Analysis and Design of Molecular Machines

C. Angione[a,1,*], J. Costanza[b,1], G. Carapezza[c], P. Lió[a,*], G. Nicosia[c,*]

[a]*Computer Laboratory, University of Cambridge, UK*
[b]*Italian Institute of Technology, Milan, Italy*
[c]*Department of Mathematics and Computer Science, University of Catania, Italy*

Abstract

Biologically inspired computation has been recently used with mathematical models towards the design of new synthetic organisms. In this work, we use Pareto optimality to optimize these organisms in a multi-objective fashion. We infer the best knockout strategies to perform specific tasks in bacteria, which involve concurrent maximization/minimization of multiple functions (codomain) and optimization of several decision variables (domain). Furthermore, we propose and exploit a mapping between the metabolism and a register machine. We show that optimized bacteria have computational capability and act as molecular Turing machines programmed using a Pareto optimal solution. Finally, we investigate communication between bacteria as a means to evaluate their computational capability. We report that the density and gradient of the Pareto curve are useful tools to compare models and understand their structure, while modelling organisms as computers proves useful to carry out computation using biological machines with specific input-output conditions, as well as to estimate the bacterial computational effort for specific tasks.

Keywords: Pareto optimality, *E. coli* modelling, Turing machine, Molecular machine, Biological complexity, Petri Nets, Register Machines, von Neumann Architectures, Trade-off Genetic Strategies

1. Introduction

In the last decades, computer science and mathematics have been widely used to understand the behavior of biological systems or to analyze high throughput data. There are different methods to represent a biological system, for instance by using a system of equations. In metabolic systems, each variable represents the variation of a metabolite concentration in a compartment, in a dynamic or steady state, where the metabolite

*Principal corresponding authors
Email addresses: `claudio.angione@cl.cam.ac.uk` (C. Angione), `jole.costanza@iit.it` (J. Costanza), `carapezza@dmi.unict.it` (G. Carapezza), `pl219@cam.ac.uk` (P. Lió), `nicosia@dmi.unict.it` (G. Nicosia)
URL: `http://www.cl.cam.ac.uk/~pl219/` (P. Lió), `http://www.dmi.unict.it/nicosia` (G. Nicosia)
[1]These authors contributed equally to this work

concentration depends on material fluxes that enter or leave the compartment. Each flux can be also modelled by using kinetics parameters. Usually, these systems contain a large number of equations (differential or algebraic) and solving the problem analytically is often very hard, leading to the increasing use of numerical methods. Additionally, the advent of high throughput data in medicine requires computational techniques for data mining. Biologically inspired computation has been used to infer mathematical models, parameter values, or to capture states and transitions at the molecular level [1].

Metabolic engineering consists of optimizing genetic and regulatory processes within cells to increase the cell production of certain substances. The in silico analysis is the first step for designing a new synthetic organism. *Escherichia coli* is one of the most studied organisms in biology, as well as in synthetic biology and metabolic engineering. Researchers and biotechnologists focused their efforts on the study of its metabolic network, since it is simple and its strain is easy to manipulate in laboratory. In particular, in the last ten years, Palsson and colleagues have published several works about the *E. coli* network and its modelling. In 2000, Edwards and Palsson [2] published the first genome-scale metabolic network of the K12-MG1655 *E. coli*, composed of 627 reactions, 438 metabolites and 660 genes. By considering the steady-state, the flux balance analysis (FBA) is used to solve the system with a linear programming approach, obtaining the flux distribution in the metabolic network. The fluxes distribution depends on the environment, for instance the glucose feed or the presence of oxygen, and on genetic manipulations (knockouts). More recently, the genome-scale reconstruction of this organism has been augmented to include 904 genes, 625 metabolites and 931 reactions [3], then 1260 genes, 1039 metabolites and 2077 reactions [4] and finally 1366 genes, 1136 metabolites and 2251 reactions [5]. FBA is a useful framework in that it allows to understand the behavior of large networks and perform the knockout analysis at a low computational cost [6]. FBA-based approaches reveal more efficient than other mathematical modelling techniques, such as those based on ordinary differential equations [7], at tackling genome-scale metabolic networks, which have been extensively used to characterize energy production in cells [8], and to design synthetic pathways in silico (e.g. for production of biofuels [9]).

In this research work, we analyze two genome-scale metabolic networks of *E. coli.* By using a multi-objective optimization method called Genetic Design by Multi-objective Optimization (GDMO) [6], we maximize several pairs of biological functions, such as acetate production and biomass formation. The biomass reaction is scaled so that the flux through it is equal to the exponential growth rate of the organism. In the optimization procedure, we search for the best genetic strategies that maximize the selected objectives. The results are represented in the Pareto curve. The area under the curve, the extension and the points of the front, the *knees* and the *jumps* are features that summarize the characteristic phenotype of the organism, and are useful tools to compare different models or different organisms.

Further, we propose a mapping between a living organism and the von Neumann architecture, where the metabolism executes reactions mapped to instructions of a Turing machine. A Boolean string found by GDMO represents the optimal genetic knockout strategy and also the executable program stored in the "memory" of the organism. We adopt our framework to investigate scenarios of communication among bacteria, gene duplication, and lateral gene transfer events. Finally, we use this mapping to estimate the computational effort for a specific metabolic task, and the computational capability of the organism as function of communication outcomes, e.g. gene duplication events.

The remainder of the article is organized as follows. In section 2, we review Pareto optimality and the main ideas underlying GDMO [6]. Then, we report a comparison between two different models used to represent the *E. coli* network. In the FBA framework, we adopt the GDMO algorithm to optimize genetic manipulations in order to maximize several biological functions. We also perform the sensitivity and robustness analyses [10] for the two models, and rank nutrient metabolites according to their influence on the output (the distribution of fluxes). Additionally, based on Pareto genetic strategies, we infer neutral, trade off and destructive manipulations. In section 3, we introduce a relation between computation and metabolism explained through a formalism that associates the structure of any bacterium with a von Neumann architecture. In section 4, we discuss this mapping thinking of the metabolism as a Minsky register machine with universal computational capability. In Sections 5-6, we discuss the effect that various events (e.g. motility, communication, gene duplication) may have on the computation performed by a bacterium. We also remark the changes occurring in the computation capability as a consequence of a duplication event followed by a mutation.

2. Optimization of Gene Sets

A Pareto front is the set of points in a given objective space such that there does not exist any other point that dominates them in all the objectives. It is obtained as a result of a multi-objective optimization technique needed when a system (a given phenotype) cannot be optimal at all the tasks it performs, and particularly when tasks are in contrast with each other [11]. For instance, given the task of optimizing an organism, the Pareto front allows to maximize or minimize two or more target metabolites, thus obtaining new optimal strains specialized in many aims simultaneously.

Formally, given r objective functions $f_1, ..., f_r$ to maximize/minimize, the problem of optimizing in a multi-objective fashion can be rephrased as the problem of finding a vector x^* that satisfies all the constraints and optimizes the vector function $f(x) = (f_1(x), f_2(x), ..., f_r(x))^\intercal$, where x is the variable (vector) to be optimized in the search space. Without loss of generality, in the definition all the functions are maximized (however, minimizing a function f_i is equivalent to maximizing $-f_i$). The output of a multi-objective routine is a set of Pareto optimal points, which constitute the *Pareto front*. A solution x^* in the search space X is *Pareto optimal* if $\nexists\ x \in X$ such that $f(x)$ *dominates* $f(x^*)$, or more formally if

$$\nexists\ x \in X \quad \text{s.t.} \quad f_i(x) > f_i(x^*), \forall i = 1, ..., r, \tag{1}$$

where f is the vector of r objective functions that have to be maximized in the objective space. Since the multiple targets f_i are usually in conflict with each other, the term *optimizing* means finding all the solutions that represent a trade-off for the designer.

The many-objective Pareto optimality is a useful and powerful tool to understand the phenotype of organisms in different environmental conditions and genetic strategies. By adopting a trade-off strategy, an organism is able to maximize/minimize simultaneously several biotechnological targets, e.g. the output of the computation it carries out. In the Pareto fronts and surfaces there are distinct regions that cluster into biologically meaningful groups, as proved by Schuetz et al. [12]. The genome of the bacterial organism can be thought of as a string of binary variables. Each variable represents a gene set, i.e. the set of genes, linked by a Boolean relationship, which synthesizes for catalytic enzymes. In the gene set we can have only one gene, or two or even more genes linked with "AND" or "OR" relations. For instance, when a gene set is composed by two genes in an "AND" relation, both the genes are necessary to catalyze the corresponding reaction, and knocking out only one gene is sufficient to block the reaction. In this case, the gene set represents an *enzymatic complex.* Conversely, when a gene set is composed of two genes in an "OR" relation, the two genes synthesize for *isoenzymes*: these enzymes differ in the amino acid sequence, but catalyze the same reactions. One gene is thus sufficient to catalyze the corresponding reaction, and knocking out only one gene is not sufficient to block the reaction.

By using Pareto analysis, we maximize several biological functions in two genome-scale *E. coli* networks: that of Feist et al. [4] and that of Orth et al. [5]. In the same environmental conditions (glucose uptake rate of 10 mmol h^{-1} gDW^{-1} and anaerobic conditions), we first maximize the succinate production and the biomass formation concurrently, and then we maximize the acetate production and the biomass formation concurrently. The same experiments are also carried out in aerobic conditions, with 10 mmol h^{-1} gDW^{-1} of oxygen uptake rate (DW stands for "dry weight"). We perform a two-objective maximization, searching for the best genetic manipulations (knockouts). Knocking out a gene set means making its genes inoperative. Therefore, this gene set does not synthesize anymore for one protein (or more) catalyzing reactions of the metabolic network. Since genes are made inoperative, proteins are not present, and the corresponding biochemical reactions do not occur. Therefore, the material fluxes through those reactions are null. Maximizing acetate/succinate production and biomass formation, by searching for knockout manipulations, is a *conflict problem* of maximization, since the increase of one objective function (e.g. acetate) implies the decrease of the other (biomass). The solution of this multi-objective optimization problem is not unique, but consists of a set of non-dominated optimal solutions called Pareto front [11].

The multi-objective optimization algorithm, inspired by NSGA-II [13] is described in [6]. In Figure 1 we report the Pareto fronts obtained when maximizing succinate production and biomass formation in *anaerobic and aerobic* conditions in the *E. coli* network of Feist et al. [4] (black points), and in the *E. coli* network of Orth et al. [5] (blue points). In Figure 2 we report the Pareto fronts for the maximization of acetate production and biomass formation. Furthermore, in Figure 3 and 4 we report the Pareto front resulting from the four-objective maximization of acetate, succinate and biomass, and the simultaneous minimization of the knockout cost.

Although the organism modelled in the two flux balancing networks is the same and the environmental conditions are identical, the Pareto fronts are different. There are also differences with the *aerobic conditions*, especially for succinate. In wild type conditions, in the *E. coli* network of Orth et al., the larger and more recent network, the succinate in the cytoplasm compartment is involved in 26 metabolic reactions, and in anaerobic conditions only five reactions are activated. In particular, succinate is produced by the reactions succinyl-diaminopimelate desuccinylase, O-succinylhomoserine lyase and Fumarate depended Dihydroorotate, and consumed by succinate dehydrogenase and succinyl-CoA synthetase. The succinate can be transferred in the periplasm and in the extracellular space through ten transport fluxes. In anaerobic conditions, succinate is transported out via proton antiport. In this case, succinate production is equal to

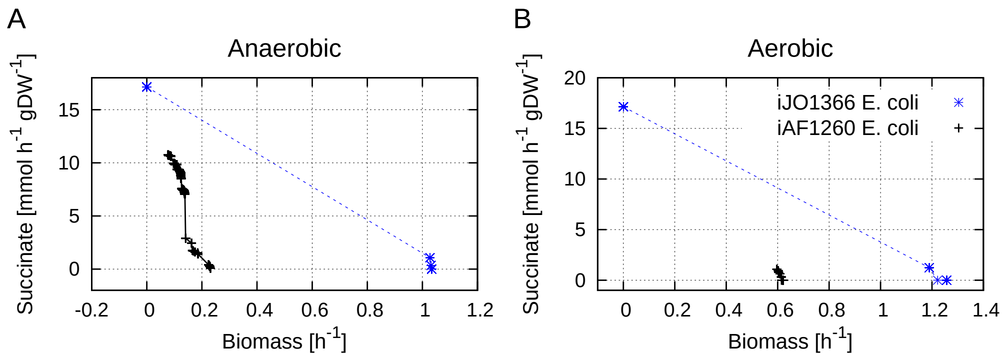

Figure 1: Pareto fronts obtained with the multi-objective optimization of the genetic strategies in *E. coli* to maximize succinate and biomass in anaerobic and aerobic conditions. In each plot, we show the results obtained by applying our optimization algorithm to the genome-scale metabolic networks of the *E. coli* models published by Feist et al. [4] (iAF1260, previously obtained in [6]) and by Orth et al. [5] (iJO1366, obtained in this research work).

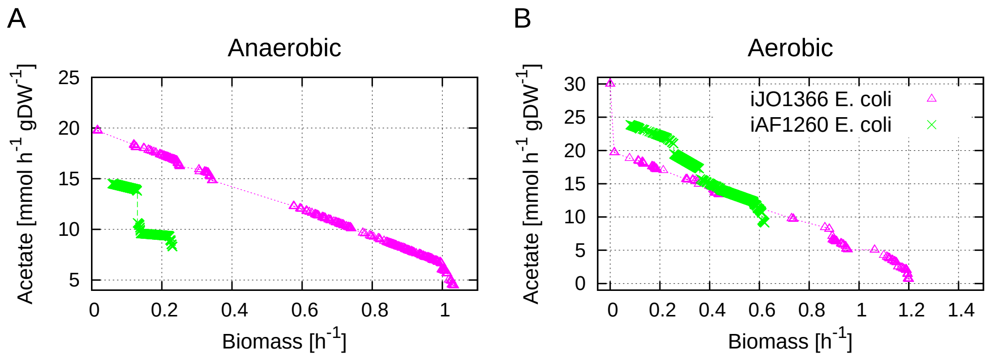

Figure 2: Pareto fronts obtained with the multi-objective optimization of the genetic strategies in *E. coli* to maximize acetate and biomass in anaerobic and aerobic conditions. We compare the results obtained using *E. coli* the model by Feist et al. [4] (iAF1260, previously obtained in [6]) with those obtained using the model by Orth et al. [5] (iJO1366, obtained in this research work).

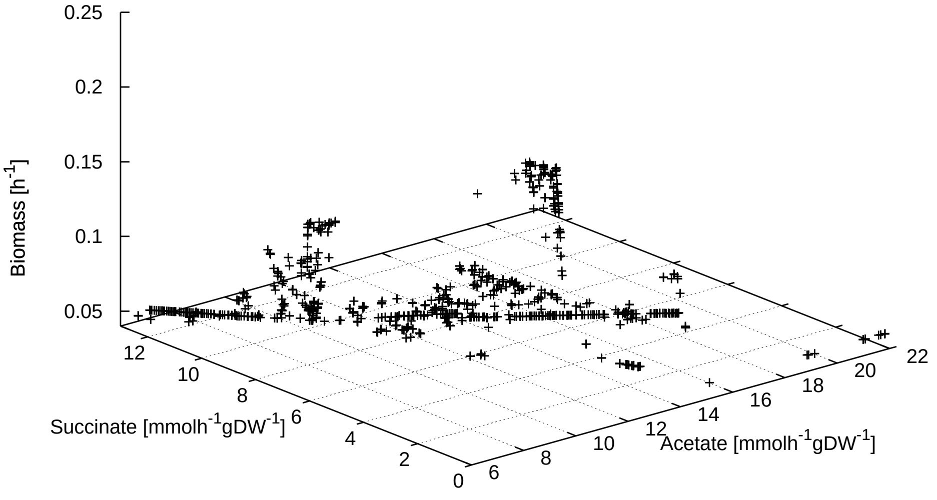

Figure 3: Pareto front obtained in the acetate-succinate-biomass space as a result of a four-objective optimization problem consisting of the simultaneous maximization of acetate, succinate and biomass while minimizing the knockout cost (four-objective optimization) in the *E. coli* model iAF1260 by Feist et al. [4].

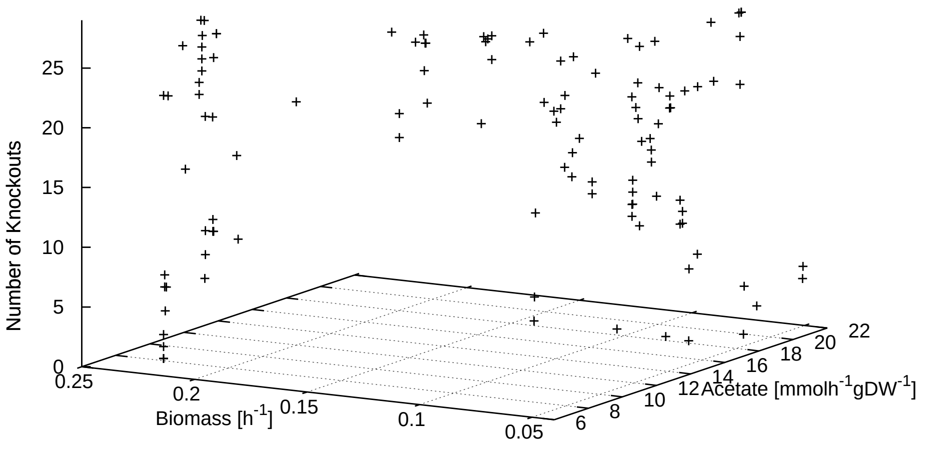

Figure 4: Pareto front obtained in the acetate-biomass-knockout space after the four-objective optimization of the genetic strategies in the *E. coli* model iAF1260 by Feist et al. [4] to maximize acetate, succinate and biomass while minimizing the knockout cost.

zero. Additionally, searching knocked out genes for maximizing succinate production gives only three Pareto solutions (showed in Figure 1 in blue): the maximum level of succinate is 17.142 mmol h^{-1} gDW^{-1}, but the biomass formation is null (the bacterium dies). So this solution cannot be considered as biologically feasible. Another solution is equal to the wild type condition: succinate is zero, and biomass 1.033 h^{-1}. The third solution reaches 1.071 mmol h^{-1} gDW^{-1} of succinate and 1.027 h^{-1} of biomass, knocking out the gene set "FumA OR FumB OR FumC", linked to the reaction "Fumarase" of the Citric Acid Cycle. All the other single gene deletions are not able to produce succinate.

Instead, for the *E. coli* network of Feist et al. [4], succinate in cytoplasm is involved in 22 reactions, and in anaerobic conditions five reactions are activated: fumarate reductase, succinyl-diaminopimelate desuccinylase, O-succinylhomoserine lyase, succinate transport out via proton antiport and succinyl-CoA synthetase. In this model, the reaction Fumarate depended Dihydroorotate is missing and the succinate production in the wild type condition is equal to 0.077 mmol h^{-1} gDW^{-1}. Unlike the *E. coli* network of Orth et al., optimizing succinate and biomass produces a Pareto front with a high number of non-dominated solutions (see Figure 1 in black). The most recent network contains additional reactions with respect to the older model, and a glucose feed equal to 10 mmol h^{-1} gDW^{-1} is not sufficient for producing succinate; indeed, by incrementing glucose feed, the Pareto front contains more points and is more dense. Unlike the acetate production, which depends mostly on the oxygen provision, the succinate production is sensitive to the glucose feed.

2.1. Inferring neutral, trade off and destructive strategies

By using the Pareto solutions obtained from the multi-objective optimization, a statistical analysis has been performed in order to cluster genetic strategies in three groups: (i) *neutral*, (ii) *trade-off* and (iii) *destructive strategies* [14]. A genetic strategy can be considered *neutral* when the objective functions do not improve (in terms of increment or decrement, in the maximization or minimization problem respectively) with respect to the nominal value. In our analysis, the nominal value is the wild type configuration of the metabolic network, i.e. when all the genes are working.

The *trade-off genetic strategies* are knockout combinations that improve an objective function and disadvantage the other. The *destructive genetic strategies* are those involving the essential genes, i.e. all the genes that are key to the biomass formation. In the knockout optimization, *constructive genetic strategies* do not exist, since a knockout cannot improve the wild type biomass. Conversely, when the decision variables are the uptake rates (nutrients optimization) we also have constructive solutions, since all the objective functions can be improved.

2.2. Case study 1: E. coli model by Feist et al.

The network of the *E. coli* model of Feist et al. [4] contains 1260 genes, and 913 gene sets. In the wild type condition, the organism grows with a biomass 0.231 h^{-1}. When we maximize the production of acetate (or succinate) searching for the best knockout strategy, the biomass rate always decreases, as shown in Figure 2 and Figure 1.

The maximum level of acetate in anaerobic conditions, is 14.519 mmol h^{-1} gDW^{-1}, corresponding to the minimum value of biomass (0.057 h^{-1}). The overall *knockout cost*, namely the minimum number of genes to be turned off in order achieve this result, is 19. This result is obtained when the following eleven gene sets are turned off:

1. b0351 OR b1241
2. b0870 OR b2551
3. b0323 OR b0521 OR b2874
4. b1773 OR b2097 OR b2925
5. b1198 AND b1199 AND b1200 AND b2415 AND b2416
6. ((b2481 AND b2482 AND b2483 AND b2484 AND b2485 AND b2486 AND b2487 AND b2488 AND b2489 AND b2490) AND b4079) OR ((b2719 AND b2720 AND b2721 AND b2722 AND b2723 AND b2724) AND b4079)
7. b2500
8. b1380 OR b2133
9. b2913
10. b3708
11. b0171

In particular, we set to '1' eleven elements of the knockout vector y. For instance, by turning off the gene set (b0351 OR b1241) we have a knockout cost associated with this gene set equal to 2, i.e., we have to delete both the genes in order to arrest the flow through the corresponding reaction(s). Instead, the knockout cost associated with (b1198 AND b1199 AND b1200 AND b2415 AND b2416) is 1, since we have to turn off at least one of the genes of the entire set to ensure the arrest of flow through the corresponding reaction(s). The total knockout cost (19) associated with the proposed knockout strategy is the sum of the knockout costs associated with each gene set involved in the strategy.

Instead, if we want to consider a trade-off between biomass and acetate, we suggest the solution that has a knockout cost equal to 5 and reaches 13.9220 mmol h^{-1} gDW^{-1} of acetate and 0.1277 h^{-1} of biomass. Another suitable solution is that with a knockout cost 7 and 13.9801 mmol h^{-1} gDW^{-1} of acetate and 0.1222 h^{-1} of biomass. In the wild type bacterium, the acetate production is 8.301 mmol h^{-1} gDW^{-1}.

By considering the point (0.1303; 13.7911) of the Pareto front in green of Figure 2 that has a knockout cost 3 (YdfG; MhpF OR AdhE), we find that 88 gene sets can be considered neutral genetic strategies. For instance, Aas; rffT; AtoB; rfe; Acs; Lpd and SucA and SucB; Amn; and AdiA are just some of the neutral genetic strategies. Instead, (YdfG; MhpF OR AdhE) represents a trade-off strategy. For the acetate maximization problem, we find that the genes MraY and murG are essential (destructive strategies), i.e. turning them off causes a null biomass, and the organism dies. Additionally, the gene tnaA is involved in 796 Pareto manipulations (out of 1000) when the acetate is maximized in anaerobic conditions. Moreover, this gene is also involved in most genetic manipulations in aerobic conditions (721 out of 1000 strategies).

As regards the succinate versus biomass optimization, we obtain the maximum level equal to 10.757 mmol h^{-1} gDW^{-1}, with a biomass 0.076 h^{-1} and a knockout cost equal to 15. A suitable solution can be obtained knocking out the gene set (isoenzymes) "AdhP OR AdhE OR FrmA", reaching 9.0373 mmol h^{-1} gDW^{-1} of succinate and 0.1231 h^{-1} of biomass.

An interesting result in anaerobic conditions is that the gene set "(SapD AND TrkA AND TrkH) OR (Kch) OR (SapD AND TrkA AND TrkG) OR (Kup)" is the most frequent both for maximizing acetate and for maximizing succinate. In Table 2.2, we report the most significant results. Pareto results in aerobic conditions have been reported in Figure 2 and Figure 1.

Acetate	Anaerobic	Aerobic
tnaA	0.796	0.721
GuaB	0.728	0.310
SurE	0.568	0.077
Succinate	Anaerobic	Aerobic
Apt	0.950	0.510
DeoD	0.610	0.230
DeoD or deoA	0.260	0.320

Table 1: Frequency of some gene sets in the knockout genetic strategies for acetate maximization and succinate maximization in the *E. coli* metabolic network [4]. The values reported in the second and third columns, between 0 and 1, represent the percentage of occurrences of the gene set (first column) in all the Pareto manipulations obtained with our method.

2.3. Case study 2: E. coli model by Orth et al.

The network of *E. coli* model of Orth et al. [5] contains 1366 genes, and 1041 gene sets. In the wild type configuration, the organism grows with a biomass of 1.033 h^{-1}. The maximum level of acetate in anaerobic conditions, is 19.789 mmol h^{-1} gDW^{-1}, corresponding to the minimum value of biomass (0.016 h^{-1}). The knockout cost is 19. Instead, if we consider a trade-off between biomass and acetate, we suggest the solution with a knockout cost equal to 1 (Mdh, malate dehydrogenase) and reaches 16.209 mmol h^{-1} gDW^{-1} of acetate and 0.252 h^{-1} of biomass. Other suitable solutions are showed in Figure 5. As discussed in the previous paragraph, we obtain only one acceptable solution that maximizes succinate in anaerobic conditions.

Analyzing all genetic strategies obtained by the Pareto front in purple in Figure 2, the gene set SerA results the most involved in the knockout manipulations (554 manipulations out of 1000). In the *E. coli* model by Feist et al., this gene has a frequency equal to 510 out of 1000. Additional details about the solutions found are also reported in Table 2. Other experiments have been performed for the following maximization: (i) ethanol production versus biomass (Figure 6), (ii) CO_2 production versus biomass (Figure 6), (iii) ATP production versus biomass (Figure 7). All the experiments have been performed using a population of 1000 individuals, and performing 500 generations of the multi-objective optimization algorithm GDMO. For the details about the algorithm, see [6].

2.4. Epsilon dominance, Sensitivity and Robustness Analyses

Epsilon Dominance Analysis. The *Epsilon dominance analysis* is inspired by the work of Laumanns et al. [15], namely to use a condition of approximated dominance for their evolutionary multi-objective algorithm with the aim of improving the diversity of solutions and the convergence of the algorithm. The "relaxed" condition of dominance, called *ϵ-dominance*, is defined as follows. Assuming that all the objective functions are positive and must be maximized, given $\epsilon > 0$ and a non-dominated point y^*, they select all the solutions y such that $f_i(y^*) - \epsilon \leq f_i(y), \forall i = 1, \ldots, r$, where f is the vector of the r objective functions. This set will contain both the "ϵ-non-dominated" solutions and the non-dominated ones.

As we will see, given a biological task and a set of reactions that shape the bacterium thought of as a Turing Machine (see Section 3), GDMO is able to produce molecular machines. In particular, it finds the string y, which represents a genetic strategy, able

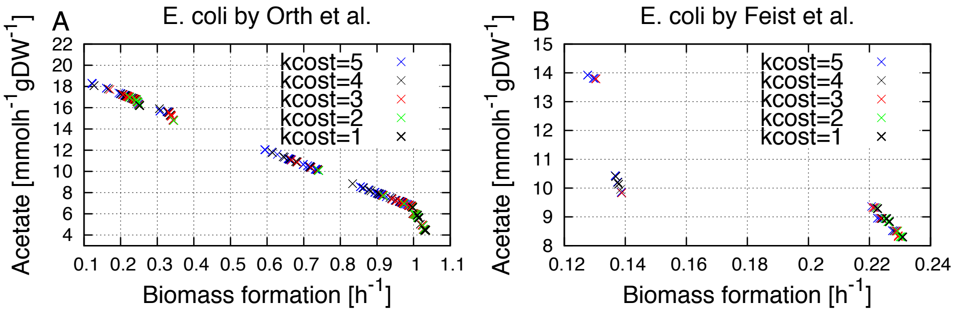

Figure 5: Pareto solutions obtained with the optimization of the genetic strategies in the *E. coli* network by Orth et al. [5] in A, and *E. coli* network by Feist et al. [4] in B, to maximize acetate and biomass in anaerobic conditions. Different colors have been used to indicate the knockout cost associated with each point. In these plots we consider solutions with a maximum knockout cost equal to 5.

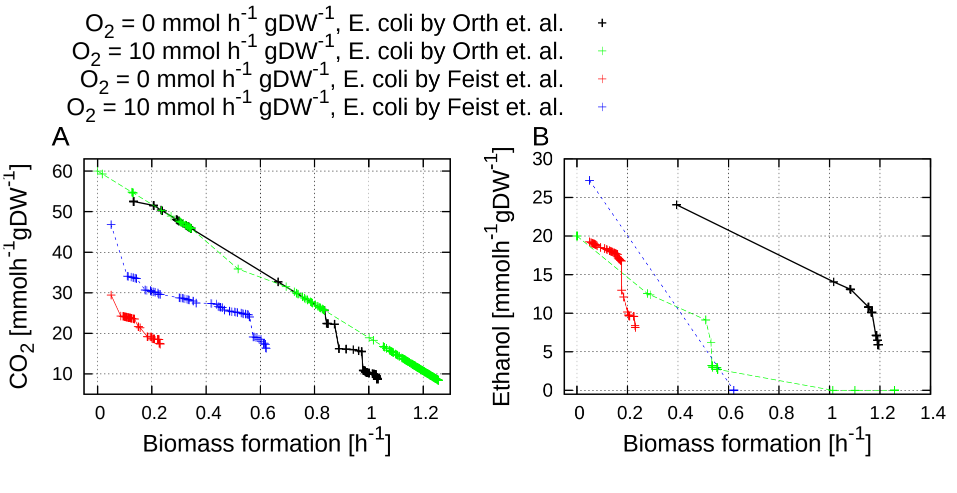

Figure 6: Pareto fronts obtained with the optimization of the genetic strategies in the *E. coli* models [4, 5] to maximize CO_2 (A), ethanol (B) and biomass.

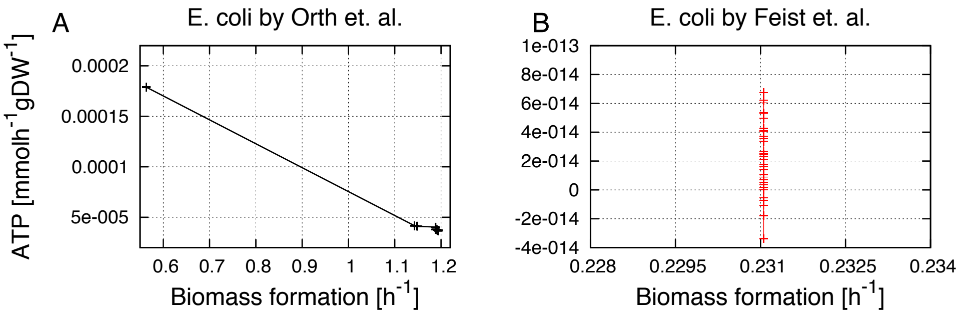

Figure 7: Pareto fronts obtained with the optimization of the genetic strategies in the *E. coli* models [4, 5] to maximize ATP and biomass.

to program the bacterium to perform the desired task. For each string y, the bacterium performs a task in a more or less efficient manner. According to the objective functions, the solutions are collected in a Pareto front consisting of non-dominated points. Other solutions and therefore molecular machines (each Pareto point is a strain, and therefore a particular molecular machine), can be derived by applying the condition of ϵ-dominance. Specifically, rather than considering only the non-dominated solutions, we consider also those solutions obtained by applying a condition of dominance approximated by an ϵ value. Hence, if a solution had been discarded because one of its objective functions was dominated by a value smaller or equal to ϵ, by applying this principle that solution is taken into consideration.

An example of this analysis was conducted on *E. coli* [5], as shown in Figure 8. The ϵ-dominance analysis explores all the Pareto ϵ-dominated solutions calculated in all the iterations of GDMO, and finds the genetic strategies y able to maximize acetate (A) or succinate (B) production and the biomass formation simultaneously. In addition, we show the points obtained by performing this analysis for values of epsilon equal to 10^{-6}, 10^{-3} and 10^{-1} (Figure 8, pink, red and green points). The larger is the value of epsilon, the more relaxed is the condition of dominance, and the larger is the number of molecular machines obtained, and able to perform the maximization of acetate/biomass or succinate/biomass.

By considering the succinate and biomass maximization, the ϵ-dominance analysis reveals an interesting solution (the green point of Figure 8-B), which reaches a succinate production equal to 1.013 mmol h^{-1} gDW^{-1} and 0.971 h^{-1}, with a knockout cost equal to 4.

Sensitivity Analysis. If the knockout string y has been set, the output(s) of a molecular machine depends on the inputs, i.e. nutrient metabolites. The sensitivity analysis allows us to analyze, which input(s) have, a strong influence on the output(s). To perform this analysis, we used the Morris method [16], which consists of an experimental plan composed of individually randomized one at time (OAT) designs. The data analysis is then based on the so called *elementary effects*, namely the changes in an output due to changes in a particular input factor in the OAT design. The Morris method can determine if the effect of the input factor x_i on the output v is negligible (v is the vector of the fluxes), linear and additive, nonlinear or involved in interactions with other input factors $\mathrm{x}_{\sim i}$.

The analysis was conducted on the *E. coli* models discussed in the previous paragraphs, y is set to the null vector (no knockouts, i.e. in wild type conditions) and the inputs x_i are the k lower bounds of the exchange fluxes. The output of the molecular machine is the distribution v of fluxes calculated by the FBA. Eventually, for each ith input, we have a distribution of elementary effects. Two sensitivity indices are calculated for each input, the mean μ^* and the standard deviation σ^* of the distribution of elementary effect. The analyzed input fluxes are sorted according to their influence on the output. In Figure 9, we report the graphical results of the analysis.

Robustness Analysis. Since the string y and the inputs have been set, a further analysis allows us to obtain the robustness of the molecular machine able to perform a specific task (e.g. maximizing acetate and biomass). Indeed, the molecular machine (or strain) can receive noise either from the environment or from the inside. We perform the robustness analysis [10] on the *E. coli* (model by Orth et al. [5] and model by Feist et al. [4]). The y strings used are those obtained by the optimization of acetate (or succinate) production and biomass formation (Figure 1 and Figure 2). The inputs are the upper v_j^U and lower v_j^L bounds, $j = 1, \dots, n$ of the metabolic fluxes. In particular, for each strain, the fluxes corresponding to knocked out gene sets, are maintained equal to zero.

There are two types of robustness analysis, called Global Robustness (GR) and Local Robustness (LR) [10]. In the first case, we apply the perturbation concurrently to the inputs, so as to evaluate the overall fragility of the molecular machine, while in the second case the perturbation is carried out for each input at a time, obtaining a robustness value for each input [17]. We also implement the analysis described in the work of Hafner et al. [18], where the authors implement a procedure that calculates the normalized volume (R) occupied by those parameters such that a system maintains the desired characteristics. The volume is computed in the parameter space. In our case, it is computed in the parameter space (given y, the upper v_j^U and lower v_j^L bounds, $j = 1, \dots, n$ of the nonzero metabolic fluxes), such that the molecular machine remains robust. For all the cases, we use the acetate (or succinate) production and the biomass formation as metrics. The results are shown in Table 2. In A strains, we report the genetic strategies that reach the maximum acetate or succinate amount; in B-C-D strains, we show the genetic strategies that represent trade off solutions. Instead, by W we indicate the wild type configuration, in order to assess if the genetic strategies cause an increase or a decrease of the molecular machine robustness.

3. Molecular Machines and Turing Machines

In 1952, Turing outlined computational processes in the morphogenesis [19], thus thinking of the biological development of an organism as a consequence of the computation that it can perform. Following Turing's idea on morphogenesis, many biological processes have been recently analyzed from a computational standpoint. In 1995, Bray [20] argued that *a single protein is a computational or information carrying element*, being able to convert input signals into an output signal. Specifically, Turing's idea is that the computation carried out by an organism allows it to move from one development pattern into another. For instance, multi-cellular organisms can be thought of as

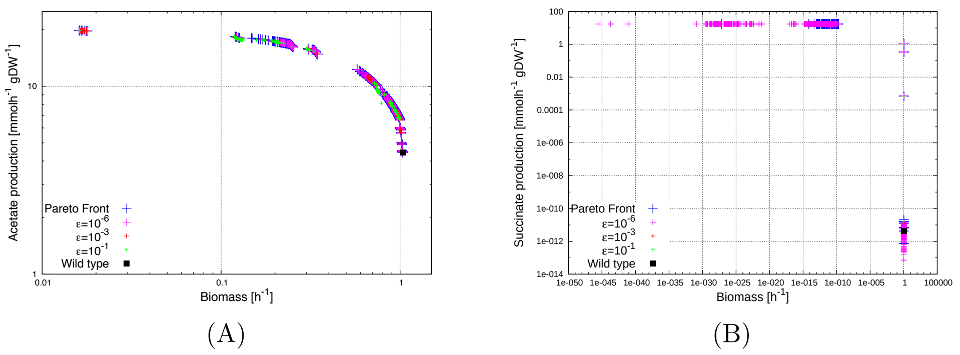

Figure 8: ϵ-dominance analysis results for *E. coli* [5]. An increased number of genetic strategies is found when constructing the Pareto Fronts with a relaxed dominance condition. The number increases with increasing ϵ. In (A) the results for the optimization of acetate and biomass, in (B) the results for the optimization of succinate and biomass.

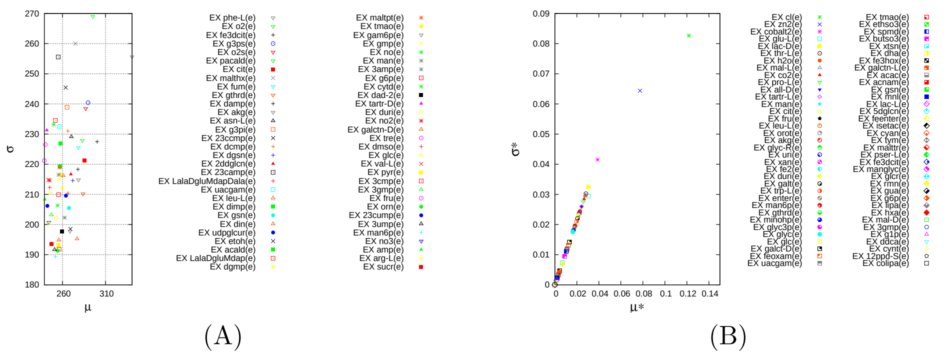

Figure 9: Sensitivity analysis results for *E. coli* models by Orth et al. [5] in (A) and by Feist et al. [4] in (B). The analyzed input fluxes are ranked according to their influence on the output. In (A), only the 60 most sensitive fluxes are shown.

the product of the computation started from a single cell, which is capable of running a program like that of a computer [21]. This leads to speculate that the instructions of the code run in an organism are responsible for driving its behavior and evolution.

Evolution had already been associated with computation many years before, by von Neumann and Burks [22], who constructed a self-replicating cellular automaton with the aim of developing synthetic models of a living organism. Recently, the theory of self-reproducing machines has been thoroughly analyzed by means of text register machines [23] or self-modifying register machines [24]. Several programming languages have been developed and analyzed for modelling biological computational processes. A computational process has been discovered also in ciliates during the unscrambling of genes [25], showing many points in common with the Adleman's algorithm [26], where a DNA computer is able to solve the instances of the NP-complete Hamiltonian path problem. Therefore, a computation having the *micronuclear sequence* as input can yield a functional *macronuclear gene*, meaning that a guided genome recombination system can simulate a Turing Machine (TM) [27]. In a molecular machine, there is increasing evidence

Strains by model Orth et al.	GR (%)	LR (%)	R	Acetate ($mmolh^{-1} \cdot gDW^{-1}$)	Biomass (h^{-1})	KC
W	22.88	17.29	1.36	4.446	1.033	0
A	12.81	15.03	1.29	19.790	0.016	19
B	25.56	38.35	1.79	10.644	0.702	8
C	54.61	56.39	1.90	16.208	0.252	1
Strains by model Orth et al.	**GR (%)**	**LR (%)**	**R**	**Succinate ($mmolh^{-1} \cdot gDW^{-1}$)**	**Biomass (h^{-1})**	**KC**
W	4.46	0	1	0	1.033	0
A	94.58	98.49	1.23	1.072	1.028	3
B	8.91	9.77	1.63	1.013	0.971	4
Strains by model Feist et al.	**GR (%)**	**LR (%)**	**R**	**Acetate ($mmolh^{-1} \cdot gDW^{-1}$)**	**Biomass (h^{-1})**	**KC**
W	54.76	54.00	1.30	8.301	0.231	0
A	28.04	36.09	1.329	14.519	0.058	19
B	38.93	48.12	1.171	13.791	0.130	3
C	33.48	44.36	1.575	13.922	0.128	5
D	42.23	50.38	0.257	13.980	0.122	7
Strains by model Feist et al.	**GR (%)**	**LR (%)**	**R**	**Succinate ($mmolh^{-1} \cdot gDW^{-1}$)**	**Biomass (h^{-1})**	**KC**
W	53.68	54.67	1.34	0.077	0.231	0
A	100	36.09	1.17	10.757	0.077	15
B	100	50.37	1.54	9.037	0.123	3

Table 2: Results of the Global Robustness (GR), Local Robustness (LR) and the normalized volume of the robust parameters (R) related to acetate and succinate optimization. Points have been selected from Pareto fronts of Figure 2 and Figure 1. Details about the amount of acetate, succinate, biomass and knockout cost (KC) are reported in the columns. In A strains, we report the genetic strategies with the maximum acetate or succinate amount; in B-C and D strains, genetic strategies that represent trade off solutions. By W we indicate the wild type configuration.

that the DNA is the part of the cell simulating a memory storage [28].

Here we propose a relation between computation and metabolism explained through an effective formalism. In particular, we associate the structure of a bacterium with a von Neumann architecture, showing that the components of a bacterium can be mapped to a processing unit, a control unit, a memory storing the “program” of the bacterium, and an input-output section. In this way, the bacterium becomes a molecular machine with computation capability. Furthermore, the set of all its chemical reactions represents a processing unit, and we show that the entire metabolic network works as a TM [29]. An optimal molecular machine that executes a particular task can be obtained using GDMO [6]. Indeed, Pareto fronts represent optimal organisms that are output of a multi-objective optimization carried out in the metabolic network. Each point provided by GDMO is a Pareto optimal molecular machine whose computation is aimed at maximizing the concentration of two or more metabolites (outputs) simultaneously.

Running a program in a molecular machine can represent an effective intervention in a cell, driving it towards the modification of its behavior according to the available inputs and the desired outputs. More specifically, taking into account external variables, the current cell state, or user-imposed goals, a code can instruct the cell to make decisions. Although modelling the whole life of an organism as a TM would certainly be computationally unfeasible, our approach is aimed at explaining the single operation executed by a bacterium in light of a computational instruction. This approach can be readily used to evaluate the computational effort for a specific task, or the computational capability of the whole organism under investigation.

4. Bacteria as von Neumann architectures

A useful metaphor to frame a biological systems as a von Neumann computer [30] hinges on the representation of the metabolism as a TM [29], as summarized in Table 3.

Biological organism	von Neumann	Bacterium
input metabolites	input	input metabolites
output metabolites	output	output metabolites
control	control	pathway handling
genome	memory	knockout string
metabolism	processor	TM

Table 3: Dictionary translating the general biological organism (left) into the computational concept of the von Neumann architecture (center) and the equivalent in a bacterium, according to our framework (right).

The sections of the von Neumann architecture mapped to an evolving bacterium are shown in Figure 10. The substrates required for the growth of the bacterium are represented by the input module of the architecture. The memory module contains the knockout string y obtained by GDMO in Section 2, which plays the role of the "program" responsible for protein production in the cell. The chemical reaction network reads and executes the program y, converting input metabolites into output products [29]. The control unit g activates or deactivates gene sets translating the binary string y (syntax) into knockout instructions (semantics). The network of the bacterial metabolism drives this process.

Interestingly, the metabolism of a bacterium can be regarded as the processing unit of the von Neumann architecture, as it has been proved to be able to perform computation [31]. More specifically the metabolism can be mapped to a Minsky's register machine (RM), where each register is a left-handed tape that stores non-negative integers by writing stacks of marks on the tape (a blank tape represents the count '0'). The Minsky's RM is equivalent to a multi-tape TM in which tapes are restricted to work as simple counters [32]. The RM registers are monosymbolic and not bounded, i.e. they are stacks containing the same symbol repeated. Each register can be either incremented or decremented (if it is nonzero, otherwise the instruction is ignored and the machine proceeds to the next instruction).

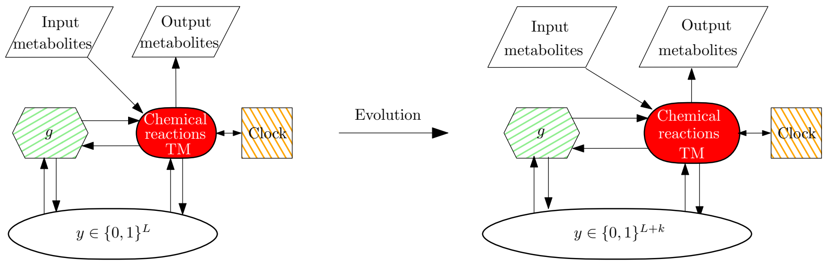

Figure 10: The sections of a von Neumann computer can be found in evolving bacteria. The string y is a knockout genetic strategy, i.e. the program stored in the RAM and responsible for protein synthesis. The function g represents the control unit of the computer, interprets the program y and handles gene sets so as to produce desired amount of proteins. The metabolism is the processing unit, composed of all the chemical reactions of the bacterium and mapped through a Minsky machine. The clock module communicates with the register logic module in a way that ensures a low probability of error per step. Metabolites of interest are produced as output of the molecular machine. The evolution and growth of a bacterium depend on duplication and mutation events occurring in the program (genome) and in its length.

The formal definition of Minsky machine

$$\mathcal{M} = (D, H, i_0, i_1, \varphi) \tag{2}$$

includes a finite set D of states, a finite set $H = \{H_r\}_r$ of registers, and a multivalued mapping

$$\varphi : D \backslash \{i_0\} \longrightarrow \{(H_r, i), (H_r, j, k) \mid H_r \in H,\ j, k \in D\}. \tag{3}$$

The initial and the halting states are two distinguished elements $i_0, i_1 \in D$. The RM executes two basic increment/decrement instructions: (i) $inc(i, r, j)$ to increment register r by 1 and move from state i to state j according to $\varphi(i) = j$, and (ii) $dec(i, r, j, k)$, with $H_r > 0$, to decrement register r by 1 and move from state i to state j ($\varphi(i) = j$); if $H_r = 0$, the machine moves from state i to state k ($\varphi(i) = k$). The RM also has a *halt* instruction that halts its operation, setting the state i_1.

The chemical reaction network of a bacterium can be simulated by a RM defining sets for state species and register species [31]. In the set of register species $\{H_r\}$, each H_r represents the molecular count of species r and corresponds to the rth register of the RM. In the set of state species $\{D_i\}$, each D_i corresponds to the ith state of the RM. More specifically, D_i is an auxiliary species representing only one of the states of the metabolic network, with no correspondence with its molecules. For every state of the RM, the molecular count of the associated state species will be 1, and all the others will be 0. The instruction $inc(i, r, j)$ is mapped to the chemical reaction $D_i \to D_j + H_r$, and the instruction $dec(i, r, j, k)$ is mapped to $D_i + H_r \to D_j$ if $H_r > 0$, or to $D_i \to D_k$ if $H_r = 0$. The reaction $D_i \to D_k$ can occur only if H_r is over, since the rth register cannot be decreased and the reaction $D_i + H_r \to D_j$ is not feasible. In this way, the bacterium executes a "test for zero". Finally, the *halt* instruction is equivalent to the cell death, when no further chemical reactions take place [33].

In order to enforce the order of reactions that take place in the register machine, a clock module can be used to communicate with the register logic module [34]. If the bacterium executes chemical reactions in a fixed volume V, given two reactions *inc* and *dec* associated with fluxes v_1 and v_2 respectively, its metabolism has a probability of error per step equal to $\epsilon = v_2/(v_1/V + v_2)$. The clock sends a species C to the register module, and when a step is performed in that module, C is converted into another species T, which is sent back to the clock. After a given delay, the clock sends C again to perform another step. The delay can be controlled in a way that ensures a low probability of error per step. More specifically, an error of the register machine consists of disabling a decrement reaction $D_i + H_r \to D_j$ involving a species H_r that is still available but with only one molecule left. This error can happen if the clock signal C is sent before the reaction $D_i + H_r \to D_j$ is completed during a decrement step, causing the reaction $D_i + C \to D_0 + T$ to take place instead, where D_0 is the status to indicate that the register H_r is empty. Since the decrement reaction has a rate independent of the clock (e.g., related to the FBA flux rate at steady state, or to the number of molecules of reactants), the role of the clock is to delay the release of C. This imposed delay maximizes the probability that the correct decrement reaction $D_i + H_r \to D_j$ occurs first, and therefore minimizes the probability of error per step. If this happens correctly, D_i is consumed and consequently $D_i + C \to D_0 + T$ is not allowed to take place, while the reaction $D_j + C \to D_0 + T$ will take place if another decrement is requested for H_r.

As regards the computation rate in a cell, given that the total number of proteins per cell is approximately $5 \cdot 10^7$ (e.g. for the budding yeast [35]), and on average enzymes carry out 10 s^{-1} reactions (far lower than the common examples found in literature) [36], we have $5 \cdot 10^8$ instructions performed every second by the molecular machine. Each instruction consists of multiple decrement and increment instructions for the registers involved in the reaction, e.g. executed through a multi-core architecture. This leads us to estimate a $5 \cdot 10^8$ Hz (or 0.5 GHz) computation rate in each cell. Reversible reaction can be translated into two different instructions, therefore increasing the computational capability of the molecular machine. A program executed in an organism could be able to implement the genetic strategy proposed by GDMO [6]. Furthermore, being the simulated TM an universal machine, our mapping would allow a bacterium to perform any computation performed by a computer, using its species and reactions characterized by their flux.

4.1. Petri Nets and Register Machines to simulate pathways

A Minsky machine can be simulated by a Time Petri Net (TPN), i.e. a Petri net in which each transition is associated with a time interval. Petri nets (PN) are directed bipartite graphs that link *places* to *transitions*. A state of the PN consists of a configuration with specific number of tokens at each place. A chemical reaction network can be modelled with a PN, in order to track the number of molecules for each species. Each place represents a molecular species of the chemical reaction network, with tokens representing the number of molecules of that species. The incidence matrix of the PN becomes the stoichiometric matrix of the network, with arc weights depending on the stoichiometric coefficients. The firing of a transition simulates a chemical reaction that takes place in the network, converting reactants into products. In order to simulate a RM, the TPN must be augmented with strong semantics, i.e. a transition can be fired or inhibited only when the time reaches the upper bound of the reaction firing interval. This allows to give priorities to conflicting transitions using the same tokens, and therefore makes it possible to implement a test for zero [37].

A PN can be defined using three main types of transition: *stochastic*, *continuous*, or *immediate*. The definition of propensity depends on the transition type, with the common property that only transitions with nonzero propensity are enabled [38]. The propensity of a stochastic transition is the probability density of the firing in the next time step Δt. In a continuous transition, the propensity is used as reaction rate, i.e. the number of firings k of every transition is the product of its propensity function and the time-step Δt; therefore, the number of tokens that each continuous transition moves from reactants to products is proportional to its propensity. Conversely, in an immediate transition, a nonzero propensity indicates the firing of the transition.

Petri nets have been widely used to represent metabolic pathways or specific parts of chemical reaction networks [39]. As a result, a RM is able to model any metabolic pathway [40]. Metabolic pathways (e.g. glycolysis) have been also modelled using Hybrid Functional Petri nets, where a function can control the speed or condition of firing of continuous transitions [41]. The rate of a transition represents the speed of the transformation from reactants (input places) to products (output places). At steady state (T-invariant of a PN), the flux rates can act as an interface between FBA and the simulation of the system through a PN. More specifically, the flux rates found with our optimization algorithm can play the role of the propensity coefficients of the associated reactions, thus representing the probability of firing. Likewise, in Hybrid Functional Petri nets, the function can be defined taking into account the flux rates provided by the optimization algorithm.

In order to fine tune this approach, biologically meaningful weights can be used on the arcs (e.g., for allowing a firing only when the reaction rate reaches a specific threshold). This type of RM can produce arbitrarily large number of molecules, and therefore the reaction volume needs to be proportional to the total number of molecules in the pathway [34]. Finally, due to the uncontrollable order in which firings take place, a unique final state is not guaranteed.

5. Multi-objective optimization of energy and motility in molecular machines

In this section we study the trade-off between energy production (required for motility) and biomass formation (required for growth) in the bacterial metabolism, which is known to modulate also chemotaxis and motility behavior. Specifically, a reduced biomass due to starvation implies consuming energy resources in order to reach new sources of food. This is done by transduction of specific signal, through direct modulation of flagellar rotation. Since energy production and biomass are contrasting objectives, a multi-objective optimization approach would be able to optimize both simultaneously, providing us with a set of optimal trade-off solutions.

Recently, a simulation program called the WholeCell [42] provided a fine description of many life cycle processes of a small bacterium (the genome is about 1/10th of that of Escherichia coli) *Mycoplasma genitalium*: metabolism, replication of the genome, and cell division (9 hours or about 32000 repetitions of the simulation loop). The WholeCell model is composed by 28 distinct modules and includes a metabolic model of 441 chemical reactions. The metabolism is analyzed using FBA, which provides the solutions (reactions fluxes and metabolite concentrations) satisfying the optimization of the most efficient use of available resources, such as nutrients. Organic compounds are converted to carbon skeletons for the synthesis of various cell components and for the production of energy.

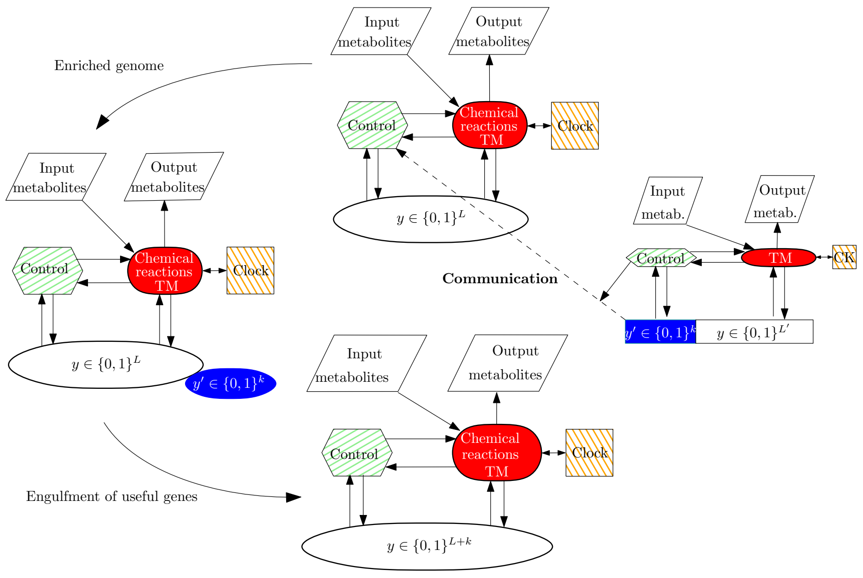

Figure 11: The communication between two bacteria. Communication between genes (top and right pictures) is useful if one bacterium needs pieces of genome from another bacterium in order to engulf them and increase its computational capability (left). This process allows a genome enriched with respect to the objectives that the bacterium aims at optimizing (bottom). In a later step, the enriched bacterium may communicate with the bacterium on the right sending other DNA fragments.

This is possible through the regulation of the reactions of anabolism and catabolism. The fitness of a bacterium is particularly concentrated on the speed of dividing, but the division time depends on reaching a certain biomass. In order to achieve a biomass, the bacterium needs to locate a source of food and move towards it. In general, we assume that the energy for the location of food source is in high demand and has higher priority than the biomass growth and division, and therefore the Pareto front would vary accordingly.

6. Gene duplication and transfer in communicating molecular machines

Gene duplication events are important sources of novel biochemical and metabolic functions. The presence of orthologous or homologous genes is often a result of the adaptation to major environmental changes. After gene duplication, mutations cause the gene copies to diverge, and the survival of the mutated gene depends on the selective pressure. Likewise, new proteins evolve from existing proteins by mutation. The classical model predicts that these mutations will generally lead to the loss of function of one gene copy; rarely, new functions will be created and both duplicate genes are conserved. Most known genes belong to large families with extensive DNA sequence similarities.

The neofunctionalization model suggests that after a gene duplicates, since the two resulting copies are functionally redundant, one can accumulate mutations leading to a new function, while the other copy remains conserved [43]. The duplication, degeneration and complementation model proposes that if the ancestral gene possesses several subfunctions, these could be alternatively distributed among its duplicate descendants by neutral mutations. Although both models assume that the duplication itself has no intrinsic advantage, clearly an increased gene dosage (i.e. increased concentrations of a protein) after duplication can be beneficial in itself and alter the robustness and sensitivity of the bio-system [44]. There is a clear relation between gene duplication and metabolic complexity. Many, if not most, enzymes can promiscuously catalyze reactions, or act on substrates, other than those for which they evolved. The process of gene duplication could lead to structural changes resulting in the modification of the infidelity of molecular recognition.

Bacterial conjugation is a genetic transfer that involves a cell-to-cell interaction between donor and recipient cells. The importance of horizontal/lateral gene transfer (LGT) in shaping the genomes of prokaryotic organisms has been recognized in recent years as a result of analysis of the increasing number of available genome sequences. LGT is largely due to the transfer and recombination activities of mobile genetic elements.

Let us consider the case that genes on some chromosome can circulate and be transmitted among interacting bacteria. (This can occur, for instance, when a bacterium tries to become immune to some antibiotic drug.) Here we try to address the question whether the computation occurring in each bacterium is affected, and to what extent. In this regard, we envisage two possible scenarios: the computation needed to process external stimuli and metabolites can be carried out either by a dedicated section of the multi-tape TM, or by the whole TM. An exchange of genes modifies the program running in each molecular machine, and consequently affects the protein production in all the bacteria involved. This leads to a change in the evolution of the bacterial network [45].

Remarkably, in a communication session between two bacteria (Figure 11), a bacterium duplicates one piece of its genome and sends it to another bacterium, which engulfs that piece of genome to increase its computational capability. Therefore, the computation in the second bacterium changes accordingly, whereas the operations taking place in the first bacterium are not affected. Indeed, in the mapping between a bacterium and a von Neumann architecture, the metabolites are mapped to the variables of the code. Since the engulfed genes were needed by the second bacterium, they are likely to be responsible for reactions involved in the production or consumption of its key metabolites (i.e. increment or decrement of internal variables). Hence, they are likely to have a significant effect on the next instructions performed by the TM, as well as on the whole metabolic network. Conversely, since the first bacterium performs a duplication that is not needed by itself, the operations associated with the gene duplication are expected to be carried out by dedicated tapes of the TM.

In the evolution process, given a sequence of genes, a subsequence of genes (e.g. an operon), or even a single gene are often duplicated and inserted somewhere else in the sequence. This process is referred to as gene amplification or gene duplication. Given an organism with L gene sets, and assuming each gene set is composed by a single gene, without loss of generality, let us denote by y the array representing the sequence of its genes. Let us assume that the duplication of the last k genes is performed:

$$y = (y_1, ..., y_L) \quad \longrightarrow \quad y = (y_1, ..., y_L, y_{L+1}, ..., y_{L+k}). \tag{4}$$

Initially, the *condition of duplication* holds, i.e. $y_l = y_{l-k}$, $\forall l = L+1, ..., L+k$. In general, the duplication is a stochastic process and the condition of duplication is not

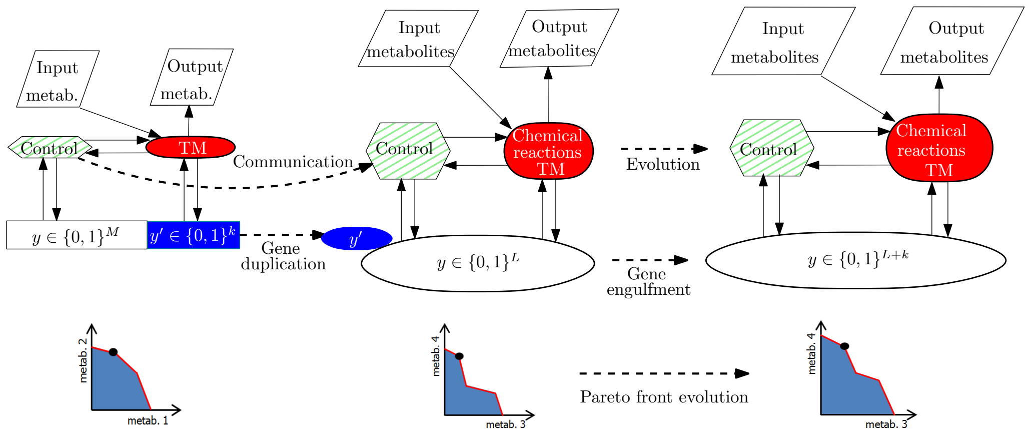

Figure 12: Interacting bacteria can be thought of as communicating von Neumann architectures (left, center). The processing unit is the metabolic network of a bacterium, which can be mapped to a TM. The string y is a program stored in the memory, while g is a function acting as a control unit that interprets the binary string y and controls the status of the gene sets. The goal is to produce desired metabolites as output of the molecular machine. Due to the communication between the two bacteria (left, center), the second bacterium engulfs a piece of genome (right) duplicated by the first bacterium. In a multi-objective optimization analysis, this results in an increment of the area underlying the Pareto front. In other words, the evolution and the growth of a bacterium can be the result of duplication and mutation events in the genome of another bacterium. This process increases the genome length, allowing more knockout strategies and causing not merely an increased complexity of the control function, but also a larger capability of intake and output production.

always true. After the duplication, the string becomes $y = (y_1, ..., y_{L'})$, where $L' = L+k$, due to the fact that mutations affect both new and existing genes. Each gene y_l codes for a reaction, say $D_i \to D_j + H_r$, and therefore for the instruction $inc(i, r, j)$ in the RM. After the duplication, both y_l and y_{l+k} will be responsible for the same reaction $D_i \to D_j + H_r$. Conversely, after the mutation, y_{l+k} will be responsible for another reaction, say $D_{i'} \to D_{j'} + H_{r'}$. As a consequence, a new reaction $inc(i', r', j')$ is now operating in the RM, namely there has been an increment in the complexity of the metabolic machine.

Each duplication followed by a mutation in an organism, defines its computational capability, i.e. the performance of the metabolic machine associated with it. This increases the number of feasible solutions in the multi-objective optimization, and therefore the area underlying the Pareto front (Figure 12). A mutation can be thought of as a stochastic process proposing a new instruction for the machine, while the natural selection, acting as a biological ratchet gear, can keep it or discard it. Consequently, the combination of stochastic processes and natural selection shapes the computational complexity of an evolving organism. In particular, the genome amplification increases the number of available chemical reactions, creating new increment and decrement instruction in the RM associated with the metabolism, thus increasing the computational power of the whole metabolic machine.

A technique for the analysis of the computational power of molecular computers able to make decisions has been proposed by Soloveichik and Winfree [46], while the methods to determine the complexity of an universal TM have been reviewed in [47]. Notably, these processes may be inserted in a network of interacting organisms where genes can be also sent to neighbors after duplication. In this scenario, a central bacterium may easily interact with many neighbors, and therefore duplicate and send several copies of a given set of genes. This leads to speculate that the increment in the computational complexity of the genome and the change in its performance depend on the position of the gene in the interaction network. After these events, the computational complexity can be computed by taking into account the propensity coefficient given to each reaction (i.e. instruction) of the molecular RM [34].

In bacterial genomes, the pathways are encoded as gene clusters termed operons [48], which are transcribed and translated together into proteins folding nearby in space, in order to assure their simultaneous availability. This condition suggests that the pathways are the computational units of the molecular machine. Many units of communicating bacteria, which share genes using LGT techniques, may also work as parallel computers, each of which runs a genetic optimization algorithm.

7. Conclusion

In this research work, we have highlighted the links between bacterial optimization, communication, gene duplication, lateral gene transfer and metabolic complexity in light of a mapping between a biological organism and the von Neumann architecture. Specifically, the metabolism executes reactions as instructions of a Turing machine. Given a Boolean string y (the executable program stored in a "programmable memory"), a control function hinges on it to control and set the knockout strategies of the bacterium. Mapping the bacterial metabolism to a Turing machine proves useful to evaluate the computational effort needed for executing a specific task, and also the overall computational capability of the bacterium.

For the search of an optimal program y, we have used Pareto optimization focusing on two genome-scale metabolic networks of *Escherichia coli.* After the concurrent maximization of biological functions, (e.g. succinate and biomass, or acetate and biomass), we have proposed the Pareto front as a means to summarize the phenotype of the organism under investigation. The shape of the front, as well as its *knees* and *jumps*, enable not merely to analyze a single organism and find its trade-off programs y, but also to perform cross-comparisons among organisms with different Pareto fronts.

Framing the bacterial metabolism as a molecular machine allows for computational analysis at a microscale/macroscale network level (e.g., biomass and production of succinate or acetate). To obtain these cellular objectives, the genetic activation/deactivation strategy is thought of as stored in a programmable memory added to the microorganism. Through the "control" level shown in Figure 10, the programmable memory can drive the generation of single-stranded DNA in response to external and internal signals, e.g. using recently developed DNA writing technologies (SCRIBE [49]). The optimal strategy needed for SCRIBE or CRISPR-Cas9 for genome engineering [50], i.e. the executable program loaded on the memory of the microorganism, can be produced by GDMO after a computational evolutionary process. This pipeline can be easily extended to find optimal overexpression/underexpression strategies.

Finally, using the idea of Pareto optimality, we have discussed the energy, motility and communication in bacteria. In a multi-objective perspective, we have analyzed the gene duplications and transfers taking place after the bacterial conjugation. This approach suggests an increasing computational capability in communicating and evolving molecular machines.

References

[1] H. Fellermann, L. Cardelli, Programming chemistry in dna-addressable bioreactors, Journal of The Royal Society Interface 11 (99) (2014) 20130987.

[2] J. S. Edwards, B. O. Palsson, The escherichia coli mg1655 in silico metabolic genotype: Its definition, characteristics, and capabilities, Proceedings of the National Academy of Sciences 97 (10) (2000) 5528–5533.

[3] J. Reed, T. Vo, C. Schilling, B. Palsson, An expanded genome-scale model of escherichia coli k-12 (ijr904 gsm/gpr), Genome Biology 4 (9) (2003) R54.

[4] A. M. Feist, C. S. Henry, J. L. Reed, M. Krummenacker, A. R. Joyce, P. D. Karp, L. J. Broadbelt, V. Hatzimanikatis, B. Ø. Palsson, A genome-scale metabolic reconstruction for escherichia coli k-12 mg1655 that accounts for 1260 orfs and thermodynamic information, Molecular Systems Biology 3 (121) (2007) 291–301.

[5] J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, B. O. Palsson, A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011, Molecular Systems Biology 7 (Article number 535) (2011) 1–9.

[6] J. Costanza, G. Carapezza, C. Angione, P. Lió, G. Nicosia, Robust design of microbial strains, Bioinformatics 28 (23) (2012) 3097–3104.

[7] M. W. Covert, N. Xiao, T. J. Chen, J. R. Karr, Integrating metabolic, transcriptional regulatory and signal transduction models in escherichia coli, Bioinformatics 24 (18) (2008) 2044–2050.

[8] C. Angione, J. Costanza, G. Carapezza, P. Lió, G. Nicosia, A design automation framework for computational bioenergetics in biological networks, Molecular BioSystems 9 (10) (2013) 2554–2564.

[9] H. Yim, R. Haselbeck, W. Niu, C. Pujol-Baxley, A. Burgard, J. Boldt, J. Khandurina, J. D. Trawick, R. E. Osterhout, R. Stephen, J. Estadilla, S. Teisan, H. B. Schreyer, S. Andrae, T. H. Yang, S. Y. Lee, M. J. Burk, S. Van Dien, Metabolic engineering of Escherichia coli for direct production of 1,4-butanediol, Nature Chemical Biology 7 (7) (2011) 445–452.

[10] G. Stracquadanio, G. Nicosia, Computational energy-based redesign of robust proteins, Computers & chemical engineering 35 (3) (2011) 464–473.

[11] V. Cutello, G. Narzisi, G. Nicosia, A multi-objective evolutionary approach to the protein structure prediction problem, Journal of the Royal Society Interface 3 (6) (2006) 139–151.

[12] R. Schuetz, N. Zamboni, M. Zampieri, M. Heinemann, U. Sauer, Multidimensional optimality of microbial metabolism, Science 336 (6081) (2012) 601–604.

[13] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, Evolutionary Computation, IEEE Transactions on 6 (2) (2002) 182–197.

[14] V. Cutello, G. Narzisi, G. Nicosia, Computational studies of peptide and protein structure prediction problems via multiobjective evolutionary algorithms, in: Multiobjective Problem Solving from Nature, Springer, 2008, pp. 93–114.

[15] M. Laumanns, L. Thiele, K. Deb, E. Zitzler, Combining convergence and diversity in evolutionary multiobjective optimization, Evol. Comput. 10 (3) (2002) 263–282.

[16] M. Morris, Factorial sampling plans for preliminary computational experiments, Technometrics 33 (2) (1991) 161–175.

[17] C. Angione, G. Carapezza, J. Costanza, P. Lió, G. Nicosia, Pareto optimality in organelle energy metabolism analysis, Computational Biology and Bioinformatics, IEEE/ACM Transactions on 10 (4) (2013) 1032–1044.

[18] M. Hafner, H. Koeppl, M. Hasler, A. Wagner, "glocal" robustness analysis and model discrimination for circadian oscillators, PLoS Comput Biol 5 (10).

[19] A. M. Turing, The chemical basis of morphogenesis, Bulletin of mathematical biology 52 (1) (1990) 153–197.

[20] D. Bray, et al., Protein molecules as computational elements in living cells, Nature 376 (6538) (1995) 307–312.

[21] B. Bryant, Chromatin computation, PloS one 7 (5) (2012) e35703.

[22] J. Von Neumann, A. W. Burks, et al., Theory of self-reproducing automata.

[23] L. Moss, Confusion of memory, Information Processing Letters 107 (3) (2008) 114–119.

[24] J. Marion, From turing machines to computer viruses, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 370 (1971) (2012) 3319–3339.

[25] L. Landweber, L. Kari, Universal molecular computation in ciliates, Evolution as Computation (2003) 257–274.

[26] L. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (5187) (1994) 1021–1024.

[27] M. Amos, Cellular Computing, Series in Systems Biology, Oxford University Press, 2004.

[28] P. Siuti, J. Yazbek, T. K. Lu, Synthetic circuits integrating logic and memory in living cells, Nature Biotechnology.

[29] C. Angione, G. Carapezza, J. Costanza, P. Lió, G. Nicosia, Computing with metabolic machines, in: A. Voronkov (Ed.), Turing-100, Vol. 10 of EPiC Series, 2012, pp. 1–15.

[30] R. Brent, J. Bruck, 2020 computing: Can computers help to explain biology?, Nature 440 (7083) (2006) 416–417.

[31] D. Soloveichik, M. Cook, E. Winfree, J. Bruck, Computation with finite stochastic chemical reaction networks, Natural Computing 7 (4) (2008) 615–633.

[32] M. Minsky, Computation, Prentice-Hall, 1967.

[33] G. Franco, P. H. Guzzi, V. Manca, T. Mazza, Mitotic oscillators as mp graphs, in: Membrane Computing, Springer, 2006, pp. 382–394.

[34] M. Cook, D. Soloveichik, E. Winfree, J. Bruck, Programmability of chemical reaction networks, Algorithmic Bioprocesses (2009) 543–584.

[35] B. Futcher, G. Latter, P. Monardo, C. McLaughlin, J. Garrels, A sampling of the yeast proteome, Molecular and Cellular Biology 19 (11) (1999) 7357–7368.

[36] A. Bar-Even, E. Noor, Y. Savir, W. Liebermeister, D. Davidi, D. Tawfik, R. Milo, The moderately efficient enzyme: evolutionary and physicochemical trends shaping enzyme parameters, Biochemistry 50 (21) (2011) 4402–4410.

[37] P.-A. Reynier, A. Sangnier, Weak time petri nets strike back!, in: CONCUR 2009-Concurrency Theory, Springer, 2009, pp. 557–571.

[38] C. P. Fisher, N. J. Plant, J. B. Moore, A. M. Kierzek, Qsspn: dynamic simulation of molecular interaction networks describing gene regulation, signalling and whole-cell metabolism in human cells, Bioinformatics 29 (24) (2013) 3181–3190.

[39] M. Heiner, D. Gilbert, R. Donaldson, Petri nets for systems and synthetic biology, in: Formal methods for computational systems biology, Springer, 2008, pp. 215–264.

[40] P. Baldan, N. Cocco, A. Marin, M. Simeoni, Petri nets for modelling metabolic pathways: a survey, Natural Computing 9 (4) (2010) 955–989.

[41] A. Doi, S. Fujita, H. Matsuno, M. Nagasaki, S. Miyano, Constructing biological pathway models with hybrid functional petri nets, In Silico Biology 4 (3) (2004) 271–291.

[42] J. Karr, J. Sanghvi, D. Macklin, M. Gutschow, J. Jacobs, B. Bolival, N. Assad-Garcia, J. Glass, M. Covert, A whole-cell computational model predicts phenotype from genotype, Cell 150 (2) (2012) 389–401.

[43] T. Sikosek, H. Chan, E. Bornberg-Bauer, Escape from adaptive conflict follows from weak functional trade-offs and mutational robustness, Proceedings of the National Academy of Sciences 109 (37) (2012) 14888–14893.

[44] T. Massingham, L. Davies, P. Lió, Analysing gene function after duplication, Bioessays 23 (10) (2001) 873–876.

[45] A. Ben-Hur, H. Siegelmann, Computation in gene networks, Chaos: An Interdisciplinary Journal of Nonlinear Science 14 (1) (2003) 145–151.

[46] D. Soloveichik, E. Winfree, The computational power of benenson automata, Theoretical Computer Science 344 (2) (2005) 279–297.

[47] D. Woods, T. Neary, The complexity of small universal turing machines: A survey, Theoretical Computer Science 410 (4) (2009) 443–450.

[48] F. Jacob, D. Perrin, C. Sánchez, J. Monod, S. Edelstein, The operon: a group of genes with expression coordinated by an operator. cr acad. sci. paris 250 (1960) 1727-1729]., Comptes rendus biologies 328 (6) (2005) 514.

[49] F. Farzadfard, T. K. Lu, Genomically encoded analog memory with precise in vivo dna writing in living cell populations, Science 346 (6211) (2014) 1256272.

[50] P. D. Hsu, E. S. Lander, F. Zhang, Development and applications of crispr-cas9 for genome engineering, Cell 157 (6) (2014) 1262–1278.