}$. We define $\mathcal{P}_0(\mathcal{A})$ by the membership in the same elementary cycle without 0(that is a cycle containing 2 adjacencies). Parts of $\mathcal{P}_0(\mathcal{A})$ are either singletons or sets of adjacencies where every pair is in cycle contradiction. For a given set of adjacencies A, the highest

frequency of its elements is denoted $u(A) = \max_{a \in A} u(a)$ and is called *set frequency*. We denote by G the adjacency graph containing all the adjacencies of $\mathcal{A}$.

We define the merging of parts $\sqcup : \mathcal{P}(\mathcal{A}) \rightarrow \mathcal{P}(\mathcal{A})$ as follows.

Definition 15 *$\sqcup(\mathcal{P}(\mathcal{A}))$ is a partition of $\mathcal{A}$ such that for any $p \in \sqcup(\mathcal{P}(\mathcal{A}))$*

- *$\exists\, p_1 \in \mathcal{P}(A)$ s.t. $p = p_1$ or*
- *$\exists\, p_1, p_2 \in \mathcal{P}(\mathcal{A})$ s.t. $p = p_1 \cup p_2$ and moreover $\exists\, a \in p_1$ and $\exists\, b \in p_2$ s.t. $u(a) = u(b) = u(p_1) = u(p_2)$ and either a and b are dependent or a and b participate in a cycle $c \in G$ without vertex $v = 0$ s.t. $\forall v \in c$ we have $u(v) \geq u(a)$.*

Starting from $\mathcal{P}_0(\mathcal{A})$, merging of parts $\sqcup$ defines a sequence of partitions $\mathcal{P}_i(\mathcal{A})$ where $\forall i > 0, \mathcal{P}_i(\mathcal{A}) = \sqcup(\mathcal{P}_{i-1}(\mathcal{A}))$. Obviously, there exists an n for which $\sqcup$ reaches its fixed point denoted by $\sqcup^n(\mathcal{P}(A))$, that is $\mathcal{P}_n(\mathcal{A}) = \sqcup(\mathcal{P}_n(\mathcal{A}))$.

Definition 16 *A* group *g is a part of $\sqcup^n(\mathcal{P}(A))$.*

Example 5 *The adjacencies of the example 3 are partitioned into groups as shown in table 3.2.*

grp. freq.	adjacencies
4	$6.0(4)$
3	$3.4(3),\ 4.0(3)$
3	$0.5(3)$
2	$0.1(2),\ 1.2(2),\ 2.1(1),\ 2.3(2)$
2	$0.6(2)$
2	$5.0(2)$

Table 3.2: Partition of adjacencies from example 3 into groups. The adjacencies are noted with their frequency in parenthesis, and the groups are sorted by decreasing group frequency. Only groups with $u(g) > 1$ are represented.

Super-blocks and partial assemblies

Definition 17 *A* super-block *is a set S of $n \geq 1$ adjacencies such that $\forall a, b \in S$, a does not contradict b, and there exists an order over S such that $\forall i \in [1, n)$, a_i complements a_{i+1}, and a_1, a_n are either independent or $a_1 = a_n = 0$. A* partial assembly *$\mathcal{P} = \{S_k\}$ is a set of super-blocks such that $\forall k, l$ with $k \neq l$ if $S_k \cap S_l \neq \emptyset \Rightarrow S_k \cap S_l = \{0\}$.*

Super-blocks, and thus partial assemblies, are formed by going through the groups of adjacencies by decreasing order of their frequencies. For a given partial assembly $\mathcal{P} = \{S_k\}$ and a current group g, any adjacency $b \in g$ is removed from it if there exists an adjacency $a \in S_k \in \mathcal{P}$ in contradiction with b. This operation is called *clean* and produces a $g_c \subseteq g$, $g_c = \mathit{clean}(g, \mathcal{P})$. However, when inspecting the current group g_c we do not have any means to prefer some of its adjacencies over the others.

Let us denote the operation of adding a group g to $\mathcal{P}$ by $\oplus$. This operation produces all possible partial assemblies $\{\mathcal{P}_i\} = \mathcal{P} \oplus g$ and can be realized by the algorithm *add_group* (algorithm 3). The complexity of this algorithm is bounded by the research of maximal independent sets over g_c (for details see [101]).

Algorithm 3 add_group($g, \mathcal{P}$)

Require: a group g, a partial assembly $\mathcal{P}$
Ensure: $\mathbb{P}$ is a set of partial assemblies
let $G_{\mathcal{P}}$ be the adjacency graph for $\mathcal{P}$
let $\mathbb{P} = \emptyset$ and $g_c = clean(g, \mathcal{P})$
let $\mathcal{M}$ be the set of all maximal independent sets over g_c
for all $\mu \in \mathcal{M}$ **do**
  let G_μ be the adjacency graph for μ
  let T be the set of all connected components of G_μ^0
  let $G_{new} = \{V_{\mathcal{P}} \cup V_\mu, E_{\mathcal{P}} \cup E_\mu\}$
  let $C = \emptyset$
  while G_{new}^0 has a cycle c **do**
    let $V = \emptyset$ be the set of adjacencies from μ participating in c
    for all $t \in T$ **do**
      if $t \cap c \neq \emptyset$ **then**
        let $V = V \cup adjacencies(t)$
        let $G_{new} = \{V_{new} \backslash \{t[0]\}, E_{new}\}$
      end if
    end for
    let $C = C \cup \{V\}$
  end while
  let $G = \{V_{\mathcal{P}} \cup V_\mu, E_{\mathcal{P}} \cup E_\mu\}$
  let $\mathbb{G}_\mu = \{G\}$
  for all $c \in C$ **do**
    $\mathbb{G} = \emptyset$
    for all $a \in c$ **do**
      for all $G_\mu \in \mathbb{G}_\mu$ **do**
        $\mathbb{G} = \mathbb{G} \cup \{\{V_\mu, E_\mu \backslash \{a\}\}\}$
      end for
    end for
    $\mathbb{G}_\mu = \mathbb{G}$
  end for
  let $\mathbb{P} = \mathbb{P} \cup$ *partial_assemblies*($\mathbb{G}_\mu$)
end for
return $\mathbb{P}$

The construction of all partial assemblies for genomes $G_1, ..., G_N$ proceeds as shown in algorithm 4. Notice that we do not consider groups where $u(g) = 1$ since these adjacencies do not have any additional support in any other genome.

Fusions of super-blocks

Algorithm 4 builds all partial assemblies by resolving conflicts between adjacencies up to group frequency 2. Groups of frequency 1 are excluded since there is no evidence if they are present by chance or not.

Definition 18 *A* fusion *of super-blocks* $S_1 = (a_1, ..., a_n)$ *and* $S_2 = (b_1, ..., b_m)$ *is a super-block* S *such that the order of definition 17 is either* $S = (a_1, ..., a_n, b_1, ..., b_m)$*, or* $S = (a_1, ..., a_n, -b_m, ..., -b_1)$*, or* $S = (b_1, ..., b_m, a_1, ..., b_n)$*, or* $S = (b_1, ..., b_m, -a_n, ..., -a_1)$*.*

This definition implies that a super-block S such that $a_1 = 0.\pi_i$ and $a_n = \pi_j.0$ can not participate in a fusion. Indeed, such a super-block already forms a chromosome telomere to telomere.

Let $\{\mathcal{P}\}$ be the set of all partial assemblies up to group frequency 2 for genomes $G_1, ..., G_N$ and $\mathcal{P} \in \{\mathcal{P}\}$ a partial assembly. The number of super-blocks in $\mathcal{P}$ can be relatively high. This is due to the fact that some elements can not be inter-connected because of the low frequency (equal to 1) of corresponding adjacencies. Such elements are located at the extremities of the super-blocks. We connect them in order to form chromosomes by fusions of super-blocks without worsening the distance and breakpoint bounds (see theorem 5).

Algorithm 4 partial_assemblies($G_1, ..., G_N$)

Require: $G_1, ..., G_N$ genomes over the same set of gene markers
Ensure: $\mathbb{P}$ is a set of partial assemblies
let $\mathcal{A}$ be the set of all adjacencies for $G_1, ..., G_N$
let $G = \{g\}$ be the set of all groups for $\mathcal{A}$
let $n = max_G u(g)$
let $\mathbb{P} = \{\emptyset\}$
for all g_i s.t. $n \geq i \geq 2$ **do**
  let $\mathbb{P}' = \emptyset$
  for all $\mathcal{P} \in \mathbb{P}$ **do**
    $\mathbb{P}_{\mathcal{P}} = \mathcal{P} \oplus g_i$
    $\mathbb{P}' = \mathbb{P}' \cup \mathbb{P}_{\mathcal{P}}$
  end for
  $\mathbb{P} = \mathbb{P}'$
end for
return $\mathbb{P}$

Theorem 5 *For any $\mathcal{P} \in \{\mathcal{P}\}$ of $G_1, ..., G_N$ such that $\mathcal{P} = \{S_k\}$, there exists a genome M such that for any chromosome π of M either $\exists\, S_k \in \mathcal{P}$ such that $\pi = S_k$, or $\exists\, \{S_k\} \subseteq \mathcal{P}$ such that π is formed by a series of fusions $\pi = S_1 ... S_k$. Moreover,*

$$\sum_i^N d(M, G_i) - \sum_i^N d(P, G_i) \leq 0 \text{ and } \sum_i^N b(M, G_i) - \sum_i^N b(P, G_i) \leq 0.$$

To find an optimal sequence of fusions, we classify them by their effect on global rearrangement distance (the sum of rearrangement distances between median genome and $G_1 \ldots G_N$). A greedy randomized algorithm is used to find ancestral candidates obtained after a limited number of fusions. By parsimony criterion, solutions that minimize global rearrangement distance are conserved.

3.3.4 A Median Genome for non-WGD yeasts

We have applied our method to analyze ancestral architectures for Génolevures project ([51]) in the case of non-WGD Hemiascomycete yeasts. The data consists in 5 completely sequenced yeasts the *Kluyveromyces* and related clades: *Kluyveromyces lactis*, *Saccharomyces kluyveri*, *Zygosaccharomyces rouxii*, *Ashbya (Eremothecium) gossypii* and *Kluyveromyces thermotolerans*[3]. These genomes have little genome redundancy and a relatively high (for yeasts) conservation of synteny.

Signed permutations representing each genome were computed using pairwise syntenic blocks obtained by i-ADHoRE method [181] from orthology and synteny relations identified using Génolevures protein families [143]. These syntenic blocks contain 8–200 genes (mean size 14 genes, estimated) and cover roughly 60% of each genome. Basing these permutations only on protein-coding genes is sufficient, since yeast genomes are highly compact (protein-coding genes cover approximately 80% of the genome), and gene relics are quite rare (approximately 4%) [50]. By combining pairwise syntenies, each genome was factored into a sequence of ordered syntenic blocks, from which a set of distinct blocks common to all genomes was determined. An arbitrary reference genome was chosen, and all the blocks forming this genome were numbered by unique sequential identifiers from 1 to n.

The permutations computed by this *in silico* chromosomal painting contained 487 blocks (mean size 51 genes, estimated); keeping the longest blocks brought the permutations to 120 identifiers[4] (mean size, 94 genes, estimated). We were able to place active and inactive centromeres in each genome permutation by locating the flanking genes (personal communication by Jacky de Montigny).

[3] Abbreviations: Klla, *K. lactis*; Sakl, *S. kluyveri*; Zyro, *Z. rouxii*; Ergo, *A. gossypii*; Klth, *K. thermotolerans*.

[4] The number of retained markers does not allow one to obtain an ancestral permutation candidate by using the public version of MGR.

Each of 9 contemporary centromeres was encoded by two successive identifiers, resulting in 15 additional blocks. Thus, each genome was represented as a signed permutation of 135 elements (see [101]), in which chromosomal rearrangements (fusion, fission, translocation, inversion) were studied. The pairwise rearrangement distances between these genomes are shown in table 3.3.

	Zyro	Klth	Sakl	Klla	Ergo
Zyro	0	84	79	115	101
Klth		0	45	105	88
Sakl			0	98	85
Klla				0	109
Ergo					0

Table 3.3: Pairwise rearrangement distances between non-WGD Hemiascomycete genomes as calculated from common synteny blocks representing 135 major conserved segments.

Comparative genome maps were painted (see figure 3.7) with *K. thermotolerans* as reference. Active centromeres are represented by red ovals, telomeres are represented by triangles. The assigned letter indicates the agreement of this centromere across the five species. Markers are well distributed on the chromosomes, so the choice of these markers is representative of the architecture of the contemporary genomes. A high degree of synteny, and a limited number of large-scale rearrangements, is observed between *K. thermotolerans* and *S. Kluyveri*; they share many common adjacencies and their rearrangement distance is half of that seen between other pairs of genomes. Note that *K. lactis* presents two syntenic breaks in centromere areas: the centromere of *Klla0F* is located between the flanking genes of centromeres h and b, and the centromere of *Klla0A* is located between the flanking genes of centromeres h and e. Moreover, *S. kluyveri* has an active centromere (the centromere i), that was disabled in all the other studied genomes.

We computed 16 sets of super-blocks each containing either 34 or 35 super-blocks. These super-block sets are highly similar. Indeed, 29 super-blocks are common among all of the sets, and there are only 4 conflicts (see figure 3.8). A given partial assembly of super-blocks $\mathcal{P}$ represents a potential structure of an ancestral architecture. Finally, it is possible to construct a final assembly from these super-blocks by successive fusions. Two sets of assemblies were computed: with and without the constraint on centromere position. For both of these cases 90 final assembly candidates were generated. In the first case the global sum of distances $\sum(M, G_i)$ varies between 281 and 285 (283,4 on average); in the second case it varies between 281 and 283 (282,2 on average). The latter represents biologically plausible architectures whose rearrangement distances are close to minimal. The whole set of solutions shows a high convergence in terms of rearrangement distances reinforcing the intuition that the computation of ancestral architectures by super-blocks assembly results in a reduced neighborhood in the search space. Further filtering of the results was done by studying distributions of chromosome sizes and of centromere locations on the chromosome. Figure 3.7 shows the candidate for ancestral architecture which has the best compromise between these parameters and minimal value for $\sum(M, G_i) = 284$.

3.3.5 Discussion

Computing the median for a given set of genomes is informative when the sample set of genomes is carefully chosen and the interpretation of the common features that are so identified is performed with caution. As with any statistical study, if the sample is too small or not representative of the population under study, then the median may be biased. It is not the object of this work to provide guidance into

Figure 3.7: Reconstructions of genome-scale homology from common synteny blocks representing major conserved segments. Med is the proposed ancestral architecture with $\sum d(Med, G_i) = 284$. Each unique numbered synteny block is given a color indicating its chromosome in the reference genome (*Klth*), and a diagonal bar indicating its relative position on the chromosome. Other genomes are signed permutations of these colored blocks; a change of slope in the diagonal bar indicates an inversion. Block widths are to scale and the size of interleaving nonsyntenic regions is shown by large grey lines. Red circles: centromeres; gray triangles: telomeres.

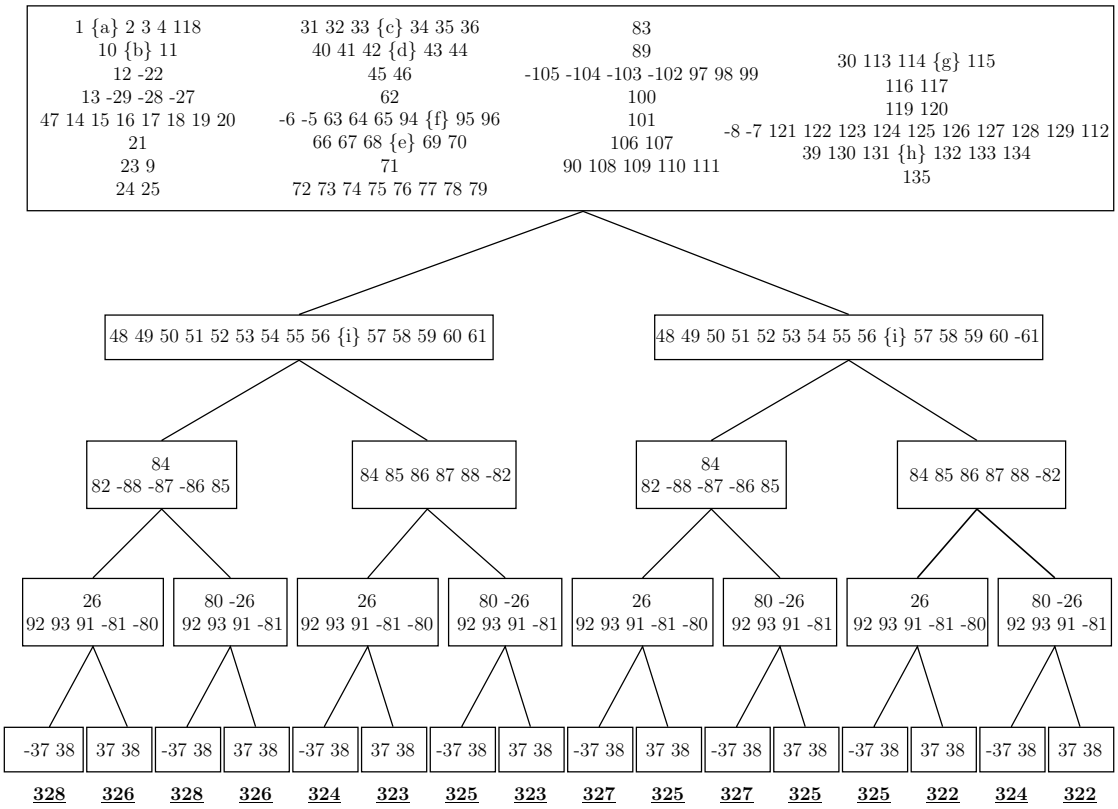


Figure 3.8: Sharing tree of super-blocks from the 16 sets of super-blocks obtained from non-WGD Hemiascomycete yeasts genomes. The root contains the super-blocks shared among all the 16 sets. Each path from the root to a leaf represents a set of super-blocks. The number inside the leaf nodes indicates the sum of the distance between this set of super-blocks and the contemporary genomes. Colors and marker numbers were chosen using *Klth* as a reference. The diagonal line in each box indicates the relative position and orientation of the marker on the reference genome.

sampling strategies for genome comparisons, but to provide robust mathematical tools for performing the comparisons. Practical studies ([58], [74], for example) concur that the set of plausible medians is quite large and that it is misleading to present just one as "the" ancestral architecture of a set of genomes.

The focus of this work is on the identification of common structural features that are likely to be inherited from ancestral genomes. These super-blocks can be seen as complex traits in the sense of Dollo parsimony, whose conservation and possibly loss from a common ancestor is more likely than independent gain in separate lineages. They are identified without use of a hypothesized phylogeny, and indeed nothing suggests that recombinatory evolution coincides with mutational evolution.

This use of phylogeny is an important feature of the work of [120]. Super-blocks share certain aspects of the motivation behind CARs: that is assembling only adjacencies having sufficient support in contemporary genomes.

The sharing tree of super-blocks (such as seen in figure 3.8) encodes all the possibilities of ancestral genome architectures by including in the super-blocks the adjacencies common to at least 2 genomes, and leaving the super-block extremities as the only places where no semantically sound assembly is possible. This final assembly is then just a question of optimization under some metric, and in this work we use the Hannenhalli-Pevzner rearrangement distance.

The super-blocks themselves implement a compromise between the rearrangement and breakpoint distances, and thus, thanks to the latter, encode the ancestral semantics, while leaving room for optimization thanks to the former.

In practice, our method realizes two successive search-space reductions. First, the super-blocks themselves diminish the number of unresolved adjacencies (left for the optimization step). Second, we rely on the biological constraints for further search-space reduction, as well as solution filtering. In particular, in our application to the non-WGD yeasts we use the centromere positions, yielding biologically plausible solutions only.

Gene and Segmental Duplication

Accounting for gene and segmental duplication is an important challenge, that we do not address in this work. In [123] Martin et al. use the interleaving patterns of gene orders to study rearrangements before and after the hypothesized whole genome duplication (WGD) event in the *Saccharomyces* lineage [212]. Interestingly, they claim that a series of partial genome duplications leads to more parsimonious rearrangement scenarios that does a single whole genome duplication. In their study they combined rearrangement events with duplication and deletion events; during a preprocessing step their method renumbers duplicated elements in gene orders to produce a permutation compatible with the Hannenhalli-Pevzner rearrangement algorithms that they use. For computational reasons, only a single chromosome of *A. gossypii* is studied in detail. For this example our results agree; indeed, the segments in their figure 5 (and supplemental file S1 provided by reviewer 2) are found in our adjacent markers 52 and 51 (Figure 3.7), conserved in our median and all genomes we consider except *Z. rouxii*. Our study is otherwise quite different. Since we deliberately only consider species outside of the WGD lineage, we are not concerned with the large-scale duplications and deletions that mask the underlying rearrangement events. Our method works efficiently on complete genomes, and is not reliant on the Hannenhalli-Pevzner method, but rather proposes a partial reconciliation between it and the breakpoint method. Our super-blocks method does not take duplications into account, since it is not obvious how to weigh duplicated adjacencies when counting their frequency. This is a direction for future work.

Towards Ancestor Construction in Yeasts

Comparative genomics in the hemiascomycete yeasts has proven extremely informative about the basic mechanisms of eukaryotic molecular evolution, both using genetic tools and computer analysis. These species represent a homogeneous phylogenetic group with small and compact genomes, but nonetheless a large diversity at the physiological and ecological levels, and an evolutionary range comparable to the Chordate phylum [51, 50]. They provide a kind of 'evolutionary playground' in which various genome-modifying mechanisms have been tested over and over. Building a mathematical description of this rich history will provide important insight.

In this work we have used our super-block method to construct a plausible ancestral architecture for a phylogenetically circumscribed group of non-WGD yeasts, using ordered markers derived from all-against-all search for conserved syntenic segments. Surprisingly, highly similar sets of super-blocks are constructed from these markers, reinforcing the idea that the ancestral semantics can be recovered using adjacencies observed in contemporary genomes. Final assemblies of these super-blocks were constructed by an optimization procedure using the Hannenhalli-Pevzner rearrangement distance as a metric. A strength of our method is that such final assemblies can be made to respect biological constraints on chromosome architecture, in this work centromere position.

Since our method can efficiently handle hundreds of markers in dozens of genomes simultaneously, these results open the way to a more in-depth study of the rearrangement history of the yeasts. This will require technical advances, for detecting synteny in the presence of segmental duplication, for masking the effects of highly mobile elements, and for improved respect of biological constraints.

3.4 Efficient meta-heuristic for the Median Genome Problem

In this section we present a novel population-based local search algorithm for the *median genome problem*. The primary result is the improvement of the performance of ancestral genome reconstruction compared to existing methods, making it possible to tackle problems where the contemporary genomes may contain many hundreds of markers. Moreover, our method is not limited to triples of genomes, and thus solves the median genome problem in its generality. We show that in real application cases the computational results are highly robust, suggesting that we can interpret the computed median genomes as candidates carrying the semantics of ancestral architectures.

Two computational approaches for construction of ancestral genome architectures were proposed. They were formulated as the *Median Genome Problem* (MGP) and the *Multiple Genome Rearrangement Problem* (MGRP). Given a set of genomes $\{\Pi_i\}$, the former consists in computing a permutation Π minimizing the sum of distances to $\{\Pi_i\}$, while the latter aims at computing the Steiner tree thus minimizing the sum of distances along its edges. The Median Genome Problem has been shown to be NP-hard for both of distance functions even in the case of only 3 genomes (see [28, 152] for breakpoint distance and [33, 34] for rearrangement distance). Nevertheless, exact resolution of MGP have been attempted, yielding optimal solutions for very small instances [180].

Approximate algorithms for MGP have been proposed for both distances. In the breakpoint case, Sankoff and Blanchette formulated the solution through a reduction to the Travelling Salesman Problem [162]. These authors proposed an algorithm that guarantees a reasonable lower bound on the sum of distances.

In the case of rearrangement distance, MGP and MGRP have been tightly linked since for real-sized cases the proposed solutions for the latter rely on successive triangulations, and thus on the solving of the former in the 3-genome case. Two existing software packages *MGR* [23] and *rEvoluzer* [20] implement partial solutions of MGP. Indeed, MGR solves the problem for triples of multichromosomal genomes, while rEvoluzer can treat more than 3 genomes, but only in the unichromosomal

case. Both of the proposed solutions are heuristics based on the detection of "good" reversals, operations that are guaranteed to improve the solution. The genomes $\{\Pi_i\}$ are re-written step by step by applying reversals until two (or three) of them become equal. There are two differences in the proposed solutions. First, the definition of what constitutes a good reversal is not exactly the same. Second, when no good reversals remain, MGR performs a k-depth search to find a best reversal, while rEvoluzer allows for backtracking.

In this study, we present FAUCILS, a new approximate algorithm for MGP in the general case, that is, for an unrestricted number of multichromosomal genomes, while improving performances of existing approaches on restricted instances. The mainly originality of our approach is the definition of a probabilistic neighborhood which evolve within a population-based local search according to observations made on the population. This mechanism allows us to greatly accelerate the search and ensures more convergence, especially for real or structured instances.

We consider *multichromosomal genomes* over the set of n markers, and we say that such a genome is of size n. For example, $\{(5,-8),(1,2,-10,6,4),(9),(-3,7)\}$ is a genome of size 10. Given a set of size-n genomes $\{\Pi_i\}$ and a genome distance function d, an instance of the combinatorial minimization problem MGP is defined by two elements $\langle \tau_n, \phi \rangle$:

1. a search space, τ_n, composed of the set of all possible size-n genomes, and
2. an objective function $\phi : \tau_n \rightarrow \mathbb{N}$ (*score*) defined by $\phi(\Pi) = \sum_{\Pi_i} d(\Pi, \Pi_i)$.

A *median genome* for a given set of genomes $\{\Pi_i\}$ is a genome Π that minimizes $\phi(\Pi)$. Every optimal solution to MGP is a median genome.

3.4.1 An original population-based local search for MGP

For addressing NP-complete problems like MGP in the general case and reaching acceptable solutions in reasonable time, approximate algorithms provide the most practical approach. We present a population-based local search algorithm using an original and evolutive neighborhood reduction mechanism for the resolution of MGP in the case of rearrangement genome distances. It gives excellent results in terms of the quality of the solutions it obtains, the speed of the computation, its robustness, and its scalability.

A descent algorithm for MGP

Stochastic Local Search (SLS) [90] is a well-known class of metaheuristics, used for the resolution of many difficult combinatorial optimization problems. SLS algorithms are iterative methods which start from an initial configuration (candidate solution of the search space) and improve it by successive local modifications. In this section we define a simple descent algorithm to MGP, where:

1. the *initial configuration* is taken from $\{\Pi_i\}$,
2. the *evaluation function* is the same as the objective function ϕ: the rearrangement distance d,
3. the *neighborhood relation* we call $\mathcal{R}^1$ is a 1-step rearrangement: $\mathcal{R}^1(\Pi) = \{\Pi' \in \tau_n, d(\Pi, \Pi') = 1\}$,
4. the *move strategy* is a *first-improve selection* (FI) which accepts better and equivalent configurations (*side-walk* mechanism, SW [170]), given a specified number of iterations *nbit*.

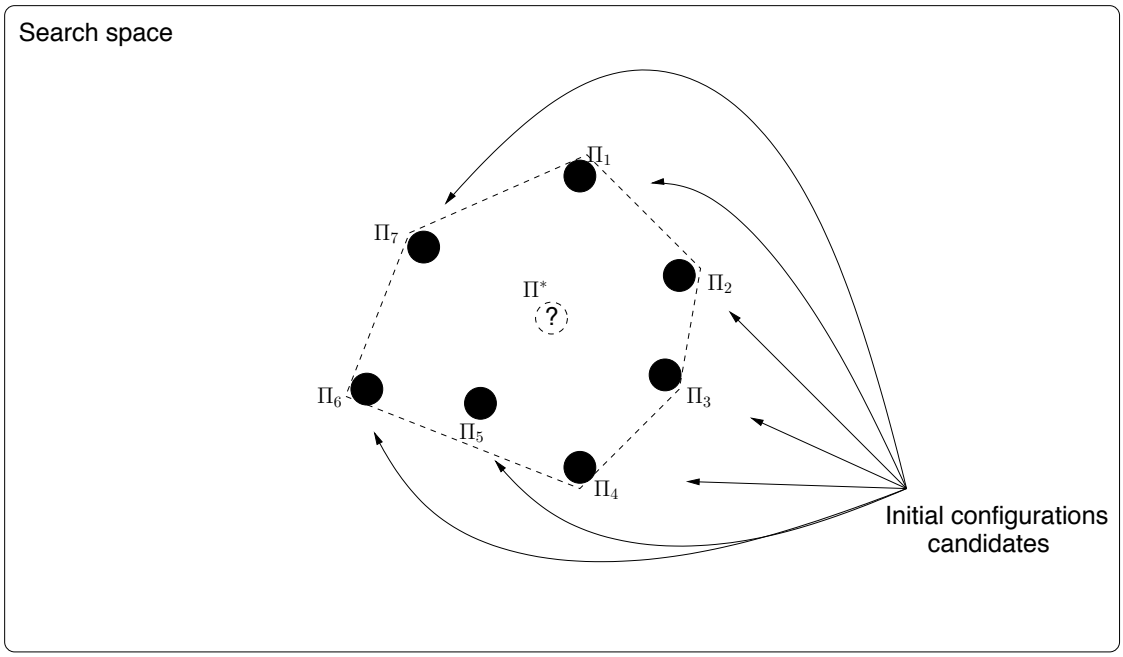


Figure 3.9: Geometric analogy of MGP: median genomes are within the convex hull of $\{\Pi_i\}$ in the space of genomes. Starting the search from a Π_i on the perimeter greatly reduces the search space.

The performance of a descent algorithm essentially depends on the neighborhood relation used [131]. In order to avoid slow processes and local optima difficulties, we use the FI+SW selection strategy combined with the large and straightforward neighborhood $\mathcal{R}^1$.

Figure 3.9 shows that configurations taken from $\{\Pi_i\}$ may be interesting initial candidates for the beginning of the search. Considering the evaluation function and the neighborhood relation $\mathcal{R}^1$, the descent will explore only configurations from the schematic area delimited by the Π_i. If the Π_i are close (for example in the case of real applications), then the resulting search space is significantly reduced.

Probabilistic population-based local search

Traditionally, descent algorithms are sensitive to either stochastic factors or initial configurations and consequently may not be sufficiently robust – that is, different executions may diverge – although the SW mechanism and the use of a large neighborhood can reduce this drawback. A commonly used solution is to perform several descents from different initial configurations (different *replications* in a *multi-start descent* process). In genetic local search algorithms, local search processes and crossovers between elements (*individuals*) of a set or multiset of current configurations (*population*) provide intensification and diversification phases.

A local search process applied to many independent replications is sometimes called a *population-based local search* even though there is no interaction between individuals [150]. Here we do not use any crossover operations, but simulate an alternative evolutionary process in order to accelerate the searches and to make multi-start descents more convergent.

We introduce a probabilistic population-based local search algorithm which favours, at each step of the search, the selection of most pertinent neighbors [75] with respect to the population. Structural

information about each individual is used to estimate a selection probability at each step of the search. In this process, all replications are dependant, while the descents are carried out simultaneously.

In this section we present a multi-start descent for MGP. We use the descent mechanism presented in section 3.4.1, adding to each neighbor a selection probability.

Let $\mathcal{P}$ be the population of our population-based descent, which initially contains individuals taken from $\{\Pi_i\}$. Now let us consider a probabilistic function $p : \tau_n \times \tau_n \times \tau_n^{|\mathcal{P}|-1} \rightarrow [0, 1]$, such that $p(\Pi, \Pi', \mathcal{P} \setminus \{\Pi\})$ gives a selection probability of $\Pi' \in \mathcal{R}^1(\Pi)$. Such a probabilistic function is quite similar to the one used for simulated annealing move strategy. The difference here is that only better or equivalent neighbors are accepted by the move strategy, whereas neighbors are generated by a probability distribution (*probabilistic neighborhood* [131]). The aim is not to escape to local optima, but to favour neighbors which share properties with other individuals in $\mathcal{P}$.

This probabilistic function is connected to the notion of *adjacencies*, that we define in the way analogous to Nadeau and Taylor [133].

Definition 19 *Two consecutive elements π_i and π_{i+1} of a chromosome $\pi \in \Pi$ are said to be* adjacent *in Π. We note this adjacency by $(\pi_i.\pi_{i+1})$.*

We consider additional adjacencies at the extremities of each chromosome by introducing marker 0. For a chromosome $(\pi_1, \ldots, \pi_m)$, two adjacencies are added: $(0.\pi_1)$ and $(\pi_m.0)$. Notice that $(\pi_i.\pi_j) = (-\pi_j. - \pi_i)$ and $(0.\pi_i) = (-\pi_i.0)$. Finally, we note $\mathcal{A}(\Pi)$ the set of all adjacencies in Π. We have $|\mathcal{A}(\Pi)| = n + N$ (n is the number of markers and N the number of chromosomes).

Each move (rearrangement) breaks one (fission) or two (reversal, fusion, reciprocal translocation) adjacencies. The probabilistic neighborhood encourages adjacencies which are not, or are less, represented in the population to be broken. The probabilistic neighbor selection operates as follows: let $\Pi' \in \mathcal{R}^1(\Pi)$; if $\phi(\Pi') \leqslant \phi(\Pi)$, then Π' replaces Π in $\mathcal{P}$ in function of the proportional representation of the broken adjacencies in $\mathcal{P} \setminus \{\Pi\}$:

$$p(\Pi, \Pi', \mathcal{P} \setminus \{\Pi\}) = 1 - \frac{|\{\Pi'' \in \mathcal{P} \setminus \{\Pi\}, (\mathcal{A}(\Pi) \setminus \mathcal{A}(\Pi')) \cap \mathcal{A}(\Pi'') \neq \emptyset\}|}{|\mathcal{P}| - 1}$$

Algorithm 5 provides an overall view of our probabilistic population-based descent we called FAUCILS for *Fast Ancestor (inference) Using Convergent and Intelligent Local Search*.

3.4.2 Experiments

For experiments we use different kinds of instances: real and random ones, with different numbers of genes and chromosomes by genome.

Real instances

First we assess our algorithm FAUCILS on two sets of 10 triplets of yeast genomes. The data, provided by Génolevures Consortium[5] (GDR CNRS 2354), consists in five sequenced yeasts from the Kluyveromyces clade: *Kluyveromyces lactis* (Klla), *Saccharomyces kluyveri* (Sakl), *Zygosaccharomyces rouxii* (Zyro), *Ashbya gossypii* (Ergo) and *Kluyveromyces thermotolerans* (Klth). From these data, two sets of permutations have been computed: the first one with 135 markers (K135), and the second one with 499 markers (K499). For the comparison with MGR, which calculates only 3-genomes medians ($N = 3$), we separate in ten instances each possible triplet of genomes: Klla-Sakl-Zyro is K135-1 and K499-1, Klla-Sakl-Ergo is K135-2 and K499-2, ... These five genomes have

[5] `http://www.genolevures.org`

Algorithm 5

Require: $\{\Pi_1, \ldots \Pi_k\}$: a set of k multichromosomal genomes of size n; $l.k$: the size of the population; *nbit*: the number of descent iterations.
Ensure: an approximate median genome set $\hat{\mathcal{P}}$
  let $\mathcal{P}$ be the multiset of the current genomes population, which initially contains l copies of each Π_i.
  let *numit* $\leftarrow 0$ be the number of performed local search iterations
  while *numit* < *nbit* **do**
    for all $\Pi \in \mathcal{P}$ **do**
      loop
        randomly select $\Pi' \in \mathcal{R}^1(\Pi)$
        break with probability $p(\Pi, \Pi', \mathcal{P} \setminus \{\Pi\})$
      end loop
      if $\phi(\Pi') \leqslant \phi(\Pi)$ **then**
        $\Pi \leftarrow \Pi'$
      end if
    end for
    numit $\leftarrow$ *numit* $+ 1$
  end while
  return $\hat{\mathcal{P}} = \text{argmin}_{\Pi \in \mathcal{P}}\, \phi(\Pi)$

Instance	k	n	FAUCILS						MGR		MGR-H1		Δ
			ϕ_b	ƒ	ϕ_m	ϕ_w	σ	CPU	ϕ	CPU	ϕ	CPU	
K135	5	135	**281**	7/20	281.7	282	0.5	42m	-	-	-	-	-
K135-1	3	135	**168**	1/20	170.6	172	1.1	15m	177	355m	178	44m	-9
K135-2	3	135	**115**	5/20	116.1	117	0.8	14m	119	188m	120	6m	-4
K135-3	3	135	**150**	1/20	151.9	153	0.7	15m	157	348m	160	60m	-7
K135-4	3	135	**132**	14/20	132.3	133	0.5	14m	135	377m	136	13m	-3
K135-5	3	135	**166**	1/20	168.0	169	0.9	15m	173	400m	175	48m	-7
K135-6	3	135	**110**	5/20	111.1	112	0.8	15m	112	148m	114	8m	-2
K135-7	3	135	**160**	1/20	161.7	164	0.9	13m	162	300m	172	26m	-2
K135-8	3	135	**185**	4/20	186.6	188	1.1	13m	193	527m	194	197m	-8
K135-9	3	135	**145**	3/20	146.6	148	1.0	13m	154	296m	154	45m	-9
K135-10	3	135	**159**	2/20	160.6	163	0.9	15m	167	355m	165	30m	-6
K499	5	499	**564**	6/20	565.9	568	1.7	98m	-	-	-	-	-
K499-1	3	499	**407**	5/20	408.1	410	0.9	44m	/	max	413	457m	-6
K499-2	3	499	234	1/20	235.0	236	0.3	38m	/	max	**233**	105m	+1
K499-3	3	499	**338**	6/20	339.1	341	0.9	43m	/	max	339	297m	-1
K499-4	3	499	263	4/20	263.9	265	0.6	39m	/	max	**262**	106m	+1
K499-5	3	499	**372**	2/20	373.7	375	1.0	43m	/	max	375	391m	-3
K499-6	3	499	181	10/20	181.6	183	0.6	38m	**179**	2 days	**179**	74m	+2
K499-7	3	499	**375**	1/20	377.1	379	1.0	37m	/	max	381	317m	-6
K499-8	3	499	**476**	3/20	479.2	481	1.6	40m	/	max	484	823m	-5
K499-9	3	499	**307**	2/20	310.3	310	0.9	35m	/	max	309	230m	-2
K499-10	3	499	**338**	3/20	340.0	343	1.4	42m	/	max	**338**	251m	=
HCM	3	114	**48**	20/20	48.0	48	0	<1m	**48**	10m	**48**	<1m	0

Table 3.4: Comparison between FAUCILS and MGR on real instances.ϕ_b is the best score returned by FAUCILS, ƒ is its frequency, ϕ_m is the mean score, ϕ_w is the worst score, σ is the standard deviation based on 20 different executions.

respectively 6, 8, 7, 6 and 8 chromosomes. We add a real test instance composed by the genomes of Human, Cat and Mouse, and available on the MGR web page[6].

[6] `http://nbcr.sdsc.edu/GRIMM/mgr.cgi`

Table 3.4 shows performances of FAUCILS and MGR on these real instances. FAUCILS is a stochastic algorithm, and two executions may return different results; for each instance we perform multiple executions. Table 3.4 indicates the best results ϕ_b of 20 executions, their frequency ƒ, the mean scores ϕ_m, the worst scores ϕ_w, the standard deviations σ and the mean computation times of one execution. FAUCILS was run with its default parameters: $l = 3$ (*i.e.* a population size of 9 when $k = 3$), and one million LS iterations (*nbit*); MGR was first run with its default parameters, and secondly with the heuristic option H1 (MGR-H1) for speeding up the search. Each execution was performed on a node of Grid'5000[7] and the computational time limit per compute node was fixed to one week. In all tables, Δ gives the difference between the best score returned by MGR and the best score returned by FAUCILS.

From Table 3.4 one observes that FAUCILS computes better median genomes than MGR. For all the K135 instances, FAUCILS performs better than MGR with 5.9 rearrangements less per instance for the bests runs, and 4.4 rearrangements less in the mean for all the 200 runs (10×20) with low computation times (about 15 minutes against hours for MGR). The MGR H1 heuristic does speed up the program, but the returned solutions are less competitive (except for K135-10 where MGR-H1 beats MGR with default settings). When the number of markers is big (499), MGR needs to be used with its speed resolution heuristic to complete the search, and for these instances FAUCILS and MGR are more comparable in term of scores, although FAUCILS remains better in mean and faster.

Finally, FAUCILS is also robust with more than three genomes (instances K135 and K499) since the returned solutions have very close scores. The increase of the computation time mainly depends on the population size parameter, which can be reduced.

Random instances

Instance	n	N	FAUCILS						MGR		MGR-H1		Δ
			ϕ_b	ƒ	ϕ_m	ϕ_w	σ	CPU	ϕ	CPU	ϕ	CPU	
R50-1	50	1	**79**	2/20	80.6	82	0.8	4m	82	6m	83	2m	-3
R50-2	50	1-9	**80**	1/20	81.5	82	0.6	6m	87	5m	87	2m	-7
R50-3	50	3-7	**80**	6/20	80.8	82	0.6	6m	84	7m	85	1m	-4
R50-4	50	5-5	**79**	5/20	80.2	81	0.8	6m	83	8m	84	1m	-4
R100-1	100	1	**171**	1/20	173	175	1.0	8m	175	801m	177	126m	-4
R100-2	100	2-10	**166**	1/20	168.8	170	1.1	10m	176	250m	178	226m	-10
R100-3	100	3-7	**170**	2/20	171.9	174	1.2	10m	174	310m	179	139m	-4
R100-4	100	5-5	**166**	1/20	169.8	172	1.5	10m	169	464m	171	170m	-3
R200-1	200	1	**354**	1/20	357.2	362	1.9	16m	/	max	/	max	-
R200-3	200	10	**351**	1/20	356.1	360	2.0	19m	/	max	/	max	-
R200-4	200	11-20	**344**	2/20	347	350	1.7	21m	/	max	366	4 days	-22
R500-1	500	1	**924**	1/20	928.2	932	2.1	37m	/	max	/	max	-
R500-2	500	10	**942**	2/20	945.4	950	2.3	42m	/	max	/	max	-
R500-2	500	18-44	**936**	1/20	942.7	947	3.1	40m	/	max	/	max	-

Table 3.5: Comparison between FAUCILS and MGR on random instances

In order to assess the performance of our population-based local search algorithm with respect to the structure and the size of the instance, we generate two types of random instances.

First, we use completely random instances (R) containing a specified number of markers, and a minimum and maximum number of chromosomes by genome (N). On these instances of size 50 and 100, FAUCILS obtains better results than MGR systematically (see Table 3.5). For larger instances, only one MGR run ended, with an uncompetitive result ($\Delta = -22$). These instances seem to be difficult because of their structure: each genome is a random point of τ_n, and the MGR

[7] https://www.grid5000.fr

Instance	n	N	div	FAUCILS						MGR		MGR-H1		Δ
				ϕ_b	f	ϕ_m	ϕ_w	σ	CPU	ϕ	CPU	ϕ	CPU	
S100-10-1	100	1	10	**10**	20/20	10.0	10	0	<1m	**10**	<1m	**10**	<1m	=
S100-10-2	100	5	10	**10**	20/20	10.0	10	0	<1m	**10**	<1m	**10**	<1m	=
S100-10-3	100	10	10	**10**	20/20	10.0	10	0	<1m	**10**	<1m	**10**	<1m	=
S100-50-1	100	1	50	**50**	20/20	50.0	50	0	<1m	**50**	8m	51	<1m	=
S100-50-2	100	5	50	**49**	20/20	49.0	49	0	<1m	**49**	8m	**49**	<1m	=
S100-50-3	100	10	50	**49**	20/20	49.0	49	0	<1m	**49**	13m	**49**	1m	=
S100-100-1	100	1	100	**95**	7/20	95.7	97	0.6	<1m	97	190m	98	2m	-2
S100-100-2	100	5	100	**95**	2/20	96.4	98	0.7	<1m	96	55m	98	2m	-1
S100-100-3	100	10	100	**96**	4/20	96.9	98	0.5	<1m	99	70m	99	4m	-3
S100-200-1	100	1	200	**155**	1/20	158.6	160	1.4	5m	163	978m	166	65m	-8
S100-200-2	100	5	200	**145**	1/20	146.6	148	0.7	5m	151	331m	151	67m	-6
S100-200-3	100	10	200	**143**	1/20	145.7	154	2.2	5m	150	114m	154	78m	-7

Table 3.6: Comparison between FAUCILS and MGR on simulated instances

algorithm seems very dependent on the structure of each instance (see the divergences between all computational times on tables 3.4, 3.5 and 3.6).

In order to estimate the impact of the structure of the instance, we generate simulated instances (S), for which distances between genomes are bounded. An arbitrary ancestral genome is generated from which a specified number of random rearrangements are applied to give three genomes. We specify the number of genes (n) and chromosomes (N), and the number of rearrangements done during the simulation (r); this parameter is an upper bound of the optimal median genome score.

The results are given in table 3.6. We can see that, with $r = 10$ or $r = 50$, instances are very easy to solve. But when the distances between genomes increase ($r = 100$ and $r = 200$), FAUCILS is very competitive and can find in short computation time solutions considerably better than MGR. Moreover, the algorithm is robust as small values of σ show. For these instances (S), we have to reduce the number of local search iterations to $2000.r$ for an equivalent efficiency.

The evolution of the ratio ϕ/r gives an empirical indication of the structure of the search space. Indeed, for $r = 200$, the minimal number of rearrangements required for reconstructing an evolutionary scenario is about 25% lower than the number of rearrangements made during the simulation. Adding to the relative difficulty to find near-optimal genomes for these instances, we can presume that this ratio represents the quantity of lost information and can be a good indicator for comparing the difficulty of simulated instances.

Finally, we have executed rEvoluzer [20] on each unichromosomal instance: S100-10-1, S100-50-1, S100-100-1, S100-200-1, R50-1, R100-1, R200-1, R-500-1. Except for the three first instances, where rEvolzer founds in few seconds or minutes the sames scores as FAUCILS (10, 50, 95), the program did not return any solution for the five other instances, even given one week of computation.

Influence of the probabilistic neighborhood One of the main originalities of FAUCILS is that neighbors are selected with a non-uniform probability. The foremost aim is to select more pertinent neighbors as a function of the similarities between individuals in the current population. In [74] we show that since the population is initialized by the given genomes (instance), the probabilistic selection will have a larger impact on structured instances, that is when genomes share adjacencies; it is notably the case of real data instances. On the contrary, on completely random instances both mechanisms have the same efficiency. Indeed, such instances have insignificant numbers of shared adjacencies, and the probabilistic selection has no effect.

3.4.3 Conclusion

In this work we proposed a new efficient algorithm for the resolution of the Median Genome Problem (MGP) in the general case. We have notably introduced an novel way for speeding up and making more convergent multi-start descents for the resolution of MGP, especially for real structured instances. The key idea is to use a probabilistic neighborhood which evolves during the search according to the partial results of all descents performed simultaneously.

Experiments realized both on real and random instances show that our software FAUCILS is able to find largely better solutions than MGR, the current reference in the domain. Moreover, this local search approach is very fast and scalable: contrary to other existing techniques, FAUCILS can treat an unbounded number of multichromosomal genomes, which may contain hundreds or thousands of markers. Future work will involve finding ways to evaluate the quality of solutions in the case of big instances, and to extend this MGP algorithm for the resolution of MGRP. The Median Genome Rearrangement Problem is a very hard computational problem for the resolution of which existing algorithms calculate multiple median genomes.

Chapter 4

BioRica: dynamic modeling formalism and platform

4.1 Introduction

A general goal of systems biology is to acquire a detailed quantitative understanding of the dynamics of living systems. Often this goal is tackled through computer simulation. Quite a number of different formalisms and simulation techniques are currently used to construct numerical representations of biological systems, and a certain wealth of models is proposed using specific and *ad hoc* methods.

There arises an interesting question of whether these models are reusable and composable between themselves or in a larger framework. As a means to circumvent the difficulty of incorporating disparate approaches in the same numerical study, ae propose BioRica, a high-level modeling method coupled with an underlying unified efficient simulator (for a complete paper refer to [79]). On the practical level, BioRica models are compiled into a discrete event formalism capable of capturing discrete, continuous, stochastic, nondeterministic and timed behaviors in an integrated and non-ambiguous way.

Recent advances in biological experimentation have increased the gap between data collection and information extraction. To understand biological systems corresponding to these collected datasets, bottom up approaches in *systems biology* aim at building mathematical models whose emergent properties mimic *in vivo* observable properties. For example, integration of molecular interactions predicts and models yeast cell cycle control [37]. At a higher level, integration of cell to cell variability in cell cycle predicts evolution of a yeast culture.

Building these integrative models is a challenge since most systems biology models concentrate on only one well defined cellular process. In a higher-level system, such models become sub-systems. For example, these models could form a logical hierarchy, such as a yeast culture *that is composed of* yeast cells, *that are in turn composed of* a plasma membrane, a nucleoid and multiple regulation processes *that are in turn composed of* specific regulation processes... Each sub model interacts and cooperates with its siblings and contained sub models, and these interactions have themselves to be modeled.

Furthermore, for the far-off goal of building a comprehensive whole cell model would ideally re-use already published and verified models of sub-systems. However, since different authors use different and apparently incompatible formalisms, such an approach requires a framework for *hybrid models* able to combine and capture inherently different phenomena (e.g. continuous, discrete and stochastic models).

However, defining and simulating such hybrid models presents three challenges. Most biologi-

cally precise models are described using ODEs and in practice, combining uncoupled ODEs is not obvious. The first difficulty arises since multiple identical ODE systems evolving in parallel result in biologically unrealistic total time synchronicity [128]. Second, co-existence of different kinds of ODEs can lead to a multi-scale model, that is computationally inefficient [59]. Third, composition of models that use discrete formalisms leads to semantic incompatibility.

Facing this heterogeneity, what can be a solution for such integrative analysis? The BioRica framework we present here can combine different formalisms within a single framework that can efficiently simulate the resulting model. BioRica is a high-level modeling framework integrating discrete and continuous multi-scale dynamics within the same semantics domain, while offering a easy to use and computationally efficient numerical simulator. It is based on a generic approach that captures a range of discrete and continuous formalisms and admits a precise operational semantics.

Related work The E-Cell package [191] is a multi-algorithm simulation framework, where each component is associated with a distinct stepping function. This function is chosen according to the simulation algorithm for the component's model. Time is advanced using a dependency order that designates the next stepping function that is allowed to advance.

T.R. Kiehl et al. integrate both discrete and continuous cellular processes under the assumption that continuous events can occur between two discrete events [107]. Discrete events such as transcription, translation and molecular signaling are allowed to be stochastic, which is not the case for their continuous counterparts. In a way analogous to K. Takahashi, more than one solver co-exist [191]. The existence of multiple solvers allows each solver to automatically adapt to the most appropriate time-scale, as described by J.M. Esposito where the system is integrated asynchronously one component at a time [59]. The issue of uniform behavior for multiple instances of the same process modeled by ODEs has been initially studied in [128] where the authors use pre-processing to generate models where the base unit is repeated in fixed topologies by varying parameters. Such an approach differentiates each unit but leads to varying qualitative properties since parameters play a predominant role in the dynamics of a continuous process (see e.g. in application to cell cycle [200, 37, 36]). That in turn can fundamentally alter the average behavior of the population.

4.2 The BioRica platform

In order to provide an unified and efficient analysis framework for multi models systems, the BioRica approach (see figure 4.1) is based on an initial modeling step that expresses each model as a *BioRica node*; these latter are the main units of a system. The nodes are then hierarchically composed and connected to build a *BioRica system* that is itself transformed by our simulation toolkit into a native and optimized computer program, that can be run to simulate the system.

4.2.1 Systems description

BioRica nodes are the base description unit of our models and are designed to have the capacity of modeling phenomena of different natures, from discrete instantaneous transition systems to randomized continuous processes. Based on the level of detail needed for the study, one can gradually move from a simple and abstract view to a more refined model, as will be illustrated in the following by the successive refinement of a cell division cycle example.

Discrete systems. Nodes in BioRica are composed of a finite number of states and of transitions between those states. They are described in BioRica by *constrained events* of the type $G \xrightarrow{e} A$ where G is a constraint denoting the conditions activating this transition and A is an assignment to state

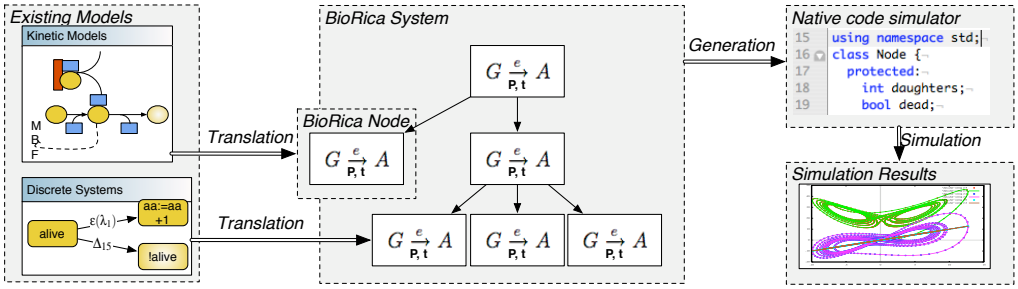


Figure 4.1: The BioRica framework. Existing hetergeneous models are expressed as BioRica nodes, consisting of constrained events. These nodes comprise a BioRica system which is then simulated.

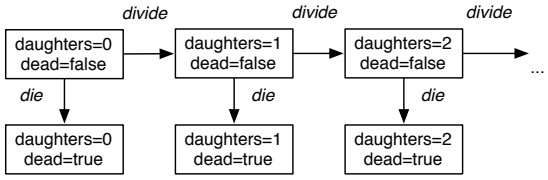


Figure 4.2: A cell division cycle, modeled as a transition system built with two variables "daughters" and "dead" respectively denoting the number of daugthers and the state of the cell (e.g. dead or alive). It is defined in BioRica by a node containing those variables and the constrained events: $dead = false \xrightarrow{divide} daughters := daughters + 1;\ \ true \xrightarrow{die} dead := true.$

variables. Since two events need not have disjoint constraints, they could be activated for the same configuration of variables and thus describe *non deterministic behavior*. This construction captures a vast range of existing discrete formalisms (Petri nets, finite automata etc.). The semantics of these nodes can be given in terms of *transition systems*, such as the one shown in figure 4.2. that depicts a gross and discrete abstraction of a cell division cycle.

Stochastic behavior can be added to a BioRica node to describe the likelihood that an event fires when it is activated concurrently with another event. This is done via optional *probability measures* attached to events. Since a probability measure can be omitted for some (or all) events, Markov chains or Markov decision processes can be concisely described in BioRica. The previous cell cycle example could thus be refined by assigning to the event *die* the probability $(1 - 1/\text{daughters})$ to describe a cell whose probability of dying increase with the numbers of division it went through.

Timed behavior can be added to a BioRica node to describe the delay between an event's activation and the moment its transition occurs. This is done via *timing functions* attached to events, which can be of three kinds. First, they can be normal functions from node's variables configurations to the real numbers; second, probability distributions parametrized by the current configuration; and third, linear combinations of timing functions. Given the current node's variables values, these timing functions output the delay needed for the associated event to complete. Note that two events can *race* since two events can be activated concurrently and since the firing of an event can lead to the deactivation of another pending event. This enables the description of subtle concurrency issues. These timing functions can be omitted for some events, thus allowing a node to mix different types of randomness (event likelihood and delays) with non deterministic behaviors. Furthermore, we make no restriction on the kind of probability distribution that can be used (e.g. discrete Dirac comb, ex-

ponential, binomial etc.).This allows for flexible and concise description of a vast range of models coming from control theory (stochastic Petri nets, generalized Markov decision processes etc.). The cell cycle example can be refined by attaching to the *divide* event a delay of 1 time unit while the die event can have a random delay following an exponential distribution parametrized by the number the division, such as $\varepsilon(1/\text{daughters})$, that consequently shortens as the number of divisions increases.

Continuous systems. ODEs systems can be described in BioRica nodes and can have randomized perturbation or random parameter switches added to them. This is done by considering that every ODE system is potentially an *hybrid system*, that is, a system whose internal state flows continuously while having discrete jumps. The discrete jump is modeled by constrained events while the internal state is kept hidden in a timing function, which outputs the time before a constrained event will happen. For convenience, ODE systems can be directly described into a BioRica node or can be imported from a SBML file.

In the cell cycle example, the dynamics of the cycle can be described by the variation of protein concentrations, described by using differential equations. Constrained events triggered by crossing protein concentration thresholds can describe what precisely happens at division time (cell mass is divided, some proteins are transmitted to the daughter etc.). The visible part of this system (and thus the discrete variables of the BioRica node describing it) can be the number of division or the current phase of the cell division cycle.

At a higher level, consider now a cell population system, where each cell is represented by a node having its dynamics described by an ODE system. Since ODE systems are inherently deterministic, once initial conditions and parameters are set, every cell will behave exactly like its neighbor, thus leading to an artificially synchronized population [128]. In BioRica, since ODE systems are translated in timing functions, we can easily add random perturbations to timing functions by adding to the numerical integration result an exponentially distributed random variable. Thus, each node can be slightly shifted in time while preserving the qualitative properties of its ODE system. Such a cell population can be described from a cell node in BioRica by using *node composition*.

Node composition. Complex and multi models systems can be described in BioRica via hierarchical descriptions, denoting composition and interactions relation between any kind of nodes. In a hierarchical description, some nodes will interact and cooperate, for example by being in the same compartment and thus sharing the same source of product. To this end, *flow connections* between nodes can be established to allow exchange of information. Another kind of node interaction taken into account in BioRica is *event synchronization*, which can impose that events in different nodes have to happen simultaneously. By default, if no node interactions are defined, nodes will just evolve independently in an asynchronous way, that is evolve one component and one event at a time.

4.2.2 System simulation

BioRica systems are hierarchical and modular descriptions: each node can use the most suitable modeling approach and may interact with nodes using other modeling approaches. This results in great flexibility to describe complex models, and also enables a simple and efficient simulation scheme.

Numerical simulation of a BioRica system is based on a discrete event approach, where changes occur only when an event is fired. To this end, the system is initially fully scanned for initially active events. These events are then scheduled either immediately (for untimed events) or in the future (when a timing function is associated to the event). Afterwards, the stepper chooses randomly or non deterministically one of the events whose scheduled time is the nearest, and updates the system accordingly to the chosen event's assignments. Since these assignments can alter variables occurring

in the constraints or timing functions of other events, some scheduled or inactive events need to be reexamined and, if needed, rescheduled. This step does not involve a full system exploration since we can infer from the system's description a static relationship between events and variables, and therefore only examine the set of events affected by the assignment. Afterwards, the system steps and chooses the next event.

Multiscale integration. The BioRica framework can simulate systems having nodes whose underlying ODEs systems use different time scales. This is done by assigning to each node a private numerical integrator, which can use a local step size and thus can be adapted to the node local configuration and ODE stiffness. The multi scale problem arises mostly when composing ODE nodes whose time scales range over different order of magnitudes.

Furthermore, the BioRica simulator can speed up simulation involving similar ODEs by using a memoization scheme. This is done by reusing previously stored ODE integrations when detecting that two ODE systems have reached the same trajectory or that an ODE system reached an oscillatory state. This approach is mathematically sound since solutions to ODE systems enjoy a memory-less property. Basically, for a given set of parameters, the solution of an ODE system is completely determined by its initial conditions. This implies, among other properties, that once an ODE node reaches a state that was previously seen (either in the same node or in another node having the same ODE and parameters), then it will behave exactly in the same way. More formally, consider an ODE system of dimension n and its solution, the trajectory function f from $\mathbb{R}$ (time) to $\mathbb{R}^n$ (variables values). Let t and t' be two point in time, with $t < t'$. Whenever f takes the same value in two separate point of time, then it will take the same value for every corresponding successive point. That is, if there exist two real numbers t and t' such that we have $f(t) = f(t')$, then for any real number ϵ, we have $f(t + \epsilon) = f(t' + \epsilon)$.

4.3 Transient Behavior in Parametrized Dynamic Models

Quantitative models in Systems Biology depend on a large number of free parameters, whose values completely determine behavior of models. These parameters are often estimated by fitting the system to observed experimental measurements and data. The response of a model to parameter variation defines qualitative changes of the system's behavior. The influence of a given parameter on the system's behavior can be estimated by varying it in a certain range. Some of these ranges produce similar system dynamics, making it possible to define general trends for trajectories of the system (e.g. oscillating behavior) in such parameter ranges. Such trends can be seen as a qualitative description of the system's dynamics within a parameter range. In this work, we define an automata-based formalism to formally describe the qualitative behavior of systems dynamics. Qualitative behaviors are represented by finite transition systems whose states contain predicate valuations and whose transitions are labeled by probabilistic delays. Biochemical system dynamics are automatically abstracted in terms of these qualitative transition systems by a random sampling of trajectories. Furthermore, we use graph theoretic tools to compare the resulting qualitative behaviors and to estimate those parameter ranges that yield similar behaviors. We validate this approach on published biochemical models and show that it enables rapid exploration of models' behavior, that is, estimation of parameter ranges with a given behavior of interest and identification of some bifurcation points.

Dynamic models in System Biology rely on kinetic parameters to represent the range of possible behaviors when enzymatic information is incomplete. Analysis of these parametrized models aims at identifying either parameter ranges yielding similar qualitative behaviors, or parameter values yielding a given behavior of interest. Qualitative transient behavior can be successfully analyzed by model checking algorithms applied on models admitting a computable path semantics. However, in

Systems Biology, state explosion and negative decidability results limit the scope of model checking to a certain subset of models. Moreover, some published and curated Systems Biology models lack explicit semantics, and for these "black box" models not much can be assumed, except the possibility of generating simulation results. Mining these simulation results to identify parameter regions yielding similar behaviors is hindered by the size of the parameter space to explore, numerical artifacts and the lack of formal definition of what it means for simulation results to be similar.

In this section, we propose the new formalism of *qualitative transition systems* for abstracting simulation results in terms of discrete objects that admit efficient similarity measures.

Indeed, simulation results for ODEs are obtained using numerical integration schemes operating on floating point numbers. The resulting approximation is problematic for the identification of precise transient properties since transient properties of interest are mathematically characterized by using equality between real numbers (e.g. $f'(x) = 0$ is necessary for a local maximum), which is inconsistent in floating point arithmetic[38]. Consequently, even analysis of basic properties, like the detection of the first time a deterministic system is in a previously visited state, fails in practice due to this inconsistency. Furthermore, different integration schemes (n-th order, implicit/explicit) yield different and incomparable numerical approximations of the same trajectory. Although using normalized sampling and fixed precision decimal numbers would seem to solve this problem, the multiplicity of time scales in ODEs show that this solution is not completely satisfying.

For dynamic models admitting a computable path semantics, the impact of numerical artifacts is absent. Indeed, it is possible to compute a finite description of the set of trajectories of the model. Consequently, for these models, model checking algorithms can decide if a logical representation of a behavior holds, and if not, can provide a counter example. Recently, a probabilistic model checking approach was successfully used to solve the inverse problem: given a logical representation of a transient behavior, return a parameter space in which any trajectory satisfies the specified behavior with sufficiently high probability[159]. For dynamic models suitable for model checking, the intuitive notion of "similar behavior" is thus fully formalized and generally decidable.

Contributions In the next section we introduce *Qualitative Transition Systems* (QTS) and define their probabilistic semantics. A novel abstraction operation is defined in section 3 with the goal of building QTSs from simulation results. We then show in section 4 that when constructing a QTS from an ODE, the QTS construction can be made independent of the numerical integration scheme. In section 5, we show that trajectory comparison using QTS can be made more resistant to noise by detecting points of interest (extremums and inflection) through the construction of a piecewise linear approximation (PLA). In section 6, we validate our approach on models from the literature.

4.3.1 Qualitative Transition Systems

Given a set Σ, we denote by Σ^* the set of all (finite) *words* $s_0 \cdots s_k$ over Σ. A (finite) *timed word* over Σ is any word $W = (t_0, s_0) \cdots (t_k, s_k) \in (\mathbb{R}_{\geq 0} \times S)^*$ such that $t_i < t_{i+1}$ for all $0 \leq i < k$. The non-negative real numbers t_i are interpreted as the absolute *observation times* and the s_i are the *observed values*. We will focus in the paper on the particular case of $\Sigma = \mathbb{R}^n$, where observed values are vectors of reals. In this case, timed words are called (multivariate) *time series*, and are denoted $S = (t_0, \vec{x_0}) \cdots (t_k, \vec{x_k})$.

We define a *Qualitative Transition System* (QTS) in the following way.

Definition 20 *A* qualitative transition system $\mathcal{A}$ *is a tuple* $\mathcal{A} = \langle Q, E, \mu, \sigma, w \rangle$ *where* Q *is a finite set of set of qualitative* states*,* $E \subseteq Q \times Q$ *is a finite set of* transitions*,* $\mu, \sigma : E \to \mathbb{R}$ *are* mean *and* standard deviation *labelings, and* $w : E \to \mathbb{N}$ *is a* weight *labeling.*

For any transition $e \in E$, $\mu(e)$ and $\sigma(e)$ are respectively interpreted as being the mean and the standard deviation of a normal distribution that is followed by a random variable called *sojourn time*. The weight labeling w induces transition probabilities for transitions. Formally, the *transition probability* labeling $p : E \to [0, 1]$ induced by w is defined by

$$p(q, q') = \frac{w(q, q')}{\sum_{(q,q'')\in E} w(q, q'')}.$$

A QTS is thus a transition system where each transition is labeled with the amount of time the system needs before moving to another state. The delay between two state changes follows a parametrized normal distribution. This has to be contrasted with continuous Markov chains, where the sojourn time in a state must be exponentially distributed.

Suppose that a QTS is in the state q, and that there exists an outgoing transition e from q to q' (i.e., $e = (q, q')$). The probability of moving from state q to state q' is $p(e)$, the transition probability of e. Suppose that the transition e has been selected in favor of other outgoing transitions; then the system will stay in the state q for a delay that is normally distributed with mean $\mu(e)$ and with variance $\sigma(e)$. Let X be such a normally distributed random variable that denotes the sojourn time in the state q, and let F_X be its cumulative distribution function. The probability to move from the state q to the state q' between t_1 and t_2 time units is thus given by $F_X(t_2) - F_X(t_1)$. Contrary to the standard semantics of continuous time Markov chains (see e.g. [114] for a complete definition), our semantics does not involve a race condition. That is, in a given state, the probability for the successor state is not conditioned by the delays but solely by the transition weights, similarly to a discrete time Markov Chain.

4.3.2 Abstraction of a time series in terms of Qualitative Transition Systems

Abstraction of a time series in terms of timed words

By representing the characteristic qualitative features of a trajectory in an abstract domain that is countable, qualitative similarity can be detected by a simple equality test over integers or integer vectors. To this end, each concrete observation $(t, \vec{x})$ of a time series S is transformed into an abstract observation (t, a) where the observation time t is unchanged and a is an abstract value in a finite domain A called the *abstract domain*. The rationale behind abstraction is that two concrete observations that are transformed into the same abstract observation are assumed indistinguishable with respect to qualitative properties.

Formally, an *abstraction function* is any function $\alpha : \mathbb{R}^n \to A$ where A is a finite domain. For any time series $S = (t_0, \vec{x_0}) \cdots (t_k, \vec{x_k})$ in $(\mathbb{R}_{\geq 0} \times \mathbb{R}^n)^*$, the abstraction of S is the timed word $\alpha(S) = (t_0, \alpha(\vec{x_0})) \cdots (t_k, \alpha(\vec{x_k}))$ in $(\mathbb{R}_{\geq 0} \times A)^*$. Note that abstraction functions may be combined by cartesian product. In practice, as in the following example, it is often desirable to use multiple-arity abstraction functions that are defined on a fixed-width "window" of observations, that is, functions from $(\mathbb{R}^n)^{d+1}$ to A where $d \in \mathbb{N}$ is the width. For such a function α, the abstraction of S would be defined as the timed word $\alpha(S) = (t_d, \alpha(\vec{x_0}, \ldots, \vec{x_d})) \cdots (t_k, \alpha(\vec{x_{k-d}}, \ldots, \vec{x_k}))$. Observe that $\alpha(S) = \alpha(S')$ where $S' = (t_d, (\vec{x_0}, \ldots, \vec{x_d})) \cdots (t_k, (\vec{x_{k-d}}, \ldots, \vec{x_k}))$ is a time series over $\mathbb{R}^{nd}$. For simplicity, and without loss of generality, we only formalize our approach for unary abstraction functions (with zero width).

As examples of abstraction function, the *sign* function can abstract a time series into a timed word over the domain Plus, Zero, Negative. The *rank* function can abstract a timed word over the domain $\mathbb{N}$ by mapping to each component of a vector its index in the corresponding sorted vector. For example, $sort((11, -2, 1, 2)) = (-2, 1, 2, 11)$ therefore $rank((11, -2, 1, 2)) = (4, 1, 2, 3)$. In

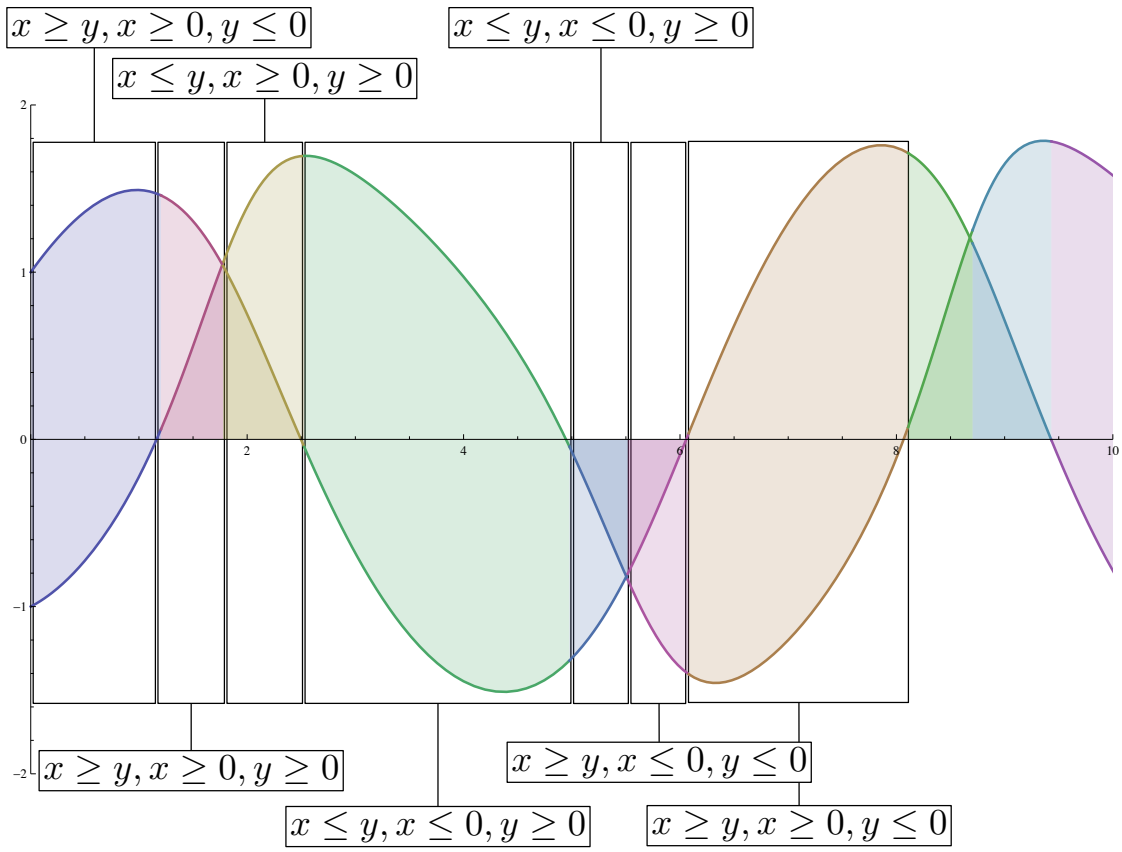


Figure 4.3: Decomposition of a trajectory of a two variables (x, y) oscillator. By considering the sign and rank of the variables, we map to each point of the trajectory an abstract value (here denoted by a formula). Boxes in the figure encompass successive points that are mapped to the same abstract value. Successive points of the trajectory are collapsed whenever they have the same abstract value, that is whenever they have the same sign and rank. This trajectory can thus be abstracted as a seven state QTS.

the same way, the sign of the first (resp. second) derivative can distinguish between increasing (resp. rapidly increasing) and decreasing (resp. rapidly decreasing) intervals. The evaluation of this abstraction function requires two points (resp. three points).

Since the abstract domain is finite, it is often the case that the abstract time series $\alpha(S)$ has successive observations that are equal. These repeated observations are removed by collapsing successive abstract observations having the same abstract value. Formally, for any timed word $W = (t_0, a_0) \cdots (t_k, a_k)$ over A, we write $collapse(W)$ the timed word $(t_{i_0}, a_{i_0}) \cdots (t_{i_h}, a_{i_h})$ where $i_0 < \cdots < i_h$ are such that $i_0 = 0$ and $a_{i_j} = a_{i_j+1} = \cdots = a_{i_{j+1}-1} \neq a_{i_{j+1}}$ for every $0 \leq j < h$. Observe that collapsing is idempotent: for any timed word W, it holds that $collapse(W) = collapse(collapse(W))$. The *reduced abstraction* of any time series S is then defined as the timed word $collapse(\alpha(S))$.

Abstraction of a timed word in terms of Qualitative Transition System

The abstraction of a time series in terms of timed words abstracts the value component of the time series. In order to adequately compare two timed words, we also need to abstract the time of observations. Consider a timed word $W = (t_0, a_0) \cdots (t_k, a_k)$ over an abstract domain A. To qualitatively abstract this timed word, it is represented as a transition system by considering that for any integer $0 \leq i < k$, the pair $((t_i, a_i), (t_{i+1}, a_{i+1}))$ of successive abstract observations of W is induced by a timed transition $a_i \rightarrow a_{i+1}$ between two *states* of a transition system with a delay of $t_{i+1} - t_i$ time units. We can then consider the set of all transitions between two given states. From such a set of transitions with identical source and target, we suppose that the delays are approximately normal, and thus estimate the mean and variance of the supposed underlying normal distribution. This way, the set of concrete transitions can be abstracted by a single stochastic transition in a qualitative transition system. Formally, a timed word is abstracted in terms of QTS with the following definition. For any finite subset $X \subseteq \mathbb{R}$, we denote by $\mathbf{E}[X]$ the *mean* of X and by $\mathbf{V}[X]$ its *variance*.

Definition 21 *The* QTS abstraction *of any timed word $W = (t_0, a_0) \cdots (t_k, a_k)$ over A is the qualitative transition system $\mathcal{A} = \langle Q, E, \mu, \sigma, w \rangle$ with*

$$\begin{array}{rclcrcl} Q & = & \{a_i \mid 0 \leq i \leq k\} & & \mu(q,q') & = & \mathbf{E}[\Delta(q,q')] \\ E & = & \{(a_i, a_{i+1}) \mid 0 \leq i < k \wedge a_i \neq a_{i+1}\} & & \sigma(q,q') & = & \mathbf{V}[\Delta(q,q')] \\ & & & & w(q,q') & = & |\Gamma(q,q')| \end{array}$$

where for any $(q, q') \in E$, $\Gamma(q, q')$ is the set of pairs (i, j) with $0 \leq i < j \leq k$ such that $a_i = q$, $a_j = q'$, and $a_{i-1} \neq a_i = a_{i+1} = \cdots = a_{j-1}$, and $\Delta(q, q') = \{t_j - t_i \mid (i, j) \in \Gamma(q, q')\}$.

Note that in the definition, the set $\Gamma(q, q')$ contains pairs of indices (i, j) such that all observations between i and j are removed by collapsing. Therefore, any two timed words W and W' over A satisfying $collapse(W) = collapse(W')$ have the same QTS abstraction.

Deterministic parametrized models, such as ODE systems, can exhibit different qualitative behaviors depending on the value of the parameters. When these systems admit a simulation algorithm (e.g. numerical integration), they generate time series. We show in [187] that, under assumptions concerning the simulation algorithms, properties of interest of a given system are preserved by the abstraction in terms of qualitative transition systems.

Periodic orbits detection Oscillations are ubiquitous qualitative behaviors found in systems admitting a feedback loop. Although bifurcation analysis provide numerical methods to establish the presence of periodic orbits for ODEs, these methods cannot be applied to a general deterministic system such as an ODE with events. However, we show in this section that a QTS can be used efficiently to estimate the likelihood of a periodic orbit in a time series.

Under an adequate abstraction function, a QTS that abstracts the transient behavior of a system with a periodic orbit has cycles in its transition relation. Consider a QTS obtained by applying the abstraction function α to an α-adequate time series S obtained by sampling a continuous function $f : t \rightarrow \mathbb{R}^n$. By definition, f admits an orbit if and only if there exists a time point t and a period π such that $f(t) = f(t+\pi)$. Furthermore, f admits a periodic and non constant orbit if and only if there exists an intermediate time step $t' < t+\pi$ such that $f(t') \neq f(t+\pi)$. Since S is adequately sampled for α, there exists at least three successive different values in $collapse(\alpha(S))$ and consequently the resulting QTS has at least a cycle of length 1.

Since equality is not coherent between the real numbers and their floating point approximation, detection of periodic orbits for a time series must rely on estimations. To find a periodic orbit in a time series S it is sufficient to find a period $\pi \in \mathbb{R}_{\geq 0}$ such that there exist two elements of

$(t_i, \vec{x_i}), (t_j, \vec{x_j}) \in S$ such that $(t_j, \vec{x_j}) \approx (t_i + \pi, \vec{x_i})$ for an adequate approximation relation $\approx$. However, for ODE systems integrated with an adaptive time step algorithm, this scheme produces mainly false positives (successive integration steps in a quasi steady region of an ODE) and false negatives (regions with high variability).

The existence of a periodic orbit of period π also implies that for any value $k \in \mathbb{N}, f(t) = f(t + k * \pi)$. Thus, if the system reaches a periodic orbit at point l, then the nearest points (according to an euclidean distance on $\mathbb{R}^n$) of l contain points from all possible periods.

Therefore, we estimate the likelihood of a periodic orbit by considering a point $l = (t_l, \vec{x_l})$ of S that we suppose being in the periodic orbit, and a set of sample points P from S such that for any point $p' = (t_{p'}, \vec{x_{p'}})$ in $S - P$, for any point $p = (t_p, \vec{x_p})$ in P, we have $\mid x_{p'} - x_l \mid > \mid x_p - x_l \mid$. Less formally, P is a set containing the points that are the nearest of l with regard to the euclidean distance. The likelihood $\mathcal{L}((t_l, x_l), \pi, P)$ of π being the period of the orbit of x_l given a sample P of neighbors of x_l is then defined by

$$\mathcal{L}((t_l, x_l), \pi, P) = \left(\sum_{\delta \in \Delta} ([\delta] - \delta)^2 \right)^{-1}$$

with $\Delta = \{(t_p - t_l)/\pi \mid t_p \in P\}$ and $[\delta]$ being the integer part of δ. In other words, we seek a period π as to minimize the distance between an integer sequence and the sequence of delays between neighbors of l.

Finding the period π that maximizes $\mathcal{L}$ is difficult in practice, since this function admits local maxima that are far from the global maximum. However, the sum of the mean of the longest cycle containing the last observation in a QTS provides a good initial guess of this period. (See the case study 4.3.4).

4.3.3 Accounting for noise by comparing critical points

Qualitative transition systems can capture the dynamics of a time series, even if the time series contains numerical errors that are only local. In the case of time series admitting global noise, abstraction functions that were adequate for a smooth time series may not be resistant to noise and can generate a QTS that inadequately captures the dynamics of noise. For example, abstracting with the sign of the first derivative can adequately detect oscillations[159] but fails for time series even with little noise. Although a moving average can smooth a time series and seem to circumvent this problem, the size of the window must be fixed *a priori* and this approach is thus neither general nor adaptive.

We propose here an adaptive approach to capture the most important points w.r.t. the shape of a time series. The *critical points* of continuous function $f : \mathbb{R} \to \mathbb{R}$ are the set of points where $f'(x) = 0$. These are points where the function f either has a peak and changes direction (local or global extremum) or presents a curvature change (inflection points). In both cases, the shape of f changes around the point. We generalize this definition to time series in the following way.

Definition 22 *The* critical point *of a time series* $S = (t_0, \vec{x_0}) \cdots (t_k, \vec{x_k})$ *is the point* $(t_c, \vec{x_c}) \in S$ *maximizing the function* $\Lambda(t_c, \vec{x_c}) = |\vec{x_c} - \vec{x_0}| + (t_c - t_0) * (\vec{x_k} - \vec{x_0})/(t_k - t_0)$.

The critical point of a time series is the point of maximal distance with the linear interpolation between the first and last points of the series. In a numerical context, this point is uniquely defined.

A critical point splits the time series into two time series. Since a critical point is also defined for these series, we can recursively approximate a time series by considering a piecewise function which is linear between critical points.

Definition 23 *The* piecewise linear approximation of order i *(hereafter PLA) of a time series is the piecewise linear function on the intervals $I_0, ..., I_k$ where any interval I_j has a lower bound (resp. upper bound) corresponding to the location of the j^{th} (resp. $j+1$) critical point.*

In order to compute the PLA of a time series, we define the piecewise linear interpolations of a set of points as the union of the linear interpolation between two successive points. The computation of the PLA of order i is then performed as follows.

PLA (S, i) returns a list of critical points of $S = (t_0, \vec{x_1}) \cdots (t_k, \vec{x_k})$

1. For each dimension of $dim(\vec{x})$
   (a) Initialize the critical points with the first and the last point of S projected on the current dimension
   (b) While the number of critical points is less than i
      i. Build a piecewise linear interpolation between each pair of successive critical points
      ii. Append to critical points the unique point of the time series maximizing the distance with the piecewise linear interpolation

The previous algorithm cannot append a point twice to the list of critical points. Indeed, once a point is appended to the list it becomes a bound for the piecewise linear interpolation that is used for determining the next critical point. Consequently, at this point the distance between the next piecewise linear interpolation and the time series is 0, and the distance can not be maximized. Note this does not hold for the piecewise linear regression. Which implies that, for any unidimensional time series, the segmented linear regression of i intervals minimizing the residuals with the time series can be obtained by considering the critical points as bounds of the interval.

Examples of critical points Critical points are highly related to the shape of the time series. Consider for example the sigmoid shape: a simple shape descriptor may simply specify that we start from a low plateau, follow an almost vertical increase before reaching another high plateau. Such a sigmoid shape exhibits two critical points, one at the end of the first (low) plateau, and one at the start of the second (high) plateau. Similarly, consider an oscillatory time series such as the one depicted in figure 4.4: its critical points will contains the successive highest local maximum and the lowest local minimum.

4.3.4 Case Studies and experimental results

In this section we show how our approach cab be used in practice by solving four problems related to qualitative behavior analysis. Although each problem and solution is illustrated on a specific model, the methods that are used are general purpose. All the models used in this section were downloaded from the BioModels database[145] in the SBML V2 L1 format[93], simulated using MathSBML[173] and were used without any modification. Simulations were performed on an Intel Core2 3,2GH personal computer and each algorithm was allowed to run for at most five minute. In the case where parameter values are not specified in the case studies, we used the parameter values provided in the SBML file. For each of the simulation results, we compared the

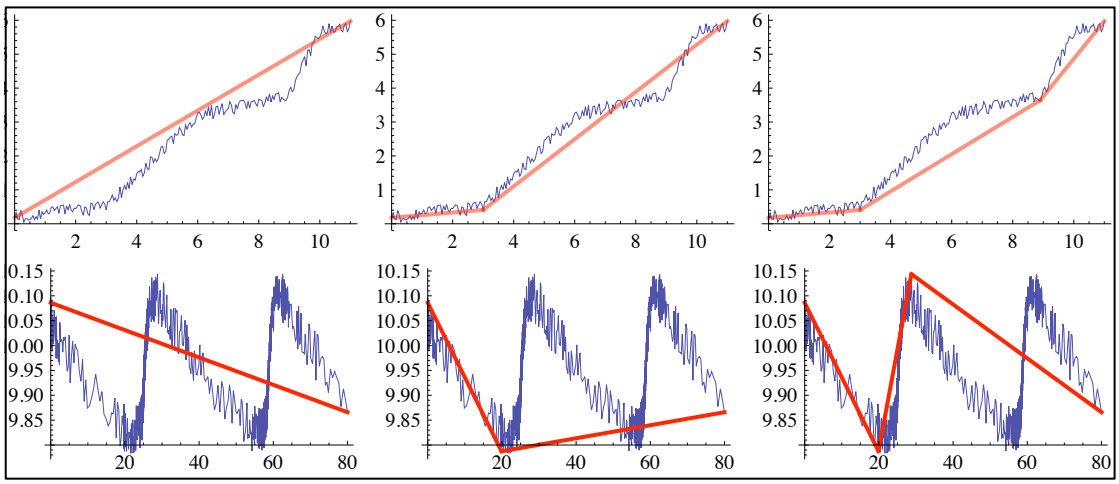


Figure 4.4: Example of piecewise linear approximation applied to two randomly generated noisy time series. The first of the three successive plots represent the time series while the last two plot represent the result of the first two iterations of the PLA algorithm. With two iterations, the PLA algorithm choose two points (and the first and last point of the time series) that are considered as being representative of the global shape of the time series.

Searching a trajectory with a given periodic orbit

The first model we consider is a model of the cell cycle based on the interactions between the cyclin dependent kinase cdc2 and cyclin [200]. The model is comprised of six variables and ten parameters. We consider the following problem: Given the representative trajectory and its associated parameter described in the original article (left in figure 4.5), what kind of similar trajectory can we find in the parameter space ?

Abstracting the behavior of the left figure with a rank abstraction function yields the QTS depicted in the right part of figure 4.5. Notice the non deterministic states zoomed by a circle. These states and transitions are due to numerical errors and happen while the system reaches its periodic orbit. Consequently, the weights of the outgoing highlighted transitions are 1 while the incoming transitions are 17. All other transitions in the single cycle of the QTS have a weight of 18.

We obtained 500 random samples for the six parameters considered as being critical by the original author. For each parameter sample, we computed the trajectory, abstracted it in terms of QTS by applying the rank function and computed the Sorensen similarity index over the set of transitions to compare the sampled QTS with the representative QTS. Figure 4.6 depicts a subset of the results. Note that we chose parameter values exhibiting "similar" sustained oscillations, but of *different transient behaviors*. We then compared these results with the one obtained with a stochastic simulation algorithm. For each simulation result, we used the PLA algorithm to reduced each noisy trajectory to its 50 most critical points, and abstracted these points in terms of QTS by applying the rank function. Trajectory similar to the one simulated with numerical integration were found for comparable parameters values.

Estimating the period of orbits

The model of MAPK cascade from [106] describes the effect of negative feedback and ultra-sensitivity in the emergence of oscillations. We thus investigated the dynamic of the period of the orbits under

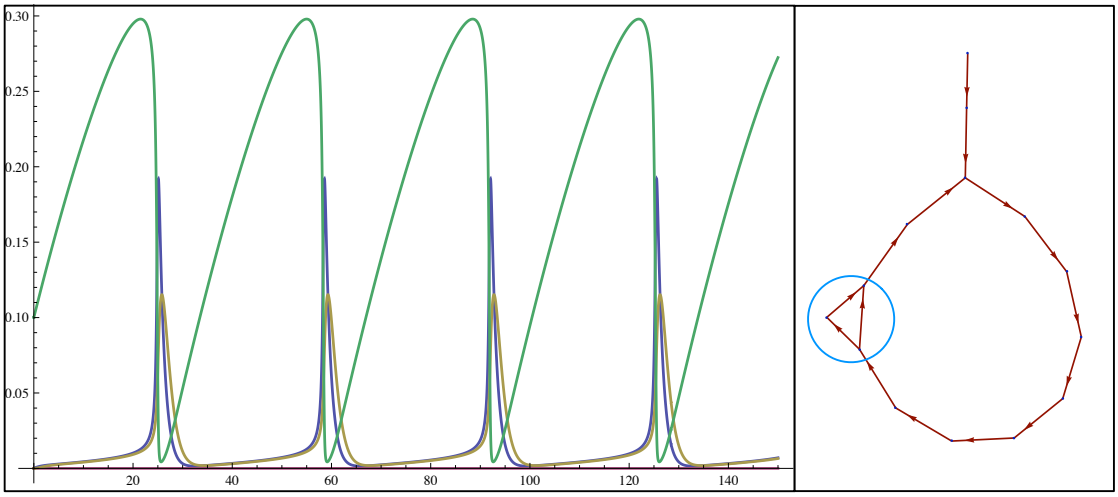


Figure 4.5: Dynamic behavior of the cell cycle model for default parameter values. **Left**: An example of trajectory obtained by numerical integration of the Tyson cell cycle model [200]. **Right**: Abstraction of this trajectory in terms of QTS by using the rank function as the abstraction function. Transition labels are omitted. The total variance of this QTS is 0.07. The states and transitions highlighted under the circle correspond to stochastic transitions and represent numerical integration errors.

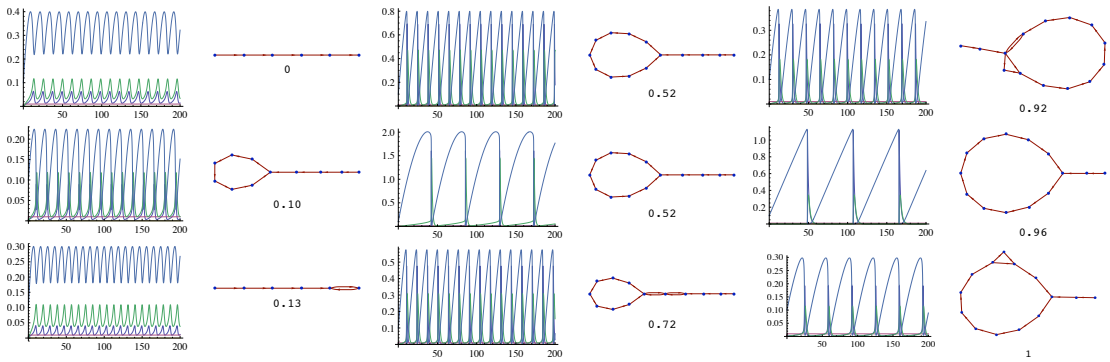


Figure 4.6: Result of trajectory comparison on the Tyson cell cycle model. For 500 randomly sampled parameter spaces, we abstracted the simulated trajectory in terms of QTS by using the rank function. Trajectory were clustered in 9 bins by measuring the Sorensen similarity index between each of the 500 QTS with the QTS of the figure 4.5. From each of the nine clusters, a representative trajectory was chosen and represented here, alongside of its QTS and similarity value. These trajectory are sorted (column wise, increasing) by their similarity value.

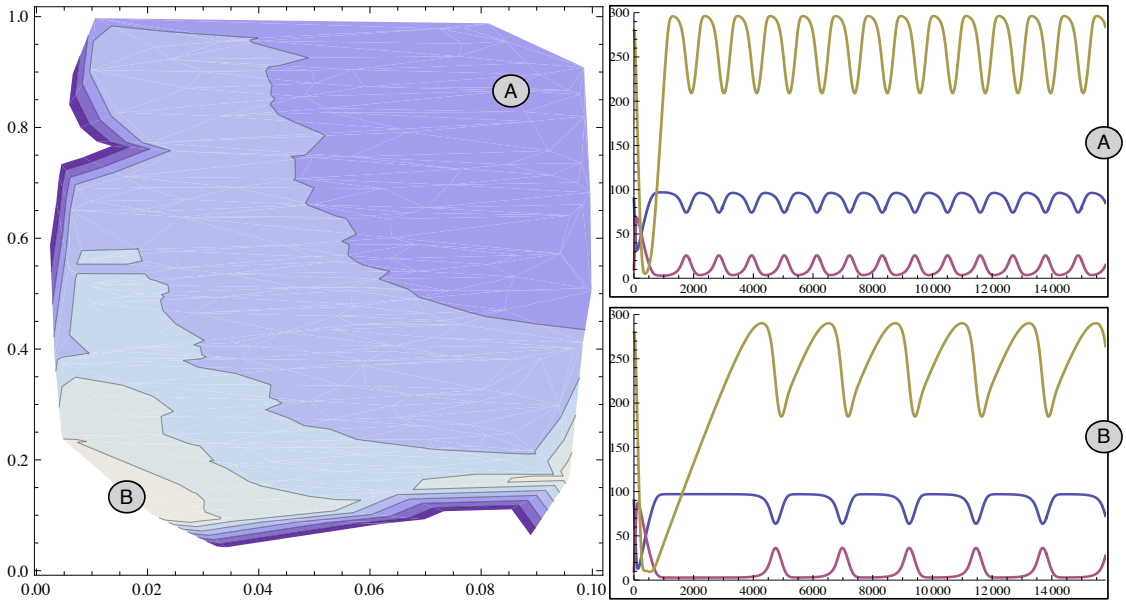


Figure 4.7: Oscillation period for the MAPK cascade model. **Left**: Contour plot representing the period of oscillations. The bottom axis (resp. left axis) represents values of the $k4$ (resp. $v5$) parameter. The contour plot was built with 500 simulations with random parameters. Regions with comparable periods are represented by a uniform region. **Right**: Two example trajectories exhibiting oscillations of extreme period A:1094 time units and B:2236 time units.

parameter changes. To estimate these periods, we considered the parameters $\{k_4, v_5\}$ as random variables following an uniform distribution over the intervals $[0, 1]$ and $[0, 0.1]$. For 500 parameters sample, we abstracted the corresponding time series in terms of QTS. These QTS were then reduced by removing transitions whose probability decreased as the simulation advanced. We then approximated the orbit's period with the sum of means of the transitions of the longest cycle of the QTS. This approximation was then fed in a local maximization procedure to identify the exact period value maximizing the likelihood function. In our tests, providing this initial "educated guess" of the period value to the maximization procedure yielded the global maximum in 98% of cases.

We can see from the results (figure 4.7) that, for this parameter subspace, oscillating behavior is very common and that the dynamics of the period does not exhibit abrupt changes.

Searching for any periodic orbits

We consider again the MAPK cascade model but with a more general objective. We consider the problem of detecting the possible oscillating behaviors and of computing the probability of finding an oscillating behavior in a larger parameter subspace. The parameters of interest are $\{k_3, k_4, k_7, k_8, V_5, V_6, V_9, V_{10}\}$ and are considered as random variables following an uniform probability over the interval $[0, 1] \subset \mathbb{R}$. We built a QTS as in previous sections. In this study, only periods in the range $[200, 5000]$ with a likelihood greater than 10 were considered genuine. Although all the resulting trajectories exhibit transient or limit cycle oscillations, they follow different transient dynamics. The four example trajectories of figure 4.8 show a subset of the possible dynamics: each of these time series admits a specific alternation of species at their maximum concentration. Multiple instances of each of these dynamics were successfully identified by applying the method from section 4.3.4. The

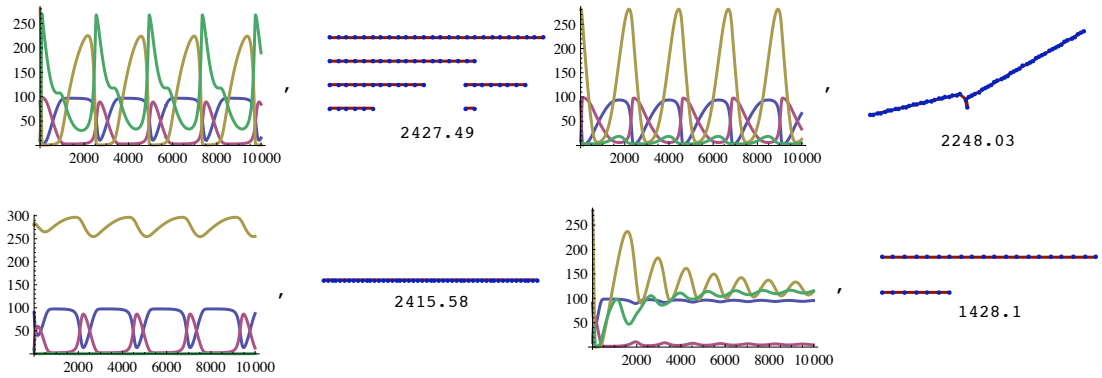


Figure 4.8: Example trajectories of the MAPK cascade exhibiting oscillating behavior found with a random sampling of ten parameters of the MAPK cascade model. On the right of each plot, the associated QTS reduced to its periodic form; under it, the period value with maximum likelihood.

number of samples needed before finding an oscillating behavior was 57 on average. For comparison, when k_3, k_4, k_7 and k_8 were sampled in the interval $[0, 0.1]$, the average number of samples needed dropped to 10.6.

Searching for given transient behavior in a parameter subspace

The extracellular signal regulated kinase (ERK) pathways plays a role in a hidden oncogenic positive feedback loop via a crosstalk with the Wnt pathway [108]. The pathological cases identified by the authors involve “an irreversible response leading to a sustained activation of both pathways". Applying our QTS construction with random samples of the β-catenin synthetic rate (V_{12}) yields results depicted at figure 4.9.

This model involves 28 species, 58 parameters, and 2 discrete events. Applying the rank abstraction yields transition systems with a state space of 600 states on average out of the possible 28! state configurations.

4.3.5 Discussion and conclusions

In this work, we have described the formalism of qualitative transition systems. A QTS is a transition system where each transition is labeled with the amount of time the system needs before moving to another state. The delay between two state changes follows a parametrized normal distribution. The probabilistic timed semantics of QTS has been defined by using a cylinder set construction. We have shown how QTS can be used to study qualitative properties of parametrized models. This is achieved by defining an appropriate abstraction function. By representing the characteristic qualitative features of a trajectory in an abstract domain that is countable, qualitative similarity can be detected by a simple equality test.

We have shown that the soundness of this approach depends on the adequacy of sampling with respect to the abstraction function. In particular, we have shown that for convex abstraction functions, if the sampling is “precise enough”, then the QTS obtained from any oversampling has the same transitions.

Finally, we applied this approach to some well known models. QTS were used to explore the parameter space and to detect uniform behaviors (oscillations etc.).

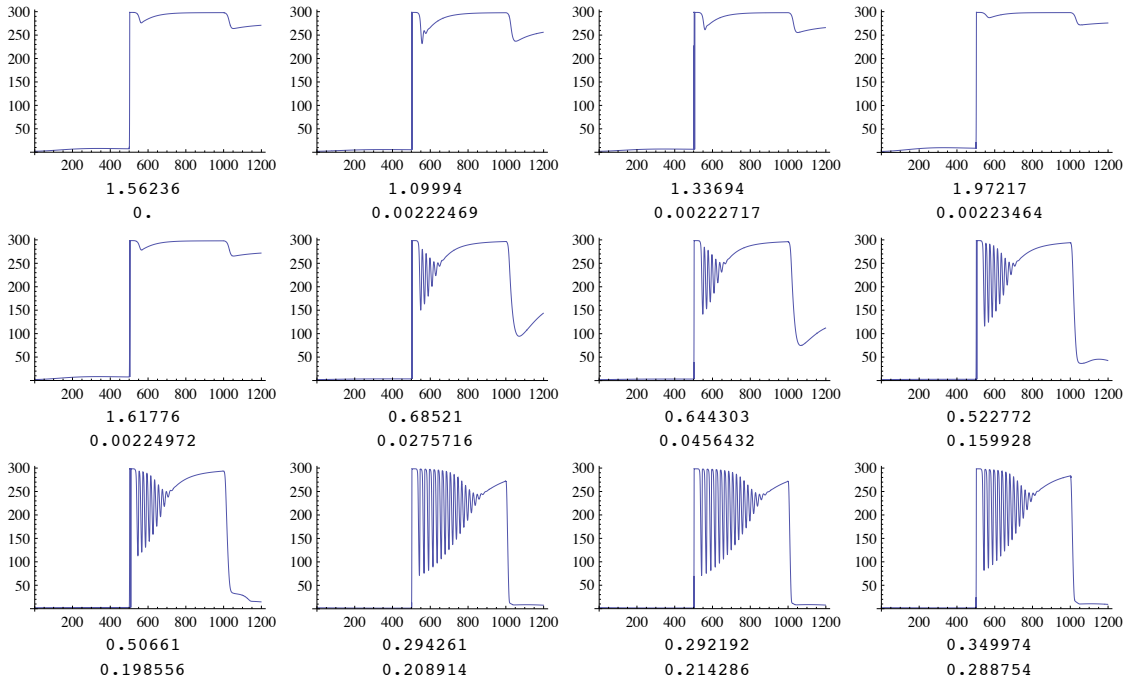


Figure 4.9: ERK Crosstalk simulation results. From top left to bottom right, simulation results are sorted by similarity with the non pathological case(reversible activation). The sampled value of v_{12} and the similarity index with the non pathological case are under each plot.

The limits of our approach as compared to model checking is the lack of exhaustivity. This has to be counterbalanced by the fact that our method is applicable to a large panel of formalisms, even those lacking a precise semantics. Consequently, we can avoid any model transformation. Finally, our approach can be applied independently to the data and to the model.

The areas of future research for qualitative transition systems can be declined on the both technical and practical plans. As for the former, we envision a more thorough study of similarity measures of QTS and how QTS similarity relates to language equivalence. As for the latter, we plan to develop clustering techniques in order to detect the resulting behavior similarity in an experimental context.

4.4 Exploratory simulation of cell ageing

Thorough knowledge of the model organism *S. cerevisiae* has fueled efforts in developing theories of cell ageing since the 1950s. Models of these theories aim to provide insight into the general biological processes of ageing, as well as to have predictive power for guiding experimental studies such as cell rejuvenation. Current efforts in *in silico* modeling are frustrated by the lack of efficient simulation tools that admit precise mathematical models at both cell and population levels simultaneously. We developed a novel hierarchical simulation tool that allows dynamic creation of entities while rigorously preserving the mathematical semantics of the model. We used it to expand a single-cell model of protein damage segregation to a cell population model that explicitly tracks mother-daughter relations. Large-scale exploration of the resulting tree of simulations established that daughters of older mothers show a rejuvenation effect, consistent with experimental results. The combination of a single-cell model and a simulation platform permitting parallel composition and dynamic node creation has proved to be an efficient tool for in silico exploration of cell behavior.

A recurring challenge for *in silico* modeling of cell behavior is that hand-tuned, accurate models tend to be so focused in scope that it is difficult to repurpose them. *Hierarchical modeling* [4] is one way of combining specific models into networks. Effective use of hierarchical models requires both formal definitions of the semantics of such compositions, and efficient simulation tools for exploring the large space of complex behaviors. In this study, we propose the use of a hierarchical model to reduce the complexity of analysing cell ageing phenomena such as cell rejuvenation. To this end, we extend a single-cell model of inheritance of protein damage to a structured population where mother-daughter relations are tracked. This requires definition and implementation of an exploratory simulation software system. Using this system we validate the model, discover a cell rejuvenation effect consistent with the experimental literature, and derive testable hypotheses on cell ageing.

Unlike most microorganisms or cell types, the yeast *Saccharomyces cerevisiae* undergoes asymmetrical cytokinesis, resulting in a large mother cell and a smaller daughter cell. The mother cells are characterized by a limited replicative potential accompanied by a progressive decline in functional capacities, including an increased generation time [182]. Accumulation of oxidized proteins, a hallmark of ageing, has been shown to occur also during mother cell-specific ageing, starting during the first G1 phase of newborn cells [1]. Both asymmetric and symmetric division exist in different yeast species. In particular, *S. cerevisiae* is known to divide asymmetrically, although symmetrical division is observed in about 30% of cells at the end of their replicative lifespan [104]. Another yeast model organism, *Schizosaccharomyces pombe*, divides symmetrically by fission (see [146] for review). The following is a mathematical model we have developed to explain how the accumulation of damaged proteins influences fitness and ageing in yeast. In this paper we consider the two theoretically possible scenarios, namely asymmetrically and symmetrically dividing cells in different damaging environments. To explore any and all branches of the pedigree tree of a cell population, we will use a hierarchical model that allows us to track mother-daughter relations. We can therefore

explore lineage-specific properties, such as the rejuvenation property.

4.4.1 From single cell to population model

Single cell model A minimal single-cell model of inheritance of damaged proteins can be formalized by the following three equations:

$$\frac{dP_{\text{int}}}{dt} = \frac{k_1}{k_s + P_{\text{int}} + P_{\text{dam}}} - k_2 P_{\text{int}} - k_3 P_{\text{int}} \tag{4.1}$$

$$\frac{dP_{\text{dam}}}{dt} = k_3 P_{\text{int}} - k_4 P_{\text{dam}} \tag{4.2}$$

$$\frac{dP}{dt} = \frac{k_1}{k_s + P_{\text{int}} + P_{\text{dam}}} - k_2 P_{\text{int}} - k_4 P_{\text{dam}} \tag{4.3}$$

The size of the cell is the sum (P) of intact (P_{int}) and damaged (P_{dam}) proteins, $P = P_{\text{int}} + P_{\text{dam}}$. Protein temporal dynamics are determined by five rate constants k_1, k_2, k_3, k_4, and k_s. Protein production rate, k_1, has been adjusted by hand allowing for a steady state to be reached and has been assigned a final value of 10^7. We choose values of k_2 and k_4, the degradation rates of P_{int} and P_{dam}, respectively, so that $k_2 < k_4$. Degradation rates are computed using the half-life formula $t_{1/2} = \ln 2/k$, where k is the degradation rate; setting the half-life of intact proteins to be 1 time unit, $k_2 = \ln 2$. Since degradation of damaged proteins is faster, k_4 needs to be greater than k_2 and it has been set to $\ln 5$. To simulate different rates of conversion, k_3 has been given a range of values, from 0.1 to 2.3. Finally, k_s is a half-saturation constant in the model, not used in this study. We assume that cells grow until they have attained a critical cell size, P_{div}, which triggers a cell division. A cell may divide symmetrically (halving its mass) or asymmetrically, as defined by size coefficients s_{mother} and $s_{daughter}$. These different types of divisions are modeled by varying the two coefficients $s_{mother}$and $s_{daughter}$. In the case of symmetrical division, the size of both progeny and progenitor is equal, so $s_{mother} = s_{daughter} = 0.5$. In asymmetrical division, cells in the next generation will have different sizes such as when $s_{mother} = 0.75$ and $s_{daughter} = 0.25$. In the following study, rate constants k_1, k_2, and k_4 received fixed values, k_3 was given a range of values with step size 0.1, and s_{mother} and $s_{daughter}$ were given two pairs of values representing symmetric and asymmetric growth strategies, namely $\langle s_{mother}, s_{daughter}\rangle$ being $\langle 0.5, 0.5\rangle$ or $\langle 0.75, 0.25\rangle$.

The proteins distribution between the progenitor and the progeny after division is described by the following set of transition assignments:

$$P_{\text{int}} := P_{\text{int}} \cdot s_P \tag{4.4}$$

$$P_{\text{dam}} := P_{\text{dam}} \cdot s_P \tag{4.5}$$

$$P := P_{\text{int}} \cdot s_P + P_{\text{dam}} \cdot s_P, \tag{4.6}$$

where s_P is s_{mother} for the progenitor and $s_{daughter}$ for the progeny.

Population Model Based on this single-cell model, we first define a hierarchical model of a structured population where complete mother-daughter relations are recorded, using the BioRica formalism. BioRica [79] is a high-level modeling framework integrating discrete and continuous multi-scale dynamics within the same semantic domain. It is in this precise sense of mixing different dynamics that BioRica models are *hybrid* following the classical definitions [4]. Moreover, BioRica models are built hierarchically. In [4] two types of hierarchy are defined: *architectural* and *behavioral*. While BioRica admits both, in this paper we are only concerned with the former. This type of hierarchy allows for both concurrency and parallel composition.

In this work each cell is encoded by a BioRica node that has a 2-level hierarchy: a discrete controller and a continuous system. The former determines the distribution of proteins at division time

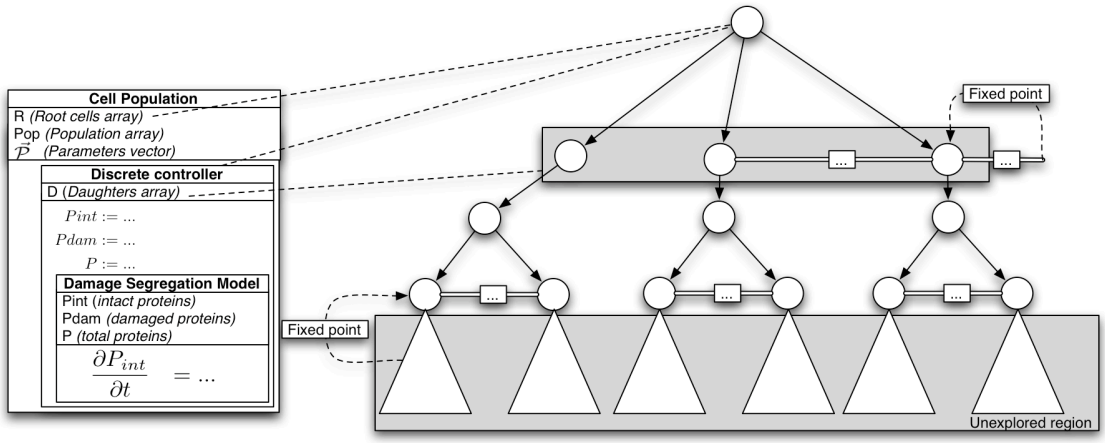


Figure 4.10: Three level hierarchical model, showing the discrete cell population and cell division controllers, and the continuous single-cell model. This model generates pedigree trees during simulation, instantiating new single-cell models for each cell division. Infinite width and depth are represented finitely by relaxing the tree constraints to permits loops from the leaves. These *fixed points* represent immortal cells or immortal lineages.

using the discrete transition assignments (4–6), while the latter determines the evolution of protein quantities during one cell cycle and is realized by the equations (1–3). More precisely, the discrete controller is encoded by a *constraint automaton* [64] defining the discrete transitions between states. A *state* of a cell c^i is a tuple $\langle P^i_{\text{int}}, P^i_{\text{dam}}, D^i \rangle$, where P^i_{int} and P^i_{dam} are protein quantities as before, and D^i is a single dimension array of integers representing the identifiers of every daughter of c^i. A *transition* between states is a tuple $\langle G, e, A \rangle$, where G is a guard, e is an event, and A is a parallel assignment. For this model, the discrete transitions are atomic operations and consequently take zero time. In our case we have: for mitosis (event e), if the threshold of the cell size is attained $P_{\text{int}} = 1500$ (guard G), then create a new BioRica node c^j for the daughter of the current cell c^i, append c^j to the vector D^i, and perform the assignments (4–6) (assignments A of state variables). A second discrete event representing clonal senescence is triggered whenever protein production reaches zero, that is $\partial P_{\text{int}} < 0$.

The cell population is encoded by a BioRica node using the mechanism of parallel composition. This node contains the population array Pop, the root of the lineage tree R and the parameter vector $\vec{\mathcal{P}}$. Since our model focuses on the division strategy, we consider the growth medium as a non limiting factor, and consequently we do not account for cell to cell interactions. This absence of interaction is directly modeled by parallel composition of independently evolving cell nodes. For illustration see figure 4.10. The algorithmic challenges related to dealing with multiple time scales and event detection, and our solutions, are described in section 4.4.2.

4.4.2 Algorithm

We now describe our method for efficient simulation of the cell population model (section 4.4.1), starting with an overview of the general simulation schema (algorithm 6) followed by a concrete specialization for damage segregation. The simulation schema for a given BioRica node is given by a hybrid algorithm that deals with continuous time and allows for discrete events that **roll back** (see figure 4.11) the time according to these discrete interruptions. Time advances optimally either

by the maximal stepsize defined by an adaptive integration algorithm [155], or by discrete jumps defined by the minimal delay necessary for firing a discrete event. As shown in algorithm 6, the simulation advances in a loop that is interrupted when either the simulation time expires, or the *alive* flag indicates that this node has died in the current or previous state. The node evolves continuously by calling *advance_numerical_integration*, after which we check whether any guard G of some event $\langle G, e, A\rangle$ was satisfied. In which case a number of updates is performed: the time is set to the firing time of e, e is stored in the trace database, the current state S is set according to the algebraic equation A, and the numerical integrator is reset to take into the account the discontinuity.

Algorithm 6 General simulation schema

Require: current state S, current simulation time t, maximal simulation time $t_{\max}$

$S' = S$
while alive(S, S') = 1 and $t < t_{\max}$ **do**
  $S' = S$
  $t, S =$ advance_numerical_integration()
  if e = discrete_events() **then**
    $t =$ get_discrete_event_time()
    store_event(e)
    $S =$ update(S, e)
    reset_numerical_integrator()
  end if
  store_state(S)
end while

As illustrated in figure 4.11, the step size proposed by the numerical integrator guarantees that the continuous function is linear between the current time t and the maximal step size. In this way the location of discrete events whose guards have been satisfied in this interval is reduced to computing the **first intersection** (see figure 4.11). It is the event e with the smallest firing time that is retained for the next discrete transition. After this transition the numerical integrator must restart from the point defined by A.

Correction of the stepping algorithm For asynchronous simulation of multi-agent hybrid systems, the correctness of a stepper algorithm mainly concerns numerical stability and event detection [60], both being in general very difficult problems [26]. For this specific model, numerical stability of the stepper described in algorithm 6 has been checked by evaluating the stiffness of the single cell ODE system by comparing the accumulated integration error between various order explicit and implicit methods. For the final implementation, the embedded Runge-Kutta-Fehlberg was retained, giving good tradeoff between efficiency and precision.

Failure of an event detection can be caused either when sub systems are coupled but not correctly synchronized or when a guard is falsely assumed as monotonic between two successive simulation steps. For the former, since the clonal senescence and the mitosis events only refer to the state and derivative of the currently integrated cell, the composed population model remains uncoupled and we do not need to synchronize any of the states. For the latter, failure of an event detection can be caused when a guard is falsely assumed as monotonic between two successive simulation step. In BioRica, since the guard of an event can not refer to the current simulation time, detecting the occurrence of an event is reduced to an intersection test between an $n + 1$ dimensional segment and a n dimensional region or polytope, where n is the number of state variables of the integrated node. More specifically for the guards of the cell population model, only intersection between the interpolated segment between two successive integration step and a downward closed region is used,

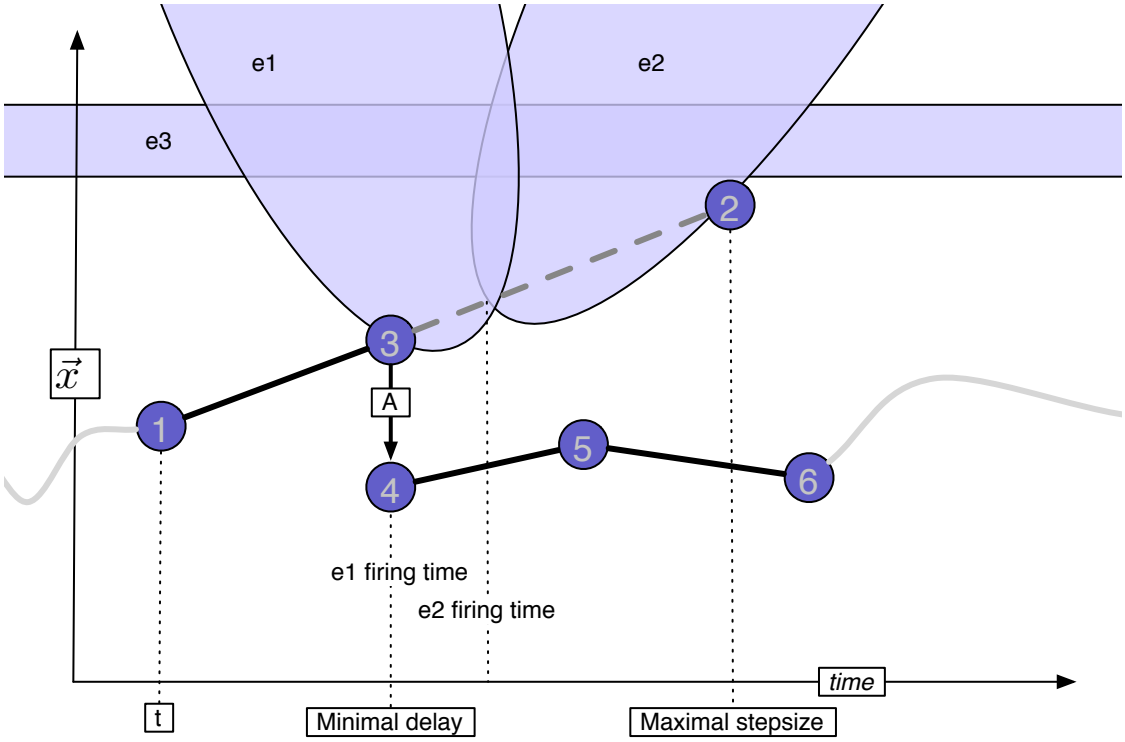


Figure 4.11: The numerical integrator advances between t (point 1) and the maximal stepsize (2). The guards of events e_1, e_2 are satisfied. The regions where these guards are satisfied are shaded. The firing time of e_1 (3) is used to reset the simulator after the discrete transition A (4).

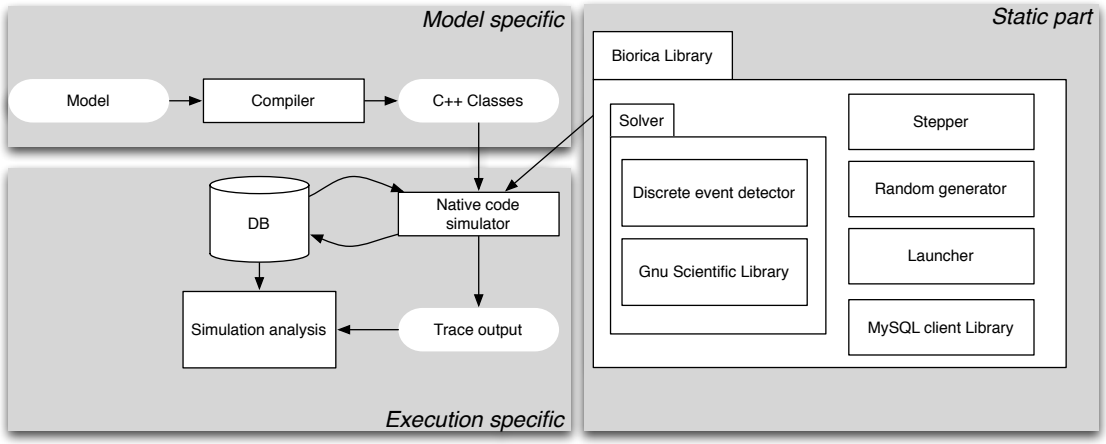


Figure 4.12: BioRica software architecture.

and is performed by evaluating the guard at the end point.

Specialization The generic simulation algorithm 6 was specialized for the damage segregation study. In particular, *alive* and *update* had to be redefined in a specific way. The *alive* predicate verifies three conditions. First, the cell is checked for immortality, which is realized by fixed point detection. Second, we verify whether the cell is in the state of clonal senescence, by evaluating the two guards described in section 4.4.1. *Update* has the role of managing new cell creation. For the current cell c it updates its state variables, according to the algebraic equations (4-6 for progenitors), and its statistics (fitness, generation time, etc). It creates a new cell node (daughter of c) according to the equations (4-6 for progeny) and inserts it into the population array Pop.

Population simulation On top of this specific stepping algorithm, another algorithm drives the whole population simulation by selectively starting simulations for pending cells in Pop. Given a depth n, a root cell c and an extent value e, this algorithm first selects pending nodes required to get a complete binary pedigree tree of depth n rooted at cell c. Afterwards, e leftmost and e rightmost leaves are used as root cells in recursive calls of this algorithm with a decremented value of e. Fix points are detected by testing before simulation if a candidate cell's initial values P_{int} and P_{dam} are equal to a previously simulated cell, in which case we get a pedigree graph by adding a loop edge.

Determining parameter values that exhibit optimal population fitness is based on averaging individual cell statistics (defined in section 4.4.3) to compute the mean fitness. In fact, this averaging maps a real value denoting the population fitness to each parameter vector. A coarse computation of this mapping is then built by varying the parameter vector using fixed step size. This coarse estimation is used to determine initial guess of optima position, that are then established by using Brent Principal Axis method on the mapping.

4.4.3 Results

Initial calibration To calibrate and validate the system, complete simulations were run to depth 4 in the pedigree tree for a large range of parameter values. Rate constants k_1, k_2, and k_4 received fixed values, k_3 was given a range of values with step size 0.1, and s_{mother} and $s_{daughter}$ were given two pairs of values representing symmetric and asymmetric growth strategies. A total of 625 simulations were run, summing to 9375 different initial conditions and parameters values. Sample results for pedigree tree are illustrated on figure 4.13. Successful comparisons with a small number

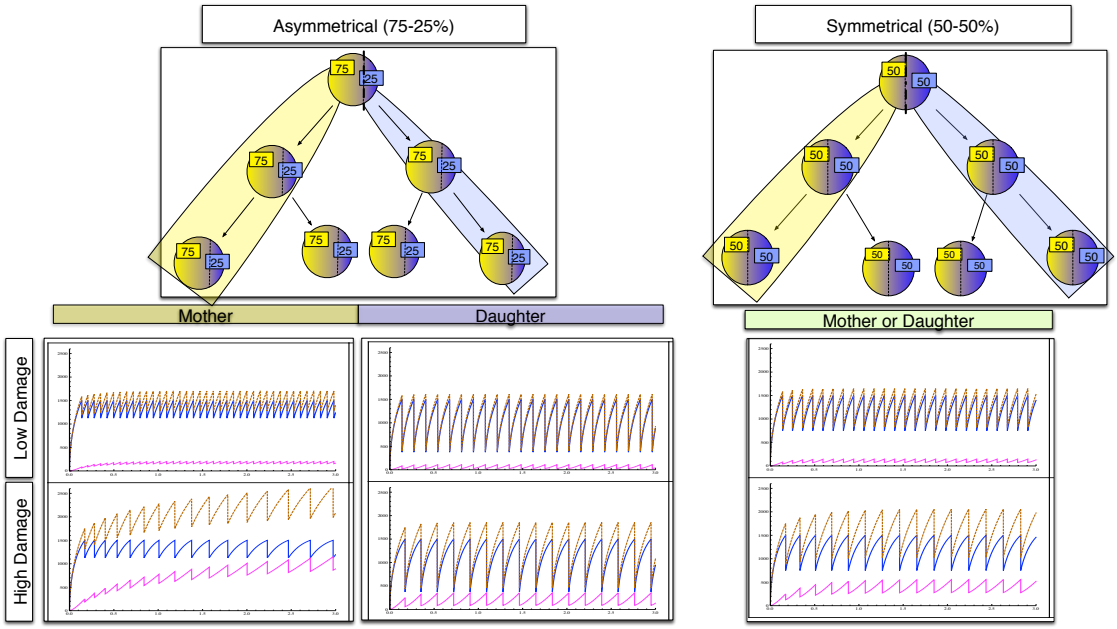


Figure 4.13: Sample pedigree tree results for asymmetrical (left) and symmetrical (right) division strategies. Pedigree tree (Top) showing mother-daughter relations; and simulation results (Bottom) showing single cell protein amounts over time: normal proteins (blue), damaged proteins (pink), total proteins (dashed). For example, in the asymmetrical case with high damage (lowest left plot), from time zero, the amount of normal proteins (blue) in the mother eventually cross the division threshold at time approx. 0.15. At this time, the proteins repartition is approx. 250 damaged proteins for 1750 total proteins. Once division is triggered, the progeny separates, and a new simulation is started for the mother (resp. the daughter) with initial normal proteins set at $1500 \times 0.75 = 1125$ (resp. $1500 \times 0.25 = 375$) and damaged proteins set at $250 \times 0.75 = 187.5$ (resp. $250 \times 0.25 = 62.5$). Since in this case the mother accumulates damage over divisions (compare damaged proteins amount between time 0.15 and 3.0), it will eventually reach a senescence point after 27 divisions. Comparison of its life span with the life span of each of its daughter and the life span of each daughter of its daughters (as tracked by the upper pedigree tree) shows a rejuvenation effect.

of experimental cell growth results were also performed (data not shown).

Parameter Exploration. Using parameter exploration (section 4.4.2) we identified sets of parameters that exhibited a given emerging high-level behaviour, both at the single-cell and whole pedigree tree levels. For example, for the former we are interested in detecting cells that have a certain number of daughters (say, 24), and for the latter we are looking for parameters giving high rejuvenation value across the whole population. These two values are computed by a trace simulation analysis script.

Thus, for each of scenarios studied here, a representative simulation was chosen by inspecting properties of the initial mother. From the whole parameter space, we selected simulations where the mother cell produces a number of daughters that is both finite and large enough (20-24 divisions depending on the case, since the average life span of wild type budding yeast is 24 divisions). For each of these simulations, the pedigree tree was calculated up to depth 30, and for each cell in the tree we calculated five values: *initial damage* and *terminal damage* levels (corresponding respectively to the amounts of damage P_{dam}at the beginning of cell cycle, and at the end of the cycle when division is about to occur), *generation time* (time between two divisions), *absolute date of birth* (in arbitrary time units, measured from the moment when mother starts its first division) and the *fitness* (defined as number of divisions during first time unit).

Model analysis. The hierarchical model we have defined explicitly tracks mother-daughter relations in pedigree trees of simulations. This allows us to study lineage-specific properties, which are properties associated with connected subgraphs of the pedigree tree. Pedigree trees and typical simulation results are shown in Figure 4.13.

In the pedigree tree, a given mother cell generates a series of daughter cells; these siblings are ordered in time, and the younger a sibling, the older the mother at the time of division. We observe in simulation results that younger siblings have higher damage, consistent with inheritance from an older mother that has accumulated more damaged proteins, and these younger siblings are thus born "prematurely old." This increase in damage accumulation is reflected in the decrease of fitness values, shown in the first level of Figure 4.14.

Extending this analysis one level further in the pedigree tree shows, expectedly, that daughters born early to the same mother have low damage, and their daughters have normal fitness. Daughters born late to the same mother have high damage and lower fitness, but remarkably, in simulations with asymmetric division, their own daughters are born with lower damage and higher fitness. This increase in fitness in the second generation is a *rejuvenation effect*, in part explaining how populations maintain viability over time despite inheritance of protein damage.

The testable hypothesis is thus that there exists a mechanism for segregation of damaged proteins during cell division, that attenuates the accumulation of such proteins in descendants, and that the asymmetry coefficients (s_{mother} and $s_{daughter}$) in the model determines the scale of the rejuvenation effect.

These predictions are consistent with *in vivo* experimental results reported in the literature: Kennedy, et al. [104] report that daughter cells of an old mother cell are born prematurely old, with lower replicative potential, but that the daughters of these daughters have normal life spans.

However, this rejuvenation effect is not present in symmetric division case, since inheritance of damaged proteins should be proportional in both mother and daughter cells, and indeed is what is observed.

Finally, in simulations we observe that fitness and viability are sensitive to precise values of k_3, the rate by which proteins are damaged (see figure 4.14). This provides a series of testable hypotheses that could be investigated experimentally in different damaging environments, such as oxidative damage or radiation damage.

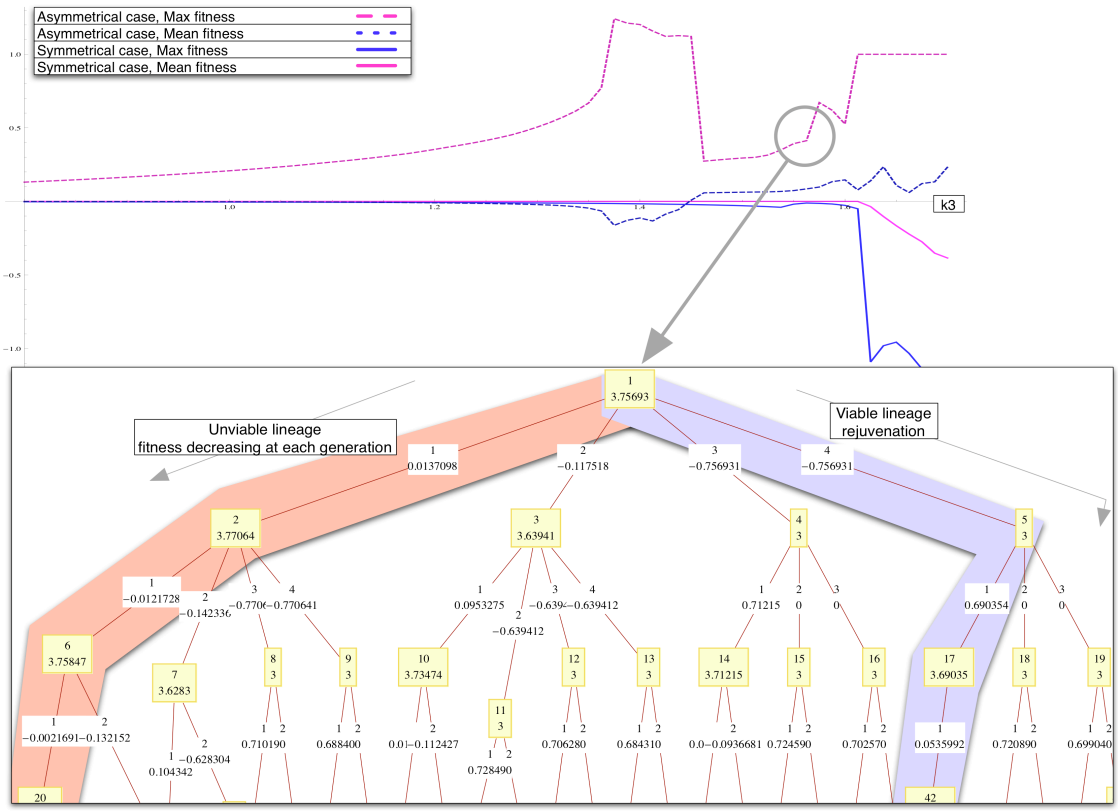


Figure 4.14: Sample parameter exploration result showing the high sensitivity and non linearity of the rejuvenation effect w.r.t. precise values of the damage rate k_3. Only some ranges of the parameter value (approx. between 1.53 and 1.59) exhibits an increase of both the maximal and the mean fitness difference between every mother and daughter of a population. **Top**: Estimation of the maximal and mean rejuvenation amount for damage rate ranging from 1.2 to 1.7 for the asymmetrical and symmetrical case. The higher the values, the fitter some cells of the lineage are compared to their direct mother. **Bottom:** Close up of a lineage tree used to compute one point of the previous rejuvenation plot. Each node (yellow box) is labeled with an numeric id and the floating-point fitness value of the cell, and each edge (white label) is labeled with the index of the daughter relatively to its mother and labeled with the difference of fitness between daughter and mother. For such a given tree, we can compute the mean and max value of the edges, which is represented as one point on the rejuvenation plot. The rejuvenation effect in young daughters of old mothers (right blue colored branch) is consistent with the experimental results of [104].

4.4.4 Conclusions

Although purely continuous systems such as ODEs have long been used for quantitative modeling and simulation of biological systems (for example [127]) and are commonly thought to be powerful enough, they do not suffice for highly structured models where emerging properties result from dynamic changes to the model.

For this study, the BioRica hybrid formalism and the related framework proved to be powerful enough to model, simulate and analyze the rejuvenation property of a hierarchical damage segregation process, by extending an existing continuous cell model to a population model. Since our hybrid formalism allows a BioRica node to describe and import an ODE system, we maintain the low computational cost and biological soundness of ODEs.

Hybrid simulation in BioRica scheme enjoys the traditional advantages of numerical integration, since the computational overhead for the hybrid stepper is proportional to the number of discrete events in the model; thus hybrid simulation of a purely continuous model is as fast as integrating it. For hybrid models mixing both continuous and discrete dynamics, our clear semantics permits concise description and reproducibility of simulation results in other simulation frameworks. For example, while the division strategy could be described in a continuous model by adjusting sigmoid functions, it is more naturally described by algebraic equations and their description in the model ought to be kept algebraic. The resulting gain in clarity has been observed elsewhere, for example in the complete cell cycle model of [36]. While most existing simulation tools admit a programming interface that allows for the modeler to simulate discrete events, the lack of a precise semantics renders the simulation predictions questionable and merely reproducible, since allowing such discrete events in a model has semantics issues[1]. Indeed, two discrete events can be enabled at the same time, but nothing defines whether in such cases the simulator should fire neither event, both events, or some random choice; and different strategies imply radically different simulation results.

In BioRica, we use the mathematical definition of non-determinism already used in discrete formalisms, thus giving any BioRica models a precise mathematical and unambiguous semantics. Furthermore, while not explicitly used in this study, BioRica leverages and extends the compositional operators initially defined in the AltaRica languages family [7] to allow for parallel, partially synchronous and data sharing compositions of hybrid, stochastic, multi-models and external abstract processes. The class of formalism that can be composed this way ranges from constraint automata to hybrid stochastic differential systems. Furthermore, since such compositions are mathematically defined in BioRica, we can exactly identify subclasses of programs admitting modern model analysis such as model checking, compositional reasoning, functional module decomposition and automatic simplification; all of which were spotted as grand challenges for modeling and simulation in system biology [165]. For compatibility with other systems biology software, BioRica imports SBML files through libSBML [93]. In addition to SBML support, BioRica exports the model as software independent C++ code, that can be compiled on any POSIX compliant system. This approach allows initial model prototyping in user friendly workbenches such as xCellerator [174], followed by use of optimized command line simulators for large scale analyses.

More specifically for population studies, since discrete variables and dynamic node creation are allowed in BioRica, our cell model can explicitly track a dynamic mother-daughter relationship. A realistic population model needs such a dynamic topology. Even when restricting ourselves to the biologically realistic case of dying cells, the number of daughters that any cell can have is *a priori* unbounded; thus, simply replicating the ODE equations to get a continuous population model as in [86] is not scalable. Furthermore, when simulations were carried up to depth 30, approximately 2^{30} cells were evolving in parallel, adding up to a 2^{32}-variable differential system that is untractable using

[1] See for example `http://www.sys-bio.org/sbwWiki/compare/themysterysolved`

a classical ODE approach. Instead, our population model clearly separates each cell behavior from the population by using hierarchical composition, and uses this modularity to provide a hierarchical simulation scheme, thus ensuring that each individual cell continuous part will be integrated with the most efficient step size.

Furthermore, the properties of parallel composition render study of population model with up to 2^{30} individuals still partially tractable by our scheme since we can linearize this population tree to simulate each cell independently. This approach is efficient since the cost of simulating a population is linearly proportional to the cost of simulating an individual, while flat and unstructured models have a quadratic complexity [60]. Finally, since we use discrete variables to track the mother-daughter relationships, we can directly estimate the rejuvenation effects, which would otherwise be buried in a flat and unstructured model.

Large scale exploration to detect the rejuvenation effect required a tree coverage that is out of reach of naive exploration algorithms such as breadth-first or depth-first. In fact, neither the population tree width nor its height are bounded, and thus these algorithms do not terminate. An *ad hoc* exploration algorithm partially solves this problem by alternating evaluation of first born daughters and evaluation of late born daughters, but does not provide the required coverage to detect significant rejuvenation. However, substantial acceleration is provided by the fix point detection scheme encoded in our tree visitor pattern, whose soundness is ensured by the deterministic nature of a cell behavior. In the continuous model, initial values of $P_{\mathrm{int}}$and $P_{\mathrm{dam}}$for given parameter values entirely determine a unique single cell behavior, and we proved that this property is preserved in the hybrid model. In fact, this is a special case of a more general result stating needed and sufficient conditions for this determinism to be preserved when adding discrete transitions to a continuous model. These conditions, beyond the scope of this paper, are not stringent and thus this acceleration can be used in most dynamic populations models built upon individuals.

Bibliography

[1] H. Aguilaniu, L. Gustafsson, M. Rigoulet, and T. Nystrom. Asymmetric inheritance of oxidatively damaged proteins during cytokinesis. *Science*, 299(5613):1751–1753, 2003.

[2] P. Akiva, A. Toporik, S. Edelheit, Y. Peretz, A. Diber, R. Shemesh, A. Novik, and R. Sorek. Transcription-mediated gene fusion in the human genome. *Genome Research*, 16:30–36, 2005.

[3] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol*, 215:403–410, 1990.

[4] R. Alur, F. Ivancic, J. Kim, I. Lee, and O. Sokilsky. Generating embedded software from heirarchical hybrid models. In *Proceedings of LCTES*, pages 171–82, 2003.

[5] K. Arakawa, Y. Yamada, K. Shinoda, Y. Nakayama, and M. Tomita. Gem system: automatic prototyping of cell-wide metabolic pathway models from genomes. *BMC Bioinformatics*, 7, 2006.

[6] A. Arkin, J. Ross, and H.H McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics*, 149:1633–1648, 1998.

[7] A. Arnold, A. Griffault, G. Point, and A. Rauzy. The altarica formalism for describing concurrent systems. *Fundamenta Informaticae*, 40:109–124, 2000.

[8] D.A. Bader, B. Moret, and M. Yan. A linear-time algorithm for computing inversion distances between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[9] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking algorithms for continuous time markov chains. *IEEE TSE*, 29(6):524–541, 2003.

[10] C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, M. Siegle, and F. Vaandrager. *Validation of Stochastic Systems: A Guide to Current Research Springer*. Lecture Notes in Computer Science 2925. Springer-Verlag, 2004.

[11] R. Balakrishnan, K. R. Christie, M. C. Costanzo, K. Dolinski, S. S. Dwight, S. R. Engel, D. G. Fisk, J. E. Hirschman, E. L. Hong, and R. Nash. *Saccharomyces* genome database, 2005.

[12] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, Massachusetts, 1998.

[13] A. Barre, V. Jouffe, C. Lartigue, M. Nikolski, A. Blanchard, and P. Sirand-Pugnet. Annotation transfert based on orthology relationships: reannotation of mycoplasma genomes from the pneumoniae group. JOBIM'06 poster, 2006.

[14] J.-P. Barthélemy and B. Leclerc. The median procedure for partitions. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1995.

[15] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S.R. Eddy, S. Griffiths-Jones, K.L. Howe, M. Marshall, and E.L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Res.*, 30:276–280, 2002.

[16] G. Bejerano and G. Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43, 2001.

[17] E.T. Bell. Exponential numbers. *Amer. Math. Monthly*, 41:411–419, 1934.

[18] A. Ben-Hur and I. Guyon. Detecting stable clusters using principal component analysis. *Methods Mol Biol.*, 224:159–82, 2003.

[19] H.M. Berman, J. Westbrook, Z. Feng, G. Gillil, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucleic Acids Res.*, 28:235–242, 2000.

[20] M. Bernt, D. Merkle, and M. Middendorf. Genome rearrangement based on reversals that preserve conserved intervals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(3):275–288, 2006.

[21] H.P.J. Bonarius, G. Schmid, and J. Tramper. Flux analysis of underdetermined metabolic networks: the quest for the missing constraints. *Trends in Biotechnol.*, 15(8):308–14, 1997.

[22] L. Bonen. Trans-splicing of pre-mRNA in plants, animals, and protists. *FASEB J.*, 7:40–46, 1993.

[23] G. Bourque and P.A. Pevzner. Genome-Scale Evolution: Reconstructing Gene Orders in the Ancestral Species. *Genome Research*, 12:26–36, 2002.

[24] G. Bourque, G. Tesler, and P.A. Pevzner. The convergence of cytogenetics and rearrangement-based models for ancestral genome reconstruction. *Genome Res.*, 16(3):311–313, 2006.

[25] G. Bourque, E.M. Zdobnov, P. Bork, P.A. Pevzner, and G. Tesler. Comparative architectures of mammalian and chicken genomes reveal highly variable rates of genomic rearrangements across different lineages. *Genome Res.*, 15:98–110, 2005.

[26] R. Brankin. Reliable solution of special event location problems for odes. *ACM Transactions on Mathematical Software*, 17:11–25, Jan 1991.

[27] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Classics in Applied Mathematics 14. Elsevier, 1989.

[28] D. Bryant. The complexity of the breakpoint median problem. Technical Report CRM2579, Centre de Recherches Mathematiques, Universite de Montreal, 1998.

[29] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Inf. Comput.*, 98(2):142–70, 1992.

[30] D.M. Burnst, V. Horn, J. Paluh, and C. Yanofsky. Evolution of tryptophan synthetase in fungi. *J. Biol. Chem.*, 265:2060–2069, 1990.

[31] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting system. *J. Chem. Phys.*, 121:4059–67, 2004.

[32] Y. Cao and L. Petzold. Slow scale tau-leaping method. *Computer Methods in Applied Mechanics and Engineering*, 97:3472–3479, 2008.

[33] A. Caprara. Formulations and Complexity of Multiple Sorting by Reversals. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings RECOMB'3*, pages 84–93, Lyon, 1999. acmp.

[34] A. Caprara. The Reversal Median Problem. *INFORMS J. on Computing*, 15(1):93–113, 2003.

[35] M. Carapeti, R.C. Aguiar, A.E. Watmore, J.M. Goldman, and N.C. Cross. Consistent fusion of MOZ and TIF2 in AML with inv(8)(p11q13). *Cancer Genet. Cytogenet.*, 113:70–72, 1999.

[36] K.C. Chen, L. Calzone, A. Csikasz-Nagy, F.R. Cross, B. Novak, and J.J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell*, 15(8):3841–62, 2004.

[37] K.C. Chen, A. Csikasz-Nagy, B. Gyorffy, J. Val, B. Novak, and J.J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular Biology of the Cell*, 11:369–391, Jan. 2000.

[38] J. Chesneaux. The equality relations in scientific computing. *Numerical Algorithms*, 7(2):129–143, Jan 1994.

[39] P. Cliften, P. Sudarsanam, A. Desikan, L. Fulton, B. Fulton, J. Majors, R. Waterson, B. A. Cohen, and M. Johnston. Finding functional features in *Saccharomyces* genomes by phylogenetic footprinting. *Science*, 301:71–76, 2003.

[40] The Genolevures Consortium. Comparative genomics of protoploid genomes of saccharomycetaceae defines the orthologous gene set and basic yeast proteome repertoire. *accepted for publication in Genome Research*, 2009.

[41] A. Courseaux and J. L. Nahon. Birth of two chimeric genes in the hominidae lineage. *Science*, 291:1293–1297, 2001.

[42] M. Cvijovic, H. Soueidan, D. Sherman, E. Klipp, and M. Nikolski. Exploratory simulation of cell ageing using hierarchical models. In *Genome Informatics Series, ISSN: 0919-9454*, volume 21, pages 114–125, 2008.

[43] A.C. Darby, N.-H. Cho, H.-H. Fuxelius, J. Westberg, and S. G. E. Andersson. Intracellular pathogens go extreme: genome evolution in the rickettsiales. *Trends in Genetics*, 23:511–520, 2007.

[44] I.J. Davis, B.L. Hsi, J.D. Arroyo, S.O. Vargas, Y.A. Yeh, G. Motyckova, P. Valencia, A.R. Perez-Atayde, P. Argani, M. Ladanyi, J.A. Fletcher, and D.E. Fisher. Cloning of an Alpha-TFEB fusion in renal tumors harboring the t(6;11)(p21;q13) chromosome translocation. *Proc. Natl. Acad. Sci. U S A*, 100:6051–6056, 2003.

[45] Jean-Antoine-Nicolas de Caritat marquis de Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. *The French Revolution research collection*, 1995. Reprod. of édition de l'Impr. royale, 1785.

[46] C. d'Enfert, S. Goyard, S. Rodriguez-Arnaveilhe, L. Frangeul, L. Jones, F. Tekaia, O. Bader, A. Albrecht, L. Castillo, A. Dominguez, et al. Candidadb: a genome database for *Candida albicans* pathogenomics. *Nucleic Acids Research*, 33:D353–D357, 2005. Sequence data for *Candida albicans* was obtained from the Stanford Genome Technology Center website at `http://www-sequence.stanford.edu/group/candida`. Sequencing of *Candida albicans* was accomplished with the support of the NIDR and the Burroughs Wellcome Fund.

[47] S. Donatelli, S. Haddad, and J. Sproston. CSLTA: an expressive logic for continuous-time markov chains. In *Proc. 4th Int. Conf. Quantitative Evaluation of Systems (QEST'07)*, pages 31–40, Edinburgh, Scotland, 2007. IEEE Computer Society.

[48] R.F. Doolittle. Similar amino acid sequences: chance or common ancestry? *Science*, 214:149–59, Oct 1981.

[49] R.F. Doolittle. The multiplicity of domains in proteins. *Annu. Rev. Biochem.*, 64:287–314, 1995.

[50] B. Dujon. Yeasts illustrate the molecular mechanisms of eukaryotic genome evolution. *Trends in Genetics*, 22:375–387, 2006.

[51] B. Dujon, D. Sherman, and G. Fischer et al. Genome evolution in yeasts. *Nature*, 430(6995):35–44, July 2004.

[52] P. Durrens, M. Nikolski, and D. Sherman. Fusion and fission of genes define a metric between fungal genomes. *Plos Computational Biology*, 4(10), 2008.

[53] E.E. Eichler. Recent duplication, domain accretion and the dynamic mutation of the human genome. *Trends in Genetics*, 17:661–669, 2001.

[54] A. J. Enright, I. Iliopoulos, N. C. Kyrpides, and C. A. Ouzounis. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402:86–90, 1999.

[55] A. J. Enright and C. A. Ouzounis. Functional associations of proteins in entire genomes by means of exhaustive detection of gene fusions. *Genome Biology*, 2:00341–00347, 2001.

[56] A.J. Enright, S. Van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30:1575–1584, 2002.

[57] A.J. Enright and C.A. Ouzounis. Generage: A robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16:451–457, 2000.

[58] N. Eriksen. Reversal and transposition medians. *Theoretical Computer Science*, 374(1-3):111–126, 2007.

[59] J. Esposito and V. Kumar. Efficient dynamic simulation of robotic systems with hierarchy. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 3, pages 2818–2823, 2001.

[60] J. Esposito and V. Kumar. An asynchronous integration and event detection algorithm for simulating multi-agent hybrid systems. *ACM Transactions on Modeling and Computer Simulation*, 14:363–388, 2004.

[61] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. In *Proc. 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*, pages 418–425, 2003.

[62] D.A. Fitzpatrick, M.E. Logue, J.E. Stajich, and G. Butler. A fungal phylogeny based on 42 complete genomes derived from supertree and combined gene analysis. *BMC Evolutionary Biology*, 6:99–114, 2006.

[63] A. Fred and A.K. Jain. Data clustering using evidence accumulation. In *In Proc. of the 16th Intl. Conference on Pattern Recognition (ICPR 2002)*, pages 276–280, 2002.

[64] L. Fribourg and M. Veloso Peixoto. Concurrent constraint automata. In *international symposium on Logic programming (ILPS)*, page 656, 1993.

[65] L. Froenicke, M.G. Caldés, A. Graphodatsky, S. Muller, L.A. Lyons, J.T. Robinson, M. Volleth, F. Yang, and J. Wienberg. Are molecular cytogenetics and bioinformatics suggesting diverging models of ancestral mammalian genomes? *Genome Res.*, 16(3):306–310, March 2006.

[66] J. E. Galagan, S. E. Calvo, C. Cuomo, L. J. Ma, J. R. Wortman, S. Batzoglou, S. I. Lee, M. Batürkmen, C. C. Spevak, J. Clutterbuck, et al. Sequencing of *Aspergillus nidulans* and comparative analysis with *A. fumigatus* and *A. oryzae*. *Nature*, 438:1105–1115, 2005.

[67] J.E. Galagan, S.E. Calvo, K.A. Borkovich, E.U. Selker, N.D. Read, D. Jaffe, W. FitzHugh, L.J. Ma, S. Smirnov, S. Purcell, et al. The genome sequence of the filamentous fungus *Neurospora crassa*. *Nature*, 422:859–868, 2003.

[68] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[69] L.Y. Geer, M. Domrachev, D.J. Lipman, and S.H. Bryant. CDART: Protein homology by domain architecture. *Genome Res.*, 12:1619–1623, 2002.

[70] R. German and C. Lindemann. Analysis of stochastic petri nets by the method of supplementary variables. *Performance evaluation*, 20:317–35, 1994. Special issue: Performance'93.

[71] M.A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem.*, 104:1876–1889, 2000.

[72] S. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81:2340–2361, 1977.

[73] E. Glemet and J. J. Codani. LASSAP, a large scale sequence comparison package (note: LASSAP is now biofacet). *Comput Appl Biosc*, 13(2):137–43, 1997.

[74] A. Goëffon, M. Nikolski, and D. Sherman. An efficient probabilistic population-based descent for the median genome problem. In *Proceedings of GECCO 2008*, pages 315–321, 2008.

[75] A. Goëffon, J.-M. Richer, , and J.-K. Hao. Progressive tree neighborhood applied to the maximum parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(1):136–145, 2008.

[76] M.L. Green and P. Karp. A bayesian method for identifying missing enzymes in predicted metabolic pathway databases. *BMC Bioinformatics*, 5(76), 2004.

[77] M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming B*, 59–96, 1989.

[78] W.N. Grundy, T.L. Bailey, C.P. Elkan, and M.E. Baker. Meta-MEME: Motif-based hidden markov models of protein families. *Computer Applications in the Biosciences*, 3(4):397–406, 1997.

[79] D.J. Sherman H. Soueidan and M. Nikolski. Biorica: A multi model description and simulation system. In *Proceedings of the 2nd Foundations of Systems Biology in Engineering (FOSBE)*, pages 279–287, Sttugart, 2007. Fraunhofer IRB Verlag. ISBN 978-3-8167-7436-5.

[80] S. Haddad, L. Mokdad, and P. Moreaux. A new approach to the evaluation of non markovian stochastic petri nets. In *Proc. 27th Int. Conf. Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2006)*, Lecture Notes in Computer Science 4024, pages 221–40. Springer-Verlag, 2006.

[81] S. Haddad and P. Moreaux. Approximate analysis of non-markovian stochastic systems with multiple time scale delays. In *Proc. 12th IEEE Int. Symp. Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2004)*, pages 23–30. IEEE Computer Society, 2004.

[82] C. Hall, S. Brachat, and F. S. Dietrich. Contribution of horizontal gene transfer to the evolution of *Saccharomyces cerevisiae*. *Eukaryot. Cell*, 4:1102–1115, 2005.

[83] S. Hannenhalli, C. Chappey, E.V. Koonin, and P.A. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: a test case. *Genomics*, 30(2):299–311, November 1995.

[84] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proceedings of twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 178–189, 1995.

[85] S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[86] M. Henson, D. Muller, and M. Reuss. Cell population modelling of yeast glycolytic oscillations. *Biochem. J.*, 368(Pt 2):433–446, Jan 2002.

[87] L. Hermida, S. Brachat, S. Voegeli, P. Philippsen, and M. Primig. The *Ashbya* genome database (AGD)–a tool for the yeast community and genome biologists. *Nucleic Acids Research*, 33:D348–D352, 2005.

[88] H. Hermjakob, L. Montecchi-Palazzi, G. Bader, J. Wojcik, L. Salwinski, A. Ceol, S. Moore, S. Orchard, U. Sarkans, C. von Mering, B. Roechert, S. Poux, E. Jung, H. Mersch, P. Kersey, M. Lappe, Y. Li, R. Zeng, D. Rana, M. Nikolski, H. Husi, C. Brun, K. Shanker, SG. Grant, C. Sander, P. Bork, W. Zhu, A. Pandey, A. Brazma, B. Jacq, M. Vidal, D. Sherman, P. Legrain, G. Cesareni, I. Xenarios, D. Eisenberg, B. Steipe, C. Hogue, and R. Apweiler. The hupo psi's molecular interaction format–a community standard for the representation of protein interaction data. *Nature Biotech*, 22(2):177–83, Feb 2004.

[89] H.L. Holm and S. Sander. Mapping the protein universe. *Science*, 273:595–602, 1996.

[90] H. H. Hoos and T. Stützle. *Stochastic Local Search : Foundations & Applications (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, 2004.

[91] S. Hua, T. Guo, J. Gough, and Z. Sun. Proteins with class α/β fold have high-level participation in fusion events. *J. Mol. Biol.*, 320:713–719, 2002.

[92] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[93] M. Hucka, A. Finney, H.M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B.J. Bornstein, D. Bray, A. Cornish-Bowden, A A. Cuellar, S. Dronov, E.D. Gilles, M. Ginkel, V. Gor, I.I. Goryanin, W.J. Hedley, T.C. Hodgman, J.-H. Hofmeyr, P.J. Hunter, N.S. Juty, J.L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L.M. Loew, D. Lucio, P. Mendes, E.D. Mjolsness, Y. Nakayama, M.R. Nelson, P.F. Nielsen, T. Sakurada, J.C. Schaff, B.E. Shapiro, T.S. Shimizu, H.D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–31, 2003.

[94] F. Iragne, M. Bertrand, M. Nikolski, D. Auber, and D. Sherman. ProViz: Protein internation vizualisation and exploration tool. Poster, European Conference on Computational Biology 2003.

[95] F. Iragne, M. Nikolski, B. Mathieu, D. Auber, and D. Sherman. ProViz: protein interaction visualization and exploration. *Bioinformatics*, 21(2):272–274, 2005.

[96] F. Iragne, M. Nikolski, and D. Sherman. Extrapolation of metabolic pathways as an aid to modelling completely sequenced non-saccharomyces yeasts. *FEMS Yeast Res.*, 8:132–139, 2007.

[97] F. Iragne, M. Nikolski, and D. Sherman. Extrapolation of metabolic pathways as an aid to modelling completely sequenced non-*saccharomyces* yeasts. *FEMS Yeast Res.*, 8(1):132–9, February 2008. Epub August 22, 2007.

[98] P. Jaccard. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, 1908.

[99] G. Jean and M. Nikolski. Genome rearrangements: a correct algorithm for optimal capping. *Information Processing Letters*, 104(1):14–20, 2007.

[100] G. Jean, D. Sherman, and M. Nikolski. Reconstruction and visualization of genome rearrangements within the kuyveromyces. Proceedings of the ESF-EMBO Symposium on Comparative Genomics of Eukariotic Microorganisms, Poster, 2008.

[101] G. Jean, D. Sherman, and M. Nikolski. Mining the semantics of genome super-blocks to infer ancestral architectures. *accepted for publication in the Journal of Computational Biology*, 2009.

[102] A. Kamburov, L. Goldovsky, S. Freilich, A. Kapazoglou, V. Kunin, A.J. Enright, A. Tsaftaris, and C.A. Ouzounis. Denoising inferred functional association networks obtained by gene fusion analysis. *BMC Genomics*, 8:460, 2007.

[103] P.D. Karp, S. Paley, and P. Romero. The pathway tools software. *Bioinformatics*, 18, 2002.

[104] B.K. Kennedy, N.R. Austriaco Jr., and L. Guarente. Daughter cells of saccharomyces cerevisiae from old mothers display a reduced life span. *J Cell Biol*, 127(6 part 2):1985–93, 1994.

[105] WJ. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler. Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc. Natl. Acad. Sci. USA.*, 100(20):11484–9, 2003.

[106] B.N. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen -activated protein kinase cascades. *FEBS Journal*, 267(6):1583–1588, 2000.

[107] T.R. Kiehl, R.M. Mattheyses, and M.K. Simmons. Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322, 2004.

[108] D. Kim, O. Rath, W. Kolch, and K. Cho. A hidden oncogenic positive feedback loop caused by crosstalk between wnt and erk p athways. *Oncogene*, 26(31):4571–9, Jan 2007.

[109] D.E. Knuth. Two notes on notation. *Amer. Math. Monthly*, 99:403–422, 1992.

[110] P. Koehl and M. Levitt. Sequence variations within protein families are linearly related to structural variations. *J Mol Biol.*, 323(3):551–62, Oct 2002.

[111] R. Kohavi and F. Provost. Glossary. *Mach. Learning J.*, 30:271–274, 1998.

[112] E.V. Koonin, L. Aravind, and A.S. Kondrashov. The impact of comparative genomics on our understanding of evolution. *Cell*, 101:573–576, 2000.

[113] S.K. Kummerfeld and S.A. Teichmann. Relative rates of gene fusion and fission in multi-domain proteins. *Trends in Genetics*, 21:25–30, 2005.

[114] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. *Lecture Notes in Computer Science*, 2324:200–204, 2002.

[115] B. Larsen and H. Aone. Fast and effective text mining using linear-time document clustering. In *Proc. of the 5th ACM SIGKDD International Conference*, pages 16–22, 1999.

[116] I. Lesur, M. Chiaverini, M. Nikolski, and D. Sherman. Malako, a system for type-safe comparison of microarray data from multiple sourcessaccharomyces siliceus. In *proc. International Meeting of the Microarray Gene Expression Data Society*, pages 127–129, 2003.

[117] C. Lindemann. *Performance modelling with Deterministic and Stochastic Petri Nets*. John Wiley and Sons, 1998.

[118] C. Lindemann, A. Thommler, A. Klemm, M. Lohmann, and O. Waldhorst. Quantitative system evaluation with dspnexpress 2000. In *Proc. 2nd Int. Workshop on Software and Performance (WOSP*, pages 12–17. ACM, 2000.

[119] B. J. Loftus, E. Fung, P. Roncaglia, D. Rowley, P. Amedeo, D. Bruno, J. Vamathevan, M. Miranda, I. J. Anderson, J. A. Fraser, et al. The genome of the basidiomycetous yeast and human pathogen *Cryptococcus neoformans*. *Science*, 307:1321–1324, 2005.

[120] J. Ma, L. Zhang, B.B. Suh, B.J. Raney, R.C. Burhans, W.J. Kent, M. Blanchette, D. Haussler, and W.Miller. Reconstructing contiguous regions of an ancestral genome. *Genome Research*, 6:1557–1565, 2006.

[121] E.M. Marcotte, M. Pellegrini, N.Ho-Leung, D.W. Rice, T.O. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions. *Science*, 285:751–753, 1999.

[122] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.

[123] N. Martin, E. Ruedi, R. LeDuc, F.-J. Sun, and G. Caetano-Anollés. Gene-interleaving patterns of synteny in the saccharomyces cerevisiae genome: are they proof of an ancient genome duplication event? *Biology Direct*, 2(1):23, 2007.

[124] H. Matsuda, T. Ishihara, and A. Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theor. Comput. Sci.*, 210:305–325, 1999.

[125] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. In *Proceedings National Academy of Science, USA*, volume 94, pages 814–819, 1997.

[126] M. Meila. Comparing clusterings by the variation of information. In *Proceeding of COLT'2003*, pages 173–187, 2003.

[127] P. Mendes. Biochemistry by numbers: simulation of biochemical pathways with gepasi 3. *Trends in Biochemical Sciences*, 22(7):361–363, 1997.

[128] P. Mendes and D.B. Kell. Meg (model extender for gepasi): a program for the modelling of complex, heterogeneous, cellular systems. *Bioinformatics*, 17:288–289, 2001.

[129] B. Mirkin. *Mathematical classification and clustering*. Kluwer Academic Press, 1966.

[130] M.Nikolski, E. Beyne, P. Durrens, and D. Sherman. Learning rules for predicting homologues in hemiascomycetous yeasts using genolevures manually-curated alignments. In *Proc. of XXI International Conference on Yeast Genetics and Molecular Biology*, pages 22–37, June 2003.

[131] H. Muhlenbein and J. Zimmermann. Size of neighborhood more important than temperature for stochastic local search. In C. Cotta and J. I. van Hemert, editors, *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 1017–1024, 2000.

[132] N.J. Mulder, R. Apweiler, T.K. Attwood, A. Bairoch, D. Barrell, A. Bateman, D. Binns, M. Biswas, P. Bradley, P. Bork, and et al. The InterPro database, 2003 brings increased coverage and new features. *Nucleic Acids Res.*, 31:315–318, 2003.

[133] J.H. Nadeau and B.A. Taylor. Lengths of Chromosomal Segments Conserved since Divergence of Man and Mouse. *Proceedings of the National Academy of Sciences of the United States of America, Part 1: Biological Sciences*, 81(3):814–818, 1984.

[134] M. Nikolskaia. A systematic study of heuristic analysis. In *Proc. of Mathematical Methods in Reliability Conference, MMR 2000*, pages 203–206, 2000.

[135] M. Nikolskaia and L. Nikolskaia. Size of OBDD representation of 2-level redundancies functions. *Theoretical Computer Science*, 255(1-2):615–625, 2001.

[136] M. Nikolskaia and A. Rauzy. Heuristics for BDD Handling of Sum-of-products Formulae. In Balkema, editor, *Proceedings of the European Safety and Reliability Association Conference, ESREL'98*, June 1998.

[137] M. Nikolskaia and A. Rauzy. Fine-tuning of boolean formulae preprocessing techniques. In Balkema, editor, *Proceedings of the European Safety and Reliability Association Conference, ESREL'99*, June 1999.

[138] M. Nikolskaia and A. Rauzy. Application des diagrammes binaires d'expression au traitement d'arbres de défaillance. In *Proc. Lambda-Mu 2000*, pages 363–367, 2000.

[139] M. Nikolskaia, A. Rauzy, and D. Sherman. Almana: A BDD Minimization Tool Integrating Heuristic and Rewriting Methods. In Springer Verlag, editor, *Proceedings of the Formal Methods for Computer Aided Design Conference, FMCAD'98*, volume 1522 of *LNCS*, pages 100–114, November 1998.

[140] M. Nikolski. Gene regulatory networks : Hybrid models vs. timed automata. JOBIM'2001 poster, 2001.

[141] M. Nikolski and P. Durrens. Gene accretion and fission events in hemiascomycete genomes. ESF-EMBO Symposium on Comparative Genomics of Eukaryotic Microorganisms, Poster, 2005.

[142] M. Nikolski, P. Ferraro, P. Durrens, and M. Aigle. Saccharomyces siliceus. In *Proc. of the International Workshop on Systems Biology of Yeast*, pages 55–56, 2003.

[143] M. Nikolski and D. Sherman. Family relationships: should consensus reign? - consensus clustering for protein families. *Bioinformatics*, 23(2):e71–e76, 2007.

[144] M. Nikolski, D. Sherman, and P. Williams. Unifying two formula rewriting techniques for circuit verification and risk assessment. Tech. Report TR-1293-03, LaBRI, University of Bordeaux-1, 2001.

[145] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, and B. Shapiro et al. BioModels Database: a free, centralized database of curated, published, quantitati ve kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(Database Issue):D689, 2006.

[146] P. Nurse. Fission yeast morphogenesis–posing the problems. *Mol. Biol. Cell.*, 5(6):613–616, June 1994.

[147] M. Ozery-Flato and R. Shamir. Two notes on genome rearrangement. *J. Bioinformatics Comput. Biol.*, 1(1):71–94, 2003.

[148] J.A. Papin, J. Stelling, N.D. Price, S. Klamt, S. Schuster, and B.O. Palsson. Comparison of network-based pathway analysis methods. *Trends in Biotechnol.*, 22(8):400–5, August 2004.

[149] S. Pasek, J.-L. Risler, and P. Brézellec. Gene fusion/fission is a major contributor to evolution of multi-domain bacterial proteins. *Bioinformatics*, 22:1418–1423, 2006.

[150] J.M. Pasia, K.F. Doerner, R.F. Hartl, and M. Reimann. A population-based local search for solving a bi-objective vehicle routing problem. In C. Cotta and J. I. van Hemert, editors, *Proceedings of EvoCOP*, volume 4446 of *Lecture Notes in Computer Science*, pages 166–175. Springer, 2007.

[151] C. A. Paulding, M. Ruvolo, and D. A. Haber. The tre2 (USP6) oncogene is a hominoid-specific gene. *Proc Natl Acad Sci U S A*, 100:2507–2511, 2003.

[152] I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 5(071), 1998.

[153] P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Research*, 13:37–45, 2003.

[154] R. Popa. *Between Necessity and Probability: Searching for the Definition and Origin of Life*. Springer-Verlag, Heidelberg, Germany, 2004.

[155] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, third edition, 2007.

[156] W.M. Rand. Objective criteria for evaluation of clustering methods. *J. of the American Statistical Association*, 66:846–850, 1971.

[157] S. Régnier. Sur quelques aspects mathématiques des problèmes de classification automatique. *ICC Bulletin*, 4:175–191, 1965.

[158] M.C. Rivera, R. Jain, J.E. Moore, and J.A. Lake. Genomic evidence for two functionally distinct gene classes. *Genetics*, 95(11):6239–6244, May 1998.

[159] A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. *Lecture Notes in Computer Science*, 5307:251–268, 2008.

[160] M. Rocchi, N. Archidiacono, and R. Stanyon. Ancestral genomes reconstruction: an integrated, multidisciplinary approach is needed. *Genome Res.*, 16(12):1557–65, 2006.

[161] R.Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. Wiley Series in Probability and Statistics. John Wiley and Sons, 1998.

[162] D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In *COCOON '97: Proceedings of the Third Annual International Conference on Computing and Combinatorics*, pages 251–264, London, UK, 1997. Springer-Verlag.

[163] D. Sankoff, G. Sundaram, and J. D. Kececioglu. Steiner points in the space of genome rearrangements. *International Journal of Foundations of Computer Science*, 7(1):1–9, 1996.

[164] D. Sankoff and P. Trinh. Chromosomal breakpoint reuse in genome sequence rearrangement. *J Comput Biol*, 12(6):812–821, 2005.

[165] H.M. Sauro, D. Harel, M. Kwiatkowska, C.A. Shaffer, A.M. Uhrmacher, M. Hucka, P. Mendes, L. Stromback, and J.J. Tyson. Challenges for modelling and simulation methods in systems biology (panel discussion). In *Proceedings of the 38th Conference on Winter Simulation*, pages 1720–30, 2006.

[166] J. Schacherer, Y. Tourette, J. L. Souciet, S. Potier, and J. de Montigny. Recovery of a function involving gene duplication by retroposition in *Saccharomyces cerevisiae*. *Genome Res.*, 14:1291–1297, 2004.

[167] H. Scherthan, T. Cremer, U. Arnason, H.U. Weier, A. Lima de Faria, and L. Fronicke. Comparative chromosome painting discloses homologous segments in distantly related mammals. *Nat Genet.*, 6(4):342–7, 1994.

[168] C.H. Schilling, D. Letscher, and B.O. Palsson. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *J. Theor. Biol.*, 203(3):229–48, April 2000.

[169] S. Schuster, D.A. Fell, and T. Dandekar. A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nat. Biotechol.*, 18(3):326–32, March 2000.

[170] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, 1992.

[171] F. Servant, C. Bru, S. Carrere, E. Courcelle, J. Gouzy, D. Peyruc, and D. Kahn. Prodom: Automated clustering of homologous domains. *Brief Bioinform.*, 3:246–251, 2002.

[172] J.S. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing, 1997.

[173] B. Shapiro, M. Hucka, A. Finney, and J. Doyle. Mathsbml: a package for manipulating sbml-based biological models. *Bioinformatics*, 20(16):2829–31, Jan 2004.

[174] B. Shapiro, A. Levchenko, E. Meyerowitz, and B. Wold. Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics*, 19(5):677–678, Jan 2003.

[175] D. Sherman, P. Durrens, E. Beyne, M. Nikolski, and J.-L. Souciet. Génolevures: comparative genomics and molecular evolution of hemiascomycetous yeasts. *Nucleic Acids Research*, 32 Database issue:D315–D318, 2004.

[176] D. Sherman, P. Durrens, E. Beyne, M. Nikolski, and J.-L. Souciet. Génolevures: comparative genomics and molecular evolution of hemiascomycetous yeasts. *Nucleic Acids Research*, 32 Database issue:D315–D318, 2004.

[177] D. Sherman, P. Durrens, E. Beyne, M. Nikolski, and J.-L. Souciet. Genolevures complete genomes provide data and tools for comparative genomics of hemiascomycetous yeasts. *Nucleic Acids Research*, 34(Database Issue):D432–D435, 2006.

[178] D. Sherman, P. Durrens, F. Iragne, E. Beyne, M. Nikolski, and J.-L. Souciet. Génolevures complete genomes provide data and tools for comparative genomics of hemiascomycetous yeasts. *Nucleic Acids Research*, 34(Database issue):D432–D435, 2006.

[179] D. Sherman, T. Martin, M. Nikolski, C. Cayla, J.-L. Souciet, and P. Durrens. Genolevures: protein families and synteny among complete hemiascomycetous yeast proteomes and genomes. *Nucleic Acids Research (NAR)*, Database Issue:D550–D554, 11 2009.

[180] A.C. Siepel and B.M.E. Moret. Finding an optimal inversion median: Experimental results. In *1st Int'l Workshop on Algorithms in Bioinformatics*, volume 2149, pages 189–203. Lecture Notes in Computer Science, Springer-Verlag, 2001.

[181] C. Simillion, K. Vandepoele, Y. Saeys, and Y. Peer. Building genomic profiles for uncovering segmental homology in the twilight zone. *Genome Res.*, 14(6):1095–106, 2004.

[182] D.A. Sinclair, K. Mills, and L. Guarente. Molecular mechanisms of yeast aging. *Trends Bioch. Sci.*, 23(4):131–4, April 1998.

[183] T. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[184] B. Snel, P. Bork, and M. Huynen. Genome evolution. gene fusion versus gene fission. *Trends in Genetics*, 16:9–11, 2000.

[185] J.L. Souciet, M. Nagy, M. Le Gouar, F. Lacroute, and S. Potier. Organization of the yeast URA2 gene: identification of a defective dihydroorotase-like domain in the multifunctional carbamoylphosphate synthetase-aspartate transcarbamylase complex. *Gene*, 79:59–70, 1989.

[186] H. Soueidan, G. Sutre, and M. Nikolsi. Model checking alltl properties over set automata. In *Proceedingds of MOdelling and VErifying parallel Processes (MOVEP)*, pages 377–383, 2006.

[187] H. Soueidan, G. Sutre, and M. Nikoski. Qualitative transition systems for the abstraction and comparison of transient behavior in parametrized dynamic models. In *accepted for publication in Lecture Notes in Bioinformatics (CMSB)*, 2009.

[188] R. Staden. Screening protein and nucleic acid sequences against libraries of patterns. *DNA Seq.*, 1(6):369–74, 1991.

[189] J. Stirling. Methodus differentialis, sive tractatus de summation et interpolation serierum infinitarium. English translation by J. Holliday, The differential method: A treatise of the summation and interpolation of infinite series. *London, 1730*, 1749.

[190] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research archive*, 2003.

[191] K. Takahashi, K. Kaizu, B. Hu, and M. Tomita. A multi-algorithm, multi timescale method for cell simulation. *Bioinformatics*, 20(4):538–546, 2004.

[192] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–160, 1972.

[193] G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, 65(3):587–609, 2002.

[194] Smith T.F. and Zhang X. The challenges of genome sequence annotation or "the devil is in the details". *Nat. Biotechnol.*, 15:1222–1223, 1997.

[195] S. A. Tomlins, D. R. Rhodes, S. Perner, S. M. Dhanasekaran, R. Mehra, X. W. Sun, S. Varambally, X. Cao, J. Tchinda, R. Kuefer, et al. Recurrent fusion of TMPRSS2 and ETS transcription factor genes in prostate cancer. *Science*, 310:644–648, 2005.

[196] A. Topchy, A.K. Jain, and W. Punch. A mixture model for clustering ensembles. In *Proc. SIAM Conf. on Data Mining*, pages 379–390, 2004.

[197] A. Topchy, M. Law, A.K. Jain, and A. Fred. Analysis of consensus partition in cluster ensemble. In *Proc. IEEE International Conference on Data Mining (ICDM'04)*, pages 225–232, 2004.

[198] K.S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., 2002.

[199] S. Tsoka and C.A. Ouzounis. Prediction of protein interactions: metabolic enzymes are frequently involved in gene fusion. *Nature Genetics*, 26:141–142, 2000.

[200] J.J. Tyson. Modeling the cell division cycle: cdc2 and cyclin interactions. *Proc. Natl. Acad. Sci. USA*, 88(16):7328–32, 1991.

[201] S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.

[202] S. van Dongen. Performance criteria for graph clustering and markov cluster experiments. Technical Report INS-R0012, National Research Institute for Mathematics and Computer Science in the Netherlands, 2000.

[203] K. Vandepoele, Y. Saeys, C. Simillion, J. Raes, and Y. Van De Peer. The automatic detection of homologous regions (ADHoRe) and its application to microcolinearity between arabidopsis and ric. *Genome Res.*, 12(11):1792–801, 2002.

[204] A. Varma and B.O. Palsson. Metabolic flux balancing: Basic concepts, scientific and practical use. *Bio/Technology*, 12:994–8, 1994.

[205] C. Vogel, S. A. Teichmann, and J. Pereira-Leal. The relationship between duplication and recombination. *J. Mol. Biol.*, 346:355–365, 2004.

[206] G. von Dassow, E. Meir, E.M. Munro, and G.M. Odell. The segment polarity network is a robust developmental module. *Nature*, 406:188–92, 2000.

[207] N. Vyahhi, A. Goeffon, M. Nikolski, and D. Sherman. Swarming along the evolutionary branches sheds light on genome rearrangement scenarios. In *accepted for publication in GECCO*, 2009.

[208] Y. Wakabayashi. The complexity of computing medians of relations. Resenhas IME-USP, 3:323-349, 1998.

[209] W. Wang, H. Yu, and M. Long. Duplication-degeneration as a mechanism of gene fission and the origin of new genes in *Drosophila* species. *Nature Genetics*, 36:523–527, 2004.

[210] D. G. Wilkinson. *In Situ Hybridization: A Practical Approach*. IRL Press, 1993.

[211] P.F. Williams, M. Nikolskaïa, and A. Rauzy. Bypassing BDD construction for reliability analysis. *Information Processing Letters*, 75(1–2):85–89, 2000.

[212] K. Wolfe and D. Shields. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387(6634):708–713, 1997.

[213] V. Wood, R. Gwilliam, M. A. Rajandream, M. Lyne, R. Lyne, A. Stewart, J. Sgouros, N. Peat, J. Hayles, S. Baker, et al. The genome sequence of *Schizosaccharomyces pombe*. *Nature*, 415:845–848, 2002.

[214] C. Wu, A. Nikolskaya, H. Huang, L. Yeh, D. Natale, C.R. Vinayaka, Z.-Z. Hu, R. Mazumder, S. Kumar, P. Kourtesis, R. Ledley, B. Suzek, L. Arminski, Y. Chen, J. Zhang, J. Cardenas, S. Chung, J. Castro-Alvear, G. Dinkov, and W. Barker. PIRSF: family classification system at the protein information resource. *Nucleic Acids Research*, 2004.

[215] I. Yanai, A. Derti, and C. DeLisi. Genes linked by fusion events are generally of the same functional category: a systematic analysis of 30 microbial genomes. *Proc Natl Acad Sci U S A*, 98:7940–7945, 2001.

[216] K. Zhang and H. Zhao. Assessing reliability of gene clusters from gene expression data. *Funct. Integr. Genomics*, 2000.

[217] Z. Zhang, H. Sun, Y. Zhang, Y. Zhao, B. Shi, S. Sun, H. Lu, D. Bu, L. Ling, and R. Chen. Genome-wide analysis of mammalian dna segment fusion/fission. *J Theor Biol.*, 240:200–208, 2006.