Large-scale identification of genetic design strategies using local search

Molecular Systems Biology 5; Article number 296; doi:10.1038/msb.2009.57
Citation: ***Molecular Systems Biology*** **5:296**

www.molecularsystemsbiology.com

molecular systems biology

REPORT

Large-scale identification of genetic design strategies using local search

Desmond S Lun[1,2,5,6], Graham Rockwell[2,3,5], Nicholas J Guido[2], Michael Baym[4], Jonathan A Kelner[4], Bonnie Berger[4], James E Galagan[1] and George M Church[2,*]

[1] Broad Institute of MIT and Harvard, Cambridge, MA, USA, [2] Department of Genetics, Harvard Medical School, Boston, MA, USA, [3] Program in Bioinformatics, Boston University, Boston, MA, USA and [4] Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
[5] These authors contributed equally to this work
[6] Present address: Phenomics and Bioinformatics Research Centre, School of Mathematics and Statistics, and Australian Centre for Plant Functional Genomics, University of South Australia, Mawson Lakes, South Australia, Australia
* Corresponding author. Department of Genetics, Harvard Medical School, 77 Avenue Louis Pasteur, NBR 238, Boston, MA 02115, USA. Tel.: +1 617 432 7562; Fax: + 1 617 432 6513; E-mail: gmc@harvard.edu

Received 9.3.09; accepted 8.7.09

In the past decade, computational methods have been shown to be well suited to unraveling the complex web of metabolic reactions in biological systems. Methods based on flux–balance analysis (FBA) and bi-level optimization have been used to great effect in aiding metabolic engineering. These methods predict the result of genetic manipulations and allow for the best set of manipulations to be found computationally. Bi-level FBA is, however, limited in applicability because the required computational time and resources scale poorly as the size of the metabolic system and the number of genetic manipulations increase. To overcome these limitations, we have developed Genetic Design through Local Search (GDLS), a scalable, heuristic, algorithmic method that employs an approach based on local search with multiple search paths, which results in effective, low-complexity search of the space of genetic manipulations. Thus, GDLS is able to find genetic designs with greater *in silico* production of desired metabolites than can feasibly be found using a globally optimal search and performs favorably in comparison with heuristic searches based on evolutionary algorithms and simulated annealing.
Molecular Systems Biology **5**: 296; published online 18 August 2009; doi:10.1038/msb.2009.57
Subject Categories: computational methods; cellular metabolism
Keywords: bi-level optimization; flux–balance analysis; metabolic engineering; mixed-integer linear programming; strain optimization

Introduction

In a number of separate studies, genome-scale flux–balance analysis (FBA) modeling has been shown to be useful for the *in silico* design of engineered strains of microbes that overproduce diverse targets. These engineered strains include *Escherichia coli* that overproduce lycopene (Alper *et al*, 2005a, b), lactic acid (Fong *et al*, 2005), succinic acid (Lee *et al*, 2005; Wang *et al*, 2006), L-valine (Park *et al*, 2007), and L-threonine (Lee *et al*, 2007), and strains of *Saccharomyces cerevisiae* that overproduce ethanol (Bro *et al*, 2006; Hjersted *et al*, 2007). FBA models allow the result of various genetic manipulations strategies to be predicted. As a result, the space of possible genetic manipulations can be computationally searched for the strategy that results in the desired metabolic network state. This space is vast, and algorithms must be designed to search the space efficiently.

Transforming bi-level optimization of FBA models to single-level mixed-integer linear programming (MILP) problems (Burgard *et al*, 2003; Pharkya *et al*, 2004; Pharkya and Maranas, 2006) 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.

The runtimes scale exponentially as the number of manipulations allowed in the final design increases. For large models—such as the latest genome-scale model of *E. coli* K-12 MG1655 (Feist *et al*, 2007), *i*AF1260—we have found that this runtime generally proves prohibitive for designs involving more than a few manipulations. Given that useful metabolically engineered strains often require many genetic manipulations (such as the artemisinic acid-producing strain of *S. cerevisiae* by Ro *et al* (2006), which required the addition of three genes and the up- or downregulation of four genes) and that the number of reactions, metabolites, and genes in metabolic models continues to grow (Feist and Palsson, 2008), it is clear that more efficient computational search techniques are required for effective *in silico* design.

We present a heuristic algorithmic method, which we call Genetic Design through Local Search (GDLS), that is capable of handling large models and allows for a much larger number of genetic manipulations in the final design, with runtime scaling only linearly with the total number of manipulations '*T*'. GDLS employs a local search approach with multiple search paths (Anderson, 1989; Pudil *et al*, 1994; Bertsimas and Weismantel, 2005) to find a set of locally optimal strategies. In brief, GDLS functions as follows. It begins by taking a certain strategy as a starting point, which can be a user-defined set of manipulations, a randomly chosen set of genetic manipulations, or no manipulations. The GDLS method then uses an MILP approach to search for the best strategies that differ from the starting point by, at most, '*k*' additional manipulations. The population of strategies GDLS maintains is limited to a maximum size '*M*', which is also the number of unique search paths maintained. These *M* best strategies are then used as the starting point for the next round of MILP search, resulting in a new set of *M* best strategies, each of which differ from the one of the starting *M* best strategies by, at most, *k* additional manipulations. This procedure is iterated until no better strategy can be found. Through this search method, GDLS is able to find strategies in which the total number of manipulations *T* is large using computationally feasible increments (determined by *k* and *M*). As a result, GDLS can discover strategies that have a larger value of *T* and of the desired flux than can feasibly be found by a global search.

A heuristic, sequential approach to finding genetic design strategies was also employed by Alper *et al* (2005a). The approach of Alper *et al* can be considered as a special case of the approach of GDLS, in which we take no genetic manipulations as the starting point and, at each iteration, search by brute-force for the best strategy that involves only one additional genetic manipulation. GDLS combines the strengths of such a sequential approach with those of the bi-level FBA approach previously mentioned (Burgard *et al*, 2003; Pharkya *et al*, 2004; Pharkya and Maranas, 2006), allowing for a graceful trade-off between the good complexity properties of the former and the good optimality properties of the latter.

Other heuristic approaches include the application of meta-heuristics such as evolutionary algorithms and simulated annealing, both of which are explored in OptFlux (Patil *et al*, 2005; Rocha *et al*, 2008). These meta-heuristics apply random local perturbations to a solution or a population of solutions and propagate those with better performance. Although such meta-heuristics can perform well on difficult global optimization problems, we find that, for genetic design, it is frequently the case that a number of manipulations have to be performed simultaneously to have an effect, whereas each carried out on its own has no effect. In such a case, if one of the manipulations is randomly chosen on its own, then it is no more likely to be propagated, as it has no effect, and the probability of choosing all the manipulations simultaneously in a random perturbation is very small. Thus, it is difficult for these meta-heuristics to identify such manipulations. GDLS overcomes this problem by systematically searching the local neighborhood. We compare the performance of GDLS with the evolutionary algorithm and simulated annealing approaches implemented in OptFlux and, using the meta-heuristics, we indeed are able only to find solutions that compare with those found by GDLS when $k=1$.

In this paper, we limit our consideration of bi-level FBA frameworks to one for gene knockouts for simplicity, but any bi-level FBA framework that can be transformed to an MILP, such as one for up- or downregulation of genes using the approach of OptReg (Pharkya and Maranas, 2006), is compatible. We do not consider FBA extensions such as energy–balance analysis (EBA) (Beard *et al*, 2004; Yang *et al*, 2005) or thermodynamics-based metabolic flux analysis (TMFA) (Henry *et al*, 2007), which account for thermodynamic constraints, or multi-objective optimization (Sendín *et al*, 2009), which allows the simultaneous optimization of multiple cellular functions to be studied. These extensions are important for accurate prediction of flux in eukaryotic cells, but are less so for prokaryotic cells (Vo *et al*, 2004; Nagrath *et al*, 2007), and developing computational methods for genetic design that incorporate such extensions is likely to be an important next step. Multi-objective optimization can also be used as an alternative to bi-level optimization to find genetic design strategies (Xu *et al*, 2009), but it is more suggestive, than prescriptive, and has received relatively little attention to date.

Beyond the local search approach, GDLS implements a number of reductions that decrease the size of the FBA model without changing its properties. In addition, to predict network changes from genetic level manipulations, GDLS employs gene–protein reaction (GPR) mappings, which provide a complete many-to-many mapping between genes and the reactions that depend on them. These relationships can be complex, making it difficult to realize the effects of genetic manipulations without computational analysis. Compared with the common approach of allowing direct manipulation of the reaction network, employing GPR mappings not only models more accurately the way in which engineering is carried out, but also potentially reduces the manipulation search space, which reduces search complexity and hence runtime.

Results

Overview of the method

GDLS provides a general method for implementing a bi-level optimization framework for genetic design that is capable of handling large, complex FBA models, and of discovering designs involving a large number of manipulations that

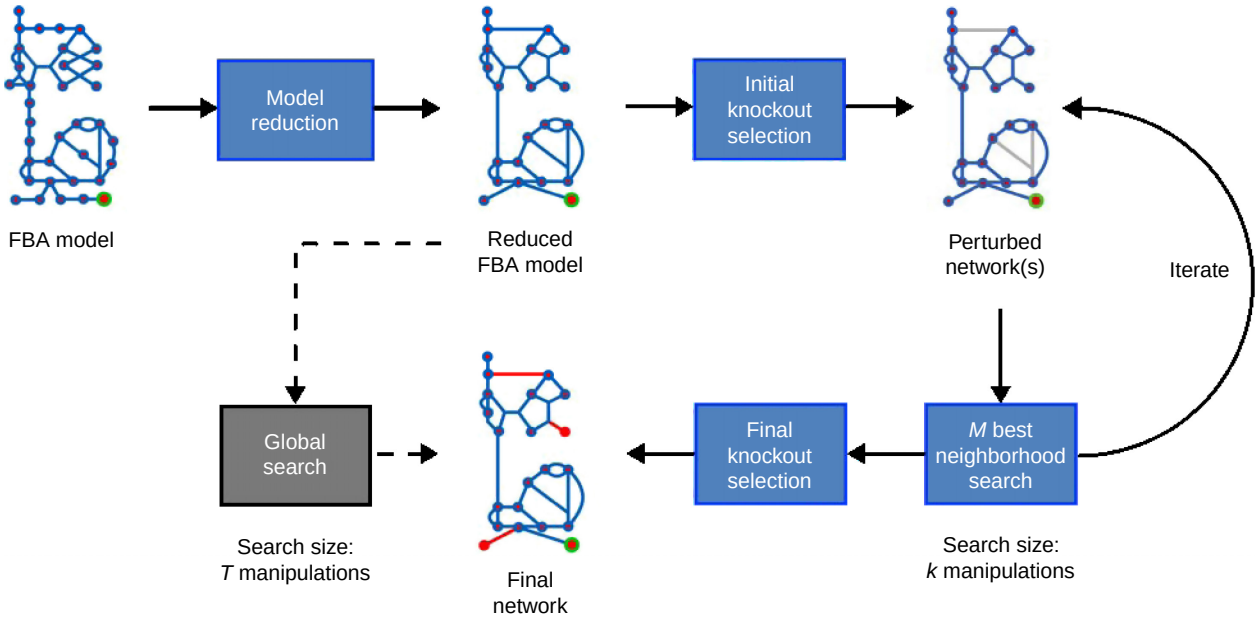

Figure 1 Overview of GDLS. The input FBA model is first reduced to yield an equivalent FBA model with fewer genes, reactions, and metabolites. Then, on this reduced FBA model, an initial knockout selection is made, yielding a perturbed network. A neighborhood search is then performed, in which MILP is used to search for the *M* best genetic manipulation strategies that differ from this starting point by, at most, *k* additional manipulations, and the *M* best perturbed networks thus obtained are used for another round of neighborhood search. We continue to iterate until no further improvement can be obtained within the full range, *T*, of allowed manipulations from the reduced FBA model. Alternatively, in global search, we simply use MILP to search for the best genetic manipulation strategy that differs from the reduced FBA model by, at most, *T* manipulations.

produce high levels of the desired flux, which we refer to as the synthetic flux. GDLS achieves its capabilities through the use of an iterative local search approach with multiple search paths. For simplicity and clarity, we frame our discussion and demonstration of GDLS in the context of optimization of knockouts but, as we have mentioned, GDLS can be applied to find genetic designs involving any type of manipulation for which there is a bi-level FBA framework that can be transformed to an MILP.

An overview of the approach employed by GDLS is shown in Figure 1. The approach begins with an initial stage in which the GDLS MILP framework is set up. First, the input FBA model is reduced to a smaller, but equivalent, model with fewer genes, reactions, and metabolites. These reductions, which include removing dead-end reactions and linked reactions, are accepted (Patil *et al*, 2005; Rocha *et al*, 2008), and though they do not substantially improve the runtime of the MILP, they do substantially improve its numerical stability. We do not remove genes from the possible knockout list based on experimental predictions of lethality, as the lethality of a single knockout may not hold when combined with other knockouts. The possible knockout list is, however, easily set by the user.

Knockouts are modeled in terms of gene sets that can affect one or more reduced reactions using GPR mappings, which give a many-to-many mapping of genes to reactions. On this reduced model, an initial set of knockouts is selected. This initial set can be chosen randomly, thus providing the heuristic with many possible starting points, each of which possibly results in a different final solution.

After the GDLS MILP framework is set up, we begin searching iteratively while maintaining a population of solutions of size *M*. Starting from the initial set of knockouts, we apply an MILP optimization routine to search for *M* sets of knockouts that result in the *M* greatest synthetic fluxes, while differing from the initial set by at most *k* knockouts. Knockouts can be removed after being selected and, in this way, GDLS is capable of backtracking. Ideally, the MILP optimization routine always returns a solution, but the software packages that are used to solve MILPs are sometimes unable to find feasible solutions or return infeasible solutions. Thus, we, in fact, obtain, at most, *M* solutions at this stage, but we do not necessarily have exactly *M* solutions. We then take the up to *M* solutions found and use them as initial sets of knockouts for the next iteration, in which we apply the same MILP optimization routine *M* times for each of the, up to, *M* currently maintained solutions. The next iteration, therefore, generates up to M^2 unique solutions, from which the best *M* solutions are selected for subsequent iterations. This process is repeated until the set of best solutions does not change or is empty.

The number of knockouts *k* that can be added to or removed from each solution at each iteration, and the number of solutions *M* maintained after each iteration are the key factors in runtime and optimality. Generally, a larger search size results in a better ability to search the space, especially for groups of changes that are only effective in concert, thus smaller values of *k* and *M* generally yield shorter runtimes at the cost of solutions with lower synthetic flux.

Performance

We tested the performance of GDLS by optimizing for acetate and succinate production in the most recent genome-scale model of *E. coli*, *i*AF1260, and comparing against a global MILP search. A global MILP search is employed by OptKnock (Burgard *et al*, 2003), though it was applied to a smaller model, it does not implement FBA model reductions or GPR mappings, and uses the proprietary solver CPLEX. In contrast, GDLS uses Solving Constraint Integer Programs (SCIP) combined with Sequential object-oriented simplex (SoPlex) as the MILP solver, which generally has slower and less numerically stable performance than CPLEX, but is freely available. This capacity to effectively use freely available solvers on large FBA models, such as *i*AF1260, is one of the strengths of GDLS. For a fair comparison, the global MILP search against which we compare GDLS includes the FBA model reductions and GPR mappings and also uses SCIP combined with SoPlex. Alternative MILP solvers can be used in GDLS and any improvements in the MILP solver will improve both global search and GDLS.

In Figure 2, we show the results obtained with a single search path (i.e. $M=1$) for search sizes k ranging from 1 to 4. The specific solutions obtained are given in Supplementary Tables S1 and S2 for acetate and succinate production, respectively. In these tests, we set the initial group of genes to be knocked out to the empty set, so GDLS runs deterministically.

The single search path case is illustrative of some aspects of the general performance of GDLS, including the scaling of search time with search size k and total gene knockouts T as well as the ability of GDLS to find solutions with greater numbers of knockouts and greater synthetic flux than global search. We observe that the CPU time required by global search increases exponentially with the number of knockouts T in the solution. In both the acetate and the succinate cases, the required CPU time quickly becomes prohibitive. By contrast, the CPU time required for the solutions found by GDLS increases much more gently with T, with an increase that is approximately linear for a fixed value of the search size k. Thus, for small T, GDLS is able to find solutions in much less time than global search, often with the same or comparable synthetic flux. For example, we are able to find the optimal three-knockout strategy for acetate production found by global search in 2566 s using an instance of GDLS, as compared with 36 924 s using global search—an over 14-fold runtime improvement. For larger T, GDLS is able to find solutions in which global search simply cannot find solutions in reasonable time.

The behavior of local search, in which we move from one local optimum to another, can be simple or complex, as shown in Figure 2. Succinate production increases fairly directly with larger search sizes k and more iterations. Acetate production, in contrast, shows the more complex nature of the search. We observe that, as expected, smaller values of the search size k result in solutions with lower synthetic flux for a given total

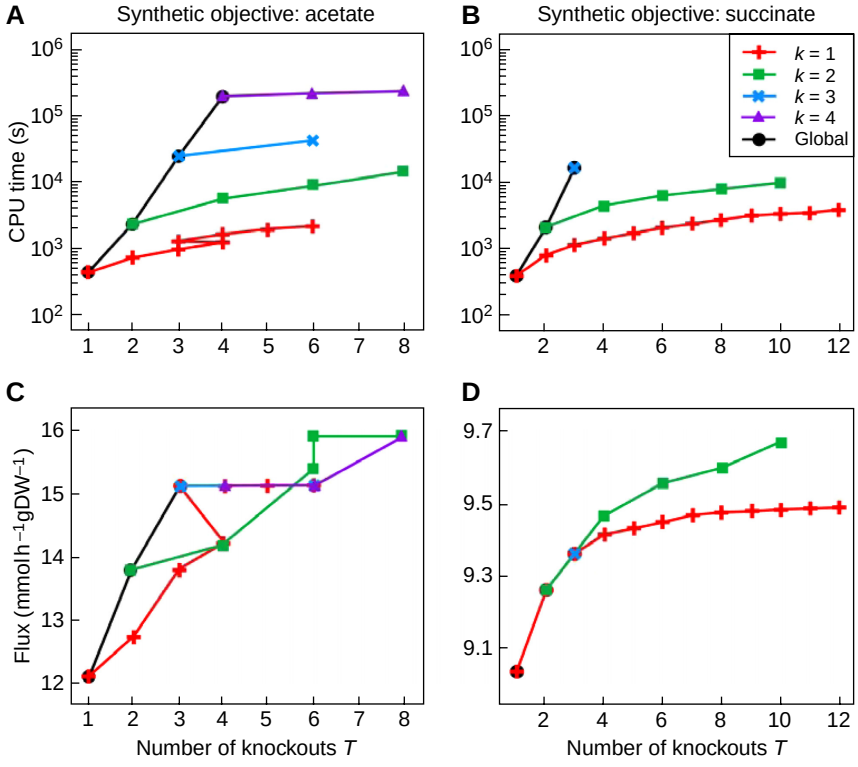

Figure 2 Performance of GDLS in *i*AF1260 with a single search path. The required CPU time and resulting synthetic flux for varying numbers of knockouts, T, are shown for solutions obtained by global search (black circles) and GDLS with search sizes, k, ranging from 1 to 4: $k=1$ (red plus signs), $k=2$ (green squares), $k=3$ (blue crosses), and $k=4$ (purple triangles). (**A**) CPU time for acetate production. (**B**) CPU time for succinate production. (**C**) Acetate flux for solutions shown in (A). (**D**) Succinate flux for solutions shown in (B). Searches were run until CPU time exceeded 70 h, until 12 iterations had passed (succinate $k=1$), or until termination (acetate $k=1$, 2, 3, 4 and succinate $k=2$, 3). In all cases in which GDLS search terminated, it terminated owing to the set of best solutions being empty.

number of knockouts T in general, but this is not necessarily true. In the acetate k=2 case, a four-knockout solution with relatively low synthetic flux, found by the search, leads to a six-knockout solution at the next iteration (which is followed by another six-knockout solution with even greater synthetic flux) that outperforms the six-knockout solutions found by the k=1, 3, and 4 searches. The k=3 and 4 searches have six-knockout solutions with lower synthetic flux because they arrive at their six-knockout solutions from different points (the k=3 search arrives at its solution from a previous three-knockout solution and the k=4 search arrives at its solution from a previous, relatively high production, four-knockout solution). From the same point, larger values of k will result in solutions with greater synthetic flux but, as the trajectories of the various searches are likely to diverge after a few iterations, there is no guarantee that one search will necessarily dominate another.

It can also be seen that local search provides a clear benefit over simple, greedy search because of the ability to simultaneously search gene knockouts and backtrack (i.e. undo manipulations from previous iterations). The knockout strategies for acetate in particular highlight the importance of backtracking to discovering the best solutions. Backtracking is clearly evident in the path of acetate k=1, but is also important if less obvious in the k=2 search at six knockouts, in which one knockout is added and one is removed, and in the k=4 search as it moves from six to eight knockouts, in which three knockouts are added and one is removed.

Apart from the succinate k=1 search, which reached the iteration maximum of 12, all other searches terminated because the set of best solutions was empty, that is, at a particular iteration, the MILP solver did not return a solution or returned an infeasible solution, which seems to become more likely as the synthetic flux increases. One method of extending the search beyond its termination points is to restart GDLS using a particular solution as the new starting point after adjusting the parameters of the search. Another method is to increase the number of search paths. As GDLS maintains a set of best solutions that grows with the number of search paths, using more search paths is less likely to result in a situation in which the set of best solutions is completely empty.

We also ran GDLS with the number of search paths M, set to 2, 3, and 4, which, as mentioned above, leads to approximately M^2 MILP searches being executed at each iteration and, hence, to an M^2 increase in CPU time (see Supplementary Tables 1 and 2). As expected, increased values of M allowed us to extend each of the searches to greater total knockouts T. For both optimization targets, reaching larger values of T resulted in correspondingly higher levels of synthetic flux. For acetate production, the increase in the number of search paths yielded solutions with equivalent levels of synthetic flux to searches with M=1 until larger values of T were reached. For succinate production, however, improvement was seen for k=1, 2, and 3 in the range of total knockouts T already encountered with M=1. In particular, for the succinate k=3 case, solutions with substantially greater synthetic flux are obtained by using two search paths (M=2), with a nine-knockout solution with a flux of 9.699 mmol h^{-1} gDW^{-1} found, which outperforms the maximum flux of 9.664 mmol h^{-1} gDW^{-1} (using ten knockouts) obtained in all cases using a single search path. This nine-knockout solution then leads ultimately to an 18-knockout solution with a flux of 9.727 mmol h^{-1} gDW^{-1}.

The knockouts in the solutions that are produced by GDLS can possibly contain redundant knockouts that are not, in fact, necessary for the achievement of the synthetic flux level of the solution. Such redundant knockouts arise because they were necessary at earlier iterations, but are no longer necessary in the solution in question. It is simple to check whether all knockouts are necessary and to remove those that are not, shrinking the knockouts to the smallest necessary set. We have found that doing so generally leads to mild improvements in GDLS, but does not do so in every case. The results of running GDLS with such shrinkage of the knockout sets are given in Supplementary Tables 3 and 4 and are shown in Supplementary Figure 1.

In addition, we compared GDLS against OptFlux (Patil *et al*, 2005; Rocha *et al*, 2008), which implements the evolutionary algorithm and simulated annealing meta-heuristics. We ran each meta-heuristic twice to find knockout strategies for both acetate and succinate production. Both meta-heuristics were run for the same number of function evaluations, taking approximately 5×10^4 s for each run (see Materials and methods for details). The best solution found for acetate production was 15.1381 mmol h^{-1} gDW^{-1} under the evolutionary algorithm meta-heuristic and 15.1291 mmol h^{-1} gDW^{-1} under the simulated annealing meta-heuristic, and the best solution found for succinate production was 9.87498 mmol h^{-1} gDW^{-1} under the evolutionary algorithm meta-heuristic and 10.0076 mmol h^{-1} gDW^{-1} under the simulated annealing meta-heuristic. For acetate production, we observe that the best synthetic fluxes obtained by these methods are only in the range of those found by GDLS with k=1 and do not rise to the levels found by GDLS when $k>1$. For succinate production, the synthetic fluxes found by OptFlux exceed those found by GDLS, but this is because OptFlux does not implement GPR mappings and allows reactions themselves to be knocked out; the solutions found by OptFlux cannot in fact be implemented using gene knockouts once GPR mappings are taken into account.

Discussion

The utility of genome-scale FBA modeling in finding promising genetic manipulation strategies for metabolic engineering has been established in numerous studies (see Patil *et al*, 2004; Feist and Palsson, 2008). The space of all possible genetic manipulation strategies, however, is vast and, even if we restrict ourselves to knocking out genes, it is a challenge to search the space of all possible strategies computationally for those that are likely to result in the desired metabolic changes. Through a principled heuristic approach, we have developed GDLS, a method that allows the space of possible genetic engineering strategies to be efficiently searched and allows the trade-off between optimality of the search and runtime. We observe that GDLS achieves over an order of magnitude improvement in computational time for solutions that yield equal or comparable values for the desired flux with as few as three manipulations.

GDLS employs a local search approach with multiple search paths, which searches through genetic design strategies by propagating a set of best solutions. Increasing the number of search paths increases the size of the set that is propagated, which, at the cost of increased runtime, has numerous advantages. First, as we have observed in the case of optimization for succinate production, it allows solutions with greater synthetic fluxes to be found. Second, it makes the search more robust to numerical inaccuracies in the MILP solver. Finally, it yields a larger set of solutions, all of which have high synthetic fluxes. These alternative solutions yield alternative genetic designs for examination and testing, which are useful because predictions from FBA models are subject to some error and prior biological knowledge may suggest that one of the alternative designs is in fact a more promising candidate for implementation.

The knockout strategies we find using GDLS on *i*AF1260 seem to be biologically sensible and consistent with previous studies. In particular, the knockout strategies for acetate production contain knockouts for alcohol and ethanol dehydrogenase and for ATP synthase, which are knockouts present in the acetate-producing strain of *E. coli* W3110 of Causey *et al* (2003). The knockout strategies we find for succinate production differ from those reported previously by Burgard *et al* (2003) using OptKnock because we use different models, but we still find knockout strategies that include removing alcohol and ethanol dehydrogenase as we do for acetate production. In the results listed in Supplementary Tables 1 and 2, all strategies start with the removal of alcohol dehydrogenase through the removal of either the gene set (adhE, frmA and adhP) (all cases except acetate k=2) or the removal of adhE and mhpF. This result is sensible because removing these enzymes eliminates the production of ethanol and other competing fermentation products. In general, the removal of competing pathways is a consistent feature of most *in silico* and *in vitro* optimization strategies.

The majority of other knockout targets affect the redox balance of the cell, which is a common result of genetic design using FBA models because many metabolites whose production we desire to increase are by-products that naturally act to help the cell maintain redox balance, and this is true of acetate and succinate. Changing the redox balance can induce the production of these metabolites as the cell acts to restore that balance. An interesting knockout in several strategies was phosphoglucose isomerase, the removal of which results in a rather drastic change in cellular function, but *in silico* it is most likely directed at forcing flux through the pentose phosphate pathway so that NADPH is produced rather than NADH. Combined with the removal of NAD(P) transhydrogenase, such that NADPH cannot be exchanged for NADH, the system is now dependent on the production of the desired metabolite to create NADH.

Our strategies also include targets to decouple the synthetic flux from amino-acid biosynthesis, biomass production, or alternative redox balancing. The superior synthetic flux achieved by the acetate k=2 search is achieved through the removal of 3-hydroxy acid dehydrogenase, which acts to break down amino acids for the production of NADH. The succinate strategies include knockouts such as glutamate dehydrogenase, the removal of which helps make the production of amino acids from succinate impossible.

In examining these knockout strategies, we see that the GPR mappings implemented in GDLS result in a realistic coupling of genes and reactions, making sure, for example, that reactions are not removed individually when they can, in fact, only be removed along with other reactions. Gene sets such as (adhE, frmA, and adhP) and individual genes such as that producing 3-hydroxy acid dehydrogenase are each associated with multiple reactions in *i*AF1260.

To discover the most complete set of design strategies possible, we recommend running GDLS with increasing values of the search size k and the number of search paths M, starting with one for both parameters, until the limit of feasible computational resources is reached (e.g. when a certain time limit is reached). Given the limitations in the prediction power of FBA modeling, such an approach will result in a variety of good design strategies that can be manually examined and will allow manipulations that are consistently chosen to be identified and prioritized.

An implementation of GDLS is publicly available, thus facilitating and advancing the use of computational tools in the design of biological organisms. With the ever-increasing complexity and accuracy of mathematical models of biological organisms, such computational tools are likely to have an ever larger role in biological engineering.

Materials and methods

Genome-scale FBA modeling

We work with the genome-scale model of *E. coli*, *i*AF1260. This model consists of three parts. From m metabolites and n reactions, we form an $m \times n$ stoichiometric matrix S, whose ijth element S_{ij} is the stoichiometric coefficient of metabolite i in reaction j. The vector of flux values of v, whose jth element v_j is the flux though reaction j, are constrained by a lower bound vector a, whose jth element a_j is the lower bound of the flux through reaction j, an upper bound vector b, whose jth element b_j is the upper bound of the flux through reaction j. Finally, the linear objective is formed by multiplying the fluxes by an objective vector f, whose jth element f_j is the weight of reaction j in the biological objective. Thus, a biologically optimal flux distribution is obtained by solving

$$\begin{array}{ll} \max & f'v \\ \text{subject to} & Sv = 0 \\ & a \le v \le b \end{array} \qquad (1)$$

Reduction of FBA models

We apply three principal reductions. First, we remove all dead-end reactions. Dead-end reactions occur when metabolites are associated with only a single reaction, which, therefore, cannot carry any flux.

Second, we find all linked reactions. Linked reactions occur when metabolites are associated with exactly two reactions. As metabolites are conserved, the fluxes of the two reactions are always in the same ratio and can be reduced to a single variable. Thus, we find all equations of the form

$$S_{ij_1} v_{j_1} + S_{ij_2} v_{j_2} = 0$$

and we reduce j_1 and j_2 to a single reaction, because we simply have

$$v_{j_1} = -\frac{S_{ij_2}}{S_{ij_1}} v_{j_2}.$$

Third, we successively maximize and minimize each flux subject to the constraints of the problem—a problem sometimes referred to as

the max–min problem (Burgard *et al*, 2001; Mahadevan and Schilling, 2003; Pharkya and Maranas, 2006)—to obtain tighter bounds on the fluxes, that is, we solve

$$\begin{array}{ll} \max/\min & v_j \\ \text{subject to} & Sv = 0, \\ & a \le v \le b. \end{array} \tag{2}$$

Let $v_{j,l}$ and $v_{j,u}$ be the result of minimizing and maximizing (2), respectively. If, for any j, we have $v_{j,u} \leqslant v_{j,l}$, then we remove reaction j from the model, as the flux through reaction j must be zero.

We iteratively apply these reductions until we cannot reduce the model any further. For the *i*AF1260 model, we start with an initial size of m=1668 and n=2382; after reduction, we obtain m=483 and n=959. These reductions reduce the number of variables in the model without changing its properties; the reduced and original models are mathematically equivalent.

Local search for genetic strategies

GDLS builds upon previous metabolic design methods by incorporating the relationship between genes and reactions and a local search of multiple genetic manipulations strategies. To allow GDLS to function at the genetic level, we have implemented GPR mappings. GPR mappings define how certain genetic manipulations affect reactions in the metabolic network. For a set of L genetic manipulations, we summarize the GPR mappings with an $L \times n$ matrix G, where the ljth element G_{lj} of G is 1 if the lth genetic manipulation maps onto reaction j and is 0 otherwise. In *i*AF1260, we have L=913 initially; after reduction, we have L=632.

We use a previously established approach for posing the problem of finding knockout strategies as a bi-level optimization problem and converting it to an equivalent MILP problem (Burgard *et al*, 2003). The bi-level optimization problem can be written as follows:

$$\begin{array}{lll} \max & g'v & \\ \text{subject to} & \sum_{l=1}^{L} y_l \le C, & \\ & y_l \in \{0,1\}, & \\ & \max & f'v \\ & \text{subject to} & Sv = 0, \\ & & (1-y)'G_j a_j \le v_j \le (1-y)'G_j b_j, \quad j = 1,\ldots,n, \end{array} \tag{3}$$

where g is the synthetic objective vector, whose jth element g_j is the weight of reaction j in the synthetic objective; y the knockout vector, whose lth element y_l is equal to 1 if the genes involved in manipulation l are knocked out and 0 otherwise; G_j denotes the jth column of G; and C is the maximum number of knockouts allowed. We convert the bi-level (3) problem to an equivalent MILP using the dual of (1) to obtain

$$\begin{array}{lll} \max & g'v & \\ \text{subject to} & \sum_{l=1}^{L} y_l \le C, & \\ & y_l \in \{0,1\}, & l = 1,\ldots,L, \\ & Sv = 0, & \\ & (1-y)'G_j a_j \le v_j \le (1-y)'G_j b_j, & j = 1,\ldots,n, \\ & f'v = \sum_{j=1}^{n} v_j b_j - \mu_j a_j, & \\ & f_j - \sum_{i=1}^{m} \lambda_i S_{ij} - v_j + \mu_j - \xi_j = 0, & j = 1,\ldots,n, \\ & -Dy'G_j \le \xi_j \le Dy'G_j, & j = 1,\ldots,n, \\ & \mu, v \ge 0, & \end{array} \tag{4}$$

where λ is the dual variable for the equality constraints of (1), μ and ν are the dual variables for the lower and upper bounds, respectively, ξ 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 ξ_j is effectively unconstrained whenever $y'G_j$ is non-zero (we set D to be 100 in our analysis).

Now suppose we start with some initial knockout vector $v^{(0)}$. At each iteration i, we maintain a set of knockout strategies $Y^{(i)}$, so let the set $Y^{(0)} := \{y^{(0)}\}$ and i=0. For each knockout vector $\tilde{y}$ in $Y^{(i)}$, we solve problem (4) M times using the following additional constraints. We first add the constraint

$$\sum_{l:\tilde{y}_l=0} y_l + \sum_{l:\tilde{y}_l=1} (1 - y_l) \le k \tag{5}$$

to problem (4) to limit the size of the search neighborhood to k, and we solve problem (4) using constraint (5). We set $Y^{(i+1)}$ to be the set that contains the optimal solution y^*. Then, for $M-1$ times, we solve problem (4) again using constraint (5) and the additional constraint

$$\sum_{l:\hat{y}_l=0} y_l + \sum_{l:\hat{y}_l=1} (1 - y_l) \ge 1 \tag{6}$$

for all $\hat{y}$ in $Y^{(i+1)}$, and we add the optimal solution y^* to $Y^{(i+1)}$. If we do not obtain an optimal solution or the solution returned is infeasible, we do not modify $Y^{(i+1)}$ and simply continue. Thus, from each $\tilde{y}$ in $Y^{(i)}$, we add up to the M best solutions to problem (4) with constraint (5) to the set $Y^{(i+1)}$, which ultimately contains up to $|Y^{(i)}|M$ elements. We then remove elements from $Y^{(i+1)}$ to reduce its size to a maximum of M, keeping the best solutions.

We increment i and repeat the above procedure until $Y^{(i+1)} = Y^{(i)}$.

GDLS implementation

We describe an implementation of GDLS using MATLAB, GLPK (GNU Linear Programming Kit), SCIP (Achterberg, 2007), and SoPlex (Wunderling, 1996). GDLS, GLPK, SCIP, and SoPlex are non-commercial software packages publicly available for download. We illustrate the capabilities of GDLS by applying it to acetate and succinate overproduction problems in *E. coli* using *i*AF1260 as the FBA model. As succinate and acetate fermentations are generally carried out under anaerobic or microaerobic conditions (Zeikus *et al*, 1999; Causey *et al*, 2003), and such conditions are consistent with good *in silico* production, we gave the model no available oxygen and 10 mmol h^{-1} gDW^{-1} available glucose. All computations were carried out using 2.4 GHz Intel Xeon processors.

Comparison with OptFlux

Following Rocha *et al* (2008), we used the set-based algorithm with a population size of 100 for the evolutionary algorithm, and we used 50 trials for simulated annealing. In both cases, runs were terminated after 100 000 function evaluations.

Software availability

GDLS is freely available for non-profit use in the supplementary information.

Supplementary information

Supplementary information is available at the *Molecular Systems Biology* website (www.nature.com/msb).

Acknowledgements

We would like to thank John Aach and Daniel Segrè for helpful comments and suggestions. MB gratefully acknowledges support from the Fannie and John Hertz Foundation.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Achterberg T (2007) *Constraint Integer Programming*. Berlin: Technische Universität Berlin

Alper H, Jin YS, Moxley JF, Stephanopoulos G (2005a) Identifying gene targets for the metabolic engineering of lycopene biosynthesis in *Escherichia coli*. *Metab Eng* **7:** 155–164

Alper H, Miyaoku K, Stephanopoulos G (2005b) Construction of lycopene-overproducing *E.* coli strains by combining systematic and combinatorial gene knockout targets. *Nat Biotechnol* **23:** 612–616

Anderson JB (1989) Limited search trellis decoding of convolutional codes. *IEEE Trans Inform Theory* **35:** 944–955

Beard DA, Babson E, Curtis E, Qian H (2004) Thermodynamic constraints for biochemical networks. *J Theor Biol* **228:** 327–333

Bertsimas D, Weismantel R (2005) *Optimization over Integers*. Belmont, Massachusetts: Dynamic Ideas

Bro C, Regenberg B, Förster J, Nielsen J (2006) In silico aided metabolic engineering of *Saccharomyces cerevisiae* for improved bioethanol production. *Metab Eng* **8:** 102–111

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

Burgard AP, Vaidyaraman S, Maranas CD (2001) Minimal reaction sets for *Escherichia coli* metabolism under different growth requirements and uptake environments. *Biotechnol Prog* **17:** 791–797

Causey TB, Zhou S, Shanmugam KT, Ingram LO (2003) Engineering the metabolism of *Escherichia coli* W3110 for the conversion of sugar to redox-neutral and oxidized products: homoacetate production. *Proc Natl Acad Sci USA* **100:** 825–832

Feist AM, Henry CS, Reed JL, Krummenacker M, Joyce AR, Karp PD, Broadbelt LJ, Hatzimanikatis V, Palsson BØ (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 AM, Palsson BØ (2008) The growing scope of applications of genome-scale metabolic reconstructions using *Escherichia coli*. *Nat Biotechnol* **26:** 659–667

Fong SS, Burgard AP, Herring CD, Knight EM, Blattner FR, Maranas CD, Palsson BØ (2005) In silico design and adaptive evolution of *Escherichia coli* for production of lactic acid. *Biotechnol Bioeng* **91:** 643–648

Henry CS, Broadbelt LJ, Hatzimanikatis V (2007) Thermodynamics-based metabolic flux analysis. *Biophys J* **92:** 1792–1805

Hjersted JL, Henson MA, Mahadevan R (2007) Genome-scale analysis of *Saccharomyces cerevisiae* metabolism and ethanol production in fed-batch culture. *Biotechnol Bioeng* **97:** 1190–1204

Lee KH, Park JH, Kim TY, Kim HU, Lee SY (2007) Systems metabolic engineering of *Escherichia coli* for L-threonine production. *Mol Syst Biol* **3:** 149

Lee SJ, Lee D-Y, Kim TY, Kim BH, Lee J, Lee SY (2005) Metabolic engineering of *Escherichia coli* for enhanced production of succinic acid, based on genome comparison and in silico gene knockout simulation. *Appl Environ Microbiol* **71:** 7880–7887

Mahadevan R, Schilling CH (2003) The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metab Eng* **5:** 264–276

Nagrath D, Avila-Elchiver M, Berthiaume F, Tilles AW, Messac A, Yarmush ML (2007) Integrated energy and flux balance based multiobjective framework for large-scale metabolic networks. *Ann Biomed Eng* **35:** 863–885

Park JH, Lee KH, Kim TY, Lee SY (2007) Metabolic engineering of *Escherichia coli* for the production of L-valine based on transcriptome analysis and *in silico* gene knockout simulation. *Proc Natl Acad Sci USA* **104:** 7797–7802

Patil KR, Åkesson M, Nielsen J (2004) Use of genome-scale microbial models for metabolic engineering. *Curr Opin Biotechnol* **15:** 64–69

Patil KR, Rocha I, Förster J, Nielsen J (2005) Evolutionary programming as a platform for *in silico* metabolic engineering. *BMC Bioinformatics* **6:** 308

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

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

Pudil P, Novovičová J, Kittler J (1994) Floating search methods in feature selection. *Pattern Recognit Lett* **15:** 1119–1125

Ro D-K, Paradise EM, Ouellet M, Fisher KJ, Newman KL, Ndungu JM, Ho KA, Eachus RA, Ham TS, Kirby J, Chang MC, Withers ST, Shiba Y, Sarpong R, Keasling JD (2006) Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature* **440:** 940–943

Rocha M, Maia P, Mendes R, Pinto JP, Ferreira EC, Nielsen J, Patil KR, Rocha I (2008) Natural computation meta-heuristics for the in silico optimization of microbial strains. *BMC Bioinformatics* **9:** 499

Sendín J-OH, Alonso AA, Banga JR (2009) Multi-objective optimization of biological networks for prediction of intracellular fluxes. In *2nd International Workshop on Practical Applications of Computational Biology and Bioinformatics (IWPACBB 2008)*, pp 197–205

Vo TD, Greenberg HJ, Palsson BØ (2004) Reconstruction and functional characterization of the human mitochondrial metabolic network based on proteomic and biochemical data. *J Biol Chem* **279:** 39532–39540

Wang Q, Chen X, Yang Y, Zhao X (2006) Genome-scale *in silico* aided metabolic analysis and flux comparisons of *Escherichia coli* to improve succinate production. *Appl Microbiol Biotechnol* **73:** 887–894

Wunderling R (1996) *Paralleler und Objektorientierter Simplex-Algorithmus*. Berlin: Technische Universität Berlin

Xu M, Bhat S, Smith R, Stephens G, Sadhukhan J (2009) Multi-objective optimization of metabolic productivity and thermodynamic performance. *Comput Chem Eng*

Yang F, Qian H, Beard DA (2005) *Ab initio* prediction of thermodynamically feasible reaction directions from biochemical network stoichiometry. *Metab Eng* **7:** 251–259

Zeikus JG, Jain MK, Elankovan P (1999) Biotechnology of succinic acid production and markets for derived industrial products. *Appl Microbiol Biotechnol* **51:** 545–552