Metabolic Design And Engineering Through Ant Colony Optimization

Stephen Lincoln
Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, USA
191 Auditorium Road
Storrs, CT 06226 USA
stephen.lincoln@uconn.edu

Ian Rogers
Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, USA
191 Auditorium Road
Storrs, CT 06226 USA
ian.rogers@uconn.edu

Ranjan Srivastava
Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, USA
191 Auditorium Road
Storrs, CT 06226 USA
srivasta@engr.uconn.edu

ABSTRACT

Due to the vast search space of all possible combinations of reaction knockouts in *Escherichia coli*, determining the best combination of knockouts for over-production of a metabolite of interest is a computationally expensive task. Ant colony optimization (ACO) applied to genome-scale metabolic models via flux balance analysis (FBA) provides a means by which such a solution space may feasibly be explored. In previous work, the Minimization of Metabolic Adjustment (MoMA) objective function for FBA was used in conjunction with ACO to identify the best gene knockouts for succinic acid production. In this work, algorithmic and biological constraints are introduced to further reduce the solution space. We introduce Stochastic Exploration Edge Reduction Ant Colony Optimization, or STEER-ACO. Algorithmically, ACO is modified to refine its search space over time allowing for greater initial coverage of the solution space while ultimately honing on a high quality solution. Biologically, a heuristic is introduced allowing the maximum number of knockouts to be no greater than five. Beyond this number, cellular viability becomes suspect. Results using this approach versus previous methods are reported.

Categories and Subject Descriptors
J.3 [**Life and Medical Science**]: Biology and genetics

General Terms: Algorithms.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ACM 978-1-4503-3472-3/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754817

Keywords: Metabolic modeling, computational biology, metabolic engineering, ant colony optimization, genome-scale, flux balance analysis, systems biology

1. INTRODUCTION

The production of molecules and biomaterials from microorganisms is a well-established practice in industrial settings, as well as bench scale experiments. For example, active pharmaceutical ingredients (APIs) that are too complex or too costly to be produced via classic organic chemistry reactions can be synthesized via metabolically engineered microorganisms due to the high value of the API of interest and low cost and low quantity of reactants needed by the engineered microorganism [10]. Another example is the production of hydrogen with *Escherichia coli*. Recent studies have been conducted for utilizing the cheap and abundant substrate glycerol to produce biofuel, or hydrogen gas. The production of H_2 via *E. coli* is of growing interest because H_2 can be used directly as a reactant in fuel cells and does not contribute to greenhouse gas emissions. H_2 also reduces the need for fossil fuel sources and can be created by *E. coli* via waste generated from agriculture and industry. However, gene insertions and deletions must be made into the wild-type *E. coli* in order to maximize the production of H_2 to feasible levels [24].

Determining the best gene knockouts and insertions to produce a molecule or biomaterial of interest in *E. coli* is a computationally and theoretically difficult task. Genome-scale metabolic models have become a powerful computational tool for simulating an organism's metabolism under a given set of genetic and environmental constraints. The term "genome-scale" simply refers to how the metabolic network is constructed. The genome annotation of an organism is used to identify genes that catalyze metabolic reactions. If such a gene is present, the metabolic reaction it catalyzes is inferred to be present. In this way, it is possible to create a genome-scale metabolic model [4; 9; 14; 15].

Attempts have been made to predict and analyze gene and reaction knockouts in metabolic networks using genome-scale metabolic modeling [16]. As an example, the genome-scale metabolic model of *E. coli, iJR904*, has 1075 reactions, 761 metabolites, and 904 genes, and models are constantly being

updated as new genes, kinetic information, and thermodynamic information is discovered [17; 18]. If the number of reactions knocked out is limited to three in *iJR904*, there are on the order of 10^9 different combinations of reactions to knock out. Increasing the maximum number of knockouts to five increases the number of possible combination to 10^{15}, and six to the order of 10^{18}. It is apparent that computational time and power become a concern as the number of knockouts is increased.

There are a number of *in silico* simulation programs to identify the best knockouts. For example, OptKnock [5] has been used to identify triplets of knockouts for overproduction of 1-butanol, 1-propanol, and 1,3-propanediol in *E. coli* in previous work [12]. However, this method is computationally expensive. For more five or more knockouts at a time, computations may take several hours to days on standard computing hardware.

Most approaches to modeling metabolism at the genome-scale use flux balance analysis (FBA) to analyze resulting fluxes based on the knockouts [3; 11; 13; 19]. The term "fluxes" refers to the throughput or "flux" of a metabolite through a given reaction pathway. Due to the number of equations and variables in an FBA model, the system is underdetermined. To calculate flux values, optimization theory is used where an objective function regarding how the cell will grow is postulated. The most commonly used objective function is maximization of growth rate. The premise for this objective function is that the cells that grow the fastest would be the most competitive. Thus those cells would be selected by evolution to be the dominant strain. However, when carrying out genetic engineering, the network that nature evolved is disrupted, casting doubt on the use of such an objective for genetically engineered organisms. To overcome this issue, another objective function has been proposed which states that the genetically engineered cell will attempt to distribute its metabolic resources in a manner similar to the wild type because that is how it evolved. This idea is referred to as "minimization of metabolic adjustment" or MoMA [20]. The MoMA concept may be formulated as a quadratic programming problem, which then may be solved to return reasonable flux distributions and identify more realistic best knockout sets. However, while MoMA can predict how metabolic resources will be distributed in genetically engineered organisms, it does not predict what genes should be knocked out to optimize production of a metabolite.

In this work, we propose a new method to identify the best sets of gene knockouts for metabolite production. Specifically we combine MoMA with ant colony optimization (ACO). We refer to this method as Stochastic Exploration Edge Reduction Ant Colony Optimization (STEER-ACO). A similar ACO/MoMA approach has recently been attempted [6]. However, the STEER-ACO algorithm improves on the previous work in several ways. The newly developed algorithm refines the search space over time to converge better solutions, as well as avoiding large reductions in the growth rate of *E. coli* by using the growth rate as a heuristic factor. It avoids a pre-processing step reduced the metabolic network that was used in the older algorithm, which had the potential to generate false positives and false negative knockouts. In addition, we have ported MoMA to Python and implemented our algorithm in Python while utilizing a Redis database system setting the stage for parallelization of the algorithm in the future. It is anticipated that this will further reduce the computational time required.

To evaluate the STEER-ACO algorithm, succinic acid production from genetically engineered *E. coli* was used as a case study. Succinic acid was chosen for two primary reasons. First, succinic acid is an important compound in industrial processes, ranging from food and pharmaceutical products, surfactants and detergents, green solvents and biodegradable plastics, and ingredients to stimulate animal and plant growth [25]. In addition, numerous studies on maximization of succinic acid production in *E. coli* have been performed, providing a large body of experimental data to compare with.

2. METHODS

2.1 Minimization of Metabolic Adjustment (MoMA)

MoMA is a constraint based programming method FBA which was first introduced by Daniel Segre, Dennis Vitkup, and George M. Church in 2002 [20]. One of the core principles of FBA is that the metabolic network of an organism of interest, such as *E. coli,* is assumed to be at steady state, which is strictly true for cells growing in chemostats and approximately true for cells growing in exponential phase [14; 23]. Under such conditions, the mass balance for each metabolite in the metabolic network of reactions must be zero. This can be denoted by the equation

$$\boldsymbol{S \cdot v = 0} \qquad (1)$$

Where **S** is the stoichiometric coefficient matrix of size *m x n* where *m* is the number of metabolites, and *n* is the number of reactions in the network. $\boldsymbol{v}$ is the vector of fluxes in the network. The system is also bound to a set of constraints for the amount of nutrients available to the system, as well as the theoretical minimum and maximum fluxes supported by the system, such that

$$\boldsymbol{a_i \leq v_i \leq b_i} \qquad (2)$$

Linear programming (LP) is used to determine each flux for the wild-type microorganism using the system described above. After the system is solved with the objective of maximizing biomass growth, MoMA is used in order to determine the new fluxes in the system once a reaction, or set of reactions is knocked out. This is simulated in MoMA by setting the lower and upper bounds for the knocked out flux to zero. MoMA then uses quadratic programming (QP) to determine the network's new fluxes, under the assumption that the mutant flux distribution will be as close to the wild-type's flux distribution as possible. In other words, the goal is to determine a flux vector, $\boldsymbol{x}$ in which the Euclidian distance is as close as possible to the wild-type flux vector $\boldsymbol{w}$:

$$\boldsymbol{D = \sqrt{\sum_{i=1}^{N} (w_i - x_i)^2}} \qquad (3)$$

D is the distance between the wild-type flux $\boldsymbol{(w)}$ and mutant flux $\boldsymbol{x}$ and **N** is the number of optimal points.

Since MoMA uses QP in order to solve the system, the goal is to minimize the standard QP problem

$$f(x) = L \cdot x + \frac{1}{2} x^T Q x \qquad (4)$$

Such that the vector L is of length **N** and the matrix Q is of size NxN. Q is used to define the linear and quadratic objective function, and x^T represents the transpose of x. Since minimizing D in Equation (3) is the same as minimizing its square and constants can be omitted from the objective function, **Q** can be set to an $N \, x \, N$ unit matrix and L can be set to $-w$, where $w = v^{wt}$.

2.2 Ant Colony Optimization (ACO)

Ant Colony Optimization is a subset of swarm intelligence that is based on the behavior of ant colonies foraging for food. Ants communicate indirectly by depositing pheromone trails that decay over time in their environment while foraging. Although ants randomly choose a path to take, there is a greater probability to take a path that contains a large amount of pheromones. This allows ants to converge towards the shortest path to food in an autocatalytic reinforcement process [22]. The metaheuristic that governs ACO is as follows:

Set parameters, initialize pheromone trails
While termination condition not met **do**
Construct solutions
Apply local search (optional)
Update Pheromones
End

The choice of solutions is guided stochastically with bias from the pheromone trails. The optional local search is performed to improve upon known solutions by locally searching the solution space, and is usually included in most ACO algorithms in order to improve upon known good solutions [7].

2.3 Stochastic Exploration Edge Reduction Ant Colony Optimization (STEER-ACO)

Although previous work has applied ACO/MoMA hybrid techniques [6] to determining the best sets of reaction knock outs in *E. coli* for succinic acid production, the approach used a completely stochastic search for knockout combinations based off of known viable reaction knock out candidates. In addition, the growth rate of mutant *E. coli* was not taken into account as a factor in engineering the final organism. The STEER-ACO algorithm is proposed as improvement to the previous approach based on both algorithmic and biological strategies. STEER-ACO consists of three stages

1. Initialization
    a. Bias generation
2. Solution Generation/Deletion
    a. Determine solution/path
    b. Update pheromone trails
    c. Forget bad solutions
3. Solution Examination

2.3.1 Initialization

During the initialization phase, the growth rate and flux of interest is determined and returned using LP FBA. Then each reaction is knocked out one at a time and the growth rate and flux of interest is determined using MoMA. If the growth rate is 0, the knockout is deemed lethal and removed from the selection of reactions that may be knocked out during solution generation. Otherwise, the value of the flux of interest is stored for that particular knockout in order to calculate the bias term that allows the algorithm to converge onto a better solution over time. Each viable knockout is stored as a key in a Redis database with the values of the current pheromone trail value, flux of interest from MoMA, growth rate returned from MoMA, and probability associated with choosing the solution. The value returned from MoMA is stored so that if an ant picks a known solution, MoMA would not have to be run again simply to return the QP value; it could just be grabbed from the database to speed up the algorithm. Besides being open source and easy to use, Redis was chosen for solution storage for two primary reasons:

1. Storing solutions with Redis is fast due to its in-memory storage
2. Redis allows for easy parallelization of STEER-ACO

The pheromone trail value is generated using the equation

$$\textbf{Pheromone amount} = \tau^{\alpha} \eta^{\beta} \qquad (5)$$

Where τ represents the flux of interest value such that

$$\tau_i = \begin{cases} \frac{v_{mut}}{v_{wt}} \; if \; v_{wt} \neq 0 \\ v_{mut} \; otherwise \end{cases} \qquad (6)$$

which is done to normalize the value returned from MoMA. The value for η was the growth rate for the knockout, and α and β are constants to influence the strength of each term. The growth rate was used for η to allow a higher probability of choosing solutions that have both a good growth rate and good flux of interest, which is one of the ways STEER-ACO integrates the algorithm and biological knowledge. It is also one of the ways in which STEER-ACO differs from previous work. In addition, by defining η this way, it also allows for STEER-ACO to score a solution with very high flux of interest value with a subpar growth rate as a good solution, as long as the growth rate is not too poor.

2.3.2 Bias Generation

Each viable single reaction knockout has its flux of interest value stored into an array. At the end of initialization, the array is divided by its sum in order to determine probabilities associated with selecting a particular gene to knock out during solution generation. In other words, each viable flux has a probability of being selected if an ant chooses to find a new solution; the higher the value from Equation (7) below, the higher the probability it has of being chosen.

2.3.3 Solution Generation

At the beginning of each epoch of solution generation, all of the current keys, or solutions, present in the Redis database are returned with their pheromone value. The probability of choosing a particular known solution, denoted p_i, is calculated for each solution:

$$p_i = \frac{\tau_i^{\alpha} \eta_i^{\beta}}{\sum_{i=1}^{K} \tau_i^{\alpha} \eta_i^{\beta}} \qquad (7)$$

where **K** is the total number of known solutions

2.3.4 Determine solutions/paths

For each ant in the system, there exists a probability denoted as **ω** to choose a new path based on the list of known solutions. This value is set high initially and decreases as the number of epochs increases. The purpose of this gradual decrease of ω is to encourage ants to explore new solutions initially, then converge on to known good solutions as time goes on. If an ant chooses a new solution, a knockout is chosen from the list of viable knockouts generated during initialization. This knockout is determined using a uniform random distribution initially in order to exhaust the search space. However, after the ants have gone through half of the allowed epochs, the knockout is chosen based on the bias determined during initialization. In other words, knockout candidates that return both a good flux value of interest and growth rate have a higher chance of being selected as time goes on.

Once a new knockout is selected, a random key is returned from the Redis database and the knockout is added to it to form a new solution. If the key already has the maximum amount of knockouts allowed, a knockout is removed based on the bias term calculated initially. Therefore, knockouts that generate a lower flux value of interest and growth rate compared to other knockouts in the solution have a higher probability to be removed and replaced. Finally, if the ant does not choose a new solution with a probability of **1- ω**, a known solution is selected from the Redis database by the probability described in Equation (7).

2.3.5 Update Pheromone Trails and Solution Deletion

After all the ants have chosen a path, the pheromone trails are updated

$$\tau_i \leftarrow (1-\rho)\tau_i + \sum_k \tau_i^k = (1-\rho)\tau_i + n\tau_i \quad (8)$$

Where **ρ** is the pheromone evaporation coefficient, and τ_i^k is the amount of pheromone deposited on solution i by the k^{th} ant, which simply reduces to $n\tau_i$ where n is the amount of ants that selected the solution. After all the pheromone trails have been updated, the solutions in the database are iterated through, and old solutions are deleted. A solution is forgotten and deleted if the **τ** of the solution is less than τ_{min}. A maximum tau is also set. See **Figure 1** for the algorithm flowchart.

Table 1. Parameters used for each trial of STEER-ACO

Parameter	Value
α	1
β	1.5
ρ	0.65
ω	0.7
Max Epochs	100
Num. Ants	200
Max Knockouts	5
τ_{min}	0.01
τ_{max}	100

2.3.5 Solution Examination

At the end of the maximum epochs allowed, the solutions in the database are sorted according to the amount of pheromone deposited at each solution. The top five results are returned according to the amount of pheromone deposited. The top five most common knockouts are also put into a separate knockout set and run through MoMA. Results that are deemed best return the largest succinic acid production, while sub-optimal results are defined as any combination of knockouts that produce a moderate amount of succinic acid with a high growth rate close to the wild-type growth rate.

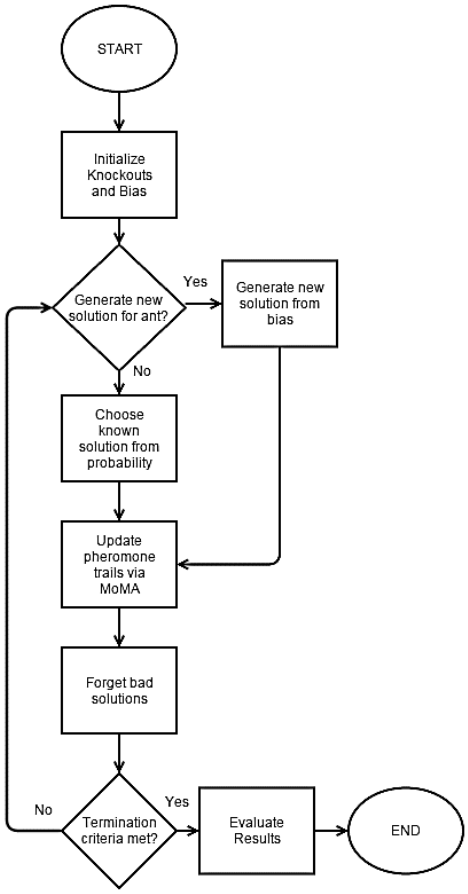

Figure 1. Flowchart for STEER-ACO algorithm

3. SIMULATIONS

The latest edition of dataset *iJR904* for *E. coli* was used [18], which contains 1075 reactions, 761 metabolites, and 904 genes, as well as gene to protein to reaction (GPR) information. For each calculation of the wild-type fluxes as well as MoMA simulations, the glucose uptake rate was set to 10 mmol $gDW^{-1}hr^{-1}$. The wild-type growth rate for the model was found to be 0.9219 h^{-1}. The objective was kept at maximizing biomass growth, while the τ value returned from MoMA was set to succinic acid production. Unlike previous work, there was no preprocessing done on the dataset in order to retain the complete model. Although this will

increase the computational time, the average time required to achieve the algorithms best solution is 8.4 minutes under current settings shown in **Table 1** for five knockouts. More importantly, the potential for false positive and false negatives is reduced by leveraging the use of the entire metabolic network rather than an abridged, albeit more tractable, version. Additionally, plans to parallelize STEER-ACO and utilize PyCUDA GPU programming will significantly decrease the computational time, so preprocessing to remove reactions and metabolites was deemed unnecessary.

The number of maximum epochs was set to 100 in order to allow enough iterations to exhaust the search space, but still converge on a good solution. The number of ants was set to 200, or about 0.2 of the total number of reactions. Simulations using more ants took more computation time and did not return a better solution. In order to take advantage of parallelization and GPU programming in future work, MoMA was ported to Python v2.7.9 utilizing CPLEX v12.6 with Python bindings for LP and QP calculations. The STEER-ACO algorithm was written in Python 2.7.9 utilizing Numpy v1.9.0. All solutions were stored in a Redis database utilizing Python bindings (v2.9.1). The model from the database was parsed into CPLEX [1] using COBRA v0.2.1 [8]. All simulations were performed on Ubuntu 12.04.5 LTS with an Intel Core 2 Duo 2.0 GHz processor and 4 GB of DDR2 RAM.

4. RESULTS

Previous approaches used an older model of *iJR904*, and the model was preprocessed. Thus those results are not directly comparable to the results presented here. To compare results generated by STEER-ACO with previous studies, the top three triplet knockouts generated by those older studies were input into MoMA using the updated, unprocessed *iJR904* model, with the results shown in **Table 2.** STEER-ACO was run 20 times, with the top five results returned for each trial shown in **Table 3**.

Table 2. Top three triplet knockouts returned from previous studies using updated & unprocessed model *iJR904*

Reactions Knocked Out	Succinic Acid Production (mmol gDW^{-1} h^{-1})	Growth Rate (h^{-1})
G6PDH2r PDH SUCD1i	2.8542	0.1139
FUM G6PDH2r PDH	2.6463	0.1162
FUM G6PDH2r PYK	2.2042	0.1244

The top five results of all the trials are listed in **Table 4.** The top ten most common knockouts for the top five sets over 20 trials were found as seen in **Figure 2.**

STEER-ACO not only performed better in terms of succinic acid production, but growth rate as well. The best solution returned from STEER-ACO had a succinic acid production and growth rate of 3.5815 mmol $gDW^{-1}h^{-1}$ and 0.1270 h^{-1} respectively. In comparison, the best result from the older algorithm had a succinic acid production rate of 2.8542 mmol $gDW^{-1}h^{-1}$ and growth rate of 0.1139 h^{-1} based on the knockouts they predicted. This is a 25.5% increase in succinic acid production, and 11.5% increase in growth rate. Even the fifth best solution returned from STEER-MOMA had an 11.5% increase in growth rate while producing approximately the same amount of succinic acid as compared to the best solution returned using the approach described in the previous studies.

Table 3. Top five sets of three knockouts returned from STEER-ACO from unprocessed model *iJR904*

Reactions Knocked Out	Succinic Acid Production (mmol gDW^{-1} h^{-1})	Growth Rate (h^{-1})
ACt2r GLCDe SUCD4	3.5815	0.1270
ACt2r SUCD4 XYLt2	3.5466	0.1273
ACNML EX_ac_(e) SUCD4	3.5466	0.1287
EX_ac_(e) Kt2r SUCD4	3.5285	0.1287
DBTSr SUCD1i PTAr	2.8288	0.1363

Given the speed at which the STEER-ACO ran, it was feasible to explore a larger knockout space. Based on our experimental experience, the viability of *E. coli* generally degraded significantly once more than five knockouts were made. A five knockout limit was thus decided upon. When comparing the predictions of STEER-ACO for five knockouts to previous work that had only three knockouts, STEER-ACO predicted genetic modifications that yielded higher levels of succinic acid. This result is unsurprising given that five knockouts provide substantially more flexibility relative to three knockouts if one is able to effectively take advantage of it. What is surprising, however, is that even with five knockouts, STEER-ACO predicts growth rates higher than previous studies did with three knockouts. The best solution returned from SEARCH-ACO had a succinic acid production and growth rate of 5.1206 mmol $gDW^{-1}h^{-1}$ and 0.1393 h^{-1} respectively, shown in **Table 4**. This is a 79.4% increase in succinic acid production and a 22.3% increase in growth rate.

The fifth best solution returned from STEER-MOMA had a 51.8% increase in succinic acid production rate and 22.5% increase in growth rate as compared to the best solution obtained using the previous study's results. This result is surprising because it would be expected that as more succinic acid is produced, few metabolic resources would be available to for cellular growth. However, it appears that by incorporating the growth rate into the value of **η**, it was possible to identify high succinic acid production rates while maintaining relatively high growth rates. Maintaining high growth rates in cell culture is

important in the biotech industry for a variety of reasons [21]. Such a result indicates the potential usefulness and value of the STEER-ACO algorithm.

Table 4. Top five sets of five knockouts returned from STEER-ACO from unprocessed model *iJR904*

Reactions Knocked Out	Succinic Acid Production (mmol $gDW^{-1} h^{-1}$)	Growth Rate (h^{-1})
ACt2r EX_leu_L_(e) NMNAT SUCD1i TALA	5.1206	0.1393
EX_ac_(e) FUM G6PDH2r SUCD4 TMPKr	4.5951	0.1253
EX_ac_(e) FUM G6PDH2r GUAPRT SUCD4	4.5889	0.1256
EX_ac_(e) EX_no2_(e) G6PDH2r PYRt2r SUCD4	4.5779	0.1277
EX_ac_(e) MTHFC RNDR3 SUCD1i SUCD4	4.3329	0.1224

The remainder of the analysis focuses on data collected from the simulations and optimization using the five knockout systems. In an attempt to create better set of knockouts, the top five most common knockouts are put into a separate set and run through MoMA, which returned a growth rate of 0.8450 h^{-1} and succinic acid production rate of 0.9317 mmol $gDW^{-1}h^{-1}$, which falls under a sub-optimal knockout set due to the moderate succinic acid production and high growth rate.

An acetic acid transport export reaction was found to be present in all of the top five knockouts combinations. In terms of the TCA cycle, this theoretically allows more acetic acid to be fed into the TCA cycle via acetyl-CoA. In addition, fumarase and succinate dehydrogenase are the most frequently knocked out reactions and appear in all of the top five knockouts. As seen in **Figure 3**, this disallows key reactions that consume succinic acid in the TCA cycle.

Table 5. Genes associated with each set of top five reactions. Note that all reactions starting with "EX" refer to transport reactions or "exchange/export fluxes" were resources are taken up by the cell or secreted from the cell.

Reactions Knocked Out	Gene Name
ACt2r	-
EX_leu_L_(e)	-
NMNAT	nadD
SUCD1i	sdh
TALA	talA or talB
EX_ac_(e)	-
FUM	fum
G6PDH2r	zwf
SUCD4	sdh
TMPKr	thiL
EX_ac_(e)	-
FUM	fumA or fumC or fumB
G6PDH2r	zwf
GUAPRT	gpt or hpt
SUCD4	sdhD, sdhC, sdhA, sdhB
EX_ac_(e)	-
EX_no2_(e)	-
G6PDH2r	zwf
PYRt2r	-
SUCD4	sdh
EX_ac_(e)	-
MTHFC	folD
RNDR3	nrd
SUCD1i	sdh
SUCD4	sdh

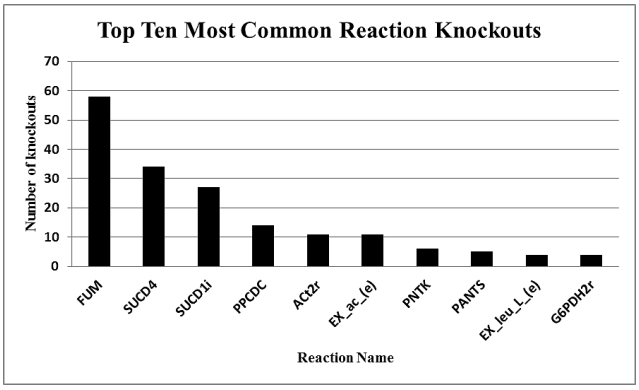

Figure 2. Top ten most common knockouts for overproduction of succinic acid in *E. coli iJR904* found in the top five results for 100 trials.

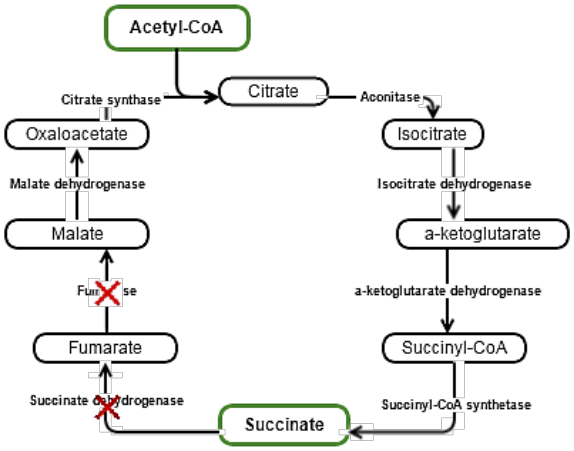

Figure 3. Basic TCA cycle diagram. Knocking out acetate exporters, succinate dehydrogenase, and fumarase allows for overproduction of succinic acid

Although fumarase was present in only two of the five top results, it is present in nearly every sub-optimal knockout set. In the sub-optimal sets, the growth rate is closer to the wild-type growth rate of 0.9219 h^{-1} but the succinic acid production is lower than in the best sets presented in **Table 3**. In the case of fumarase, it was present in 58 of the 100 top knockout sets, which includes the best and sub-optimal sets. The average succinic acid production and growth rate for the sets that contained fumarase were 1.6777 mmol $gDW^{-1}h^{-1}$ and 0.5731 h^{-1} respectively. This is further demonstrated for each of the top ten reaction knockouts in **Table 5.** In general, if a reaction knockout has a higher average succinic acid production rate, it has a lower growth rate due to the majority of the carbon available to the *E. coli* being shunted towards the succinic acid synthesis rather than production of biomass. The genes catalyzing the reactions that were recommended to be knocked out are shown in **Table 4**.

Table 5. The average succinic acid production rate and growth rate for all knockout sets containing each reaction

Reaction Name	Average Succinic Acid Production (mmol $gDW^{-1}h^{-1}$)	Average Growth Rate (h^{-1})
FUM	1.6777	0.5731
SUCD4	2.6715	0.2210
SUCD1i	2.7725	0.2096
PPCDC	0.8838	0.8434
ACt2r	6.1650	0.3074
EX_ac_(e)	3.6132	0.1908
PNTK	0.8857	0.8452
PANTS	0.8863	0.8453
EX_leu_L_(e)	2.6929	0.3202
G6PDH2r	4.3057	0.1324

5. CONCLUSION

This work focused on implementation of STEER-ACO in order to determine the best reaction knockouts to be performed for overproduction of succinic acid in *E. coli* model *iJR904*. This algorithm differs from the previous ACO/MoMA hybrid methods developed by *Chong et. al.* in that while the solution space is searched in a completely stochastic manner in the first iterations of the algorithm, it refines the search space over time towards favorable solutions or edges. This is performed with a bias towards favorable single reaction knockouts determined in the initialization phase of the algorithm, as well as a probability of local search that decays over time. The top five predicted best knockouts in five sets of twenty runs not only had a higher succinic acid production rate, but a higher growth rate as well when compared to reaction knockouts reported from *Chong et. al.*. This is due in part by using the growth rate as a heuristic factor in determining the amount of pheromone to be deposited on a solution, as well as the bias towards favorable knockouts as time goes on. This allows for solutions that return a high succinic acid production rate and a reasonable growth rate. It also allows for identification of sub-optimal solutions that return a moderate succinic acid production rate and growth rate close to the wild type, which is important if growth rate is an important factor.

In addition, good solutions can be converged upon in just over eight minutes for up to a maximum of five reaction knockouts. The methods described in this work can be applied to any flux of interest, such as H_2 production for biofuel or API's for pharmaceutical use. Furthermore, predictions made by STEER-ACO may be used as a guideline for wet laboratory experiments to guide overproduction of a molecule or biomaterial of interest, especially because custom reaction and constraints easily be inserted into the *in silico* model and replicated *in vitro*.

This algorithm is still in its early stages of development. In the future, the algorithm will be further optimized for speed and performance. This will be accomplished by parallelization across multiple CPU's, as well as utilizing PyCUDA for GPU programming, which can achieve 20x-2000x performance increase [2]. This will eliminate the need for any preprocessing of the model to cut down computation time, and allow for more ants and iterations for exhaustion of the search space and convergence on best solutions.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1137249.

7. REFERENCE

[1] CPLEX Optimizer. In *CPLEX Optimization Studio* IBM.

[2] 2015. GPU Accelerated Computing with Python. In *CUDA ZONE* NVIDIA.

[3] Bautista, E.J., Zinski, J., Szczepanek, S.M., Johnson, E.L., Tulman, E.R., Ching, W.M., Geary, S.J., and Srivastava, R., 2013. Semi-automated Curation of Metabolic Models via Flux Balance Analysis: A Case Study with Mycoplasma gallisepticum. *PLoS Comput Biol 9*, 9 (Sep), e1003208. DOI= http://dx.doi.org/10.1371/journal.pcbi.1003208.

[4] Becker, S.A., Feist, A.M., Mo, M.L., Hannum, G., Palsson, B.O., and Herrgard, M.J., 2007. Quantitative prediction of cellular metabolism with constraint-based

models: the COBRA Toolbox. *Nat Protoc 2*, 3, 727-738. DOI= http://dx.doi.org/10.1038/nprot.2007.99.

[5] Burgard, A.P., Pharkya, P., and Maranas, C.D., 2003. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol Bioeng 84*, 6 (Dec 20), 647-657.

[6] Chong, S.K., Mohamad, M.S., Mohamed Salleh, A.H., Choon, Y.W., Chong, C.K., and Deris, S., 2014. A hybrid of ant colony optimization and minimization of metabolic adjustment to improve the production of succinic acid in Escherichia coli. *Comput Biol Med 49*(Jun), 74-82. DOI= http://dx.doi.org/10.1016/j.compbiomed.2014.03.011.

[7] Dorigo, M., Birattari, M., and Stutzle, T., 2006. Ant colony optimization. *Computational Intelligence Magazine, IEEE 1*, 4, 28-39.

[8] Ebrahim, A., Lerman, J.A., Palsson, B.O., and Hyduke, D.R., 2013. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Syst Biol 7*, 74. DOI= http://dx.doi.org/10.1186/1752-0509-7-74.

[9] Feist, A.M. and Palsson, B.O., 2008. The growing scope of applications of genome-scale metabolic reconstructions using Escherichia coli. *Nat Biotechnol 26*, 6 (Jun), 659-667. DOI= http://dx.doi.org/10.1038/nbt1401.

[10] Keasling, J.D., 2010. Manufacturing molecules through metabolic engineering. *Science 330*, 6009 (Dec 3), 1355-1358. DOI= http://dx.doi.org/10.1126/science.1193990.

[11] Latendresse, M., Krummenacker, M., Trupp, M., and Karp, P.D., 2012. Construction and completion of flux balance models from pathway databases. *Bioinformatics 28*, 3 (Feb 1), 388-396. DOI= http://dx.doi.org/10.1093/bioinformatics/btr681.

[12] Ohno, S., Furusawa, C., and Shimizu, H., 2013. In silico screening of triple reaction knockout Escherichia coli strains for overproduction of useful metabolites. *J Biosci Bioeng 115*, 2 (Feb), 221-228. DOI= http://dx.doi.org/10.1016/j.jbiosc.2012.09.004.

[13] Orth, J.D., Thiele, I., and Palsson, B.O., 2010. What is flux balance analysis? *Nat Biotechnol 28*, 3 (Mar), 245-248. DOI= http://dx.doi.org/10.1038/nbt.1614.

[14] Palsson, B.O., 2006. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, New York.

[15] Price, N.D., Reed, J.L., and Palsson, B.O., 2004. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat Rev Microbiol 2*, 11 (Nov), 886-897.

[16] Reed, J.L. and Palsson, B.O., 2003. Thirteen years of building constraint-based in silico models of Escherichia coli. *J Bacteriol 185*, 9 (May), 2692-2699.

[17] Reed, J.L., Vo, T.D., Schilling, C.H., and Palsson, B.O., 2003. An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). *Genome Biol 4*, 9, R54.

[18] Schellenberger, J., Park, J.O., Conrad, T.M., and Palsson, B.O., 2010. BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics 11*, 213. DOI= http://dx.doi.org/10.1186/1471-2105-11-213.

[19] Schilling, C.H., Edwards, J.S., Letscher, D., and Palsson, B.O., 2000. Combining pathway analysis with flux balance analysis for the comprehensive study of metabolic systems. *Biotechnol Bioeng 71*, 4, 286-306.

[20] Segre, D., Vitkup, D., and Church, G.M., 2002. Analysis of optimality in natural and perturbed metabolic networks. *Proc Natl Acad Sci U S A 99*, 23 (Nov 12), 15112-15117. DOI= http://dx.doi.org/10.1073/pnas.232349399.

[21] Shuler, M.L. and Kargi, F., 2001. *Bioprocess Engineering: Basic Concepts*. P T R Prentice Hall, Englewood Cliffs.

[22] Solnon, C., 2010. *Ant colony optimization and constraint programming*. Wiley Online Library.

[23] Stephanopoulos, G.N., Aristidou, A.A., and Nielsen, J., 1998. *Metabolic Engineering: Principles and Methodologies*. Academic Press, San Diego.

[24] Tran, K.T., Maeda, T., Sanchez-Torres, V., and Wood, T.K., 2015. Beneficial knockouts in Escherichia coli for producing hydrogen from glycerol. *Appl Microbiol Biotechnol*(Jan 8). DOI= http://dx.doi.org/10.1007/s00253-014-6338-7.

[25] Zeikus, J., Jain, M., and Elankovan, P., 1999. Biotechnology of succinic acid production and markets for derived industrial products. *Appl Microbiol Biotechnol 51*, 5, 545-552.