2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

An Integrated Framework for Strain Optimization

Paulo Maia*†‡, Isabel Rocha*, Miguel Rocha†
*IBB-Institute for Biotechnology and Bioengineering
†CCTC - Computer Science and Technology Center
University of Minho, Braga, Portugal
‡SilicoLife - Computational Biology Solutions for the Life Sciences
AvePark - SpinPark, Apart. 4152, 4806-909, Guimarães, Portugal
{paulo.maia,irocha}@deb.uminho.pt, mrocha@di.uminho.pt

***Abstract*—The identification of genetic modifications leading to mutant strains able to overproduce compounds of industrial interest is a challenging task in Metabolic Engineering (ME). Several methods have been proposed but, to some extent, none of them is suitable for all the specificities of each particular strain optimization problem. This work proposes an integrated framework that allows its users to configure and fine tune all the various steps involved in a strain optimization strategy, including the loading of models in distinct formats, the definition of a suitable phenotype simulation method and the choice and configuration of the strain optimization engine. Moreover, it is designed to suit the needs of users skilled at programming, as well as less advanced users. The framework includes a GUI implemented as the strain optimization plug-in for the OptFlux workbench (version 3), a reference platform for ME (http://www.optflux.org). All the code is distributed under the GPLv3 licence and it is fully available (http://sourceforge.net/projects/optflux/).**

I. Introduction

The ability to understand the genotype-phenotype relationship began to change in the mid 1990s, after the completion of the first sequenced bacterial genome. Until then, this relationship was treated mostly based on qualitative analysis. With the recent developments in genome sequencing technologies and semi-automated annotation techniques, an increasingly large number of fully annotated microbial genomes is being made available. These full genome sequences provide comprehensive information about the genetic elements that compose an organism and, when combined with the understanding of some cellular processes such as metabolism, results in structured knowledge that is mathematically representable [1]. This knowledge explosion accelerated the development of constraint-based methods for phenotype simulation based on genome-scale stoichiometric metabolic models for a large number of microbes [2], allowing the development of prediction methods for distinct genetic and environmental conditions.

The most common approach is to consider the cell to be in a pseudo steady-state and, since the solution space for the metabolic flux state of the cell is usually very large, constraint based optimization approaches are often applied for simulating metabolic fluxes. Given this assumption, it is therefore plausible to predict cellular behaviour by solving optimization problems, as long as biologically realistic objective functions are put forward. Several methods have been developed to solve these problems. Among these, Flux Balance Analysis (FBA) [3] is a widely used phenotype prediction tool that uses a linear programming (LP) formulation for maximization of growth (synthesis of biomass constituents) as the objective function, considering the biological assumption that organisms tend to maximize their growth. FBA has been applied with success in several cases [4], [5], although mostly for wild-type or unperturbed metabolic networks [6].

However, to predict the cellular behaviour of mutant organisms, such assumption is not widely accepted and, for that specific purpose, other methods have been proposed such as Regulatory On/Off Minimization of metabolic flux changes (ROOM) [7], Minimization of Metabolic Adjustment (MOMA) [8] and more recently Minimization of Metabolites Balance (MiMBl) [6]. ROOM is based on Mixed Integer-Linear Programming (MILP) and the objective function is the minimization of the number of significantly perturbed fluxes from the mutant relatively to the wild-type. As for MOMA, the principle is similar, only the formulation is based on Quadratic Programming (QP) and the objective function is the minimization of flux variations relatively to the wild-type. The hypothesis underlying both MOMA and ROOM is that fluxes in a perturbed cell (e.g. a mutant) will be redistributed in order to be as similar as possible to the wild-type [6]. MiMBl looks at the problem from a different perspective and, instead of tackling the problem by finding linear combinations of fluxes, centers its attention on the metabolite turnovers, decoupling the formulation away from the problems related with the sensitivity to the stoichiometric representations, which can hinder the predictions.

Parallel efforts in the Metabolic Engineering (ME) arena have been exploiting these advances for the purposes of strain optimization, through the development of methods able to select genetic modifications, capable of achieving an improved production of compounds of high industrial interest [9]. These novel tools aim to contribute to the competitiveness of Biotechnological processes in the production of a number of valuable products, replacing traditional chemical synthesis with reduced environmental costs. In fact, although there are still limitations in constraint-based modelling approaches, ME increasingly relies on the use of genome scale models, combined with computational algorithms to predict and optimize cell behaviour under distinct conditions, both environmental and genetic.

In ME, strain optimization tasks can be formulated as bi-level problems, adding to phenotype simulation a layer that searches for the best mutant that can be obtained by performing genetic alterations to the wild type. The most studied problem consists in discovering a subset of genes to delete to achieve strains that maximize an objective function (e.g. yield) related with the production of a target compound,

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

while keeping the host viable. Different approaches were proposed, namely OptKnock that uses MILP [10] and metaheuristics such as Evolutionary Algorithms (EAs) [11] and Simulated Annealing (SA) [12]. Both approaches search for a set of gene (reaction) deletions that maximize the flux towards a desired product, while the internal flux distribution is still operated such that growth (or another biological objective) is optimized. The formulation of OptKnock guarantees that the optimal solution, if there is one, will be found. However, this formulation also poses some problems. Since it is MILP-based, a linear objective function is mandatory, which is an important limitation; moreover, it requires that the user decides the size of the gene (reaction) deletion set and it is not scalable. This means that, for larger sets, the resolution of the problem is not computationally tractable. Both EA-based and SA-based approaches overcome these problems, although they do not guarantee that an optimal solution is found.

A problem with these optimization methods was tied to the fact that all of them were reaction-based. This meant that the variables interplaying in the optimization process represented reactions instead of the genes encoding the proteins that promoted those reactions, disregarding available transcriptional/ translational information (gene-protein-reaction (GPR) relationships) and thus generating many biologically unfeasible solutions. This limitation has been overcome in recent work by the authors' research group [13].

On the other hand, knockout-based optimization is limited by the fact that *in silico* genetic changes are restricted to gene deletions. An alternative way to optimize the production of metabolites is to up or down-regulate the expression of genes relatively to their wild-type values. OptReg [14] is a first attempt to address this task, working directly with reaction over/under expression. A reaction is considered under (or over) regulated in a mutant if it takes a flux value considerably lower (or higher) when compared with the wild-type steady-state fluxes. OptKnock is the basis for OptReg formulation, and therefore, the limitations of this approach are similar to the ones pointed out for this method, a fact that led to the development of alternative methods based in EAs [15].

Albeit all the mentioned approaches have provided good results, they are still limited, since they usually return a single solution to the problem or, in some cases, sets of solutions with limited variability. EAs are able to provide more solutions, but typically those follow similar strategies to maximize the selected objective function. There are a few reasons to prefer algorithms that can provide diverse solutions, namely the fact that, since simulation results are used to generate the best mutants, these can have poor performances *in vivo* due to pitfalls in the model or simulation tools. Furthermore, these approaches can also provide strains with different trade-offs between the distinct factors involved, namely the production yield of the desired compound, the strain viability (measured by the biomass flux) or the predicted cost of implementing the genetic modifications. Additionally, the inclusion of information from multiple criteria in a single objective function through aggregation techniques can introduce undesired bias in the sampling process. Since the complexity of many optimization problems is often difficult to represent using a single objective, the use of multi-objective optimization (MOO) is a logical path to follow.

MOO applications have been increasing, and the development of suitable algorithms has been an active area of research. As compared to single objective, in MOO there is no longer a single optimal solution but rather a set solutions of non-comparable quality. The classical definition of a MOO problem can be formally written in the form: $minimize[f_1(x), f_2(x), \ldots, f_K(x)]$ where $f_i : \Omega \rightarrow \mathcal{R}^K$ are K objective functions. The task is to determine from the set of feasible solutions, the particular subset yielding optimum values for all objective functions [16]. Yet, it is very rare to find a solution that simultaneously optimizes all objective functions. Therefore, we usually seek the optimal set of trade-offs rather than a single solution. To this optimal set of trade-off solutions we call Pareto-front. The Pareto-front is composed solely of non-dominated solutions. A solution $a \in \Omega$ is said to dominate another solution $b \in \Omega$, when $\forall i : f_i(a) \leq f_i(b)$ and $\exists i : f_i(a) < f_i(b)$, where f_i is the *i*-th objective (assuming a minimization problem without losing generality). This relation is notated as $a \prec b$, meaning that a solution a dominates another solution b, if a is not worse than b in any of the objectives and it is at least better in one.

The populational and stochastic nature of EAs enables them to find several possible members of the Pareto-front in a single run. It must be noted, however, that to properly work in multi-objective scenarios, a new class of MOEAs has been proposed which, while keeping the main principles, introduce some changes into the selection and re-insertion steps. One advantage of using EAs over other MOO methods lies in the few constraints imposed over the objective function (e.g. they do not depend on the continuity of the Pareto-front). Also, they are relatively easy to implement, are very robust and can be easily adapted to parallel computing. A recent history of the field can be found in [17].

The fact that no single method is optimal for every strain optimization problem leads to the primary objective of this work, which is the development of a framework that allows researchers to configure and fine tune strain optimization strategies tailored to their own needs. A complementary aim is to provide users with visualization tools that allow to easily analyse the results generated by those strategies. The proposed framework includes the selection of several phenotype prediction methods and several optimization algorithms capable of performing well in both single and multi-objective tasks. Moreover, the user can choose to perform reaction or gene-based optimization and for both options, the selection of the over/under expression approach or simple knockouts is also allowed. We demonstrate the usefulness of such framework with a simple case study focused on maximizing the production of succinic acid in *Escherichia coli*.

II. Framework Architecture

The architecture of the framework is segmented into 4 distinct layers. A) The model layer; B) The simulation layer; C) the optimization layer and D) the interface layer. These layers and their hierarchy are represented in Figure 1.

A. Model Layer

This layer handles all the operations related to stoichiometric metabolic models. In this context, a model is represented by

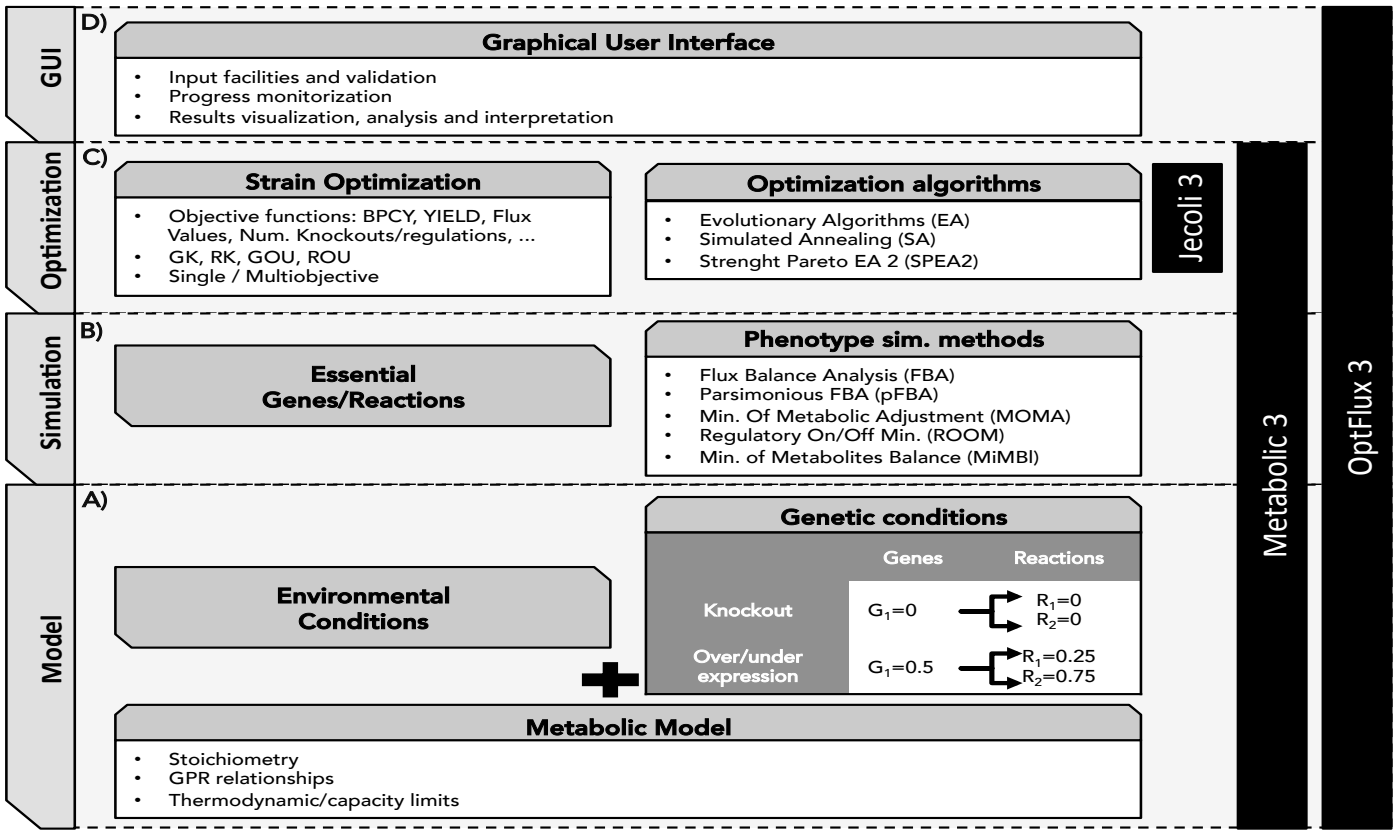

Fig. 1. The 4 layers of the framework. Black boxes represent the various libraries that compose the framework and how they relate to the various components/layers of the framework. A) The model layer represents the model and its constraints, wether they are environmental or genetic. Genetic conditions can be gene knockouts (GK), reaction knockouts (RK), gene over/under expression (GOU) or reaction over/under expression (ROU). B) The simulation layer encompasses the essentiality information computation (i.e., computation of reactions/genes without which the organism becomes unviable) and the phenotype simulation methods for both wild-type (FBA, pFBA) and mutants strains (MOMA, ROOM, MiMBl). C) The optimization layer is responsible for the formulation of the optimization problems, it allows the selection of the optimization algorithms, the selection of the optimization type (GK, RK, GOU, ROU) and the specification of the objective functions. D) The GUI layer represents the graphical user interface. This layer is supported by the OptFlux 3 application by means of a plug-in.

a stoichiometric matrix, the set of flux thermodynamic/capacity constraints and the Gene-Reaction (GR) relationship rules, as exemplifed in Figure 2.

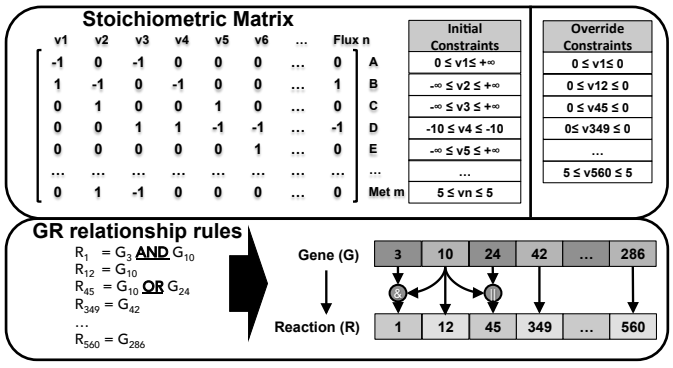

Fig. 2. The several components of a stoichiometric metabolic model.

This layer includes methods for reading and writing the models in several formats including the Systems Biology Markup Language (SBML) [18], Metatool [19], Flat-files [20] and the generic table format (csv, tsv, etc.). It also allows for the definition of environmental conditions (eg. medium components). Moreover, the definition of the genetic conditions is also supported by this layer. Genetic conditions are related to modifications in the original flux capacities of the internal stoichiometric model. These can be gene knockouts (GK), reaction knockouts (RK), gene over/under expression (GOU) or reaction over/under expression (ROU) constraints. Both the environmental and genetic conditions will act as constraints to the original model in the simulation layer.

B. Simulation layer

The simulation layer is responsible for the formulation of all the phenotype simulation problems implementing different methods for their solution. This layer allows the user to specify a simulation method and use it to predict the phenotype of both wild-type and mutant strains. It allows the prediction of the flux distributions of wild-type strains using the environmental conditions defined in the model layer. Moreover, it allows the prediction of flux distributions for mutant strains, using both the environmental and genetic conditions defined in the model layer. The currently available methods are the following: i) for wild-type strains: FBA and *parsimonious enzyme usage Flux Balance Analysis* (pFBA) [1], and ii) for mutant knockout and over/under expression phenotype prediction: FBA, pFBA, MOMA (includes its LP variant, Linear MOMA), ROOM and MiMBl.

Finally, the simulation layer also allows the computation of the essentiality information. The essentiality information comprises the set of genes/reactions that, when inactive, render the organism unviable (no biomass flux). These are computed via simulation and can be used to constrain the solution space in the optimization layer (not considering these in the optimization problem.)

C. Optimization Layer

The optimization layer is the core subject of this work. This layer allows for the definition, tuning and solving of the strain optimization problem that better suits the user requests. In the context of strain optimization, the problem is defined as: finding the optimal set of genes (or reactions) to inactivate (or over/under express) that optimize a set of objective functions. These objective functions are the most important user input, in the sense that they identify what are the optimization targets/objectives. An example is the maximization of the flux of a given reaction (over-production of a compound of interest). Several objective functions are readily available:

- *Biomass-Product Coupled Yield (BPCY) [11]*: This objective function couples (via multiplication) the biomass flux (B) with the selected product flux (P) relative to the substrate (S) consumption: $bpcy(solution) = \frac{B \times P}{S}$;

- *Product Yield with minimum biomass (YIELD)*: Computes the yield of the desired product flux (P) relative to the substrate consumption (S) ensuring that a minimum percentage of biomass (relative to the wild-type) is produced: $yield(solution) = \frac{P}{S}, B_{mut} \geq B_{wt}$;

- *Flux value*: Maximize or minimize the value of a specified reaction flux;

- *Number of knockouts (or over/under expressions)*: Maximize or minimize the number of modifications performed to the mutant strain;

- *Sum of flux measures*: Maximize or minimize the sum of all the fluxes in the flux distribution (metabolic activity).

The layer supports the selection of multiple objective functions to be optimized in parallel. In fact, even when a single objective (SO) algorithm is selected using multiple objective functions, the framework still follows the multi-objective premisses. This is accomplished by means of the archive manager, a feature that manages the solutions reached by the algorithms according to specific criteria (refer to Section III-A for more details). The optimization layer also allows the selection of environmental conditions to use in the optimization problems, as well as a set of genes/reactions that should not be considered in the optimization procedure. Regarding the optimization algorithms, 3 alternatives are provided:

- *Evolutionary Algorithms (EAs)* : EAs [21] are a popular family of metaheuristic optimization methods, inspired in biological evolution through natural selection. They work by evolving a set of individuals encoding solutions to a given problem. Each individual is evaluated via a fitness function that assigns a numerical value to it, depending on the quality of the solution. New solutions are created using reproduction operators over selected parents taken probabilistically from the current population;

- *Simulated Annealing (SA)*: SA [22] works in a distinct fashion, keeping just one solution. This is used to create new solutions through the application of a mutation operator. Each time a new solution is created, it replaces the original one, if it is better or, in some cases, even when it is worse. The probability of accepting worse solutions depends on the difference between the fitness values and on the current temperature, a variable that is reduced as the SA iterates.

- *Strength Pareto Evolutionary Algorithm 2 (SPEA2)*: SPEA2 [23] is a MOEA, being an evolution of the SPEA [24] algorithm. SPEA uses an external archive that contains non-dominated solutions (called the external non-dominated set). At each generation, non-dominated individuals are copied to this external non-dominated set. For each individual in the archive, a strength value, proportional to the number of solutions it dominates, is computed. The fitness of each individual in the current population is computed according to the strengths of all external individuals that dominate it. This strategy is used to promote the convergence of the algorithm. The fact that the external non-dominated set is used in the selection process brings the problem that, if that set grows too much, the selection pressure might be reduced, thus slowing down the global search process. To prevent this, a clustering technique called "average linkage method" was adopted to prune the external non-dominated set, thus maintaining diversity. The main improvements of SPEA2 include: (i) a better fitness assignment strategy taking into account the number of individuals that each individual dominates/ is dominated by; (ii) the use of a nearest neighbour density estimation technique to guide the search more efficiently; and, (iii) a truncation technique to prevent the loss of boundary solutions.

D. Graphical User Interface (GUI) layer

The GUI layer is implemented in the OptFlux application [20] via the Optimization plug-in. This layer provides a facility to the less programming-oriented users in which they can use all the functionalities of the framework. The GUI layer provides dialogs for setting and fine-tuning all the optimization parameters and objective functions, in a simplified/user-friendly manner. Also, during the execution of the optimization procedures, the user is provided with an interface to monitor the progress of the procedure. This interface also allows stopping the execution at any time, processing and returning the solutions found thus far. Upon successful conclusion, the results are presented in the OptFlux clipboard tree, and the user can analyze each solution individually.

III. Development

The framework was fully developed in the Java programming language. The four layers of the framework (depicted in Figure 2) are supported by three distinct software libraries, described below.

A. Jecoli 3

The Java Evolutionary Computation Library (JECoLi 3) [25] is a generic-purpose evolutionary computation library. It is an adaptable, flexible, extensible and reliable software framework implementing metaheuristic optimization algorithms, using the Java programming language. In its current version, JECoLi aims to offer a solution suited for the integration of EC-based approaches in larger applications, as well as a way to

the rapid and efficient benchmarking of EC algorithms in specific problems taking full advantage of the available hardware. JECoLi 3 allows to encode solutions using different types of representations such as: binary, integer, real, permutations, sets and trees. JECoLi also provides a panoply of almost 40 reproduction operators (crossover and mutation) as well as several distinct selection operators implemented for EAs. A number of pre-processing schemes applicable to the selection operators are also made available (e.g. scaling, ranking). Moreover, several metaheuristic algorithms are also provided: General purpose Evolutionary Algorithms, including Genetic Algorithms and Evolutionary Programming; Differential Evolution; Genetic Programming and Linear GP; Simulated Annealing; Cellular Automata GAs [1] and multi-objective optimization Evolutionary Algorithms (NSGA II and SPEA2). The library also includes an Application Programming Interface (API) that allows all its components to be easily extended. Some important features and components of the library are highlighted next.

a) ElementSetRepresentation: The representation used to encode the solutions for gene or reaction knockouts (GK, RK) problems. This is a set based representation that does not allow repetitions and guarantees the constant order of its elements. Each element in our problems represents a given gene or reaction in the model that will be deleted. This is represented in Figure 3 a).

b) HybridSetRepresentation: The representation used to encode the solutions generated for gene or reaction over/under expression (GOU, ROU) problems. It is an extension of the Element Set Representation, but provides an extra field per element that allows a relative expression level to be set. In our problems, each element is composed of an Integer, representing a gene/reaction in the model, and a Float representing the level of over/under expression relatively to the original expression level as depicted in Figure 3 b).

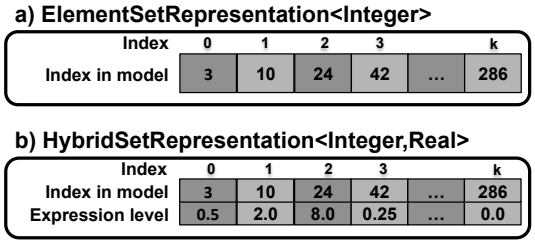

Fig. 3. The two representations used in the framework.

c) Archive Manager: The Archive Manager runs in parallel with the optimization algorithms and it is notified by the evaluation function of the algorithms. Each time a new solution is evaluated, the archive is notified and decides whether or not to keep that solution. Several operations are performed by the Archive Manager to maximize the number of interesting solutions that are kept. The user is allowed to specify the maximum number of solutions to keep, as well as the strategies to keep or remove solutions. When multiple objective functions are selected, the archive manager uses the the Zitzler truncation clustering method [23] to prune the archive to the maximum size allowed.

B. Metabolic 3

The Metabolic 3 library is the ”brain” of the framework. It is responsible for the interaction with the models in their various formats, formulation of all the previously mentioned phenotype simulation methods, the computation of the essential genes/reactions and the strain optimization procedures. The Metabolic 3 library includes an API that allows programming-oriented users to take advantage of all the provided facilities. The library is segmented in several modules, from which the 5 most relevant ones are detailed next:

-metabolic.criticality: provides facilities to compute sets of critical genes and reactions;
-metabolic.model: provides data structures to support all the information pertinent to stoichiometric metabolic models. The read/write support for the several file formats is provided by an extra library named BioComponents (also available in the sourceforge of the project);
-metabolic.simplification: includes the implementation of the model simplification procedures;
-metabolic.simulation: provides the implementation of all the phenotype prediction methods. It also includes the *SteadyStateSimulationControlCenter*, an API class that serves as the control center for every phenotype simulation process available in the library;
-metabolic.optimization: The optimization module is the core software piece of this work. It provides the implementation of all the optimization procedures of the framework. An API class named *StrainOptimizationControlCenter* allows to programmatically access all the features described in this work in a swift manner. This class provides API methods that allow the configuration of every previously described feature, including some details that are not visible in the GUI layer. Moreover, it also provides an abstraction that permits the selection of several LP/ MILP solvers (GLPK, CLP, CPLEX, GUROBI) via another library named Solvers2 (also in sourceforge).

C. OptFlux 3

The OptFlux 3 optimization plug-in provides a rational Graphical User Interface (GUI) for the described framework. The configuration dialog of the strain optimization operation allows the process to be configured and launched. It is depicted in Figure 4, providing the following options:

-Select Simulation method. The method used to perform the phenotype simulation: FBA, pFBA, MOMA, LinearMOMA, ROOM and MiMBl are available.
-Select Environmental Conditions. The list of available environmental conditions for this project.
-Select the objective functions. OptFlux supports multiple-criteria optimisation; this means that the user can select several objective functions to optimize at the same time, among the following: BPCY, YIELD, Max/Min of reaction flux value, Max/Min of the number of knockouts and Max/Min of the sum of flux measures. The user can configure each objective function individually and add them to the list (on the right).
-Select Optimization Algorithm. The optimization method to use, selected from the available ones: EA, SA and SPEA2. SPEA2 is natively multi-objective, thus prone to provide better solutions when multiple objective functions are used.
-Knockout Type. The user can select reaction (R) or gene

(G) based optimization, as well as simple knockouts (K) or the over/under expression (OU) approach.

-Optimization Basic Setup. *Maximum Number of Solution Evaluations* - The number of simulations allowed for this optimization. Increasing this parameter will also increase the chances of finding optimal solutions, but the optimization process will take more time to complete. *Maximum Number of Knockouts* - The maximum number of deletions/regulations allowed. *Variable size solution* - When selected, the size of the solutions is not fixed and can vary up to the maximum previously configured.

-Essential Information. - Depending on whether this is a Reaction of Gene optimization, the user can select a set of reactions/genes that must never be considered for deletion. This set can be computed automatically by OptFlux.

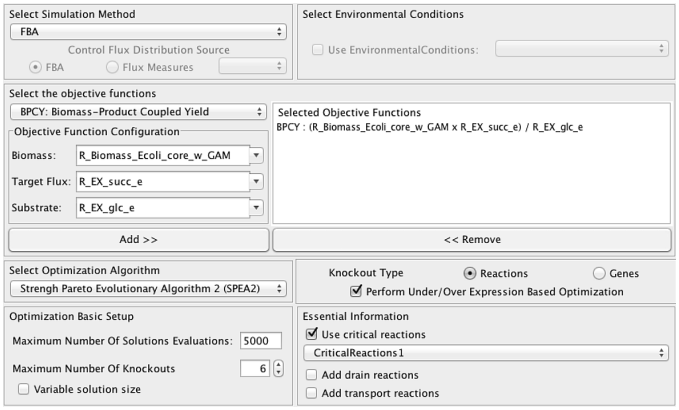

Fig. 4. The strain optimization configuration dialog.

IV. Case Study

To illustrate the framework capabilities we selected a case study, that considers the bacterium *Escherichia coli* in the production of succinate. This case study was designed to demonstrate the versatility of the framework while, at the same time, display the different outcomes that the various strain optimization strategies may originate. This will illustrate our claim, that no single strain optimization strategy meets every purpose, and that a framework that allows a flexible tuning of those strategies is a big surplus for the ME community.

Succinate is one of the key intermediates in cellular metabolism and therefore an important case study for ME (Lee et al., 2002). Succinate and its derivatives have been used to synthesize polymers, as additives and flavouring agents in foods, supplements for pharmaceuticals, or surfactants. Currently, succinate is produced through petrochemical processes that can be expensive and have significant environmental impacts. The knockout solutions that lead to an improved production are not straightforward to identify since they often involve a large number of interacting reactions [26]. *E. coli* has many advantageous characteristics as a production host, such as rapid growth and simple nutritional requirements.

The *E. coli* genome-scale metabolic model used in this work was developed by [27]. This model includes a total of N=1075 reactions, M=761 metabolites and G=904 genes. To reduce the search space, some simplification operations were conducted, described in detail in [12]. After the simplification operations, the model was reduced to a size of N=549 reactions, M=332 metabolites and G=505 genes. Moreover, sets of essential reactions and essential genes have been identified, removing them from the variables for optimization. This left the problems with 332 variables for the the RK optimization strategy and 403 variables for the GK optimization strategy.

In order to demonstrate the usefulness of such framework, a batch of tests was designed, comprising several optimization strategies and comparing two phenotype prediction methods. EA, SA and SPEA2 were tested in the four optimization strategies discussed (GK, GOU, RK, ROU) and for each, both pFBA and MOMA were used to predict the phenotypes. For every algorithm two objective functions were imposed: $OF1 = Max\ Biomass$ and $OF2 = Max\ Succinate$. It is worth mentioning that MO optimization was possible for the SO algorithms (EA, SA) due to an abstraction provided by the framework via the archive manager. The solutions are kept following the premisses of the MO strategies, but the evolution of the algorithms is not affected by this. In order to compute a fitness value for these SO approaches, an aggregation function is automatically applied ($Fit = OF1 \times OF2$). The framework provides two different aggregation functions (sum and multiplication) and allows to define new ones by implementing a simple interface.

The metric chosen for performance comparison was the previously detailed Biomass-Product Coupled Yield (BPCY). This metric is easy to compute for both SO and MO algorithms, and allows direct comparison with results from other works [12]. For each combination, 20 trials were performed and each solution was simplified (removed all the knockouts or over/under regulations that did not contribute for the objective functions). Moreover, for each combination, the results from the 20 trials were aggregated into solution supersets where non-better and repeated solutions were removed. The number of solutions in each set is detailed in Table I.

TABLE I. Number of solutions in each solution superset.

		EA	SA	SPEA2
GK	pFBA	583	825	1011
	MOMA	1985	1992	1999
GOU	pFBA	1164	1407	1211
	MOMA	1600	1699	1836
RK	pFBA	1066	1530	1532
	MOMA	1992	1990	1998
ROU	pFBA	714	792	1062
	MOMA	1666	1663	1943

The results presented in Table I allow to infer, as a first conclusion, that shifting the phenotype prediction method, results in a substantially different number of unique solutions generated. When MOMA was used as the phenotype prediction method, the number of unique solutions in the aggregated sets suffered an increase of almost two fold in some cases. When only the optimization algorithm or the optimization strategy are swaped, the results do not present the same variability. For each trial, the best BPCY and the size of the best solution was also computed and are presented in Table II.

A careful examination of Table II and the boxplots in Figure 5 shows that, albeit using MOMA as a phenotype prediction method produces a larger set of alternative solutions, these solutions are also generally larger in size. In this work, the size of the solutions was constrained to a maximum of 20 (knockouts or over/under regulations). The best solutions found

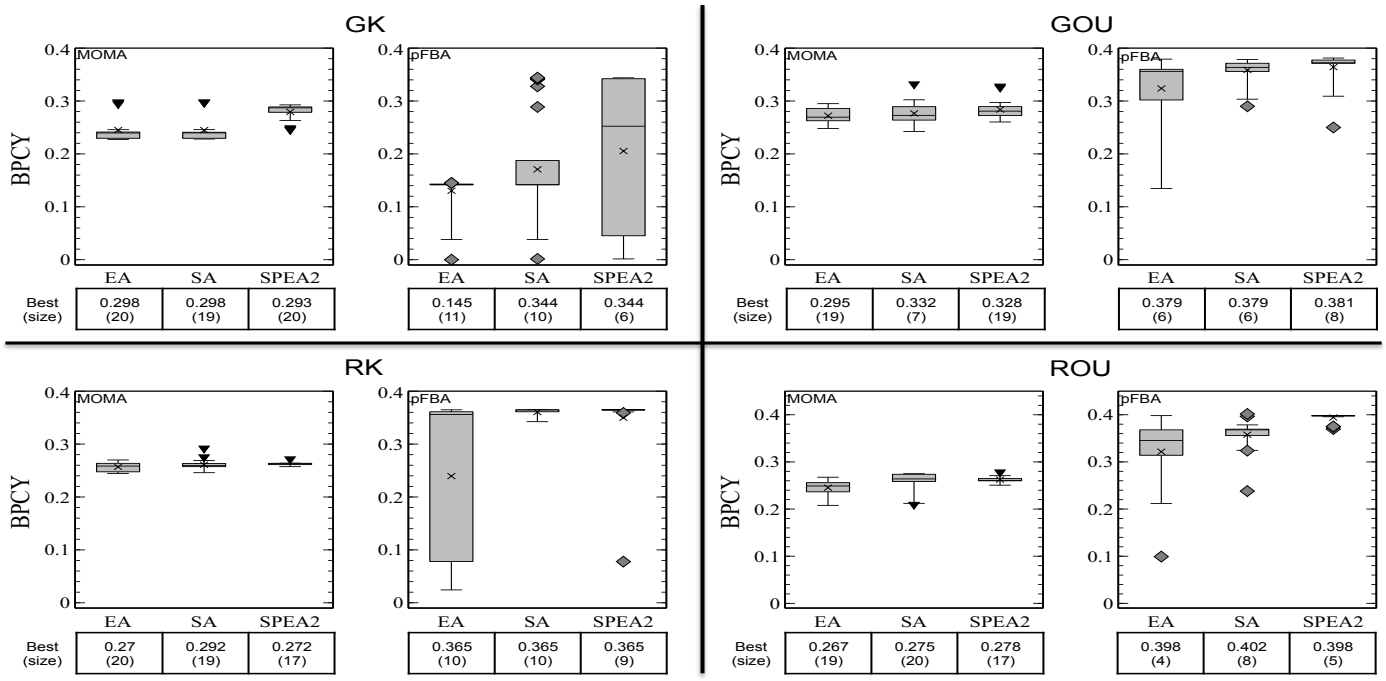

Fig. 5. Boxplots of the best BPCYs found for each run. All the optimization strategies were contemplated and for each, the two phenotype prediction methods were tested. The tables display the best BPCYs found in all runs of that specific combination and its respective size (in brackets).

by MOMA were constantly close to that limit. Exchanging the optimization algorithm translated into less visible changes, even tough the SPEA2 algorithm (with some exceptions), tends to generate better solutions that are smaller in size.

Changing from knockout-based optimization strategies to over/under expression ones (GK $\rightarrow$ GOU and RK $\rightarrow$ ROU), allowed better solutions to be found when the phenotype simulation method was pFBA, but had almost no influence in the results when MOMA was preferred. Also, when observing the effects of changing the optimization targets from genes to reactions (GK $\rightarrow$ RK and GOU $\rightarrow$ ROU), it is possible to observe that, when pFBA was used, changing from GK to RK allowed better solutions to be found but those solutions were larger on average (probably due to genes encoding for multiple reactions). Once again, MOMA-based results did not seem to be significantly affected. Comparing GOU to ROU did not reveal significant differences, suggesting that knockout-based strategies impose harder constraints to the problems, and that some solutions that are probably not viable (do not allow biomass flux) in the knockout-based strategies, become viable by under/over regulations of some of their elements.

TABLE II. AVERAGE BPCYS OF THE BEST SOLUTIONS AND 95% CIS. AVERAGE BEST SOLUTION SIZES ARE PRESENTED IN BRACKETS.

		EA	SA	SPEA2
GK	pFBA	0.1307 ± 0.01 (7.1)	0.171 ± 0.046 (5.9)	0.244 ± 0.047 (4.8)
	MOMA	0.245 ± 0.01 (19.8)	0.245 ± 0.01 (19.1)	0.279 ± 0.007 (18.8)
GOU	pFBA	0.324 ± 0.028 (4.7))	0.359 ± 0.01 (4.7)	0.364 ± 0.015 (5.3)
	MOMA	0.272 ± 0.007 (18.9)	0.277 ± 0.009 (18.1)	0.284 ± 0.008 (17.1)
RK	pFBA	0.239 ± 0.062 (7.2)	0.361 ± 0.004 (9.6)	0.35 ± 0.028 (7.8)
	MOMA	0.257 ± 0.004 (19.9)	0.261 ± 0.004 (19.5)	0.263 ± 0.001 (18.1)
ROU	pFBA	0.322 ± 0.033 (5.2)	0.358 ± 0.015 (5.7)	0.394 ± 0.004 (5.8)
	MOMA	0.245 ± 0.007 (19.4)	0.257 ± 0.009 (19.3)	0.262 ± 0.002 (17.6)

Another analysis methodology was also devised, in order to evaluate the feasibility of some solutions under distinct phenotype evaluation methods. A solution is considered feasible if its BPCY is greater than zero. The solutions generated by the EA optimization algorithm, spanning all the four optimization strategies (GK, GOU, RK, ROU) were used in this test. For each optimization strategy, the solutions generated by the EA when MOMA was the phenotype simulation method were simulated using pFBA and vice-versa. For each case, a Venn diagram was generated, displaying the solutions that were only feasible under MOMA, pFBA or were feasible for both methods. The results are presented in Figure 6.

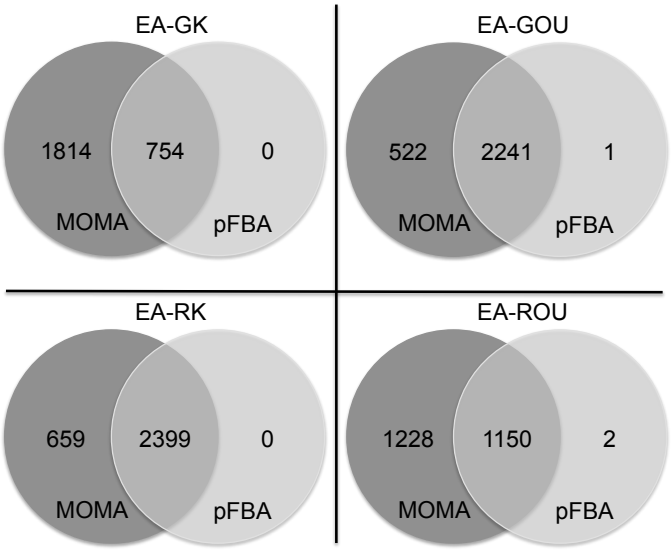

Fig. 6. Venn diagrams for the EA algorithm.

The first observable fact is that MOMA is able to achieve feasability for almost all the solutions, including the ones that were generated by optimization strategies that used pFBA

as the phenotype simulation method. On the other hand, pFBA does not have almost any exclusive feasible solutions. Comparing EA-GK to EA-GOU solutions, it is possible to observe a major increase in the number of solutions that pFBA is able to solve, but that observation is not possible when comparing EA-RK to EA-ROU solutions.

V. Conclusions

In the Metabolic Engineering arena, the possibility of selecting the appropriate genetic modifications to a given microorganism promises revolutionary advances in several other fields. However, either due to lack of information or incorrect assumptions, the uncertainty of the results provided by the current approaches renders some of the solutions not viable. Moreover, no strain optimization method developed to date is able to satisfy every particular problem that arises. This work proposes an integrated framework that allows its users to configure and fine tune strain optimization strategies tailored to their specific needs. The framework is segmented in various layers, that allow both programmers and less advanced users to take full advantage of all the methods provided, and also to extend it with new ones. The integration of this framework in the open-source OptFlux platform, makes it even more attractive to an ever increasing ME community.

Future work contemplates the generalization of the optimization layer to support other non-evolutionary optimization methods, as well as the improvement of the programming interfaces making it easier to extend the framework with third-party software.

Acknowledgment

This work is partially funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within projects ref. COMPETE FCOMP-01-0124-FEDER-015079 and PTDC/EBB-EBI/104235/2008. This work is also funded by National Funds through the FCT within project PEst-OE/EEI/UI0752/2011. The work of PM was supported by the FCT through the Ph.D. grant SFRH/BD/61465/2009.

References

[1] N. E. Lewis, H. Nagarajan, and B. O. Palsson, "Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods," *Nat. Rev. Microbiol.*, vol. 10, no. 4, pp. 291–305, 2012.

[2] C. Henry, M. DeJongh, A. Best, P. Frybarger, B. Linsay, and R. Stevens, "High-throughput generation, optimization and analysis of genome-scale metabolic models," *Nat. biotec.*, vol. 28, no. 9, pp. 977–982, 2010.

[3] K. Kauffman, P. Prakash, and J. Edwards, "Advances in flux balance analysis," *Current Op. Biotec.*, vol. 14, no. 5, pp. 491–496, 2003.

[4] R. U. Ibarra, J. S. Edwards, and B. O. Palsson, "Escherichia coli k-12 undergoes adaptive evolution to achieve in silico predicted optimal growth," *Nature*, vol. 420, no. 6912, pp. 186–189, 2002.

[5] I. Famili, J. Förster, J. Nielsen, and B. O. Palsson, "Saccharomyces cerevisiae phenotypes can be predicted by using constraint-based analysis of a genome-scale reconstructed metabolic network," *PNAS*, vol. 100, no. 23, pp. 13 134–13 139, 2003.

[6] A. R. Brochado, S. Andrejev, C. D. Maranas, and K. R. Patil, "Impact of stoichiometry representation on simulation of genotype-phenotype relationships in metabolic networks," *PLoS computational biology*, vol. 8, no. 11, p. e1002758, 2012.

[7] T. Shlomi, O. Berkman, and E. Ruppin, "Regulatory on/off minimization of metabolic flux changes after genetic perturbations," *PNAS*, vol. 102, no. 21, p. 7695, 2005.

[8] D. Segre, D. Vitkup, and G. Church, "Analysis of optimality in natural and perturbed metabolic networks," *PNAS*, vol. 99, no. 23, p. 15112, 2002.

[9] J. Nielsen, "Metabolic engineering," *Applied Microbiology and Biotechnology*, vol. 55, no. 3, pp. 263–283, 2001.

[10] A. Burgard, P. Pharkya, and C. Maranas, "Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization," *Biotechnology and bioengineering*, vol. 84, no. 6, pp. 647–657, 2003.

[11] K. Patil, I. Rocha, J. Förster, and J. Nielsen, "Evolutionary programming as a platform for in silico metabolic engineering," *BMC bioinformatics*, vol. 6, no. 1, p. 308, 2005.

[12] M. Rocha, P. Maia, R. Mendes, J. Pinto, E. Ferreira, J. Nielsen, K. Patil, and I. Rocha, "Natural computation meta-heuristics for the in silico optimization of microbial strains," *BMC bioinformatics*, vol. 9, no. 1, p. 499, 2008.

[13] P. Vilaça, P. Maia, I. Rocha, and M. Rocha, "Metaheuristics for strain optimization using transcriptional information enriched metabolic models," in *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. Springer, 2010, pp. 205–216.

[14] P. Pharkya and C. D. Maranas, "An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems," *Metabolic engineering*, vol. 8, no. 1, pp. 1–13, 2006.

[15] E. Gonçalves, R. Pereira, I. Rocha, and M. Rocha, "Optimization approaches for the in silico discovery of optimal targets for gene over/underexpression," *Journal of Computational Biology*, vol. 19, no. 2, pp. 102–114, 2012.

[16] C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information systems*, vol. 1, no. 3, pp. 129–156, 1999.

[17] K. Deb, *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001, vol. 16.

[18] 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 *et al.*, "The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.

[19] T. Pfeiffer, F. Montero, S. Schuster *et al.*, "Metatool: for studying metabolic networks." *Bioinformatics*, vol. 15, no. 3, pp. 251–257, 1999.

[20] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. Pinto, J. Nielsen, K. Patil, E. Ferreira, and M. Rocha, "Optflux: an open-source software platform for in silico metabolic engineering," *BMC systems biology*, vol. 4, no. 1, p. 45, 2010.

[21] Z. Michalewicz, *Genetic algorithms+ data structures*. Springer, 1996.

[22] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[23] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *EUROGEN 2001*, K. Giannakoglou *et al.*, Eds. CIMNE, 2002, pp. 95–100.

[24] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[25] P. Evangelista, P. Maia, and M. Rocha, "Implementing metaheuristic optimization algorithms with jecoli," in *ISDA'09*. IEEE, 2009, pp. 505–510.

[26] H. Lin, G. Bennett, and K. San, "Genetic reconstruction of the aerobic central metabolism in escherichia coli for the absolute aerobic production of succinate," *Biotechnology and bioengineering*, vol. 89, no. 2, pp. 148–156, 2005.

[27] J. L. Reed, T. D. Vo, C. H. Schilling, B. O. Palsson *et al.*, "An expanded genome-scale model of escherichia coli k-12 (ijr904 gsm/gpr)," *Genome Biol*, vol. 4, no. 9, p. R54, 2003.