BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 12 2012, pages 1619–1623
doi:10.1093/bioinformatics/bts255

Systems biology

Advance Access publication April 28, 2012

# Truncated branch and bound achieves efficient constraint-based genetic design

Dennis Egen[1] and Desmond S. Lun[1,2,*]

[1]Center for Computational and Integrative Biology and Department of Computer Science, Rutgers University, Camden, NJ 08102, USA and [2]Phenomics and Bioinformatics Research Centre and School of Mathematics and Statistics, University of South Australia, Mawson Lakes, SA 5095, Australia

Associate Editor: Olga Troyanskaya

**ABSTRACT**

**Motivation:** Computer-aided genetic design is a promising approach to a core problem of metabolic engineering—that of identifying genetic manipulation strategies that result in engineered strains with favorable product accumulation. This approach has proved to be effective for organisms including *Escherichia coli* and *Saccharomyces cerevisiae*, allowing for rapid, rational design of engineered strains. Finding optimal genetic manipulation strategies, however, is a complex computational problem in which running time grows exponentially with the number of manipulations (i.e. knockouts, knock-ins or regulation changes) in the strategy. Thus, computer-aided gene identification has to date been limited in the complexity or optimality of the strategies it finds or in the size and level of detail of the metabolic networks under consideration.

**Results:** Here, we present an efficient computational solution to the gene identification problem. Our approach significantly outperforms previous approaches—in seconds or minutes, we find strategies that previously required running times of days or more.

**Availability and implementation:** GDBB is implemented using MATLAB and is freely available for non-profit use at http://crab.rutgers.edu/~dslun/gdbb.

**Contact:** dslun@rutgers.edu

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

Received on November 18, 2011; revised on April 11, 2012; accepted on April 25, 2012

## 1 INTRODUCTION

Computational methods have been developed that allow the genomes of organisms to be designed for metabolic engineering purposes. These methods, which include OptKnock (Burgard *et al.*, 2003), OptReg (Pharkya and Maranas, 2006), OptStrain (Pharkya *et al.*, 2004), OptFlux (Rocha *et al.*, 2010) and Genetic Design through Local Search (GDLS) (Lun *et al.*, 2009), have proved to be effective for the metabolic engineering of *Escherichia coli* and *Saccharomyces cerevisiae* (Lun *et al.*, 2009; Tyo *et al.*, 2010), and the same approach can be applied to other organisms. Such a computer-based approach to metabolic engineering facilitates rapid, effective engineering and, consequently, efficient computational design techniques have significant implications for many applications including biofuel and drug production.

The computational methods we describe are based on constraint-based models of metabolism that predict the production of desired compounds under genetic manipulations. Constraint-based models can achieve a high level of accuracy in their predictions, and over 50 organism-specific genome-scale models have been developed and used in various applications to date. A comprehensive review of the applications of constraint-based modeling, including a table listing existing organism-specific genome-scale models, has been conducted by Milne *et al.* (2009). Constraint-based modeling is based on the stoichiometric matrix, or *S* matrix, which represents the metabolic network within a cell (Vemuri and Aristidou, 2005). More accurately, the *S* matrix is comprised of the stoichiometric coefficients of all metabolic reactions within a cell. Given the *S* matrix, methods such as flux balance analysis (FBA; Orth *et al.*, 2010), minimization of metabolic adjustment (Segrè *et al.*, 2002) and regulatory on/off minimization (Shlomi *et al.*, 2005) allow the effect of genetic manipulations (such as gene deletions or knockouts) to be predicted. Computer design of genomes for metabolic engineering uses *in silico* predictions to identify genetic manipulations that result in favorable metabolic phenotypes.

Although obtaining *in silico* predictions is much faster than performing genetic manipulations on the actual organism, the space of all possible genetic manipulations is nevertheless vast. Even if we only consider gene knockouts, the number of possible gene knockout configurations (i.e. specific sets of gene knockouts) increases exponentially with the total number of genes we allow to be knocked out. Thus, computational methods seeking to find an optimal set of knockouts generally have running times that increase exponentially with the number of allowed knockouts, which limits optimal genomic designs found *in silico* to a small number of knockouts (Burgard *et al.*, 2003; Lun *et al.*, 2009).

Moreover, with ever more biological knowledge being created and curated, the size and accuracy of constraint-based models is increasing, which causes an associated increase in the computational complexity of modeling and genomic design. As an example of the pace of improvement, the model we use in this study, the *i*AF1260 model of *E.coli* (Feist *et al.*, 2007), is nearly twice of the size of the previous *i*JR904 model (Reed *et al.*, 2003), published only 4 years prior. In those 4 years, the number of metabolic reactions modeled increased from 747 to 1387—a near doubling. Given the pace at which model size has grown for *E.coli* over

*To whom correspondence should be addressed.

the past 20 years (Feist and Palsson, 2008) and the application of constraint-based modeling to organisms beyond *E.coli* (Milne *et al.*, 2009)—many with larger genomes—it seems evident that models of the size of *i*AF1260 and larger will become standard in future.

It has therefore become imperative that efficient computational methods for finding genetic manipulation strategies that overcome the exponential scaling of manipulation configurations are developed, and a significant body of work has been devoted to this goal (Boghigian *et al.*, 2010; Kim *et al.*, 2011; Lun *et al.*, 2009; Melzer *et al.*, 2009; Patil *et al.*, 2005; Rocha *et al.*, 2008; Sendín *et al.*, 2010; Song *et al.*, 2011; Trinh *et al.*, 2009; Yang *et al.*, 2010). Here, we present a simple, elegant solution to the problem. Our approach, which we call Genetic Design through Branch and Bound (GDBB) uses a truncated branch and branch algorithm to tackle the bilevel optimization framework used in OptKnock (Burgard *et al.*, 2003), OptReg (Pharkya and Maranas, 2006), GDLS (Lun *et al.*, 2009) and elsewhere.

Truncated branch and bound is an adaption of the branch and bound algorithm. Branch and bound is a general algorithm for finding optimal solutions to various optimization problems. Truncated branch and bound stops processing before the algorithm determines that a solution is optimal and stops at a feasible near-optimal solution considered sufficient for practical purposes (Zhang, 1993). Truncated branch and bound can be applied to any combinatorial optimization problem, and it has been previously shown to perform well on difficult optimization problems, such as the asymmetric traveling salesman problem (Zhang, 1993) and project scheduling problems (Franck *et al.*, 2001).

We show that GDBB, using this heuristic approach, vastly outperforms previous approaches, finding near-optimal solutions in seconds to minutes that otherwise would require days or more, thus representing a significant, enabling advance in computer design for metabolic engineering.

GDBB can be applied to find near-optimal gene knockout strategies using the bilevel optimization framework used in OptKnock (Burgard *et al.*, 2003) or near-optimal up- and down-regulation strategies using the bilevel optimization framework used in OptReg (Pharkya and Maranas, 2006). More generally, it can be applied to find near-optimal genetic manipulations strategies for any setup for which there exists a bilevel optimization framework that can be converted in a single-level mixed-integer linear program (MILP). Linear programming (LP) is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. An MILP problem is a LP problem where some of the unknown variables are required to be integers. In general, MILP problems are NP-hard (Bertsimas and Tsitsiklis, 1997). Transforming bilevel optimization of FBA models to single level MILP problems (Burgard *et al.*, 2003; Pharkya and Maranas, 2006; Pharkya *et al.*, 2004) has resulted in computational methods that efficiently search the space of genetic manipulations. This approach is much more efficient than exhaustive, brute-force search, but it is nevertheless very computationally intensive. To tackle the MILP, GDBB uses a truncated branch and branch algorithm. In this article, we show that truncated branch and bound performs extraordinarily well on bilevel optimization problems associated with genomic design. For simplicity, we focus exclusively on knockouts in this article, i.e. on finding knockout strategies that lead to favorable metabolic phenotypes.

## 2 METHODS

### 2.1 Genome-scale FBA modeling

We work with the genome-scale model of *E.coli*, *i*AF1260. This model consists of three parts. First, from $m$ metabolites and $n$ reactions, we form an $m\times n$ stoichiometric matrix $S$, whose $ij$-th element $S_{ij}$ is the stoichiometric coefficient of metabolite $i$ in reaction $j$. Second, the flux distribution $v$, whose $j$-th element $v_j$ is the flux through reaction $j$, is constrained by a lower bound vector $a$, whose $j$th element $a_j$ is the lower bound of the flux through reaction $j$ (which may be negative infinity if there is no flux lower bound) and an upper bound vector $b$, whose $j$-th element $b_j$ is the upper bound of the flux through reaction $j$ (which may be positive infinity if there is no flux upper bound). Finally, a linear objective is formed by multiplying the fluxes by an objective vector $f$, whose $j$-th element $f_j$ is the weight of reaction $j$ in the biological objective.

In the analysis of biological reaction networks, we are typically interested in steady-state flux distributions that can be maintained over a period of time. Steady-state flux distributions $v$ in the biological network specified by the stoichiometric matrix $S$ satisfy the constraint $Sv=0$. For typical biological networks, the number of reactions $n$ is greater than the number of metabolites $m$ resulting in a plurality of feasible steady-state flux distributions (Palsson, 2006). The assumption of a biological objective allows specific, biologically optimal steady-state flux distributions to be found by LP, namely by solving

$$\begin{array}{rl} \max & f'v \\ \text{subject to} & Sv=0, \\ & a\le v\le b. \end{array} \tag{1}$$

### 2.2 Truncated branch and bound for finding knockout strategies

As previously described (Lun *et al.*, 2009), we reduce the optimization problem (1) to a simpler, yet mathematically equivalent problem by iteratively removing dead-end reactions, linked reactions and any reactions that cannot support flux due to the flux bounds $a$ and $b$. Dead-end reactions occur when metabolites are associated with only a single reaction, which, therefore, cannot carry any flux. Linked reactions occur when metabolites are associated with exactly two reactions and, hence, the flux through one reaction precisely determines the other.

Let $G$ be a matrix that represents gene-protein-reaction (GPR) mappings. GPR mappings define how modifications at the genetic level map to reactions. Suppose there are $L$ sets of genes, each with unique metabolic function, which can be modified to affect the reaction network. We summarize the GPR mappings with an $L\times n$ matrix $G$, where the $ij$-th element of $G_{ij}$ of $G$ is 1 if the $l$-th set of genes maps onto reaction $j$ and is 0 otherwise. We then pose the problem of finding knockout strategies as a bilevel optimization problem that is then converted to an equivalent MILP problem (Burgard *et al.*, 2003):

$$\begin{array}{rll} \max & g'v & \\ \text{subject to} & \sum_{l=1}^{L} y_l \le C, & \\ & y_l\in\{0,1\}, & l=1,\ldots,L, \\ & Sv=0, & \\ & (1-y'G_j)a_j\le v_j\le(1-y'G_j)b_j, & j=1,\ldots,n, \\ & f'v=\sum_{j=1}^{n} v_jb_j-\mu_ja_j, & \\ & f_j-\sum_{i=1}^{m}\lambda_iS_{ij}-v_j+\mu_j-\xi_j=0, & j=1,\ldots,n, \\ & -Dy'G_j\le\xi_j\le Dy'G_j, & j=1,\ldots,n, \\ & \mu,v\ge 0, & \end{array} \tag{2}$$

where $g$ is the synthetic objective vector, whose $j$-th element $g_j$ is the weight of reaction $j$ in the synthetic objective, $y$ is the knockout vector, whose $l$-th element $y_l$ is equal to 1 if the genes involved in manipulation $l$ are knocked out and 0 otherwise, $G_j$ denotes the $j$-th column of $G$, $C$ is the maximum number of knockouts allowed, $\lambda$ is the dual variable for the equality constraints of (1), $\mu$ and $\nu$ are the dual variables for the lower and upper bounds, respectively, $\xi$ is the dual variable corresponding to the constraint $v_j = 0$ if $y_j = 1$ and $D$ is a scalar chosen to be sufficiently large to ensure that $\xi_j$ is effectively unconstrained whenever $y' G_j$ is non-zero (we set $D$ to be 100 in our analysis).

We solve problem (2) using a truncated branch and bound implemented with the Gurobi solver (Gurobi Optimization, Houston, TX, USA). Gurobi is a standalone, commercial solver for LP, quadratic programming (QP) and mixed-integer programming (MIP including MILP and MIQP). Briefly, we set up problem (2), and we pass it to the Gurobi solver with the following settings: We set FeasibilityTol to $10^{-9}$ and IntFeasTol to $10^{-9}$, which defines the solution tolerances, so as to achieve accurate solutions, and we set Heuristics to 1.0 to produce more and better feasible solutions and MIPFocus to 1 to prioritize feasible solutions over optimality. ImproveStartGap was set to infinity, to again focus on producing more feasible solutions. We used TimeLimit to specify the truncation time.

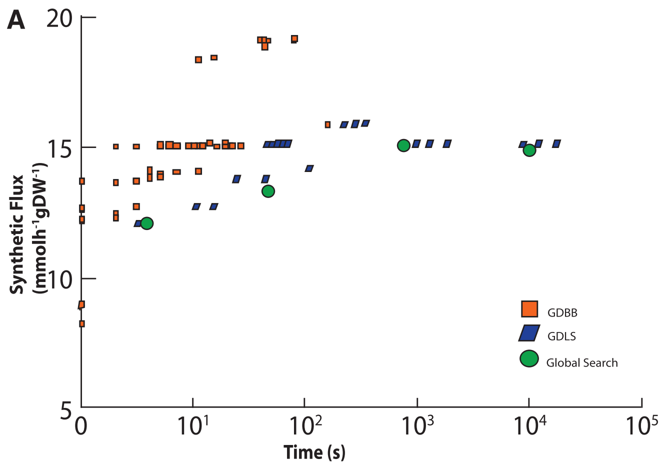
### 2.3 Comparison with GDLS and global search

For the comparison with GDLS and global search, we used the latest version of GDLS (version 1.1) available from the GDLS website, http://crab.rutgers.edu/~dslun/gdls. We ran GDLS with a single search path ($M = 1$) and search sizes $k = 1$ to $k = 7$. The search size $k$ is a positive integer and, as $k$ increases, the runtime of GDLS increases and generally so does its optimality. By running with search sizes from $k = 1$ to $k = 7$, we exhausted all search sizes that allowed GDLS to complete within 24 h. The number of search paths $M$ is also a positive integer and, as $M$ increases, the runtime and memory requirements of GDLS increase as does its optimality. In our previous experiments on the acetate and succinate production problems using *i*AF1260, we did not find significant performance improvements with values of $M > 1$, so we simply chose $M = 1$. Each search was terminated after a maximum running time of 86 400 s (24 h). Computations were performed using a computer equipped with an Intel® Core™ i5 Dual Core Processor 650 with VT running Linux version 2.6.35.

## 3 RESULTS

To illustrate the utility of GDBB, we consider the metabolic engineering of *E.coli* for acetate and succinate production. These cases serve as useful cases for comparison: they were previously used to compare the performance of GDLS with other approaches (in which GDLS compared favorably) and there is experimental experience with the metabolic engineering of *E.coli* for these purposes (Lun *et al.*, 2009). As GDLS compares favorably with other approaches, such as the evolutionary algorithm and simulated annealing meta-heuristics, implemented in the OptFlux software platform (Patil *et al.*, 2005; Rocha *et al.*, 2008), we primarily show the performance of GDBB in comparison with GDLS. For reference, we have also included the performance of the global search method, OptKnock (Burgard *et al.*, 2003). For a given number of allowed knockouts, global search is guaranteed to find an optimal solution (i.e. a knockout strategy that achieves the maximum synthetic production flux), but as such, its running time increases exponentially with the number of allowed knockouts.

In Figure 1, we show the performance of GDBB compared with that of GDLS and global search for acetate production in *E.coli* using the most recent genome-scale constraint-based model of *E.coli*, *i*AF1260 (Feist *et al.*, 2007). Using the different approaches, we search for knockout strategies with a maximum running time of 86 400 s (24 h). We chose 24 h to be the time period for evaluating performance as a reasonable but arbitrary choice. We expect that performance gains found within 24 h can be extended to any given time period. We see that GDBB finds comparable solutions to those found by GDLS and global search with vastly improved running time—often one or two or more orders of magnitude. As such, GDBB is able to find solutions within 24 h that are not found by the other two approaches. Indeed, GDBB finds a knockout strategy achieving an acetate production flux of 19.232 mmol h$^{-1}$ gDW$^{-1}$, which is an increase of 2.62278 mmol h$^{-1}$ gDW$^{-1}$ (or 14%) over the best solution found by GDLS in 24 h, and it does so in a mere 81 s.



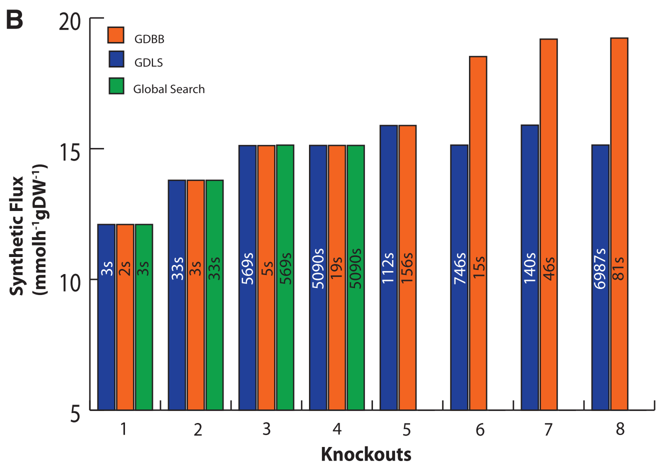


**Fig. 1.** Computational running time of GDBB compared to GDLS and global search for production of acetate using *i*AF1260. (**A**) The acetate production flux (synthetic flux) is shown as a function of running time for genetic manipulation strategies found by each of the three methods within 24 h. For GDBB, the best strategy, with synthetic flux of 19.232 mmol h$^{-1}$ gDW$^{-1}$, is found in 81 s. (**B**) The synthetic flux and running time is shown as a function of the number of knockouts for the best strategy with a given number of knockouts found within 24 h for each of the three methods

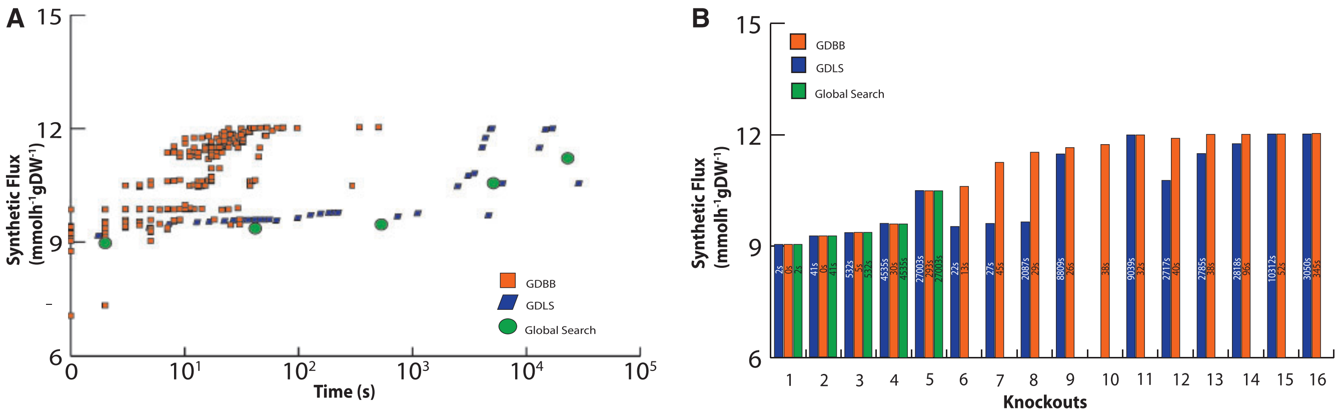


**Fig. 2.** Computational running time of GDBB compared to GDLS and global search for production of succinate using *i*AF1260. (**A**) The succinate production flux (synthetic flux) is shown as a function of running time for genetic manipulation strategies found by each of the three methods within 24 h. For GDBB, the best strategy, with synthetic flux of 12.037 mmol $h^{-1}$ $gDW^{-1}$, is found in 345 s. (**B**) The synthetic flux and running time is shown as a function of the number of knockouts for the best strategy with a given number of knockouts found within 24 h for each of the three methods

In Figure 2, we show the performance of GDBB compared with that of GDLS and global search for succinate production using *i*AF1260. For succinate production, we see similar vast improvements in running time. We notice that, while GDLS is capable of finding good solutions involving many knockouts (albeit with much longer running time than GDBB) in case of succinate production, it is inconsistent in its ability to find solutions for any given number of knockouts, while GDBB displays a more or less smooth increase in the synthetic flux as the number of knockouts increases. Overall, it appears that the genetic design problem for succinate production is easier than that for acetate production. In particular, for succinate production, the optimization problem appears to become relatively easy when large numbers of knockouts are allowed. We believe this may be because the maximum achievable synthetic flux for this problem is around 12 mmol $h^{-1}$ $gDW^{-1}$ and when many more knockouts are allowed than are necessary to achieve this flux; it becomes easier to find a favorable set of knockouts simply because more are allowed and simultaneously tried. Thus, with 14–16 knockouts, GDLS can find solutions similar to those found by GDBB within 24 h. But, because of the inconsistent behavior of GDLS with increasing numbers of knockouts, for a given, moderate allowed number of knockouts, the ability of GDLS to efficiently find a favorable genetic manipulation strategy is nevertheless variable and unreliable.

All knockout strategies found by GDBB were reached in <6 min, and we found no further improvement in synthetic flux by allowing 24 h of running time. Thus, in just several minutes, GDBB finds significantly better solutions than GDLS does when using up to an entire day of running time. As with GDLS, the knockout strategies found by GDBB make biological sense, being consistent with previous experimental findings and adopting strategies such as adjustments to redox balance. A full list of the strategies found by GDBB and their running times is given in Supplementary Table S1.

Examination of the strategies found by GDBB reveals the basis for its vastly improved performance. All prior approaches of which we are aware, such as GDLS or the evolutionary meta-heuristics used by OptFlux (Rocha *et al.*, 2010), rely fundamentally on the notion that good genetic manipulation strategies are achieved by adding manipulations to other good genetic manipulation strategies. This relates to the intuitive notion that good mutants for a specific purpose are found by taking already good mutants and mutating them further. It turns out, however, that sometimes several manipulations have to be performed simultaneously to have a significant effect and that any subsets of those manipulations have only marginal effect. For example, consider the best six-knockout solution found by GDBB, which achieves an acetate production flux of 18.5228 mmol $h^{-1}$ $gDW^{-1}$. The maximum acetate production flux achieved by any subset of the six is only 13.79 mmol $h^{-1}$ $gDW^{-1}$, which compares poorly with the best five-knockout solution found by GDBB, with an acetate production flux of 15.8859 mmol $h^{-1}$ $gDW^{-1}$. Thus, this solution does not arrive by adding manipulations to competitive mutant strains. GDBB does not search for manipulation strategies by adding manipulations to other good genetic manipulation strategies and, thus, is able to find strategies that cannot be easily found using other approaches.

In this article, we have considered a relatively simple problem, where we looked at the metabolic engineering of *E.coli* through knockouts, and we saw that GDBB was very useful. The need for GDBB will become more acute with more complex problems. There are two major ways in which the complexity of the genetic design problem can and will likely increase. First, models will become larger and more detailed as biological knowledge grows and if the organisms we seek to engineer are more complex than *E.coli*. Second, we can consider modifications other than knockouts and will likely desire to do so. We can, in particular, consider reducing or increasing gene expression instead of simply knocking genes out. The more modifications we consider, the larger the combinatorial space of possibilities that needs to be searched for genetic design. Herein, we have considered a simple case but provided a technique that can scale for future problems. We have presented here an efficient computational method for finding genetic manipulation strategies. GDBB finds, quickly and easily, strategies that are predicted by FBA to yield favorable phenotypes for metabolic engineering purposes. This approach of using FBA predictions for rational strain design has been previously experimentally validated and has much promise, but it has been significantly hindered to date by its computational complexity. With the advent of GDBB, we believe that rational, model-based design for metabolic engineering can be used much more widely, with great utility for important problems such as biofuel and drug production.

## ACKNOWLEDGEMENTS

The authors would like to thank Yalçin Kaya and Regina Burachik for helpful discussions and suggestions.

*Conflict of Interest*: none declared.

## REFERENCES

Bertsimas,D. and Tsitsiklis,J.N. (1997) *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA.

Boghigian,B. *et al.* (2010) Utilizing elementary mode analysis, pathway thermodynamics, and a genetic algorithm for metabolic flux determination and optimal metabolic network design. *BMC Syst. Biol.*, **4**, 49.

Burgard,A.P. *et al*. (2003) OptKnock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol. Bioeng.*, **84**, 647–657.

Feist,A.M. *et al.* (2007) A genome-scale metabolic reconstruction for *Escherichia coli* K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Mol. Syst. Biol.*, **3**, 121.

Feist,A.M. and Palsson,B.Ø. (2008) The growing scope of applications of genome-scale metabolic reconstructions using *Escherichia coli*. *Nat. Biotechnol.*, **26**, 659–667.

Franck,B. *et al*. (2001) Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spektrum*, **23**, 297–324.

Kim,J. *et al*. (2011) Large-scale bi-level strain design approaches and mixed-integer programming solution techniques. *PLoS One*, **6**, e24162.

Lun,D.S. *et al.* (2009) Large-scale identification of genetic design strategies using local search. *Mol. Syst. Biol.*, **5**, 296.

Melzer,G. *et al.* (2009) Flux design: in silico design of cell factories based on correlation of pathway fluxes to desired properties. *BMC Syst. Biol.*, **3**, 120.

Milne,C.B. *et al.* (2009) Accomplishments in genome-scale *in silico* modeling for industrial and medical biotechnology. *Biotechnol. J.*, **4**, 1653–1670.

Orth,J.D. *et al*. (2010) What is flux balance analysis? *Nat. Biotechnol.*, **28**, 245–248.

Palsson,B. (2006) *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, New York.

Patil,K.R. *et al.* (2005) Evolutionary programming as a platform for *in silico* metabolic engineering. *BMC Bioinformatics*, **6**, 308.

Pharkya,P. *et al*. (2004) OptStrain: a computational framework for redesign of microbial production systems. *Genome Res.*, **14**, 2367–2376.

Pharkya,P. and Maranas,C.D. (2006) An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems. *Metab. Eng.*, **8**, 1–13.

Reed,J. *et al.* (2003) An expanded genome-scale model of *Escherichia coli* K-12 (iJR904 GSM/GPR). *Genome Biol.*, **4**, R54.

Rocha,I. *et al.* (2010) OptFlux: an open-source software platform for in silico metabolic engineering. *BMC Syst. Biol.*, **4**, 45.

Rocha,M. *et al.* (2008) Natural computation meta-heuristics for the in silico optimization of microbial strains. *BMC Bioinformatics*, **9**, 499.

Segrè,D. *et al*. (2002) Analysis of optimality in natural and perturbed metabolic networks. *Proc. Natl Acad. Sci. USA*, **99**, 15112–15117.

Sendín,J.O.H. *et al*. (2010) Multi-objective mixed integer strategy for the optimisation of biological networks. *IET Syst. Biol.*, **4**, 236–248.

Shlomi,T. *et al*. (2005) Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proc. Natl Acad. Sci. USA*, **102**, 7695–7700.

Song,B. *et al.* (2011) Manipulating the steady state of metabolic pathways. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **8**, 732–747.

Trinh,C.T. *et al*. (2009) Elementary mode analysis: a useful metabolic pathway analysis tool for characterizing cellular metabolism. *Appl. Microbiol. Biotechnol.*, **81**, 813–826.

Tyo,K.E.J. *et al.* (2010) Toward design-based engineering of industrial microbes. *Curr. Opin. Microbiol.*, **13**, 255–262.

Vemuri,G.N. and Aristidou,A.A. (2005) Metabolic engineering in the -omics era: elucidating and modulating regulatory networks. *Microbiol. Mol. Biol. Rev.*, **69**, 197–216.

Yang,L. *et al*. (2010) Rapid design of system-wide metabolic network modifications using iterative linear programming. In: Kothare, M. *et al.* (eds) *Proceedings of the 9th International Symposium on Dynamics and Control of Process Systems (DYCOPS 2010)*. IFAC, Leuven, Belgium, pp. 377–382.

Zhang,W. (1993) Truncated branch-and-bound: a case study on the asymmetric TSP. *Proceedings of the AAAI-93 Spring Symposium on AI and NP-Hard Problems*. Stanford, CA, pp. 160–166.