Model-guided Analysis of Plant Metabolism and Design of Metabolic Engineering Strategies

Jiun Y. Yen

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
In
Biological Systems Engineering

Ryan S. Senger
Glenda E. Gillaspy
Chenming Zhang
David R. Bevan

February 16, 2017
Blacksburg, Virginia

Keywords: Genome-scale model, Metabolic engineering, Plant, *Arabidopsis thaliana*, Energy metabolism, Flux balance analysis

Model-guided Analysis of Plant Metabolism and Design of Metabolic Engineering Strategies

Jiun Y. Yen

ABSTRACT

Advances in bioinformatics and computational biology have enabled integration of an enormous amount of known biological interactions. This has enabled researchers to use models and data to design experiments and guide new discovery as well as test for consistency. One such computational method is constraint-based metabolic flux modeling. This is performed using **ge**nome-scale metabolic **m**odels (GEMs) that are a collection of biochemical reactions, derived from a genome's annotation. This type of flux modeling enables prediction of net metabolite conversion rates (metabolic fluxes) to help understand metabolic activities under specific environmental conditions. It can also be used to derive metabolic engineering strategies that involve genetic manipulations. Over the past decade, GEMs have been constructed for several different microbes, plants, and animal species. Researchers have also developed advanced algorithms to use GEMs to predict genetic modifications for the overproduction of biofuel and valuable commodity chemicals. Many of the predictive algorithms for microbes were validated with experimental results and some have been applied industrially. However, there is much room for improvement. For example, many algorithms lack straight-forward predictions that truly help non-computationally oriented researchers understand the predicted necessary metabolic modifications. Other algorithms are limited to simple genetic manipulations due to computational demands. Utilization of GEMs and flux-based modeling to predict *in vivo* characteristics of multicellular organisms has also proven to be challenging. Many researchers have created unique frameworks to use plant GEMs to hypothesize complex cellular interactions,

such as metabolic adjustments in rice under variable light intensity and in developing tomato fruit. However, few quantitative predictions have been validated experimentally in plants. This research demonstrates the utility of GEMs and flux-based modeling in both metabolic engineering and analysis by tackling the challenges addressed previously with alternative approaches. Here, a novel predictive algorithm, **N**ode-**R**eward **Opt**imization (NR-Opt) toolbox, was developed. It delivers concise and accurate metabolic engineering designs (i.e. genetic modifications) that can truly improve the efficiency of strain development. As a proof-of-concept, the algorithm was deployed on GEMs of *E. coli* and *Arabidopsis thaliana*, and the predicted metabolic engineering strategies were compared with results of well-accepted algorithms and validated with published experimental data. To demonstrate the utility of GEMs and flux-based modeling in analyzing plant metabolism, specifically its response to changes in the signaling pathway, a novel modeling framework and analytical pipeline were developed to simulate changes of growth and starch metabolism in Arabidopsis over multiple stages of development. This novel framework was validated through simulation of growth and starch metabolism of Arabidopsis plants overexpressing sucrose non-fermenting related kinase 1.1 (SnRK1.1). Previous studies suggest that SnRK1.1 may play a critical signaling role in plant development and starch level (a critical carbon source for plant night growth). It has been shown that overexpressing of SnRK1.1 in Arabidopsis can delay vegetative-to-reproductive transition. Many studies on plant development have correlated the delay in developmental transition to reduction in starch turnover at night. To determine whether starch played a role in the delayed developmental transition in SnRK1.1 overexpressor plants, starch turnover was simulated at multiple developmental stages. Simulations predicted no reduction in starch turnover prior to developmental transition. Predicted results were experimentally validated, and the predictions

were in close agreement with experimental data. This result further supports previous data that SnRK1.1 may regulate developmental transition in Arabidopsis. This study further validates the utility of GEMs and flux-based modeling in guiding future metabolic research.

Model-guided Analysis of Plant Metabolism and Design of Metabolic Engineering Strategies

Jiun Y. Yen

GENERAL AUDIENCE ABSTRACT

Recent advances in genetic and biochemical studies revealed the incredible complexity of cells, which generated interests in using computers to aid whole cell analyses and design cell engineering strategies to overproduce valuable commodity chemicals, such as biofuel, medicines, polymers, and many industrial materials. In order to use computers to study cells, current knowledge of cellular machinery is converted into mathematical models, such as genome-scale metabolic models. Genome-scale metabolic models are used to simulate the rates of chemical events in cells, which helps researchers predict cellular outputs of interest, such as growth rate and chemical synthesis rates. Combining genome-scale metabolic models with sophisticated computer algorithms, researchers can simulate numerous cell engineering experiments and select a few candidates to test physically, which can reduce cost and research time significantly. This computational technique has been well validated in microorganisms, such as *E. coli* and yeast; however, the ability to simulate cellular chemistry accurately in plants remains a challenge, which was a goal in my research. In addition, my research also aimed to reduce the inefficiencies in previous cell engineering design algorithms. I was able to develop a novel genome-scale model framework that enabled accurate simulation of plant growth and changes of starch content over time. I also developed a new computer algorithm that could significantly improve the efficiency in designing cell engineering strategies.

ACKNOWLEDGEMENTS

This might be the only section of this document that I can be myself. In that case, I want to tell whomever reading this, possibly my advisers, committee members, graduate school, proof-readers, parents, girlfriend, friends and future self, that thank you for all your support throughout the years and I love you. It has been a long and treacherous road filled with boredom, failure, rage, despair, and a sprinkle of excitement, success, and joy. Those of us in academia choose this path in pursuit of intellectual satisfaction and human advancement. We may have all wondered at some point whether we have chosen poorly. For me, I have a mixed bag of feelings that is best summarized as "WOW!" This experience that you have given me is certainly life changing. Statistically, it is just as life changing as any other experience that I could have had, but that is not the point. Math aside, I am glad that I pursued this experience, and I am so glad that I had all of you with me. I cannot wait to share whatever that lies ahead with you. I hope they will be just as difficult as this.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GEM	Genome-scale model
ACOMoMA	Ant colony optimization with MOMA
ADH	Alcohol dehydrogenase
ATP	Adenosine triphosphate
BAFBA	Bees algorithm and FBA
BDO	1,4-butanediol
BPCY	Biomass-product coupled yield
CESA	Cellulose synthase
CiED	Cipher of evolutionary design
CosMos	Continuous modifications
dFBA	Dynamic flux balance analysis
DIC	Dicarboxylate carrier
EOD	End-of-day
EON	End-of-night
FBA	Flux balance analysis
FBrAtio	Flux balance analysis with flux ratio
FDCA	Flux distribution comparison analysis
FSEOF	Flux scanning based on enforce objective flux
FVA	Flux variability analysis
hSOD	Human superoxide dismutase
KD	knock-down
KIKO	Knock-in/knockout
KO	Knockout
LDH	Lactate dehydrogenase
LPDA	Pyruvate dehydrogenase
MAGE	Multiplex automated genome engineering
MDH	Malate dehydrogenase
MMM	Multiscale metabolic modeling
MOMA	Minimization of metabolic adjustment
NDK1	Nucleoside diphosphate kinase

NR-Opt	Node-reward optimization
OAC	Oxaloacetate carrier
OX	Overexpression
PCR	Polymerase chain reaction
pFBA	Parsimonious flux balance analysis
PFL	Pyruvate formate lyase
RBS	Ribosomal binding site
RER	Relative leaf expansion rate
ROOM	Regulatory on/off minimization
SnRK	Sucrose non-fermenting related kinase
sRNA	Small RNA
tRNA	Transfer RNA
UDP	Uridine diphosphate
UGPase	UDP-glucose pyrophosphorylase
UTP	Uridine triphosphate
WT	Wild-type

CHAPTER 1

INTRODUCTION

Genome-scale models and their utilities

A **ge**nome-scale metabolic flux **m**odel (GEM) is a mathematical representation of the metabolic network of an organism. It is a network of biochemical reactions assembled from annotated enzyme-coding genes, and it has been used traditionally to aid the analysis of metabolism and design metabolic engineering strategies. Since its initial introduction in 1999 (Schilling et al., 1999), GEM reconstructions have been built for many well-studied organisms, including *E. coli* (Edwards and Palsson, 2000), *Helicobacter pylori* (Schilling et al., 2002; Thiele et al., 2005), *Saccharomyces cerevisiae* (Förster et al., 2003; Herrgård et al., 2008), *Mus musculus* (Sheikh et al., 2005), *Arabidopsis thaliana* (Cheung et al., 2013; de Oliveira Dal'Molin et al., 2010; Mintz-Oron et al., 2012; Poolman et al., 2009), and *Homo sapiens* (Duarte et al., 2007). A critical component of a GEM is the biomass equation, which accounts for all known biomass constituents and their fractional contributions to the overall cellular biomass (Thiele and Palsson, 2010). The biomass equation often includes carbohydrates, amino acids, lipids, nucleic acids, cell wall polymers, ions, and maintenance ATP, which is the energy requirement to drive growth and maintenance (Poolman et al., 2009). The most straight-forward method to compute flux distribution within a metabolic network is to perform **f**lux **b**alance **a**nalysis (FBA) given an objective function (usually maximized cell growth) and constraints (i.e. reaction reversibility based on thermodynamics and observed influx/efflux of substrate/products). FBA functions by first assuming cellular metabolism is at a pseudo-steady state, where changes in cell development over time are slow compared to rates of catabolic and anabolic flux (Orth et al., 2010). To obtain

meaningful information from FBA calculations, a high-quality GEM with careful manual curation is required. Comprehensive protocols for construction high-quality GEMs and resources to automatically generate scaffold GEMs are currently available (Büchel et al., 2013; Devoid et al., 2013; King et al., 2015; Thiele and Palsson, 2010). Another powerful computational method to predict flux distribution in a GEM is the minimization of metabolic adjustment (MOMA) algorithm. MOMA is used to predict the immediate flux adaptation in response to metabolic changes, such as due to genetic manipulations (i.e. gene overexpression or knockout), and it has been used to successfully predict the results of metabolic engineering effort (Agren et al., 2013; Park et al., 2007; Segre et al., 2002). Unlike FBA, MOMA recognizes that achieving global flux optimality may require evolution and results may not be immediate; thus, MOMA assumes that a mutant cell will attempt to achieve a sub-optimal metabolic flux distribution with the smallest adjustments from wild-type (WT) (Segre et al., 2002). This concept has been validated in *E. coli* with experimental fluxomic analyses (Schuetz et al., 2012). MOMA has also been suggested as a valid approach to predict plant metabolic flux and metabolic engineering strategies (Allen et al., 2009; Yen et al., 2015).

GEMs have been used to aid research in systems and synthetic biology. A recent review summarized the process of utilizing GEMs to design and examine metabolic engineering strategies experimentally (Yen et al., 2015). Microbes, with emphasis on *E. coli*, have been the prime target to validate new approaches of using GEMs and flux-based modeling due to their well-understood metabolism, uniformity in cell cultures, and ease of experimental validation. The robustness of translating these approaches to plant models has been discussed thoroughly in the literature (Collakova et al., 2012; de Oliveira Dal'Molin and Nielsen, 2013). A novel approach utilized FBA with flux ratio constraints (an algorithm called "FBrAtio") to predict the

metabolic outcomes of genetic engineering strategies had been shown to perform well with plant GEMs (Yen et al., 2013). FBrAtio utilizes flux ratios to constrain the partition of flux of a metabolite through a metabolic network branch point, where multiple reactions are competing for that metabolite (McAnulty et al., 2012; Yen et al., 2013). The metabolic branch point where flux ratio is installed is termed a "node". Unlike conventional flux constraints, flux ratio constraints can redirect metabolism without limiting flux to a specific value. FBrAtio has been shown to be effective in predicting metabolic engineering strategies in multiple species (McAnulty et al., 2012; Yen et al., 2013). The limitation of FBrAtio is it requires *a priori* knowledge on potential nodes or it requires sampling of the metabolic network to identify critical nodes for metabolic engineering. The research in this dissertation will develop and utilize a predictive algorithm inspired by the FBrAtio constraint, termed **N**ode-**R**eward **Opt**imization (NR-Opt), to rapidly design concise metabolic engineering strategies for increasing target chemical yield in a host organism.

The importance of integrating regulatory networks in GEM reconstructions is widely accepted; however, methods of implementation are still being refined. Most current approaches directly convert gene expression profile into flux constraints (Blazier and Papin, 2012). A recent study showed that none of these approaches outperform a simple FBA with a growth constraint (Machado and Herrgård, 2014). For these reasons, this research developed and validated a more conservative approach to utilizing only metabolite profiles and known plant metabolic interactions (i.e. diurnal pattern of starch metabolism) to study metabolic regulation and signaling.

Motivation to develop new metabolic engineering design algorithms

The conventional process to model-guided metabolic engineering is (i) computer automated design of metabolic engineering strategies, (ii) manual assessment of the list of designs, (iii) conversion of designs into genetic engineering strategies, and (iv) experimental implementation and validation of the selected designs. The availability of genetic engineering tools, (i.e. for gene knockout, overexpression, or down-regulation) is considered when developing design algorithms. A metabolic engineering strategy can require multiple combinations of genetic engineering tools. Currently, there are many well-developed algorithms that use GEMs to design metabolic engineering strategies for the overproduction of a target chemical in a host cell, such as OptKnock, OptGene, RobustKnock, ReacKnock, BAFBA, OptForce, and EMILiO (Burgard et al., 2003; Choon et al., 2014; Patil et al., 2005; Ranganathan et al., 2010; Tepper and Shlomi, 2010; Xu et al., 2013; Yang et al., 2011). The utility of these tools has been validated with experimental results. Until now, the aim of such design algorithms has been to generate as many designs as possible to provide researchers with multiple options. The size of a typical list of designs can be on the scale of hundreds, which can make manual evaluation labor intensive. Manual evaluation of the predicted designs allows researchers to get a better sense of the necessary metabolic modifications. However, additional genetic modifications to improve target chemical yield is often necessary, and these requires expert analysis (Yim et al., 2011). A design algorithm that predicts fewer but more accurate metabolic engineering strategies would improve the efficiency of the overall metabolic engineering process. In many cases, multiple iterations of computational predictions are necessary; thus, a fast design algorithm is favorable. With speed, accuracy, and conciseness in mind, the NR-Opt algorithm

was developed to predict a reasonable list of concise and accurate metabolic engineering strategies quickly. This will truly improve the workflow in the metabolic engineering process.

AraGEM – an Arabidopsis thaliana genome-scale model

The first genome-scale model reconstruction of the Arabidopsis genome was published in 2009, and it contains 1,406 reactions and 1,253 metabolites (Poolman et al., 2009). Most of the reactions were gathered from the AraCyc database and compartmentalization was limited. Another independent GEM reconstruction of Arabidopsis, AraGEM, was immediately published (de Oliveira Dal'Molin et al., 2010). Similar to the previous model, AraGEM models primary metabolism with 1,601 reactions associated with 1,404 annotated genes collected from public databases. Unlike the previous model, AraGEM is compartmentalized into 5 organelles: cytoplasm, mitochondrion, plastid, peroxisome, and vacuole. AraGEM was curated with biomass composition and growth rate measured in cultured protoplasts to model cells undergoing photosynthesis, photorespiration, and respiration. AraGEM was used to model photon utilization and energy distribution in metabolism undergoing photosynthesis and photorespiration well (de Oliveira Dal'Molin et al., 2010). It was also used to model redox metabolism to achieve the optimal growth and maintenance in non-photosynthetic cells (de Oliveira Dal'Molin et al., 2010).

Although AraGEM was shown to perform well in modeling multiple Arabidopsis pathways, it is still confined to modeling metabolism at pseudo-steady state when FBA is used (Collakova et al., 2012). In this research, AraGEM was used to model metabolism of plants grown in soil, which is significantly different from using protoplasts as the model plant material. A new modeling framework was developed to simulate non-steady state metabolic changes.

SnRK1 regulation of plant metabolism

Without mobility, plants must adjust metabolism, nutrient partitioning, and developmental programming via complex regulatory mechanisms in response to environmental cues (Robaglia et al., 2012). Sucrose non-Fermenting Related Kinase 1 (SnRK1) plays a central role in the global regulation of plant carbon metabolism (Li and Sheen, 2016). In Arabidopsis, there are three genes in the SnRK1 gene family, which are the functional SnRK1.1 and SnRK1.2 and the unexpressed SnRK1.3 (Baena-González et al., 2007; Hrabak et al., 2003; Williams et al., 2014). Of these, SnRK1.1 has been shown to be the predominantly expressed isoform in most plant tissue (Jossier et al., 2009; Williams et al., 2014). In vitro experiments have shown that plant SnRK1 can phosphorylate and inactivate four metabolic enzymes: (i) 3-hydroxymethyl 3-methylglutaryl-CoA reductase (Dale et al., 1995), (ii) sucrose phosphate synthase, (iii) nitrate reductase (Sugden et al., 1999), and (iv) trehalose phosphate synthase 5 (Harthill et al., 2006). These enzymes are critical to sucrose biosynthesis, nitrogen assimilation for amino acid biosynthesis, and signaling the regulation of plant metabolism and development (Halford et al., 2003; Harthill et al., 2006; Jossier et al., 2009; Tsai and Gazzarrini, 2012). SnRK1 has also been shown to phosphorylate stress response proteins, such as the small heat shock protein 17 (Slocombe et al., 2004). In addition, SnRK1 can regulate the transcription of many genes, including the α-amylase and SuSy genes, which are responsible for starch and sucrose degradation (Laurie et al., 2003; Purcell et al., 1998). It has been suggested that SnRK1 may up-regulate SuSy and ADP-glucose pyrophosphorylase under high sucrose to generate starch and activate sucrose degradation (Fu and Park, 1995; Geigenberger, 2003). A recent study in Arabidopsis seedlings treated with different sugars showed that sucrose, glucose, and fructose can lower SnRK1.1 gene expression, and trehalose can dramatically increase SnRK1.2 gene

expression (Williams et al., 2014). It has been suggested that studies of SnRK1 may offer insights in designing plant engineering strategies to improve stress tolerance and crop yield (Coello et al., 2011). The overexpression of SnRK1.1 and SnRK1.2 in Arabidopsis generated plants with increased developmental rate and biomass, which suggested SnRK1 overexpressor plants to be valuable to metabolic engineering (Baena-González et al., 2007; Williams et al., 2014).

Motivation to study metabolism of SnRK1.1 overexpression plants with modeling

As emphasized in the previous section, plant have a complex signaling pathway that can alter metabolism (Yamaguchi-Shinozaki and Shinozaki, 2006). In addition, plants with genetic modifications in the signaling pathway, such as overexpression of SnRK1.1, can manifest different phenotypes in different stages of plant development (Gazzarrini and Tsai, 2014; Williams et al., 2014). As concluded in the previous section, there is great interest in understanding the role of SnRK1.1 in leaf metabolism over plant development. However, studying the metabolic roles of genes in the signaling pathway is challenging due to the difficulties in dissecting metabolic and regulatory events, and further complicated by the temporal aspect (Sheen, 2014). It is possible to utilize the pseudo-steady state assumption in flux-based modeling to indicate whether a metabolic event is a mass-balancing interaction or if it involve regulation. Flux-based modeling with GEMs have previously been used to identify organelle interactions in rice leaves that can be modeled with only mass-balancing (Poolman et al., 2013). Flux-based modeling with GEMs has also been shown to model tomato fruit development metabolism well (Colombié et al., 2015). Unlike in fruits, metabolic changes in vegetative tissues are much faster; thus, pseudo-steady state assumption does not apply (Allen et al., 2009; Collakova et al., 2012). Previous studies that imposed pseudo-steady state assumptions

to model vegetative tissue could only predicted results that are qualitatively accurate (Gomes de Oliveira Dal'Molin et al., 2015; Grafahrend-Belau et al., 2013; Poolman et al., 2013). To gain quantitative accuracy would require modeling at non-steady state. Modeling non-steady state metabolic changes would need a framework similar to kinetic models, which requires enzyme kinetics data. Such framework is currently inapplicable at the genome-scale (Smallbone et al., 2010). With these challenges in mind, the goals of this study are to develop a novel genome-scale modeling framework that can: (i) model metabolic changes over plant development, (ii) model non-steady state metabolite changes, and (iii) calculate quantitative changes accurately. To address these goals, a novel dynamic flux-based GEM modeling framework was developed to model non-steady state changes of a target metabolic interaction. The new framework was used to model changes of starch level in SnRK1.1 overexpressor plants at multiple stages of development. Experimental validations revealed unprecedented quantitative accuracy.

Chapters and organization of this dissertation

The three main chapters of this dissertation – chapter II to IV are manuscripts that have been accepted for publication or are in preparation for submission.

Chapter 2: Designing metabolic engineering strategies with genome-scale metabolic flux modeling.

This published review discusses the utilities of all the well-accepted metabolic engineering design algorithms that are used with GEMs.

Chapter 3: Predicting metabolic engineering strategies with the Node-Reward-Optimization toolbox.

This manuscript introduces a novel metabolic engineering design algorithm to rapidly predict concise metabolite engineering strategies. The algorithms implemented in the Node-

Reward Optimization programs are described in detail. To validate its utility, the Node-Reward Optimization programs were used to design metabolic engineering strategies to overproduce 1,4-butanediol in *E. coli* and cellulose in *Arabidopsis thaliana*. Predicted design strategies were cross-validated with published data.

Chapter 4: Model-guided analysis of SnRK1.1 overexpression in Arabidopsis predicts significant changes in starch metabolism over plant development

This manuscript demonstrates the utility of flux-based modeling of plant metabolism and introduces a novel framework to simulate non-steady state starch metabolism. This novel framework was used to investigate whether the delayed plant developmental transition in SnRK1.1 overexpressor plants is due to changes in starch turnover rate. Model simulation of growth and starch changes are validated experimentally. The results showed that SnRK1.1 may regulate plant developmental transition independent of starch turnover rate.

Chapter 5: Conclusions and future directions

REFERENCES

Agren, R., Otero, J. M., Nielsen, J., 2013. Genome-scale modeling enables metabolic engineering of Saccharomyces cerevisiae for succinic acid production. Journal of industrial microbiology & biotechnology. 40**,** 735-747.

Allen, D. K., Libourel, I. G. L., Shachar-Hill, Y., 2009. Metabolic flux analysis in plants: coping with complexity. Plant, cell & environment. 32**,** 1241-1257.

Baena-González, E., Rolland, F., Thevelein, J. M., Sheen, J., 2007. A central integrator of transcription networks in plant stress and energy signalling. Nature. 448**,** 938-942.

Blazier, A., Papin, J., 2012. Integration of expression data in genome-scale metabolic network reconstructions. Frontiers in physiology. 3.

Büchel, F., Rodriguez, N., Swainston, N., Wrzodek, C., Czauderna, T., Keller, R., Mittag, F., Schubert, M., Glont, M., Golebiewski, M., 2013. Path2Models: large-scale generation of computational models from biochemical pathway maps. BMC systems biology. 7**,** 116.

Burgard, A. P., Pharkya, P., Maranas, C. D., 2003. Optknock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. Biotechnology and bioengineering. 84**,** 647-657.

Cheung, C. Y. M., Williams, T. C. R., Poolman, M. G., Fell, D. A., Ratcliffe, R. G., Sweetlove, L. J., 2013. A method for accounting for maintenance costs in flux balance analysis improves the prediction of plant cell metabolic phenotypes under stress conditions. The plant journal. 75**,** 1050-1061.

Choon, Y. W., Mohamad, M. S., Deris, S., Illias, R. M., Chong, C. K., Chai, L. E., Omatu, S., Corchado, J. M., 2014. Differential bees flux balance analysis with OptKnock for in silico microbial strains optimization. PloS one. 9**,** e102744.

Coello, P., Hey, S. J., Halford, N. G., 2011. The sucrose non-fermenting-1-related (SnRK) family of protein kinases: potential for manipulation to improve stress tolerance and increase yield. Journal of experimental botany. 62**,** 883-893.

Collakova, E., Yen, J. Y., Senger, R. S., 2012. Are we ready for genome-scale modeling in plants? Plant science. 191–192**,** 53-70.

Colombié, S., Nazaret, C., Bénard, C., Biais, B., Mengin, V., Solé, M., Fouillen, L., Dieuaide-Noubhani, M., Mazat, J. P., Beauvoit, B., 2015. Modelling central metabolic fluxes by constraint-based optimization reveals metabolic reprogramming of developing Solanum lycopersicum (tomato) fruit. The plant journal. 81**,** 24-39.

Dale, S., Wilson, W. A., Edelman, A. M., Hardie, D. G., 1995. Similar substrate recognition motifs for mammalian AMP-activated protein kinase, higher plant HMG-CoA reductase kinase-A, yeast SNF1, and mammalian calmodulin-dependent protein kinase I. FEBS letters. 361**,** 191-195.

de Oliveira Dal'Molin, C. G., Quek, L. E., Palfreyman, R. W., Brumbley, S. M., Nielsen, L. K., 2010. AraGEM, a genome-scale reconstruction of the primary metabolic network in Arabidopsis. Plant physiology. 152**,** 579-89.

de Oliveira Dal'Molin, C. G., Nielsen, L. K., 2013. Plant genome-scale metabolic reconstruction and modelling. Current opinion in biotechnology. 24**,** 271-277.

Devoid, S., Overbeek, R., DeJongh, M., Vonstein, V., Best, A. A., Henry, C., 2013. Automated genome annotation and metabolic model reconstruction in the SEED and Model SEED. Systems metabolic engineering. Springer, pp. 17-45.

Duarte, N. C., Becker, S. A., Jamshidi, N., Thiele, I., Mo, M. L., Vo, T. D., Srivas, R., Palsson, B. Ø., 2007. Global reconstruction of the human metabolic network based on genomic and bibliomic data. Proceedings of the national academy of sciences. 104**,** 1777-1782.

Edwards, J. S., Palsson, B. O., 2000. The Escherichia coli MG1655 in silico metabolic genotype: Its definition, characteristics, and capabilities. Proceedings of the national academy of sciences. 97**,** 5528-5533.

Förster, J., Famili, I., Fu, P., Palsson, B. Ø., Nielsen, J., 2003. Genome-scale reconstruction of the Saccharomyces cerevisiae metabolic network. Genome research. 13**,** 244-253.

Fu, H., Park, W. D., 1995. Sink-and vascular-associated sucrose synthase functions are encoded by different gene classes in potato. The plant cell. 7**,** 1369-1385.

Gazzarrini, S., Tsai, A. Y.-L., 2014. Trehalose-6-phosphate and SnRK1 kinases in plant development and signaling: the emerging picture. Frontiers in plant science. 5**,** 119.

Geigenberger, P., 2003. Regulation of sucrose to starch conversion in growing potato tubers. Journal of experimental botany. 54**,** 457-465.

Gomes de Oliveira Dal'Molin, C., Quek, L.-E., Saa, P. A., Nielsen, L. K., 2015. A multi-tissue genome-scale metabolic modeling framework for the analysis of whole plant systems. Frontiers in plant science. 6**,** 4.

Grafahrend-Belau, E., Junker, A., Eschenröder, A., Müller, J., Schreiber, F., Junker, B. H., 2013. Multiscale metabolic modeling: dynamic flux balance analysis on a whole-plant scale. Plant physiology. 163**,** 637-647.

Halford, N. G., Hey, S., Jhurreea, D., Laurie, S., McKibbin, R. S., Paul, M., Zhang, Y., 2003. Metabolic signalling and carbon partitioning: role of Snf1-related (SnRK1) protein kinase. Journal of experimental botany. 54**,** 467-475.

Harthill, J. E., Meek, S. E., Morrice, N., Peggie, M. W., Borch, J., Wong, B. H., MacKintosh, C., 2006. Phosphorylation and 14-3-3 binding of Arabidopsis trehalose-phosphate synthase 5 in response to 2-deoxyglucose. The plant journal. 47**,** 211-223.

Herrgård, M. J., Swainston, N., Dobson, P., Dunn, W. B., Arga, K. Y., Arvas, M., Blüthgen, N., Borger, S., Costenoble, R., Heinemann, M., 2008. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. Nature biotechnology. 26**,** 1155-1160.

Hrabak, E. M., Chan, C. W. M., Gribskov, M., Harper, J. F., Choi, J. H., Halford, N., Kudla, J., Luan, S., Nimmo, H. G., Sussman, M. R., Thomas, M., Walker-Simmons, K., Zhu, J.-K., Harmon, A. C., 2003. The Arabidopsis CDPK-SnRK Superfamily of Protein Kinases. Plant physiology. 132**,** 666-680.

Jossier, M., Bouly, J. P., Meimoun, P., Arjmand, A., Lessard, P., Hawley, S., Grahame Hardie, D., Thomas, M., 2009. SnRK1 (SNF1-related kinase 1) has a central role in sugar and ABA signalling in Arabidopsis thaliana. The plant journal. 59**,** 316-328.

King, Z. A., Lloyd, C. J., Feist, A. M., Palsson, B. O., 2015. Next-generation genome-scale models for metabolic engineering. Current opinion in biotechnology. 35**,** 23-29.

Laurie, S., McKibbin, R. S., Halford, N. G., 2003. Antisense SNF1-related (SnRK1) protein kinase gene represses transient activity of an α-amylase (α-Amy2) gene promoter in cultured wheat embryos. Journal of experimental botany. 54**,** 739-747.

Li, L., Sheen, J., 2016. Dynamic and diverse sugar signaling. Current opinion in plant biology. 33**,** 116-125.

Machado, D., Herrgård, M., 2014. Systematic evaluation of methods for integration of transcriptomic data into constraint-based models of metabolism. PLoS computational biology. 10**,** e1003580.

McAnulty, M. J., Yen, J. Y., Freedman, B. G., Senger, R. S., 2012. Genome-scale modeling using flux ratio constraints to enable metabolic engineering of clostridial metabolism in silico. BMC systems biology. 6**,** 42.

Mintz-Oron, S., Meir, S., Malitsky, S., Ruppin, E., Aharoni, A., Shlomi, T., 2012. Reconstruction of Arabidopsis metabolic network models accounting for subcellular compartmentalization and tissue-specificity. Proceedings of the national academy of sciences. 109**,** 339-344.

Orth, J. D., Thiele, I., Palsson, B. O., 2010. What is flux balance analysis? Nature biotech. 28**,** 245-248.

Park, J. H., Lee, K. H., Kim, T. Y., Lee, S. Y., 2007. Metabolic engineering of Escherichia coli for the production of L-valine based on transcriptome analysis and in silico gene knockout simulation. Proceedings of the national academy of sciences. 104**,** 7797-7802.

Patil, K. R., Rocha, I., Förster, J., Nielsen, J., 2005. Evolutionary programming as a platform for in silico metabolic engineering. BMC bioinformatics. 6**,** 1.

Poolman, M. G., Kundu, S., Shaw, R., Fell, D. A., 2013. Responses to light intensity in a genome-scale model of rice metabolism. Plant physiology. 162**,** 1060-1072.

Poolman, M. G., Miguet, L., Sweetlove, L. J., Fell, D. A., 2009. A genome-scale metabolic model of Arabidopsis and some of its properties. Plant physiology. 151**,** 1570-1581.
Purcell, P. C., Smith, A. M., Halford, N. G., 1998. Antisense expression of a sucrose non-fermenting-1-related protein kinase sequence in potato results in decreased expression of sucrose synthase in tubers and loss of sucrose-inducibility of sucrose synthase transcripts in leaves. The plant journal. 14**,** 195-202.
Ranganathan, S., Suthers, P. F., Maranas, C. D., 2010. OptForce: an optimization procedure for identifying all genetic manipulations leading to targeted overproductions. PLoS computational biology. 6**,** e1000744.
Robaglia, C., Thomas, M., Meyer, C., 2012. Sensing nutrient and energy status by SnRK1 and TOR kinases. Current opinion in plant biology. 15**,** 301-307.
Schilling, C. H., Covert, M. W., Famili, I., Church, G. M., Edwards, J. S., Palsson, B. O., 2002. Genome-scale metabolic model of Helicobacter pylori 26695. Journal of bacteriology. 184**,** 4582-4593.
Schilling, C. H., Edwards, J. S., Palsson, B. O., 1999. Toward metabolic phenomics: analysis of genomic data using flux balances. Biotechnology progress. 15**,** 288-295.
Schuetz, R., Zamboni, N., Zampieri, M., Heinemann, M., Sauer, U., 2012. Multidimensional optimality of microbial metabolism. Science. 336**,** 601-604.
Segre, D., Vitkup, D., Church, G. M., 2002. Analysis of optimality in natural and perturbed metabolic networks. Proceedings of the national academy of sciences. 99**,** 15112-15117.
Sheen, J., 2014. Master regulators in plant glucose signaling networks. Journal of plant biology. 57**,** 67-79.
Sheikh, K., Förster, J., Nielsen, L. K., 2005. Modeling hybridoma cell metabolism using a generic genome-scale metabolic model of Mus musculus. Biotechnology progress. 21**,** 112-121.
Slocombe, S. P., Beaudoin, F., Donaghy, P. G., Hardie, D. G., Dickinson, J. R., Halford, N. G., 2004. SNF1-related protein kinase (snRK1) phosphorylates class I heat shock protein. Plant physiology and biochemistry. 42**,** 111-116.
Smallbone, K., Simeonidis, E., Swainston, N., Mendes, P., 2010. Towards a genome-scale kinetic model of cellular metabolism. BMC systems biology. 4**,** 6.
Sugden, C., Donaghy, P. G., Halford, N. G., Hardie, D. G., 1999. Two SNF1-related protein kinases from spinach leaf phosphorylate and inactivate 3-hydroxy-3-methylglutaryl-coenzyme A reductase, nitrate reductase, and sucrose phosphate synthase in vitro. Plant physiology. 120**,** 257-274.
Tepper, N., Shlomi, T., 2010. Predicting metabolic engineering knockout strategies for chemical production: accounting for competing pathways. Bioinformatics. 26**,** 536-543.
Thiele, I., Palsson, B. Ø., 2010. A protocol for generating a high-quality genome-scale metabolic reconstruction. Nature protocols. 5**,** 93-121.
Thiele, I., Vo, T. D., Price, N. D., Palsson, B. Ø., 2005. Expanded metabolic reconstruction of Helicobacter pylori (iIT341 GSM/GPR): an in silico genome-scale characterization of single-and double-deletion mutants. Journal of bacteriology. 187**,** 5818-5830.
Tsai, A. Y. L., Gazzarrini, S., 2012. AKIN10 and FUSCA3 interact to control lateral organ development and phase transitions in Arabidopsis. The plant journal. 69**,** 809-821.
Williams, S. P., Rangarajan, P., Donahue, J. L., Hess, J. E., Gillaspy, G. E., 2014. Regulation of Sucrose non-Fermenting Related Kinase 1 genes in Arabidopsis thaliana. Frontiers in plant science. 5.

Xu, Z., Zheng, P., Sun, J., Ma, Y., 2013. ReacKnock: identifying reaction deletion strategies for microbial strain optimization based on genome-scale metabolic network. PloS one. 8**,** e72150.

Yamaguchi-Shinozaki, K., Shinozaki, K., 2006. Transcriptional regulatory networks in cellular responses and tolerance to dehydration and cold stresses. Annual review of plant biology. 57**,** 781-803.

Yang, L., Cluett, W. R., Mahadevan, R., 2011. EMILiO: a fast algorithm for genome-scale strain design. Metabolic engineering. 13**,** 272-281.

Yen, J. Y., Nazem-Bokaee, H., Freedman, B. G., Athamneh, A. I. M., Senger, R. S., 2013. Deriving metabolic engineering strategies from genome-scale modeling with flux ratio constraints. Biotechnology journal. 8**,** 581-594.

Yen, J. Y., Tanniche, I., Fisher, A., Gillaspy, G., Bevan, D., Senger, R., 2015. Designing metabolic engineering strategies with genome-scale metabolic flux modeling. Advances in genomics and genetics. 7**,** 149-160.

Yim, H., Haselbeck, R., Niu, W., Pujol-Baxley, C., Burgard, A., Boldt, J., Khandurina, J., Trawick, J. D., Osterhout, R. E., Stephen, R., 2011. Metabolic engineering of Escherichia coli for direct production of 1, 4-butanediol. Nature chemical biology. 7**,** 445-452.

CHAPTER 2

DESIGNING METABOLIC ENGINEERING STRATEGIES WITH GENOME-SCALE METABOLIC FLUX MODELING

Jiun Y. Yen [1,2], Imen Tanniche [1], Amanda K. Fisher [1-3], Glenda E. Gillaspy [2], David R. Bevan [2,3], Ryan S. Senger [1]

[1] Department of Biological Systems Engineering, Virginia Tech, Blacksburg, VA, USA

[2] Department of Biochemistry, Virginia Tech, Blacksburg, VA, USA

[3] Genomics, Bioinformatics, and Computational Biology Interdisciplinary Programs, Virginia Tech, Blacksburg, VA, USA

Republished with permission of Dove Press, from Yen, J. Y., Tanniche, I., Fisher, A., Gillaspy, G., Bevan, D., Senger, R., Yen, J. Y., Tanniche, I., Fisher, A. K., Gillaspy, G. E., 2015. Designing metabolic engineering strategies with genome-scale metabolic flux modeling. Clinical Epidemiology. 7, 149-160. Permission conveyed through Copyright Clearance Center, Inc.

Advances in Genomics and Genetics

Dovepress
open access to scientific and medical research

Open Access Full Text Article

REVIEW

Designing metabolic engineering strategies with genome-scale metabolic flux modeling

Jiun Y Yen[1,2]
Imen Tanniche[1]
Amanda K Fisher[1–3]
Glenda E Gillaspy[2]
David R Bevan[2,3]
Ryan S Senger[1]

[1]Department of Biological Systems Engineering, [2]Department of Biochemistry, [3]Genomics, Bioinformatics, and Computational Biology Interdisciplinary Program, Virginia Tech, Blacksburg, VA, USA

Abstract: New in silico tools that make use of genome-scale metabolic flux modeling are improving the design of metabolic engineering strategies. This review highlights the latest developments in this area, explains the interface between these in silico tools and the experimental implementation tools of metabolic engineers, and provides a way forward so that in silico predictions can better mimic reality and more experimental methods can be considered in simulation studies. The several methodologies for solving genome-scale models (eg, flux balance analysis [FBA], parsimonious FBA, flux variability analysis, and minimization of metabolic adjustment) all have unique advantages and applications. There are two basic approaches to designing metabolic engineering strategies in silico, and both have demonstrated success in the literature. The first involves: 1) making a genetic manipulation in a model; 2) testing for improved performance through simulation; and 3) iterating the process. The second approach has been used in more recently designed in silico tools and involves: 1) comparing metabolic flux profiles of a wild-type and ideally engineered state and 2) designing engineering strategies based on the differences in these flux profiles. Improvements in genome-scale modeling are anticipated in areas such as the inclusion of all relevant cellular machinery, the ability to understand and anticipate the results of combinatorial enrichment experiments, and constructing dynamic and flexible biomass equations that can respond to environmental and genetic manipulations.

Keywords: genome-scale modeling, flux balance analysis, flux variability analysis, minimization of metabolic adjustment, metabolic bottleneck, pathway optimization

A brief introduction to genome-scale metabolic flux modeling

A "genome-scale" metabolic flux model (GEM) consists of a network of biochemical reactions that is reconstructed based on the genomic sequence and annotation of a cell. Assuming a "steady-state" metabolism (ie, a snapshot of metabolism at one time point) is reached on a short time-scale, these reactions can be represented by a linear system of equations. Then, problems such as maximizing specific chemical production or growth can be solved efficiently by linear programming. GEMs and their uses have been reviewed thoroughly, and they are most basically used to predict reaction flux, which is the overall rate of metabolite conversion.[1,2] Often, laboratory measurements including the rates of substrate consumption, product formation, and growth are used as model constraints so calculations coincide with observations. Other model constraints can be derived from reaction thermodynamics,[3] cellular regulatory networks,[4] and -omics datasets.[5] GEMs have been constructed and utilized for intensively

Correspondence: Ryan S Senger
Department of Biological Systems Engineering, Virginia Tech,
301C HABB1, Blacksburg, VA, USA
Tel +1 540 231 9501
Email senger@vt.edu

studied model organisms with well-annotated genomes (eg, *Escherichia coli* MG1655 [bacteria],[6] *Saccharomyces cerevisiae* [yeast],[7] *Mus musculus* [mouse],[8] and *Arabidopsis thaliana* [plant][9]). In addition, homology algorithms have enabled GEM construction of the less-studied organisms. For example, the European Bioinformatics Institute has constructed draft GEMs for 2,630 organisms across phylogenetic domains using automated model-building methods,[10] and the Model SEED also contains several GEMs and has the ability to custom-build GEMs for annotated genomes submitted by the user.[11] The construction of high-quality models often requires expert-informed manual curation,[12] but automated reconstruction provides foundations for further improvement. Although GEMs have been built for species of all domains, microbes still dominate GEM reconstructions and studies due to their relative genomic simplicity, usefulness in biotechnology, and the pathogenicity of some species.

GEMs have extraordinary utility for biological discovery, and novel computational tools have been developed to predict metabolic engineering strategies, which are then validated in the laboratory. Much research in metabolic engineering is focusing on the synthesis of valuable chemicals, biofuels, and pharmaceuticals. Model-guided metabolic engineering presents significant advantages, notably the minimization of laboratory resource use and time required to develop productive strains. Using GEM predictions to design strains enables researchers to engineer product yield/selectivity, substrate utilization, and growth rate. Future developments are anticipated to allow engineering of toxicity responses, cellular differentiation, culture density, and cellular interactions with other cells and materials. Some of the computational tools for predicting gene targets in GEMs for metabolic engineering have been reviewed.[1] The focuses of this review are: 1) how predictions from different tools have been translated into experimental metabolic engineering strategies and 2) which of the experimental methods available are (or are not) represented in the computational (in silico) tools. Since the experimental toolset for metabolic engineering is expanding, this review also addresses how new tools can be incorporated in the in silico design strategies.

In silico metabolic engineering tools

It has been long believed that cells (especially microbes) maintain optimal growth as their primary objective. It has been shown that an additional objective of a minimal adjustment between initial and engineered states also exists.[13] Imposing the goal of chemical overproduction by metabolic engineering often conflicts with the optimal growth objective. Thus, genome-scale modeling serves to establish the relationship between target chemical production and growth. In silico metabolic engineering tools seek to identify genetic manipulations to alter this relationship so that stable strains with high chemical production and growth can be achieved. This section describes the various methods available for solving GEMs, and it highlights those used when metabolism has been engineered. In addition, this section presents the recent advances in in silico tools used with GEMs to generate metabolic engineering strategies for the overproduction of a targeted chemical. In this review, in silico metabolic engineering tools are classified as "top-down" or "bottom-up". The top-down algorithms generate/apply metabolic modifications in silico and then simulate their effects on the dual objectives (ie, productivity and growth) through genome-scale metabolic flux modeling. The procedure is repeated until optimal metabolic modifications are identified. On the other hand, bottom-up algorithms generate separate flux solutions where: 1) growth is maximized and 2) product formation of interest is maximized. Differences between the two flux distributions are identified as targets to design metabolic engineering strategies. These approaches are reviewed in detail in the following section; however, first the methods for generating metabolic flux solutions of GEMs are summarized.

Flux balance analysis and its variants

The fundamental approaches of constraint-based modeling have been reviewed,[1,2] and a subset of these applicable to metabolic engineering are described here. The essential base of almost all predictive tools is flux balance analysis (FBA), which solves the linear system of biological reactions given the "pseudo-" steady-state assumption and an objective function (eg, maximize growth or chemical production rate) using linear programming. The flux balance equation is now commonly written as $S \cdot v = 0$, where S is an m-by-n matrix containing stoichiometric coefficients for each biochemical reaction. Each compound is represented by a row of the matrix, and each reaction is represented in a column. The vector v contains flux values for all n reactions of the system. The system also contains a "biomass equation" that describes cell growth. This is often composed of stoichiometric amounts of macromolecules (eg, protein, DNA, RNA, lipids, cell wall), small molecules, and adenosine triphosphate (ATP) hydrolysis required for growth "maintenance".[14] FBA solves the system of equations given an objective function and constraints (upper and lower) for each flux contained in v. Flux constraints are imposed from laboratory measurements, thermodynamic predictions, and regulatory rules; many reaction fluxes are left unconstrained.

Two other useful approaches are parsimonious FBA (pFBA)[15] and flux variability analysis (FVA).[16] Since the number of reactions is typically greater than the number of compounds in GEMs, multiple FBA solutions exist, and techniques that explore this solution space have been reviewed. The pFBA algorithm was developed to provide the FBA solution that meets optimality with a minimized total flux in the system. In addition, FVA serves the purpose of calculating the possible flux distributions of all reactions.

As mentioned earlier, cellular metabolism changes when a genetic manipulation is introduced in vivo. However, dramatic shifts in metabolism, on a global level, toward optimality are not immediate.[17,18] Thus, a flux distribution predicted in silico that captures this initial response of a cell, instead of one that describes massive flux reorganization toward optimality, provides a better description of the cellular response to genetic changes. For this reason, the minimization of metabolic adjustment (MOMA) algorithm was developed to predict the optimal flux distribution of altered metabolism that would require the smallest change from that of wild-type metabolism.[17] This concept has since been validated by ^{13}C-isotope tracing studies.[13] Similar to MOMA, the regulatory on/off minimization (ROOM) tool hypothesizes that a cell attempts to compensate for genetic manipulations through the fewest number of enzymatic reactions by gene regulation.[18] Additional studies have shown that, in time, cells will evolve from this minimized flux redistribution state to the FBA solution.[19] This concept, introduced over a decade ago,[20] is shown in Figure 1. The goal of metabolic engineering is to alter the metabolic network of a cell so that optimal growth and target chemical production are coupled (meaning a product must be formed as the cell reaches an optimum growth rate). This approach leads to stable strains capable of industrial production. As a cell is engineered, MOMA/ROOM can predict the immediate outcome of genetic manipulations, and FBA (or pFBA) predicts the long-term evolved state of the cell. In the following sections, the top-down and bottom-up in silico metabolic engineering tools are discussed, and a summary of these tools is given in Table 1. However, it is important to note that not all tools are designed to consider evolution and long-term strain stability, which are critically important if an industrial process is going to consider chemostat cultivation over batch processing in which the microbe is replaced frequently.

Top-down in silico tools for designing metabolic engineering strategies

As mentioned previously, a top-down approach is defined here as one in which genetic manipulations are made in

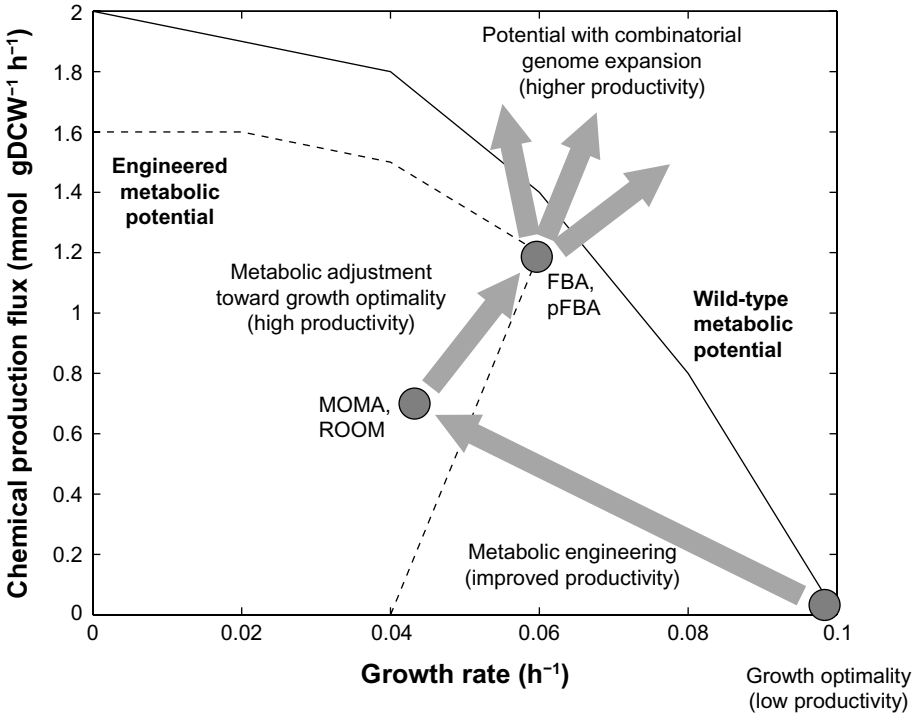

Figure 1 The relationship between the target chemical production flux and the growth rate for wild-type (solid line) and an engineered strain (dash line). The initial wild-type optima determined by FBA (bottom right) can be engineered and the resulting state predicted with MOMA/ROOM. Evolution will eventually optimize growth, which can be predicted by FBA/pFBA. Combinatorial addition of metabolic capabilities can expand the solution space beyond the wild-type potential.
Abbreviations: FBA, flux balance analysis; MOMA, minimization of metabolic adjustment; pFBA, parsimonious FBA; ROOM, regulatory on/off minimization.

Table 1 Summary of in silico tools for generating metabolic engineering strategies and the experimental tools that can be used for implementation

In silico tool	Description	Experimental tools[§,†]	Reference(s)
Top-down approaches			
Randomized gene knockouts	Fluxes associated with reactions catalyzed by one or more genes are set to zero. Simulation performed by FBA, pFBA, MOMA, or ROOM	Gene knockout[a]	21, 22
OptKnock	A bilevel optimization to find gene knockout candidates leading to product formation at optimal growth (FBA)	Gene knockout[a]	19, 20, 23
OptGene	Explores the feasible solution region using a genetic algorithm to identify the necessary gene deletions for the desired phenotype	Gene knockout[a]	24
Cipher of evolutionary design (CiED)	Uses a genetic algorithm to identify optimal mutations to maximize product formation	Gene knockout[a] Gene overexpression[b]	25
ReacKnock	Inspired by OptKnock and enables up to 20 gene deletions	Gene knockout[a]	26
OptReg	An expansion of OptKnock designed to predict up- and downregulation of reactions to achieve a desired phenotype	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	27
OptStrain	Uses a universal database of known enzyme-catalyzed reactions to determine the minimal pathway modification required to maximize product formation	Gene knockout[a] Gene overexpression[b]	28
MOMAKnock	A similar bilevel programming framework to OptKnock except MOMA assumption was adapted to determine flux redistribution	Gene knockout[a]	29
Ant colony optimization with MOMA (ACOMoMA)	A hybrid of ant colony optimization and MOMA to predict gene knockout strategies	Gene knockout[a]	30
Bees Algorithm and FBA (BAFBA)	Similar to ACOMoMA, except Bees Algorithm is used to search gene knockouts and FBA is used to determine fitness	Gene knockout[a]	31
Flux balance analysis with flux ratios (FBrAtio)	Flux ratios serve as constraints to redirect metabolism to a desired product. Resulting flux ratio constraints can be translated to metabolic engineering strategies directly	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	32, 33
Bottom-up approaches			
Flux distribution comparison analysis (FDCA)	Incremental solutions are compared to identify genes of reactions with significant flux changes	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	35, 36
Flux scanning based on enforce objective flux (FSEOF)	Identifies genes of reactions with increased flux upon maximizing product formation	Gene overexpression[b]	37, 38
OptForce	Flux variability analysis of wild-type and mutants (with a desired phenotype) are compared to identify genes of reactions with significant flux change	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	39–41
k-OptForce	An expansion of OptForce with the integration of enzyme kinetic constants to allow optimal solutions to arise from metabolic and/or enzyme engineering	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	42
Continuous modifications (CosMos)	A continuous modification to flux bounds is used to identify upper and lower bounds that can guarantee product formation. The solution space is sampled randomly to find optimum solutions	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	43
Redirector	Iteratively identifies all reactions with flux changes that accommodate for the progressive change in biomass and desired product	Gene knockout[a] Gene overexpression[b] Gene expression knockdown[c]	44

Notes: [§]Not all experimental tools apply to all species; [†]direct genome editing is likely to eventually apply to all cases; [a]gene knockout can be accomplished through insertion mutagenesis using homologous recombination (ie, λ red recombineering), transposable elements, or by genome editing with the aid of CRISPR-Cas systems; [b]gene overexpression can be performed using plasmids or genome knock-in/editing procedures, which are accomplished by inserting gene-of-interest with high expression promoter element using homologous recombination, transposable elements, or CRISPR-Cas systems. RBS and promoter engineering are recommended methods for modulating expression levels. The RBS calculator is a valuable tool for RBS design; [c]gene expression knockdown can be achieved through posttranscriptional gene silencing with sRNA, siRNA, antisense RNA, and/or microRNA. RBS redesign using the RBS calculator is also an effective strategy for gene expression knockdown.

Abbreviations: CRISPR, clustered regularly interspaced short palindromic repeats; FBA, flux balance analysis; MOMA, minimization of metabolic adjustment; pFBA, parsimonious FBA; RBS, ribosomal binding site; ROOM, regulatory on/off minimization.

silico, and then genome-scale modeling is used to determine whether the strategy is beneficial. The concept is shown in Figure 2. The simplest strategy to employ is creating single-gene knockouts. This is done in silico by constraining all reactions associated with a gene of interest to zero and performing FBA or MOMA/ROOM to look for knockouts that enhance target chemical production without compromising growth. This method was used in a well-known study to identify gene knockouts in *E. coli*, resulting in the overproduction of L-valine.[21] Here, single-, double-, and triple-gene knockouts

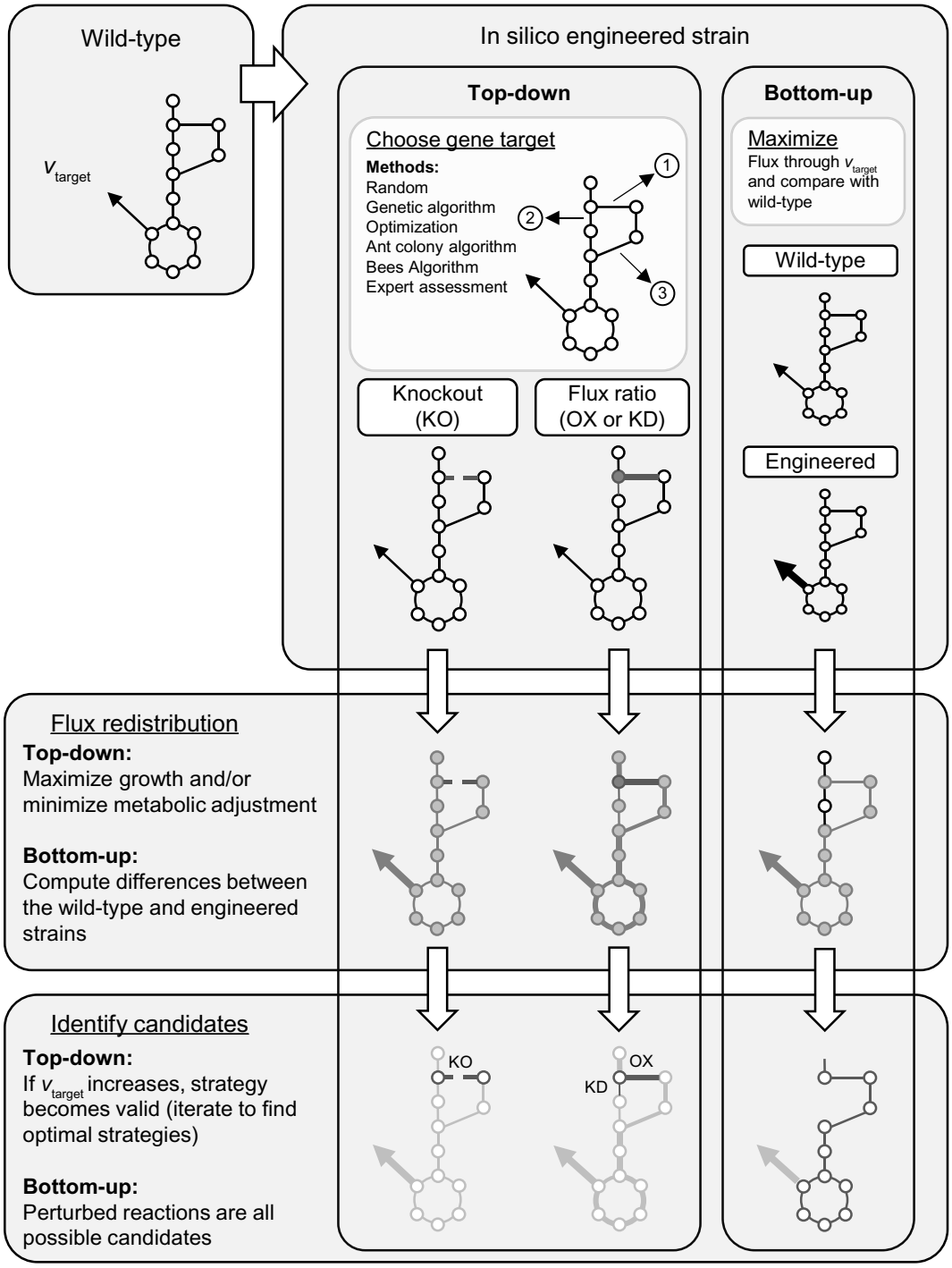

Figure 2 Example workflow to design metabolic engineering strategies using "top-down" and "bottom-up" approaches. Several different in silico tools apply these strategies in different forms. In all cases, the objective is to maximize production of a target chemical (shown here as v_{target}). The following metabolic engineering strategies are shown: (KO) gene knockout, (OX) gene overexpression, and (KD) gene expression knockdown.

were investigated in silico using MOMA to predict resulting phenotypes. Using a multifaceted approach that included the in silico gene deletion study, an industrially relevant strain capable of producing over 7.5 g/L of L-valine (2.27-fold improvement over wild-type) was engineered. This method was also used to generate all single- and double-gene knockout combinations in *S. cerevisiae* in an effort to overproduce succinate.[22] FBA was used in calculations, and three single knockouts (Δ*mdh*, Δ*oac1*, and Δ*dic1*) were selected for experimental validation. The Δ*dic1* strategy was successful, yet non-intuitive for succinate production, and this study demonstrated an important proof-of-concept for designing strains by in silico predictions followed by experimental validations. OptKnock was one of the first

in silico metabolic engineering tools, and it guides the selection of gene knockouts in order to couple maximized product formation to growth.[20] It uses FBA and identifies a limited number of gene knockouts, which serve to reshape the growth and product formation relationship as shown in Figure 1. Successful identification of gene knockout targets, followed by adaptive evolution to achieve FBA predictions, have led to industrially relevant strains capable of producing lactic acid,[19] 1,4-butanediol,[23] and others. Since its introduction, other inspired approaches have attempted to extend its capabilities (ie, increase the potential number of gene candidates for knockout) by reconsidering the bilevel optimization framework. Approaches such as OptGene[24] and the cipher of evolutionary design (CiED)[25] relied on an evolutionary algorithm to select gene targets, and improved functionality was noted. ReacKnock has emerged recently with a new approach to the mixed integer bilevel optimization problem and enables up to 20 gene deletion predictions in a short amount of computational time.[26] In their publication, the authors provide ReacKnock- and OptKnock-designed gene knockout strategies to produce succinate, ethanol, acetate, hydrogen, formate, glycolate, D-lactate, fumarate, and threonine from *E. coli*.[26] OptReg extended OptKnock to include gene overexpressions,[27] and OptStrain allowed incorporation of non-native metabolic pathways for the production of new chemicals.[28] Other approaches, such as MOMAKnock[29] have focused on the limitations of FBA and have sought to implement MOMA in the automated design of gene knockout strategies. Further modifications have combined ant colony optimization (ACO) methods with MOMA in an algorithm called ACOMoMA. The ACOMoMA approach was applied to produce an improved gene knockout strategy for succinate production from *E. coli*.[30] Another development achieved significant results using a hybrid of Bees Algorithm and FBA (BAFBA; a metaheuristic procedure) to design gene knockouts for succinate and lactate production.[31]

While most in silico designs rely on gene knockouts, others infer gene overexpression and partial gene knockdowns as metabolic engineering strategies. In general, the flux change of a reaction may be the result of: 1) directly engineering genes of the catalyzing enzymes; 2) engineering the availability of reaction precursors and substrates upstream; or 3) eliminating bottlenecks downstream. Thus, these strategies are all major contributors to a metabolic adjustment. A recent approach called FBA with flux ratios (FBrAtio) considers strategies of gene overexpression, knockout, and partial knockdown for designing metabolic engineering strategies.[32,33] FBrAtio examines how multiple enzymes compete for the same substrate and allow the distribution of this substrate to be modified and included as a flux ratio constraint in a GEM. Flux ratio constraints can be modified, and pFBA is used to predict global flux distributions. This procedure has been used to design metabolic engineering strategies for several chemicals by different organisms. The concept of the flux ratio constraint was first introduced for two enzymes that compete for the same compound.[32] However, this was later expanded to include all enzymes competing for the same compound.[33] FBrAtio has been used to model the metabolic shift in *Clostridium acetobutylicum* from acids to solvents production as well as predict a high-ethanol-producing phenotype.[32] In addition, it has been used to examine metabolic engineering strategies for: 1) cellulose overproduction by *A. thaliana*; 2) isobutanol production by yeast; 3) acetone production by *Synechocystis*; 4) hydrogen production by *E. coli*; and 5) mixed solvents production by *C. acetobutylicum*.[33] The purpose of this study was to demonstrate further improvements of experimental implementations where possible with "fine-tuned" metabolic engineering strategies derived by FBrAtio. With *Arabidopsis,* it was shown experimentally that the overexpression of a heterologous uridine diphosphate (UDP)–glucose pyrophosphorylase (UGPase) increased cellulose production by approximately 25%.[34] The FBrAtio approach predicted that further increased uridine triphosphate (UTP) consumption by the UGPase could continue to increase cellulose production up to 30%–50% (compared to wild-type) before UTP depletion impacted the growth of the plants negatively.

Bottom-up in silico tools for designing metabolic engineering strategies

The tools classified as bottom-up approaches rely on multiple objective functions in genome-scale modeling to design metabolic engineering strategies. The flux distribution comparison analysis (FDCA) provides a good example of this. First, a GEM is solved by FBA to maximize growth. Then, the GEM is solved (by linear MOMA [lMOMA]) to maximize the production of a chemical of interest. The differences between the flux distributions are considered, and rules for up- or downregulation of genes are determined based on significant changes between the flux distributions.[35] FDCA has been used to improve lycopene production by 174% in an *E. coli* strain already capable of high lycopene production,[36] and it identified 51 potential gene targets, including five novel gene knockout targets and four novel gene overexpression targets. The flux scanning based on enforced objective flux (FSEOF) approach was also developed to

enhance lycopene production.[37] This approach also begins with maximizing biomass formation of a GEM with FBA, but the flux of product formation is constrained to be equal to the experimentally observed flux in the wild-type organism. Then, the theoretical maximum product formation rate is calculated in a new simulation by setting this as the objective function. FSEOF works by maximizing the cell growth rate while the target product formation rate is increased gradually from its initial value toward its theoretical maximum. Targets for gene overexpression are identified as fluxes that increase throughout simulations without changing direction. This method identified 35 gene overexpression targets for lycopene production by *E. coli*. FVA was then employed to narrow these potential targets by selecting those showing increases outside of the ranges due to flux variability.[37] This approach can also be used with an altered biomass equation to accommodate intracellular target (eg, protein) accumulation. For example, the human superoxide dismutase (hSOD) enzyme was overproduced in *Pichia pastoris* using predicted gene knockout and overexpression strategies from MOMA and FSEOF, respectively.[38]

OptForce is another bottom-up approach that has enabled the incorporation of gene knockouts, overexpressions, and knockdowns as metabolic engineering strategies.[39] OptForce also allows (and encourages) the incorporation of experimentally measured metabolic flux data of the wild-type and a strain engineered to overproduce a target chemical. In general, flux variability is calculated for both wild-type and engineered strains, and the flux ranges are compared for each reaction. Candidates for metabolic engineering are identified as those reactions where there is no overlap between possible flux ranges. OptForce then performs a secondary optimization (a top-down procedure) where the minimal set of metabolic interventions is identified to achieve a desired goal. OptForce has been used in several applications, including the overexpression of succinate[39] and fatty acids of specified chain length in *E. coli*.[40] In addition, OptForce was used to design a metabolic engineering strategy leading to a four-fold increase in intracellular malonyl-CoA concentration in *E. coli*, which was then utilized for the production of naringenin (a valuable plant secondary metabolite).[41] The recent extension k-OptForce has enabled the incorporation of enzyme kinetic constants, where possible, and returns metabolic engineering strategies (ie, gene knockout, overexpression, or knockdown) along with kinetic parameters that could be altered by enzyme engineering.[42] This approach can consider relevant phenomena, such as substrate inhibition, that cannot be modeled using flux-based approaches alone. The continuous modifications (CosMos) approach significantly differs from OptForce in that changes to flux bounds are modified continuously, rather than by FVA results. CosMos then minimizes product formation given a constrained non-zero growth rate, and looks for modified flux constraints that still yield product formation under these conditions.[43]

Finally, the Redirector approach is different in that it relies on an artificial objective function consisting of contributions from growth and metabolic flux redirected into a product-forming pathway and does not rely on manipulating flux bounds.[44] Redirector can also design gene knockout, overexpression, or knockdown metabolic engineering strategies, and the manipulation of algorithm parameters can alter the number of manipulations returned by the algorithm. The production of fatty acids by *E. coli* MG1655 was chosen as a test case of the algorithm. The algorithm designed strategies capable of reaching 80% of the theoretical yield for myristoyl-CoA while maintaining 20% biomass yield.[44] The global implementation of FBrAtio (currently in press) is also classified as an approach that does not manipulate flux bounds to derive metabolic engineering strategies. The global FBrAtio uses flux distribution maps of maximized growth and product formation using pFBA and designs flux ratio constraints that enable product formation and growth.

Experimental metabolic engineering tools

The available in silico metabolic engineering tools return strategies consisting of gene knockout, overexpression, and/or knockdown (and enzyme engineering for k-OptForce). There are several ways in which these strategies can be implemented, but current in silico tools do not consider this level of detail. In this section, many common (but certainly not all) experimental implementation methods are reviewed along with their relationship to in silico predictions. For example, returning a gene overexpression strategy does not explain how it should be implemented. If it must be encoded on a plasmid, what type and strength of promoter/ribosomal binding site (RBS) combination should be used? What copy number of plasmid should be used? Since plasmid copy number per cell is heterogeneous, what impacts will this have? Will plasmid replication demand cellular resources that influence metabolic flux predictions? What are the impacts of antibiotic resistance genes? Should one or multiple copies of the gene of interest be knocked into the genome? Or, should a native promoter/RBS be tuned instead? If so, to what levels? Finally, what impact will this genetic manipulation have on the resulting phenotype? Will this significantly impact cell

composition, metabolic flux distribution, and predictions? These and more questions will be addressed by in silico tools that returned "fine-tuned" metabolic engineering strategies (eg, overexpress a target gene by 70% relative to wild-type) and take into account changing cell phenotype by updating the GEM biomass equation.

Manipulating gene expression

Here, basic experimental strategies for gene expression manipulations are reviewed in the context of genome-scale modeling. Clearly, not all tools and approaches can be discussed here, but the basics are identified. Manipulations can occur at the transcriptional, translational, and posttranslational levels, with emphasis on the first two in microbes. Several experimental methods exist for generating gene knockouts that involve chromosomal integration for gene disruption. Of course, chromosomal integration can also be used to knock-in useful genes/regulatory elements. One particularly popular method for single-gene targeting is the polymerase chain reaction (PCR)-based version of λ red recombineering.[45] It has also been used for the introduction of site-directed mutations, promoter tuning/replacement, and reporter genes for promoter tagging experiments.[46] The knock-in/knockout (KIKO) vectors facilitate the chromosomal integration of large DNA segments (including multigene cassettes and entire pathways) at specific well-characterized loci using λ red recombination.[47] Other means of gene knockout involve the use of transposons or homologous recombination mediated by phage-derived elements, and more advanced genetic systems are required for other microbes, such as the clostridia.[48,49] In higher plant species, such as *A. thaliana,* genomic integration is accomplished using an *Agrobacterium*-mediated method that makes use of its ability to transfer DNA from its tumor-inducing plasmid into the plant host genome.[50,51] This technology has been used for both gene disruption and knock-in in *Arabidopsis*. Gene knockouts (and knock-ins) appear to be the most benign to genome-scale modeling predictions, as long as plasmids and antibiotic resistance markers are removed. Indeed, the presence of plasmids and antibiotics (even with effective antibiotic resistance genes) has been shown to alter cell phenotypes.[52] The GEM biomass equation describes the cell phenotype, and how this equation should be altered by the presence of plasmids, antibiotics, or other genetic or environmental manipulations remains a subject for research. This makes clustered regularly interspaced short palindromic repeats (CRISPR)-Cas systems[53] attractive for genome editing from a genome-scale modeling standpoint. While the mechanisms of plasmid replication are understood, this cellular machinery is not yet encoded in GEMs, creating a divergence between the in silico and experimental systems. In addition, gene knockouts, knock-ins, and genome editing are designed to alter metabolism. When successful, this alters the cellular phenotype; thus, the biomass equation must be updated accordingly. However, this will require predictions or a simplified method of measurement, both of which are discussed later.

With this knowledge, it is easy to see why gene overexpression methods may lead to greater metabolic burden and uncertainty with genome-scale modeling, especially when a gene is overexpressed from a plasmid. Techniques that minimize the ATP maintenance requirements of a cell are preferred and are more effectively modeled. With gene overexpression, promoter and RBS engineering have enabled significant progress. Controllable gene expression has launched the field of synthetic biology and led to the quest to design genetic circuits.[54] Furthermore, promoter tuning[55] with RBS optimization can improve metabolic pathway function.[56] Tools, such as the RBS calculator,[57] are enabling RBS design based on thermodynamic principles. In these cases, it becomes clear that synthetic designs are enabling pathway overexpression by orders of magnitude, and at some point, cellular resources are depleted (eg, transfer RNA [tRNA] pools), creating competition between cell growth and pathway expression. This is not yet accounted for by genome-scale modeling and presents a unique opportunity to integrate metabolic pathway tuning with genome-wide metabolic activity.

Gene expression tuning can also be engineered at the posttranslational level, where interactions with mRNA are the major focus. Small RNA (sRNA) bind targeted mRNA (through complementary base-pairing) and modulate its translation.[58] The majority of sRNAs have been identified as translation repressors, and binding generally occurs at or near the RBS.[59] Thermodynamic-based design has enabled "fine-tuned" gene expression knockdowns,[60] and these have proven advantageous in a metabolic engineering strategy to produce phenol from glucose.[61] Similarly, artificial small interfering RNA and microRNA have been widely used in plant systems to reduce gene expression.[62] Achieving stable gene integration in higher plants can be problematic. An alternative is to employ viral-induced gene silencing approaches.[63] While these technologies enable gene knockdown, they are generally operated from plasmid-based systems, which provide the same challenges to genome-scale modeling as mentioned previously.

Combinatorial approaches

Using genome-scale modeling to predict the outcomes of combinatorial metabolic engineering experiments is an area for many future advances. As shown in Figure 1, the addition of new genetic material (either synthetic or from other organisms) can expand the product-forming capabilities of an organism. Combinatorial approaches can involve induced chromosomal mutations, random insertion of transposons, genome shuffling, transcription factor engineering,[64] or even randomized chromosomal insertion of synthetic DNA.[65] Lycopene production has been engineered successfully through: 1) a combination of model-driven and transposon-based combinatorial knockouts[66] and 2) the multiplex automated genome engineering (MAGE) platform, which relies on synthetic DNA insertion.[65] In addition, gene overexpression libraries offer the opportunity to insert the genomic capabilities of a single organism or a metagenome. This strategy has proven successful in locating genomic sequences to confer tolerance to furfural,[67] among many others. The simultaneous expression of dual libraries on a plasmid and fosmid led to a unique combination of gene enrichment that increased acid tolerance in *E. coli* by 9,000-fold.[68] Expanding the genome to confer resistance to toxins or new/improved metabolic capabilities has the potential to redefine the relationship between product formation and culture growth, as shown in Figure 1. In the case of conferring resistance to toxins, often uncharacterized or non-obvious library fragments are selected during enrichment.[69] This is often because toxicity mechanisms, as well as many cellular interactions, are multigenic and still not understood fully. While genome-scale modeling cannot provide these types of predictions, where the interaction mechanisms are uncharacterized, the metabolic potentials through the completion and addition of new pathways and enzymes are predictable. It is likely that the theoretical limits of metabolic enhancement due to library enrichment can be found through genome-scale modeling, and the emergence of metagenomic GEMs will likely contain the metabolic potentials.

Phenotyping

Phenotyping refers to the monitoring of cell chemical composition and differentiation. This is critical because the GEM biomass equation contains the cell chemical composition and is representative of the cellular phenotype, which is known to change with genetic and environmental perturbations. The role of the biomass equation has been shown to be crucial in genome-scale modeling,[14,70] creating the need for accurate and near real-time monitoring techniques to interface with GEMs. In silico optimization methods have shown promising results,[70] but it is likely that an experimental approach will be needed as a supplement. Traditional methods of biomass equation generation are laborious and involve offline analytical methods. Analysis of heterogeneous populations of differentiating (eg, sporulating) microbes is now possible using flow cytometry.[71] In addition, Raman spectroscopy has recently proven useful for near real-time phenotyping of *E. coli*. Raman spectroscopy also does not require the use of chemical labels and is nondestructive to the sample. In one application, Raman spectroscopy was used to resolve fatty acids (saturated, unsaturated, and cyclopropane), cell membrane fluidity, amino acids, and total protein content of cultures exposed to toxic 1.2% volume per volume 1-butanol (and control cultures) over a 180-minute time course.[72] In another approach, "chemometric fingerprinting", a multivariate statistical analysis involving principal component analysis and linear discriminate analysis, was used to classify the *E. coli* phenotypes resulting from exposure to different classes of antibiotics.[73] Chemometric fingerprinting is unique in that it uses the entire Raman spectrum to characterize a phenotype, whereas most approaches focus on only a few well-defined characteristic bands of the spectrum. With these types of near real-time analyses, GEM biomass equations can become dynamic and responsive to environmental and genetic changes. With current offline methods of phenotype characterization, this level of detail is not possible. However, with easily accessible phenotyping capabilities, biomass equations can be updated easily, leading to improved genome-scale modeling performance.

The path forward

New metabolic engineering targets and opportunities with plants

Deriving metabolic engineering strategies with genome-scale modeling is proving to be efficient and informative. As research continues to derive de novo metabolic pathways to synthesize valuable chemicals, optimization of product yield to meet industrial demands will be inevitable. Still, the variety of potential products from microbes remains limited and may be expanded in the near term by looking into complex eukaryotic species, such as plants. There are many valuable compounds made by plants that are not available elsewhere. For example, oil seed crops (eg, soybeans) produce edible vegetable oil that is used throughout the world. Although the pathways for lipid biosynthesis in higher plants have been studied

for years, understanding of the crucial regulatory mechanisms of these pathways remains limited. Thus, engineering plants to accumulate high levels of healthy omega-3 long-chain polyunsaturated fatty acids,[74] or modified non-native fatty acids as replacements for petroleum-derived chemicals in industrial processes[75] is desirable. Similarly, central carbon metabolism is a target for understanding the relationship between the regulation of carbon partitioning and biomass production in plants. Identifying metabolic bottlenecks in the production of cellulose, the energy-rich polymer that is targeted for consolidated bioprocessing,[76] could enhance efforts to produce more cellulose per plant in the field. Likewise, modeling is being applied to the goal of reducing plant lignin, a phenolic polymer in the secondary wall that limits our use of cellulosic biomass during industrial processing.[77] One caveat of reducing lignin is that optimal plant growth must also be preserved, and GEMs may be uniquely positioned to tackle this issue because they can theoretically integrate metabolic behavior with plant growth.[78] Enhancing the vitamin content of edible plants is another active area of research.[79] For some vitamin synthesis pathways, enough information exists to begin the application of genome-scale modeling to increase the concentration of vitamins to meet minimal requirements for humans.[80] In the future, it may be possible to use genome-scale modeling to tackle issues such as optimizing plant growth under stressful or poor nutrient growth conditions. In these cases, genome-scale models would have to account for complex interactions between stress, hormone, and other signaling pathways that impact biomass synthesis and composition.[81] In addition, a related application is to understand how to limit plant yield loss due to pests, by engineering known, disease-resistance pathways.[81] All of these approaches will require flexible biomass equations that can respond to manipulations, and a complex multicellular plant will likely require tissue-specific GEMs that will integrate to form an overall plant phenotype.

Enzyme engineering for pathway redirection

The k-OptForce in silico tool is among the first to incorporate the concept of enzyme engineering to redirect metabolic flux for the production of target chemicals. Kinetics-based approaches to genome-scale metabolic modeling are emerging,[82,83] and soon enzyme redesign will be a valid metabolic engineering strategy. Direct genome editing, which is preferred over insertion of plasmids and markers that consume cellular resources, will enable easy implementation. Enzyme engineering is a complex field itself and beyond the scope of this review, but effective in silico methods are emerging and are expected to play a role in enzyme redesign. Improvements in hardware and software performance will continue to expand the range and size of enzyme engineering problems and systems that can be studied. Current computational approaches can be divided into bioinformatics, molecular modeling, and de novo design.[84] Bioinformatics approaches are typically based on analysis of evolutionary data and can be used to change activity, selectivity, and stability within a family of enzymes. Molecular modeling approaches (eg, molecular dynamics, quantum mechanics/molecular mechanics simulations) have considerable potential to address challenges in computational enzyme design and redesign. In particular, advances in these methods may enable improved calculation of binding affinities and energy barriers, which will enhance understanding of enzyme specificity.[85] De novo design is also showing increasing promise in designing enzymes, including those that catalyze reactions for which nature has not designed a catalyst. Notably, these novel methods may be enhanced by the application of molecular modeling approaches.[86]

Increasing modeling accuracy

Finally, the path forward must focus on methods that increase the accuracy of genome-scale metabolic flux modeling and improve agreements with ^{13}C-isotopomer tracing studies. In our experience, there are four areas for immediate improvement. The first area includes the incorporation of a more detailed account of cellular machinery in GEMs. As mentioned previously, the ATP maintenance approximation of the GEM biomass equation should be replaced by mechanistic accounts. This must also allow for the identification of metabolic burdens of plasmids and altered metabolic states as a result of genome editing. There are current ongoing efforts of "whole cell modeling" that aim to include cellular machinery in modeling efforts. These models are showing promise of being able to predict phenotypes as well as better integrate and explain -omics datasets.[87] Second, more accurate biomass equations are needed. Whether these will be derived computationally or experimentally remains to be seen, and there are good arguments for both approaches. Third, a more accurate representation of flux branching at critical metabolic nodes is needed. This occurs when multiple enzymes can consume the same metabolite. Ultimately, the laws of thermodynamics (including enzyme availability) determine how that metabolite is distributed among the competing enzymes. Current methods of FBA, pFBA, FVA, ROOM (etc) do not consider this level of detail. FBrAtio provides this capability, though significant

strides are needed to first translate the biophysical constraints into flux ratio constraints. Finally, the roles of redox states in product secretion profiles and the influx/efflux of protons across the cell membrane need to be included as constraints in GEMs. In addition, efforts in these areas will supplement the many useful emerging tools that are focusing on genomic regulation and -omics dataset integrations. All of these will improve genome-scale modeling accuracy, which is needed for deriving effective metabolic engineering strategies.

Acknowledgments

Funding was provided by the National Science Foundation (NSF1243988 and NSF1254242), the US Department of Agriculture (2010-65504-20346 and the HATCH program), and the Institute for Critical Technologies and Applied Science at Virginia Tech.

Disclosure

The authors report no conflicts of interest in this work.

References

1. Tomar N, De RK. Comparing methods for metabolic network analysis and an application to metabolic engineering. *Gene.* 2013;521(1):1–14.
2. Oberhardt MA, Palsson BØ, Papin JA. Applications of genome-scale metabolic reconstructions. *Mol Syst Biol.* 2009;5:320.
3. Henry CS, Broadbelt LJ, Hatzimanikatis V. Thermodynamics-based metabolic flux analysis. *Biophys J.* 2007;92(5):1792–1805.
4. Chandrasekaran S, Price ND. Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in *Escherichia coli* and Mycobacterium tuberculosis. *Proc Natl Acad Sci U S A.* 2010;107(41):17845–17850.
5. Hyduke DR, Lewis NE, Palsson BØ. Analysis of omics data with genome-scale models of metabolism. *Mol Biosyst.* 2013;9(2):167–174.
6. Orth JD, Conrad TM, Na J, et al. A comprehensive genome-scale reconstruction of *Escherichia coli* metabolism – 2011. *Mol Syst Biol.* 2011;7:535.
7. Heavner BD, Smallbone K, Price ND, Walker LP. Version 6 of the consensus yeast metabolic network refines biochemical coverage and improves model performance. *Database (Oxford).* 2013;2013:bat059.
8. Selvarasu S, Karimi IA, Ghim GH, Lee DY. Genome-scale modeling and in silico analysis of mouse cell metabolic network. *Mol Biosyst.* 2010;6(1):152–161.
9. de Oliveira Dal'Molin CG, Quek LE, Palfreyman RW, Brumbley SM, Nielsen LK. AraGEM, a genome-scale reconstruction of the primary metabolic network in Arabidopsis. *Plant Physiol.* 2010;152(2):579–589.
10. Büchel F, Rodriguez N, Swainston N, et al. Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC Syst Biol.* 2013;7:116.
11. Aziz RK, Devoid S, Disz T, et al. SEED servers: high-performance access to the SEED genomes, annotations, and metabolic models. *PLoS One.* 2012;7(10):e48053.
12. Thiele I, Palsson BØ. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat Protoc.* 2010;5(1):93–121.
13. Schuetz R, Zamboni N, Zampieri M, Heinemann M, Sauer U. Multidimensional optimality of microbial metabolism. *Science.* 2012;336(6081):601–604.
14. Senger RS. Biofuel production improvement with genome-scale models: the role of cell composition. *Biotechnol J.* 2010;5(7):671–685.
15. Lewis NE, Hixson KK, Conrad TM, et al. Omic data from evolved *E. coli* are consistent with computed optimal growth from genome-scale models. *Mol Syst Biol.* 2010;6:390.
16. Gudmundsson S, Thiele I. Computationally efficient flux variability analysis. *BMC Bioinformatics.* 2010;11:489.
17. Segre D, Vitkup D, Church GM. Analysis of optimality in natural and perturbed metabolic networks. *Proc Natl Acad Sci U S A.* 2002;99(23):15112–15117.
18. Shlomi T, Berkman O, Ruppin E. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proc Natl Acad Sci U S A.* 2005;102(21):7695–7700.
19. Fong SS, Burgard AP, Herring CD, et al. In silico design and adaptive evolution of *Escherichia coli* for production of lactic acid. *Biotechnol Bioeng.* 2005;91(5):643–648.
20. Burgard AP, Pharkya P, Maranas CD. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol Bioeng.* 2003;84(6):647–657.
21. Park JH, Lee KH, Kim TY, Lee SY. 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 U S A.* 2007;104(19):7797–7802.
22. Agren R, Otero JM, Nielsen J. Genome-scale modeling enables metabolic engineering of Saccharomyces cerevisiae for succinic acid production. *J Ind Microbiol Biotechnol.* 2013;40(7):735–747.
23. Yim H, Haselbeck R, Niu W, et al. Metabolic engineering of *Escherichia coli* for direct production of 1,4-butanediol. *Nat Chem Biol.* 2011;7(7):445–452.
24. Patil KR, Rocha I, Forster J, Nielsen J. Evolutionary programming as a platform for in silico metabolic engineering. *BMC Bioinformatics.* 2005;6:308.
25. Fowler ZL, Gikandi WW, Koffas MA. Increased malonyl coenzyme A biosynthesis by tuning the *Escherichia coli* metabolic network and its application to flavanone production. *Appl Environ Microbiol.* 2009;75(18):5831–5839.
26. Xu Z, Zheng P, Sun J, Ma Y. ReacKnock: identifying reaction deletion strategies for microbial strain optimization based on genome-scale metabolic network. *PLoS One.* 2013;8(12):e72150.
27. Pharkya P, Maranas CD. An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems. *Metab Eng.* 2006;8(1):1–13.
28. Pharkya P, Burgard AP, Maranas CD. OptStrain: a computational framework for redesign of microbial production systems. *Genome Res.* 2004;14(11):2367–2376.
29. Ren S, Zeng B, Qian X. Adaptive bi-level programming for optimal gene knockouts for targeted overproduction under phenotypic constraints. *BMC Bioinformatics.* 2013;14 Suppl 2:S17.
30. Chong SK, Mohamad MS, Mohamed Salleh AH, Choon YW, Chong CK, Deris S. A hybrid of ant colony optimization and minimization of metabolic adjustment to improve the production of succinic acid in *Escherichia coli*. *Comput Biol Med.* 2014;49:74–82.
31. Choon Y, Mohamad M, Deris S, et al. Identifying gene knockout strategies using a hybrid of Bees Algorithm and flux balance analysis for in silico optimization of microbial strains. In: Omatu S, De Paz Santana JF, González SR, Molina JM, Bernardos AM, Rodríguez JMC, editors. *Distributed Computing and Artificial Intelligence.* Vol 151. Springer Berlin Heidelberg; 2012:371–378.
32. McAnulty MJ, Yen JY, Freedman BG, Senger RS. Genome-scale modeling using flux ratio constraints to enable metabolic engineering of clostridial metabolism in silico. *BMC Syst Biol.* 2012;6(1):42.
33. Yen JY, Nazem-Bokaee H, Freedman BG, Athamneh AI, Senger RS. Deriving metabolic engineering strategies from genome-scale modeling with flux ratio constraints. *Biotechnol J.* 2013;8(5):581–594.
34. Wang Q, Zhang X, Li F, Hou Y, Liu X, Zhang X. Identification of a UDP-glucose pyrophosphorylase from cotton (Gossypium hirsutum L.) involved in cellulose biosynthesis in Arabidopsis thaliana. *Plant Cell Rep.* 2011;30(7):1303–1312.

35. Meng H, Lu Z, Wang Y, Wang X, Zhang S. In silico improvement of heterologous biosynthesis of erythromycin precursor 6-deoxyerythronolide B in *Escherichia coli*. *Biotechnol Bioproc Eng.* 2011;16(3):445–456.
36. Wang JF, Meng HL, Xiong ZQ, Zhang SL, Wang Y. Identification of novel knockout and up-regulated targets for improving isoprenoid production in *E. coli*. *Biotechnol Lett.* 2014;36(5):1021–1027.
37. Choi HS, Lee SY, Kim TY, Woo HM. In silico identification of gene amplification targets for improvement of lycopene production. *Appl Environ Microbiol.* 2010;76(10):3097–3105.
38. Nocon J, Steiger MG, Pfeffer M, et al. Model based engineering of Pichia pastoris central metabolism enhances recombinant protein production. *Metab Eng.* 2014;24:129–138.
39. Ranganathan S, Suthers PF, Maranas CD. OptForce: an optimization procedure for identifying all genetic manipulations leading to targeted overproductions. *PLoS Comput Biol.* 2010;6(4):e1000744.
40. Ranganathan S, Tee TW, Chowdhury A, et al. An integrated computational and experimental study for overproducing fatty acids in *Escherichia coli*. *Metab Eng.* 2012;14(6):687–704.
41. Xu P, Ranganathan S, Fowler ZL, Maranas CD, Koffas MA. Genome-scale metabolic network modeling results in minimal interventions that cooperatively force carbon flux towards malonyl-CoA. *Metab Eng.* 2011;13(5):578–587.
42. Chowdhury A, Zomorrodi AR, Maranas CD. k-OptForce: integrating kinetics with flux balance analysis for strain design. *PLoS Comput Biol.* 2014;10(2):e1003487.
43. Cotten C, Reed JL. Constraint-based strain design using continuous modifications (CosMos) of flux bounds finds new strategies for metabolic engineering. *Biotechnol J.* 2013;8(5):595–604.
44. Rockwell G, Guido NJ, Church GM. Redirector: designing cell factories by reconstructing the metabolic objective. *PLoS Comput Biol.* 2013;9(1):e1002882.
45. Datsenko KA, Wanner BL. One-step inactivation of chromosomal genes in *Escherichia coli* K-12 using PCR products. *Proc Natl Acad Sci U S A.* 2000;97(12):6640–6645.
46. Thomason LC, Court D, Bubunenko M, et al. Recombineering: genetic engineering in bacteria using homologous recombination. *Curr Protoc Mol Biol.* 2007;Chapter 1:Unit 1.16.
47. Sabri S, Steen JA, Bongers M, Nielsen LK, Vickers CE. Knock-in/Knock-out (KIKO) vectors for rapid integration of large DNA sequences, including whole metabolic pathways, onto the *Escherichia coli* chromosome at well-characterised loci. *Microb Cell Fact.* 2013;12:60.
48. Heap JT, Pennington OJ, Cartman ST, Carter GP, Minton NP. The ClosTron: a universal gene knock-out system for the genus Clostridium. *J Microbiol Methods.* 2007;70(3):452–464.
49. Tracy BP, Jones SW, Papoutsakis ET. Inactivation of σE and σG in Clostridium acetobutylicum illuminates their roles in clostridial-cell-form biogenesis, granulose synthesis, solventogenesis, and spore morphogenesis. *J Bacteriol.* 2011;193(6):1414–1426.
50. Tsuda K, Qi Y, Nguyen le V, et al. An efficient Agrobacterium-mediated transient transformation of Arabidopsis. *Plant J.* 2012;69(4):713–719.
51. Alonso JM, Stepanova AN, Leisse TJ, et al. Genome-wide insertional mutagenesis of Arabidopsis thaliana. *Science.* 2003;301(5633):653–657.
52. Walter A, Reinicke M, Bocklitz T, et al. Raman spectroscopic detection of physiology changes in plasmid-bearing *Escherichia coli* with and without antibiotic treatment. *Anal Bioanal Chem.* 2011;400(9):2763–2773.
53. Jiang W, Bikard D, Cox D, Zhang F, Marraffini LA. RNA-guided editing of bacterial genomes using CRISPR-Cas systems. *Nat Biotechnol.* 2013;31(3):233–239.
54. Alper H, Fischer C, Nevoigt E, Stephanopoulos G. Tuning genetic control through promoter engineering. *Proc Natl Acad Sci U S A.* 2005;102(36):12678–12683.
55. Hammer K, Mijakovic I, Jensen PR. Synthetic promoter libraries – tuning of gene expression. *Trends Biotechnol.* 2006;24(2):53–55.
56. Pfleger BF, Pitera DJ, Smolke CD, Keasling JD. Combinatorial engineering of intergenic regions in operons tunes expression of multiple genes. *Nat Biotechnol.* 2006;24(8):1027–1032.
57. Salis MH, Mirsky EA, Voigt CA. Automated design of synthetic ribosome binding sites to control protein expression. *Nat Biotechnol.* 2009;27(10):946–950.
58. Waters LS, Storz G. Regulatory RNAs in bacteria. *Cell.* 2009;136(4):615–628.
59. Aiba H. Mechanism of RNA silencing by Hfq-binding small RNAs. *Curr Opin Microbiol.* 2007;10(2):134–139.
60. Yoo SM, Na D, Lee SY. Design and use of synthetic regulatory small RNAs to control gene expression in *Escherichia coli*. *Nat Protoc.* 2013;8(9):1694–1707.
61. Kim B, Park H, Na D, Lee SY. Metabolic engineering of *Escherichia coli* for the production of phenol from glucose. *Biotechnol J.* 2014;9(5):621–629.
62. Tiwari M, Sharma D, Trivedi PK. Artificial microRNA mediated gene silencing in plants: progress and perspectives. *Plant Mol Biol.* 2014;86(1–2):1–18.
63. Lange M, Yellina AL, Orashakova S, Becker A. Virus-induced gene silencing (VIGS) in plants: an overview of target species and the virus-derived vector systems. *Methods Mol Biol.* 2013;975:1–14.
64. Santos CN, Stephanopoulos G. Combinatorial engineering of microbes for optimizing cellular phenotype. *Curr Opin Chem Biol.* 2008;12(2):168–176.
65. Wang HH, Isaacs FJ, Carr PA, et al. Programming cells by multiplex genome engineering and accelerated evolution. *Nature.* 2009;460(7257):894–898.
66. Alper H, Miyaoku K, Stephanopoulos G. Construction of lycopene-overproducing *E. coli* strains by combining systematic and combinatorial gene knockout targets. *Nat Biotechnol.* 2005;23(5):612–616.
67. Glebes TY, Sandoval NR, Reeder PJ, Schilling KD, Zhang M, Gill RT. Genome-wide mapping of furfural tolerance genes in *Escherichia coli*. *PLoS One.* 2014;9(1):e87540.
68. Nicolaou SA, Gaida SM, Papoutsakis ET. Coexisting/Coexpressing Genomic Libraries (CoGeL) identify interactions among distantly located genetic loci for developing complex microbial phenotypes. *Nucleic Acids Res.* 2011;39(22):e152.
69. Borden JR, Jones SW, Indurthi D, Chen Y, Papoutsakis ET. A genomic-library based discovery of a novel, possibly synthetic, acid-tolerance mechanism in Clostridium acetobutylicum involving non-coding RNAs and ribosomal RNA processing. *Metab Eng.* 2010;12(3):268–281.
70. Senger RS, Nazem-Bokaee H. Resolving cell composition through simple measurements, genome-scale modeling, and a genetic algorithm. *Methods Mol Biol.* 2013;985:85–101.
71. Tracy BP, Gaida SM, Papoutsakis ET. Development and application of flow-cytometric techniques for analyzing and sorting endospore-forming clostridia. *Appl Environ Microbiol.* 2008;74(24):7497–7506.
72. Zu TN, Athamneh AI, Wallace RS, Collakova E, Senger RS. Near real-time analysis of the phenotypic responses of *Escherichia coli* to 1-butanol exposure using Raman spectroscopy. *J Bacteriol.* 2014;196(23):3983–3991.
73. Athamneh AI, Alajlouni RA, Wallace RS, Seleem MN, Senger RS. Phenotypic profiling of antibiotic response signatures in *Escherichia coli* using Raman spectroscopy. *Antimicrob Agents Chemother.* 2014;58(3):1302–1314.
74. Napier JA, Haslam RP, Beaudoin F, Cahoon EB. Understanding and manipulating plant lipid composition: metabolic engineering leads the way. *Curr Opin Plant Biol.* 2014;19:68–75.
75. Yu XH, Prakash RR, Sweet M, Shanklin J. Coexpressing *Escherichia coli* cyclopropane synthase with Sterculia foetida Lysophosphatidic acid acyltransferase enhances cyclopropane fatty acid accumulation. *Plant Physiol.* 2014;164(1):455–465.
76. Youngs H, Somerville C. Development of feedstocks for cellulosic biofuels. *F1000 Biol Rep.* 2012;4:10.

77. Chen HC, Song J, Wang JP, et al. Systems biology of lignin biosynthesis in Populus trichocarpa: heteromeric 4-coumaric acid:coenzyme a ligase protein complex formation, regulation, and numerical modeling. *Plant Cell.* 2014;26(3):876–893.
78. Collakova E, Yen JY, Senger RS. Are we ready for genome-scale modeling in plants? *Plant Sci.* 2012;191–192:53–70.
79. Wilson SA, Roberts SC. Metabolic engineering approaches for production of biochemicals in food and medicinal plants. *Curr Opin Biotechnol.* 2014;26:174–182.
80. Mintz-Oron S, Meir S, Malitsky S, Ruppin E, Aharoni A, Shlomi T. Reconstruction of Arabidopsis metabolic network models accounting for subcellular compartmentalization and tissue-specificity. *Proc Natl Acad Sci U S A.* 2012;109(1):339–344.
81. Suzuki N, Rivero RM, Shulaev V, Blumwald E, Mittler R. Abiotic and biotic stress combinations. *New Phytol.* 2014;203(1):32–43.
82. Chakrabarti A, Miskovic L, Soh KC, Hatzimanikatis V. Towards kinetic modeling of genome-scale metabolic networks without sacrificing stoichiometric, thermodynamic and physiological constraints. *Biotechnol J.* 2013;8(9):1043–1057.
83. Jamshidi N, Palsson BO. Formulating genome-scale kinetic models in the post-genome era. *Mol Syst Biol.* 2008;4:171.
84. Damborsky J, Brezovsky J. Computational tools for designing and engineering enzymes. *Curr Opin Chem Biol.* 2014;19:8–16.
85. Kiss G, Çelebi-Ölçüm N, Moretti R, Baker D, Houk KN. Computational enzyme design. *Angew Chem Int Ed Engl.* 2013;52(22):5700–5725.
86. Privett HK, Kiss G, Lee TM, et al. Iterative approach to computational enzyme design. *Proc Natl Acad Sci U S A.* 2012;109(10):3790–3795.
87. Karr JR, Sanghvi JC, Macklin DN, et al. A whole-cell computational model predicts phenotype from genotype. *Cell.* 2012;150(2):389–401.

CHAPTER 3

PREDICTING METABOLIC ENGINEERING STRATEGIES WITH NODE-REWARD-OPTIMIZATION TOOLBOX

Jiun Y. Yen [1,2], Glenda E. Gillaspy [2], Ryan S. Senger [1,3]

[1] Department of Biological Systems Engineering, Virginia Tech, Blacksburg, VA, USA

[2] Department of Biochemistry, Virginia Tech, Blacksburg, VA, USA

[3] Department of Chemical Engineering, Virginia Tech, Blacksburg, VA, USA

ABSTRACT

Overproduction of valuable metabolic compounds has been a primary goal of metabolic engineering. Identification of effective metabolic engineering strategies has proven to be an on-going challenge due to metabolic complexity. With recent advances in bioinformatics and computational biology, many computational algorithms have been developed to design metabolic engineering strategies. Predictive algorithms that utilize constraint-based genome-scale metabolic flux models (GEMs) have shown promising results. Nearly all of these algorithms aim to generate many alternative designs, which leads to redundancies and requires expert-level interpretation. Redundant strategies can hinder down-stream cross-referencing and slow the experimental validation processes, thus reducing overall research efficiency. To address this issue, we developed the **N**ode-**R**eward-**Opt**imization (NR-Opt) toolbox, which consists of a set of fast and accurate algorithms that predicts ranked non-redundant gene knockout (KO), overexpression (OX), and knock-down (KD) strategies. The core programs of the NR-Opt toolbox implements a modified steepest ascent hill climbing algorithm to enable rapid convergence to the best design. We deployed NR-Opt to design strategies for the overproduction of 1,4-butanediol (BDO) in *E. coli* and cellulose in *Arabidopsis thaliana*. Within minutes, NR-Opt designed four gene knockout strategies and two gene overexpression strategies to overproduce BDO in *E. coli*. The best strategy is a triple-knockout of alcohol dehydrogenase, lactate dehydrogenase, and pyruvate formate lyase, and this strategy has been experimentally validated previously. Although NR-Opt could not design knockout strategies to overproduce cellulose in Arabidopsis, it designed two gene overexpression strategies. The best strategy is the overexpression of cellulose synthase, and it has been shown to significantly increase cell wall

cellulose content in Arabidopsis in previous experimental studies. Overall, NR-Opt is a fast, accurate, and concise algorithm for designing metabolic engineering strategies.

INTRODUCTION

Cells produce an extraordinary variety of metabolites to support growth and fitness. Many studies have shown that it is possible to engineer cellular metabolism to overproduce specific metabolites with high commodity values, which can include biofuels, commodity chemicals, pharmaceuticals, and polymers (Kempinski et al., 2015; Park et al., 2007; Phithakrotchanakoon et al., 2013; Yim et al., 2011). Due to the complexity of cellular metabolism, the identification an optimal metabolic engineering strategies is a challenge. Recent advances in bioinformatics, computational biology, and modeling have enabled faster and more accurate designs. Algorithms involving the utilization of genome-scale models (GEMs) and flux-based modeling shown in the literature to yield effective designs (Agren et al., 2013; Burgard et al., 2003; McAnulty et al., 2012; Meng et al., 2011; Oddone et al., 2009; Yim et al., 2011). In fact, a wide range of design algorithms involving GEMs now exist essentially to achieve the same goal, but they (i) use different approaches, (ii) allow for different numbers of genetic manipulations, (iii) converge at different speeds, and (iv) consider the availability of different genetic tools for implementation.

Flux-based modeling with GEMs utilizes mass balancing, making it a valuable approach to evaluate the maximum or minimum theoretical yield of biosynthesis. All predictive algorithms that uses GEMs, such as OptKnock, OptGene, RobustKnock, ReacKnock, BAFBA, OptForce, and EMILiO, take advantage of this feature to design strategies that ensure a high theoretical target yield as an obligatory byproduct of growth (Burgard et al., 2003; Choon et al., 2014; Patil et al., 2005; Ranganathan et al., 2010; Tepper and Shlomi, 2010; Xu et al., 2013; Yang et al., 2011). It has been proposed that biomass-product coupled yield (BPCY), which is defined as the rate of target chemical formation multiplied by growth rate, is the most appropriate metric of

assessing cell productivity (Choon et al., 2014; Kim et al., 2012). Genetic tools to implement a metabolic engineering strategy essentially include gene knockout (KO), overexpression (OX), and expression knock-down (KD); although it is recognized there are several ways of accomplishing these desired effects, including altering regulatory elements. Algorithms that predict KO strategies, such as OptKnock and ReacKnock are popular due to the aggressive nature of gene KO and the relatively straightforward experimental procedure compared to OX or KD (Burgard et al., 2003; Xu et al., 2013). Recently, algorithms that can predict OX and KD or combinations of all three strategies, such as OptForce and EMILiO, have yield promising results (Ranganathan et al., 2010; Yang et al., 2011). So far, nearly all algorithms take the approach of returning many different designs, which are first screened computationally, expert-analyzed manually, and then evaluated experimentally. The caveat is that a manual assessment of the strategies is always necessary prior to experimentation, but the size of the prediction lists are usually in the hundreds (Yang et al., 2011; Yim et al., 2011). Some designs can contain redundant modifications that do not improve yield significantly. By first understanding the core strategies, researchers can then select the most rational strategies to validate (Yim et al., 2011). Considering that model predictions serve to guide strategy design, it would be more efficient to the overall workflow if there were only non-redundant designs that contain the core metabolic engineering strategy. It is also beneficial to repeat the design process multiple times, and often algorithm convergence time can be a limiting factor, especially as the design involves more genetic manipulations. Some available algorithms, such as EMILiO, ReacKnock, and DBFBA, operate on the scale of minutes, but they return many designs, which must be screened further and interpreted before implementation.

With speed, non-redundancy, and robustness as goals, we developed the **N**ode-**R**eward **Opt**imization (NR-Opt) toolbox, to enable efficient metabolic engineering strategy design that exceed the state-of-the-art standards. Similar to OptKnock, the algorithms in the NR-Opt toolbox are top-down, which means they perform modifications to the model first then examine the outcomes (Yen et al., 2015). The algorithms implemented in the NR-Opt core programs are unique from all other design algorithms because they integrate a modified steepest ascent hill climbing algorithm that allows for delayed ascension and uses a minimum improvement threshold, which reduces search iterations, enables more rapid convergence to the local maxima, and improves the chances of reaching the global maxima. The NR-Opt toolbox is primarily made of two core programs: (i) NR-Knock and (ii) NR-Ox. The NR-Knock algorithm is designed to predict KO strategies, and NR-Ox predicts OX and/or KD strategies. The NR-Opt toolbox is coded in MATLAB and optimized to run on parallel computing systems. Prediction of KO strategies and OX/KD strategies are separated because, as noted by others, gene KO is most likely to force metabolic reprogramming. The NR-Ox algorithm provides additional strategies when the predicted KO strategies are insufficient, ineffective, or absent. As a proof-of-concept, the NR-Opt toolbox was deployed to design metabolic engineering strategies for the overproduction of BDO in *E. coli* and cellulose in *Arabidopsis thaliana*. These designs are validated with experimental results available in the literature to demonstrate its utility with both microbes and eukaryotes.

MATERIALS AND METHODS

The NR-Knock and NR-Ox algorithms

Both NR-Knock and NR-Ox use a modified steepest ascent hill climbing algorithm to enable short non-optimal neighborhood searches for design strategies. In brief, metabolic flux

modifications, including: eliminating (KO), increasing (OX), and decreasing flux (KD), in a GEM and numerous combinations of these make up the entire search space. The challenge is to reduce this enormous search space without eliminating optimal solutions. To improve efficiency of experimental validation, the designed strategies with "extra" metabolic modifications (which do not improve the outcomes) are termed "redundant" strategies and are removed. The NR-Opt toolbox reduces this search space intelligently to critical reactions and combinations to predict all the non-redundant strategies with significantly improved BPCY. The NR-Opt toolbox was coded in MATLAB R2016a with the parallel computing toolbox, and it comes with custom FBA and flux variability analysis (FVA) solvers that use the MOSEK 7 optimization software (https://www.mosek.com/). All the codes to the NR-Opt toolbox can be found in the Supplemental Information (Appendix B). Parallel computing was performed at Advance Research Computing at Virginia Tech using Dragontooth, a 48-node system with a 2x Intel Xeon E5-2680v3 (Haswell) 2.5 GHz 12-core CPU and a 256 GB 2133 MHz DDR4 RAM in each node. All runs were performed with 4 nodes (96 cores) unless specified. SBML models were converted to COBRA format with the COBRA Toolbox (Schellenberger et al., 2011).

The overall logic flow of the NR-Knock and NR-Ox algorithm is shown in Figure 3-1 and the step-by-step process is described in Appendix A. The designed metabolic engineering strategies are a set of genetic modifications. Thus, the entire search space is first reduced by keeping only enzyme-catalyzed reactions with gene annotation in the GEM. After this reduction, NR-Knock and NR-Ox each treat the search space differently. NR-Knock further reduces the search space by excluding reactions that do not carry meaningful flux after evaluating with FVA. Meaningful fluxes are ones that do not resemble futile cycles and not artifacts of computational precision. In this study, flux values between 10^{-9} and 150 $mmol \cdot gDCW^{-1} \cdot h^{-1}$ were considered

meaningful. One of the features of the NR-Opt algorithms is the further reduction of search space by evaluating flux ratios. The flux ratio constraint is based on the principle of multiple reactions competing for the same limited metabolite pool. The metabolite distribution is determined by the thermodynamics and the availability of enzymes, which can be altered through genetic modifications (McAnulty et al., 2012; Yen et al., 2013). Reduction of the search space using flux ratio constraints serves as the first elimination of redundant predictions. Flux ratios are presented as reaction-node pairs, the competing reactions, and the limited metabolite pool(s). In NR-Knock, the FVA flux solutions are used to determine flux ratio ranges. Reactions that do not have associated flux ratios are removed because these represent linear metabolic pathways where metabolism cannot be re-routed. The remaining reactions now feed into the steepest ascend hill climbing search, which iteratively searches for a combination of KOs that can generate the best BPCY. In the first iteration, single-KO strategies are tested by constraining each reactions to zero flux then perform FBA with the objectives of maximizing growth (G) and minimizing target product yield (Y) to calculate BPCY, which is the product of G and Y (Choon et al., 2014). Each single-KO strategy is assigned an initial reward point value specified by the user ($p_i \leftarrow p_0$, where p_0 is the initial reward point and p_i is the points for strategy i). Strategies that have non-zero BPCY are rewarded additional points. Likewise, each ineffective strategy is deducted one point. If the point falls below zero, then the strategy is eliminated from expansion. The stack of strategies is ranked by highest BPCY after the first search iteration. The second iteration of this depth-first search continues from the top of the strategy stack. The top KO strategy, now the "parent" strategy, is combined with an additional KO of each reaction in the search space except itself to form a "child" strategy. The BPCY and p of this double-KO strategy is evaluated as described previously. Then, the strategy stack expands if the strategy has a better BPCY. Each

iteration of the depth-first search is optimized for parallel computing. The search loop ends when the stack is empty or at least one of the strategies meets the minimum BPCY goal ($BPCY_{acceptance}$) assigned by the user. The list of designs is generated and updated as the program runs to allow for real-time monitoring of results. A build of NR-Knock for a local machine is also available, and FastFVA (Gudmundsson and Thiele, 2010) is used.

The NR-Ox algorithm is fundamentally the same as NR-Knock with the notable exception that the search space is significantly larger and each strategy involves altering flux ratios of relevant reactions. The magnitude of flux ratio increase or decrease can be defined by the user. As shown in Figure 3-1, after the initial reduction, NR-Ox proceeds to its iterative search for strategies. With each iteration, FBA is performed with the top parent strategy (wild-type for the first iteration) to determine the new set of flux ratios. In contrast with NR-Knock, where the search space is the same for each iteration, NR-Ox redefines search space for each iteration. It is important to emphasize that each reaction can compete in multiple nodes, and thus be involved in multiple flux ratios. Each flux ratio is subjected to increase or decrease as evaluations of overexpression and knock-down strategies. This illustrates the large size of the search space. FBA is performed to evaluate each strategy and BPCY is determined as before. Similarly, the stack of strategies is ranked by BPCY, the p of each strategy is updated as necessary and the iterative search continues until the stack is empty or one strategy meets $BPCY_{acceptance}$.

There are five parameters that are critical to robustness of design and required CPU time. The first is the minimum score increment (dv_{cutoff}), which is the minimum required BPCY increase from the current best BPCY if a strategy is qualified for reward. If the dv_{cutoff} is too high, all valid strategies are dismissed. If it is too low, insignificant strategies are qualified for

expansion and search space can become unnecessarily large. The default value is 10% of the predicted maximum theoretical BPCY. The initial reward point (p_0), mentioned previously, and the maximum allowable points (p_{max}) are two parameters that determine how many expansions a strategy can have without improving BPCY. A p_0 of zero means that any strategy that does not improve BPCY of the wild-type (WT) model in the first search iteration is eliminated. To maintain the integrity of good strategies, p_{max} is used to limits the number of continuous sub-optimal expansions. This feature enables faster search by allowing search of optimal strategies from sub-optimal strategies (Figure 3-2). In contrast to Bees Hill Flux Balance Analysis (Choon et al., 2015), which uses a traditional hill climbing algorithm, NR-Opt modifies the hill climbing algorithm to allow for short continuation of neighborhood search in non-optimal local solution in attempt to find the global maxima. Allowing sub-optimal solution expansion does not introduce redundant strategies in NR-Knock predictions, but it can introduce redundant strategies in NR-Ox predictions, as search spaces can be different for each iteration. Thus, it is necessary to perform a reduction to eliminate unnecessary flux ratio modifications in preliminary NR-Ox strategies after the search is complete. The fourth critical parameter is the maximum number of modifications (N_{max}). A high N_{max} can increase CPU time (but not significantly) because most strategies are eliminated from expansion early on. This is, unless, dv_{cutoff} is small and p_{max} is high. The fifth critical parameter is $BPCY_{acceptance}$, which is the cutoff for $BPCY_{strategy}/BPCY_{max}$ to allow for an early termination. It is important to note, just as in all hill climbing algorithms, NR-Opt is not guaranteed find the global optima, which is why these five parameters are critical and should be adjusted if good strategies cannot be found.

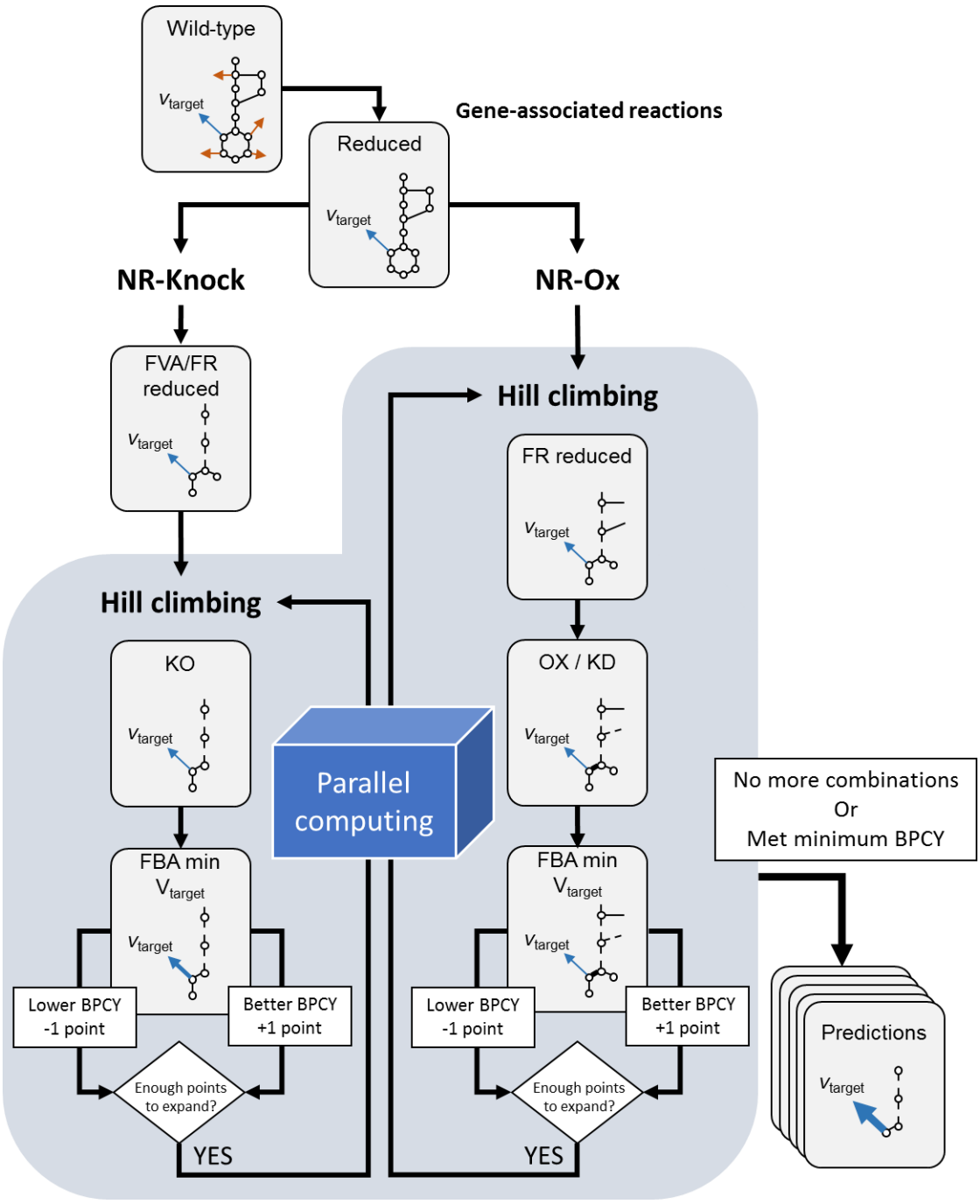

Figure 3-1. The NR-Knock and NR-Ox algorithms. Complete algorithm is described in detail in Materials and Methods.

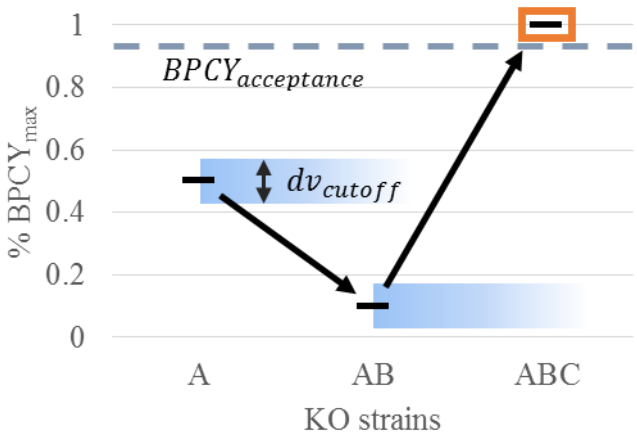

Figure 3-2. Searching for optimal strategy from suboptimal strategy in the NR-Opt algorithm. This plot shows a hypothetical scenario of NR-Knock predicting KO strains. The KO strains represents KO of reaction A (A), double-KO of reactions A and B (AB), and triple-KO of reactions A, B, and C (ABC). The $\%BPCY_{max}$ for of the predicted strategies are shown as black bars. $BPCY_{acceptance}$ (dash line) and dv_{cutoff} range are shown. The optimal strategy is highlighted in orange box.

NR-Opt setup for E. coli and Arabidopsis case studies

A previously constructed GEM of *E. coli* (Ec-iAF1260) was modified to include the BDO biosynthesis pathway (Feist et al., 2007; Yim et al., 2011). This adds 6 reactions for 2-oxoglutarate decarboxylase, CoA-dependent succinate semialdehyde dehydrogenase, 4-hydroxybutyrate dehydrogenase, 4-hydroxybutanoate CoA-transferase, 4-hydroxybutyryl-CoA reductase, and alcohol dehydrogenase (for the conversion of 4-hydroxybutryraldehyde to BDO) to the Ec-iAF1260 model, and generates BDO-WT. An additional BDO exchange was added as necessary. BPCY was calculated as the product of growth and the BDO production flux.

A previously constructed *Arabidopsis thaliana* GEM, AraGEM, (de Oliveira Dal'Molin et al., 2010), was used in the second case-study. Because cellulose is already a part of the biomass equation, a separate cellulose exchange was added and used as the target reaction to determine if excess cellulose could be produced. This generatd the Cellulose-WT model. BPCY was calculated as the product of growth and the cellulose exchange flux. The NR-Opt parameters for both case studies are shown in Table 3-1.

Table 3-1. NR-Opt parameters used in prediction.

	BDO in *E. coli*		**Cellulose in Arabidopsis**	
	NR-Knock	**NR-Ox**	**NR-Knock**	**NR-Ox**
Model	BDO-WT	BDO-Mut 8	Cellulose-WT	Cellulose-WT
Target reaction	BDO exchange		Cellulose exchange	
dv_{cutoff}	0.1	0.001	0.0001	0.0001
p_0	0	0	2	1
p_{max}	1	1	2	1
N_{max}	4	4	4	4
$BPCY_{acceptance}$	1	0.95	0.9	1
Real time, s (1)	41.2 (2)	185.7	1259.1	373.2

(1) Performance on 96 cores of the system described in Material and Methods
(2) 57.7 s with 24 cores, and 248.1 s with 1 core on a laptop.

RESULTS

Predicted metabolic engineering strategies to increase BDO yield from E. coli

A gene KO strategy to increase 1,4-butanediol (BDO) yield in *E. coli* was predicted previously using OptKnock, and validated experimentally (Yim et al., 2011). To demonstrate that the NR-Opt toolbox can predict robust metabolic engineering strategies, we examined whether NR-Opt would design the same or a similar strategy to increase BDO production from *E. coli*.

When FBA was performed on the modified Ec-iAF1260 model (BDO-WT), no BDO production was predicted as a byproduct of growth. NR-Knock was deployed on the BDO-WT model and 4 KO strategies to increase minimum BDO BPCY were designed. The conventional evaluation method of designed strategy is by examining the production envelope to identify the predicted minimum theoretical product yield as cells evolve or adjust towards maximum growth. As shown in Figure 3-3A, all 4 NR-Knock strategies ranked by BPCY were compared against the validated strategy (Yim et al., 2011). All of the NR-Knock strategies have equivalent or lower minimum BDO yield at maximum growth as compared to the validated strategy. The designed strategies do not have high minimum BDO yield at maximum growth because the NR-Knock objective was to maximize BCPY. As shown in Figure 3-3B, the calculated BPCY envelopes of all the NR-Knock strategies have greater predicted BPCY and faster growth except of BDO-Mut 4. The BDO-Mut 4 design is interesting because its calculated BPCY envelope shows a mutant with alcohol dehydrogenase (*adh*) KO can generate high BDO, but the production is not yet at its theoretical maximum, as indicated by the round head space above the maximum growth rate point. In contrast, BDO-Mut 1 and 3 designs, and the validated Yim et al.

strategy, have reached their maximum theoretical BPCYs when the strains achieve maximum growth, as indicated by the sharp peak.

Table 3-2 shows (i) the theoretical maximum BPCY of BDO, (ii) the BPCY of Yim et al. KO strategy, and (iii) the top NR-Knock designs. Although the maximum number of KOs was set to 4, the best strategies were 3 KOs or fewer. The validated 4-KO strategy implemented in Yim et al. has a BCPY that is 78.6% of the theoretical maximum, which is less than all the strategies predicted by NR-Knock. The best strategies predicted by NR-Knock is a 3-KO strategy (BDO-Mut 1) with a BCPY that is 99.94% of the BDO-WT theoretical maximum (Table 3-2). This strategy was validated experimentally in Yim et al. to produce significantly higher yield than the WT strain containing the BDO pathway (Yim et al., 2011). It is worth noting that BDO-Mut 4 is a single KO strategy that has a BPCY similar to the OptKnock strategy of Yim et al. The gene targets in all of the NR-Knock strategies are similar to those of the validated Yim et al. strategy, in which they all attempt to limit the production of ethanol, lactate, and formate to force BDO production (Yim et al., 2011). In addition, NR-Knock was able generate these predictions in 57.7 seconds with 24 cores of the computer system described in Materials and Methods. Importantly, when NR-Knock was performed using one core on a local machine with an Intel Core i7-3537U 2.0 GHz CPU and an 8 GB RAM, it required 4.1 minutes to design the same KO strategies. By contrast, this is significantly faster than OptKnock, on the scale of tens of minutes with 48 cores, and ReacKnock, on the scale of minutes with 48 cores, when solving similar problems (Xu et al., 2013). DBFBA is a very fast and robust algorithm, but it requires 34 minutes to solve a similar problem on a local machine (Choon et al., 2014). EMILiO is arguably the fastest algorithm to date with a run time as low as one minute, and it can predict KO, OX, and KD strategies simultaneously (Yang et al., 2011). The drawback is that EMILiO predicts

many redundant strategies with additional modifications to the core strategy that do not lead to significant differences. NR-Knock can perform on the same time scale as EMILiO, but redundant predictions are eliminated.

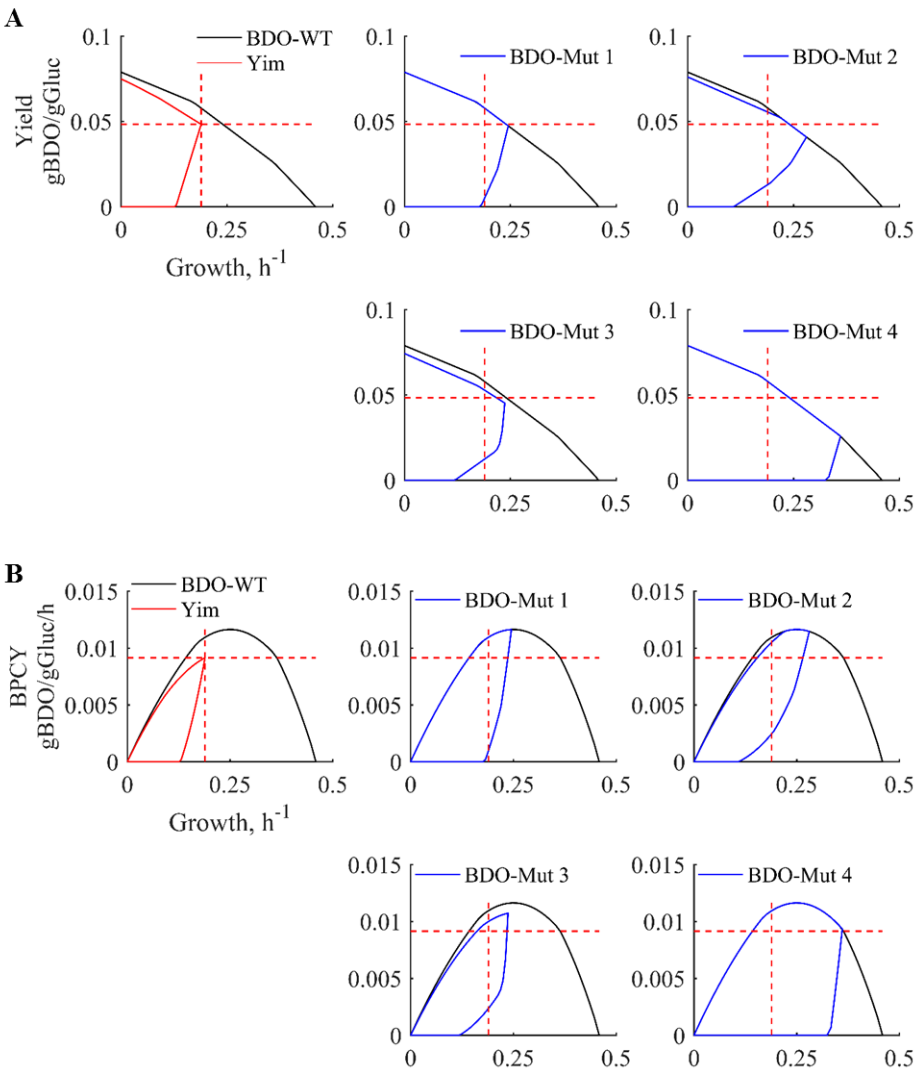

Figure 3-3. Conventional production envelopes and BPCY envelopes of predicted KO strategies to increase BDO yield. Conventional production envelopes (A) and BPCY envelopes (B) for BDO-WT (black line), Yim et al. OptKnock strategy (red line), and 4 NR-Knock strategies (blue lines). The maximum growth rate (vertical red dashed line) and BDO yield or BPCY (horizontal red dashed line) of Yim et al. Optknock strategy is shown in every plot for comparison.

Table 3-2. Predicted gene KO strategies to increase BDO BPCY in E. coli.

Strains	Reactions to KO	Genes	BPCY
BDO-WT	(No KO)		0
	(Theoretical maximum)		0.0116
Yim et al.	Ethanol + NAD^+ ↔ Acetaldehyde + H^++ NADH	*adh*	0.0091
	D-Lactate + NAD^+ ↔ Pyruvate + H^++ NADH	*ldh*	
	CoA + Pyruvate ↔ Acetyl-CoA + Formate	*pfl*	
	L-Malate + NAD^+ ↔ H^+ + NADH + Oxaloacetate	*mdh*	
BDO-Mut 1	Ethanol + NAD^+ ↔ Acetaldehyde + H^+ + NADH	*adh*	0.0116
	D-Lactate + NAD^+ ↔ Pyruvate + H^++ NADH	*ldh*	
	CoA + Pyruvate ↔ Acetyl-CoA + Formate	*pfl*	
BDO-Mut 2	Ethanol + NAD^+ ↔ Acetaldehyde + H^+ + NADH	*adh*	0.0115
	ADP + 4 H^+ + Phosphate ↔ ATP + H_2O + 3 H+	*atp*	
	CoA + Pyruvate ↔ Acetyl-CoA + Formate	*pfl*	
BDO-Mut 3	Ethanol + NAD^+ ↔ Acetaldehyde + H^+ + NADH	*adh*	0.0107
	ADP + 4 H^+ + Phosphate ↔ ATP + H_2O + 3 H+	*atp*	
	D-Glucose 6-phosphate ↔ D-Fructose 6-phosphate	*pgi*	
BDO-Mut 4	Ethanol + NAD^+ ↔ Acetaldehyde + H^+ + NADH	*adh*	0.0093

To examine the performance of NR-Ox, the BDO-Mut 4 model (which contains the *adh* KO) was used to determine if a gene overexpression or knock-down strategy exists that can further improve its BPCY. The BDO-Mut 4 model was used because it contains a single gene KO and its BPCY was less than the theoretical maximum. NR-Ox returned 2 design strategies, given the parameters listed in Table 3-1, and these are shown in Table 3-3. Interestingly, NR-Ox predicted the importance of pyruvate dehydrogenase (*lpdA*) activity in the production of BDO (Yim et al., 2011). NR-Ox also calculated that if the overexpression of *lpdA*, in the presence of the *adh* KO, can partition 90.4% of the total pyruvate through pyruvate dehydrogenase reaction the BDO BCPY can improve from 80.2% to 100% of the theoretical maximum. For this study, NR-Ox was parameterized to search for at most 4 modifications, which included overexpression and/or knock-down of enzyme catalyzed reactions. Even though the solution space became very large, NR-Ox required only 3.1 minutes (using the 96 core configuration described in Materials and Methods) to design an optimal strategy.

Table 3-3. NR-Ox predictions to increase BDO BPCY in E. coli strain with adh KO.

Strain	$\Delta FR^{(1)}$	Nodes	Reactions	Genes	BPCY
BDO-adh-Mut 1	+0.9	Pyruvate	CoA + NAD^+ + Pyruvate ↔ Acetyl-CoA + CO_2 + NADH	*lpdA*	0.0116
BDO-adh-Mut 2	+0.315	D-Erythrose 4-phosphate	D-erythrose 4-phosphate + H_2O + Phosphoenolpyruvate ↔ 2-Dehydro-3-deoxy-D-arabino-heptonate-7-phosphate + Phosphate	*aroG*	0.0094

(1) The amount of change to the initial flux ratio. Positive change indicates gene OX and negative change indicates gene KD. The maximum is ±1.

Predicted metabolic engineering strategies to increase cellulose content in Arabidopsis

The NR-Opt toolbox was used with the modified AraGEM model (Cellulose-WT) to predict metabolic engineering strategies to increase cellulose production in *Arabidopsis thaliana*. NR-Knock was unable to predict any KO strategy even when p_0 and p_{max} were large, as shown in Table 3-1. Given the size of the Cellulose-WT model, over 4 million KO strategies were evaluated in 21 minutes of real time (using the 96-core configuration described previously). This result also suggests that no metabolic characteristics are present to allow for repartitioning of cellular resources to overproduce cellulose. However, NR-Ox designed two gene overexpression strategies. As shown in Table 3-4, the strategy with the highest predicted BPCY (Cellulose-Mut 1) is the overexpression of cellulose synthase (CesA). NR-Ox predicted that if CesA can partition 97% of the total UDP-glucose, then 100% of theoretical maximum BCPY can be achieved. The drawback of this strategy is that growth is reduced by 64% compared to wild-type (WT).

The second strategy (Cellulose-Mut 2) requires upregulating both a nucleoside diphosphate kinase (Ndk1) and a UDP-glucose pyrophosphorylase (UGPase). Cellulose-Mut 2 has a significantly lower predicted BPCY, which is only 9.5% of the theoretical maximum. However, further examination found that the low BPCY is primarily due to the significantly reduced predicted growth (4.4% of WT). Using slightly lower flux ratios on UGPase|UTP and Ndk1|ATP reaction|node pair than the default assigned in the NR-Ox run, lost growth is recovered and BPCY is further improved. The optimal flux ratios for UGPase in the UTP node and Ndk1 in the ATP node to maximize BPCY are shown in Figure 3-4. Analysis of the predicted flux distribution revealed that Ndk1 is predicted to have very low flux in WT, and flux through UGPase is reduced when the Ndk1 flux ratio increases.

Table 3-4. NR-Ox predicted overexpression strategies to increase cellulose BPCY.

Strain	ΔFR[1]	Nodes	Reactions	Genes	BPCY[2]
Cellulose-WT		(No modification)[3]			0
		(Theoretical maximum)			0.0063
Cellulose-Mut 1	+0.295	UDP-Glucose	UDP-glucose ↔ Cellulose + UDP	CesA	0.0063
Cellulose-Mut 2	+0.894	ATP	ATP + UDP ↔ ADP + UTP	Ndk1	0.0006
	+0.898	UTP	D-Glucose 1-phosphate + UTP ↔ Pyrophosphate + UDP-glucose	UGPase	

(1) The amount of change to the initial flux ratio. Positive change indicates gene OX and negative change indicates gene KD. The maximum is ±1.
(2) Modified calculation: $BPCY = v_{cellulose} \cdot v_{Growth}$.
(3) Does not include the amount already in the biomass equation.

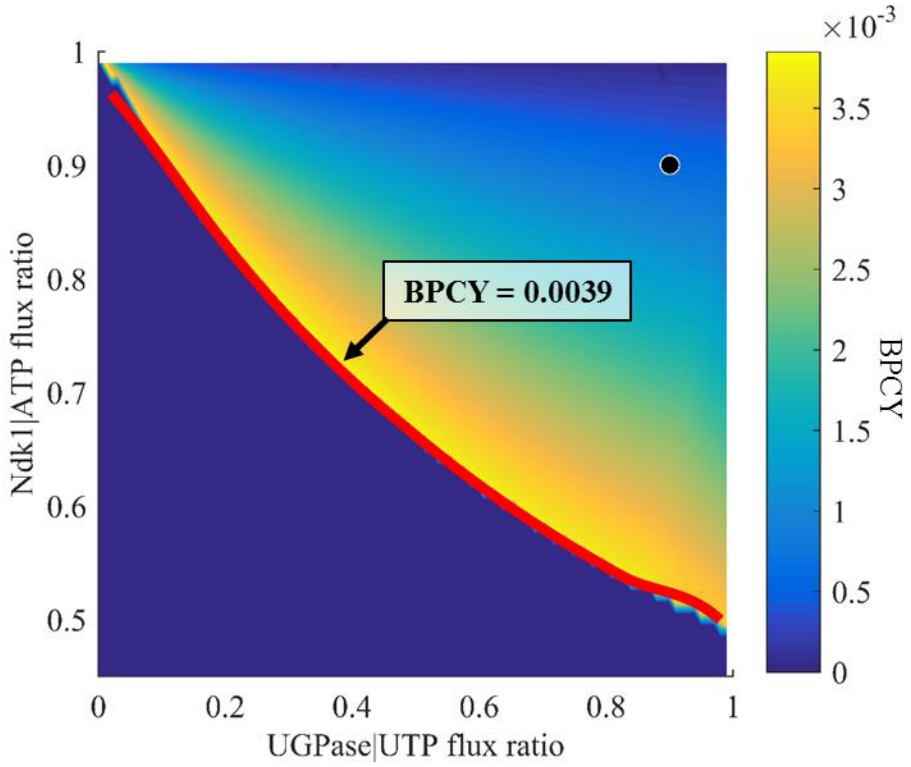

Figure 3-4. The dynamics of flux ratios of UGPase|UTP and Ndk1|ATP. BPCYs of different flux ratios of UGPase|UTP and Ndk1|ATP are shown by the color gradient. The optimal flux ratios to achieve maximum theoretical BPCY of 0.0039 is shown by the red curve. The flux ratios examined by NR-Ox during run is shown by the black dot.

DISCUSSION

Computational predictions can aid the metabolic engineering process significantly. However, predictions from new algorithms should be cross-referenced with previous experimental studies to determine their degree of validity. Adjustments to the metabolic model may be required; thus, an iteration approach may be necessary. The NR-Opt toolbox, presented here for the first time, has shown to be capable of designing accurate, complex, yet concise metabolic engineering strategies in near-record time. Furthermore, the NR-Opt toolbox incorporates the genomic tools of KO, OX, and KD and offers a dynamic set of parameters to give users full control of run time and robustness. The NR-Opt toolbox has demonstrated shorter run-times due to competing algorithms. This allows users to explore more complex strategies (i.e. include more genetic manipulations) and repeat model designs. The reward parameters allow the user to reject redundant strategies, specifically ones with additional genetic manipulations that do not improve BCPY. This effectively removes them from FBA analysis, the most time-consuming step of the algorithm. This feature also allows the resulting list of designs to be concise, simplifying downstream evaluation and experimental validations.

In the case of engineered BDO production in *E. coli*, OptKnock predicted a list of 203 metabolic engineering strategies (Yim et al., 2011). Cross-referencing these with literature, to determine which to implement experimentally, can be tedious without expert knowledge. On the other hand, NR-Knock returned 4 optimal strategies, and each retained only necessary gene KO's that can improve BPCY significantly. Thus, fewer refinements of results and experimental validations are required, and implementing any of the top NR-Knock designs will likely yield a favorable result. The differences between the sizes of the design strategy lists and the predictions themselves can be attributed to the metabolic models used in the studies; however, it is primarily

due to the differences in the design algorithms. It is worth emphasizing that the top ranking NR-Knock strategy is an experimentally validated strategy by Yim et al., but it was not top ranking strategy designed by OptKnock (Yim et al., 2011). Another important point is that Yim et al. incorporated additional modifications on top of their OptKnock design, which ultimately lead to much higher BDO production. The complete Yim et al. strategy included four KOs of *adh*, lactate dehydrogenase (*ldh*), pyruvate formate lyase (*pfl*), and *mdh*, which were predicted by OptKnock, and three other genetic modifications that were rationalized by the authors based on the OptKnock strategy (Yim et al., 2011). In this strategy, *mdh* KO played a role of decreasing oxidative TCA cycle to increasing reducing equivalent reserve for BDO biosynthesis (Yim et al., 2011). The three additional genetic modifications served to reinforce the OptKnock strategy. It is argued that with a quick evaluation of the short NR-Opt prediction list, a researcher could design the final strategy implemented in Yim et al more rapidly.

Similarly, NR-Opt returned a very short list of designs (Table 3-4) to increase cellulose production in Arabidopsis. This is interesting because the predicted strategies contain, so far, the only two experimentally validated strategies despite decades of cellulose research (Kim et al., 2013; Wang et al., 2011). This result implies the challenge of cellulose engineering owning to the robustness of Arabidopsis metabolism. Previous experimental study showed that the overexpression of a transcription factor, MYB46, in Arabidopsis can upregulate the cellulose synthase (CesA) complex and increase crystalline cellulose concentration in leaves by 30%. The up-regulation of CesA complex by overexpressing MYB46 can be considered a “true” strategy to increase cellulose, but this comes at the expense of growth (Kim et al., 2013). The reduced plant growth due to MYB46 overexpression has shown to be independent of cell wall thickening, which suggests it could be a metabolic consequence (Kim et al., 2013; Ko et al., 2009). The

reduction of growth can be compensated by inducing MYB46 overexpression later in the plant developmental stage (Kim et al., 2013). This suggests that cellulose production may still have room to increase, but this seems to be at the expense of plant growth. The analysis on the dynamics between Ndk1 and UGPase (Figure 3-4) shows that a coordinated adjustment of Ndk1|ATP and UGPase|UTP flux ratios is required to ensure optimal BPCY from this strategy. Although there is currently no evidence correlating Ndk1 expression to cellulose biosynthesis, multiple studies have found that the overexpression of UGPase can increase cellulose content (Li et al., 2014; Zhang et al., 2013). Overexpression of UGPase in Arabidopsis increases crystalline cellulose concentration in the stem, but this increase is much smaller compared to MYB46 overexpression (Wang et al., 2011). It was shown that UGPase overexpression increases plant growth; however, FBA calculated reduced growth (Wang et al., 2011). In the modeling process, metabolic flux in AraGEM is calculated by FVA and FBA under the assumption of a maximal growth objective, which means all available resources are directed to growth, or transport reactions that serve growth through mass balancing. The disagreement between predicted growth consequences and experimental growth improvement suggests that the assumption of maximizing growth may not be accurate. This growth objective inaccuracy suggests that there is an underlying biological complexity in Arabidopsis that prevents maximal uptake of nutrients and/or maximal partitioning of nutrient into growth. Although the Cellulose-Mut 2 design is not entirely consistent with experimental evidence, this becomes an excellent example of using predictive algorithms to guide the understanding of biology and to contribute in the model improvement cycle.

Finally, it is important to emphasize that regardless of the metabolic engineering strategy design algorithm being used, the quality of the design depends heavily on the quality of the

metabolic model. Cross-validation of designs with multiple GEMs and different design algorithms to explore robustness is still recommended, as noted previously (Choon et al., 2014). Because NR-Opt is fast and the predictions are more concise, it can serve as the first pass in the design process. Algorithms, such as EMILiO, ReacKnock, and DBFBA, can then be used for cross-validation and to explore alternative strategies.

CONCLUSIONS

The NR-Opt toolbox has been developed as a set of fast, accurate, and concise predictive algorithms that uses GEMs to design metabolic engineering strategies. The NR-Opt toolbox was applied in a proof-of-concept to design strategies that can theoretically increase BDO production in *E. coli* and cellulose production in Arabidopsis. Cross-validation with published experimental results showed that NR-Opt can efficiently generate complex and accurate designs. Its run time is among the fastest of any available design algorithms, and it returns a concise list of top-rated designs to accelerate the experimental validation process.

REFERENCES

Agren, R., Otero, J. M., Nielsen, J., 2013. Genome-scale modeling enables metabolic engineering of Saccharomyces cerevisiae for succinic acid production. Journal of industrial microbiology & biotechnology. 40**,** 735-747.

Burgard, A. P., Pharkya, P., Maranas, C. D., 2003. Optknock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. Biotechnology and bioengineering. 84**,** 647-657.

Choon, Y. W., Mohamad, M. S., Deris, S., Chong, C. K., Omatu, S., Corchado, J. M., 2015. Gene knockout identification using an extension of Bees Hill Flux Balance Analysis. BioMed research international. 2015.

Choon, Y. W., Mohamad, M. S., Deris, S., Illias, R. M., Chong, C. K., Chai, L. E., Omatu, S., Corchado, J. M., 2014. Differential bees flux balance analysis with OptKnock for in silico microbial strains optimization. PloS one. 9**,** e102744.

de Oliveira Dal'Molin, C. G., Quek, L. E., Palfreyman, R. W., Brumbley, S. M., Nielsen, L. K., 2010. AraGEM, a genome-scale reconstruction of the primary metabolic network in Arabidopsis. Plant physiology. 152**,** 579-89.

Feist, A. M., Henry, C. S., Reed, J. L., Krummenacker, M., Joyce, A. R., Karp, P. D., Broadbelt, L. J., Hatzimanikatis, V., Palsson, B. Ø., 2007. A genome-scale metabolic reconstruction

for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. Molecular systems biology. 3**,** 121.
Gudmundsson, S., Thiele, I., 2010. Computationally efficient flux variability analysis. BMC bioinformatics. 11**,** 489.
Kempinski, C., Jiang, Z., Bell, S., Chappell, J., 2015. Metabolic Engineering of Higher Plants and Algae for Isoprenoid Production. In: Schrader, J., Bohlmann, J., Eds.), Biotechnology of isoprenoids. Springer International Publishing, Cham, pp. 161-199.
Kim, J.-W., Chin, Y.-W., Park, Y.-C., Seo, J.-H., 2012. Effects of deletion of glycerol-3-phosphate dehydrogenase and glutamate dehydrogenase genes on glycerol and ethanol metabolism in recombinant Saccharomyces cerevisiae. Bioprocess and biosystems engineering. 35**,** 49-54.
Kim, W. C., Ko, J. H., Kim, J. Y., Kim, J., Bae, H. J., Han, K. H., 2013. MYB46 directly regulates the gene expression of secondary wall-associated cellulose synthases in Arabidopsis. The plant journal. 73**,** 26-36.
Ko, J.-H., Kim, W.-C., Han, K.-H., 2009. Ectopic expression of MYB46 identifies transcriptional regulatory genes involved in secondary wall biosynthesis in Arabidopsis. The plant journal. 60**,** 649-665.
Li, N., Wang, L., Zhang, W., Takechi, K., Takano, H., Lin, X., 2014. Overexpression of UDP-glucose pyrophosphorylase from Larix gmelinii enhances vegetative growth in transgenic Arabidopsis thaliana. Plant Cell Rep. 33**,** 779-791.
McAnulty, M. J., Yen, J. Y., Freedman, B. G., Senger, R. S., 2012. Genome-scale modeling using flux ratio constraints to enable metabolic engineering of clostridial metabolism in silico. BMC systems biology. 6**,** 42.
Meng, H., Lu, Z., Wang, Y., Wang, X., Zhang, S., 2011. In silico improvement of heterologous biosynthesis of erythromycin precursor 6-deoxyerythronolide B in Escherichia coli. Biotechnology and bioprocess engineering. 16**,** 445-456.
Oddone, G. M., Mills, D. A., Block, D. E., 2009. A dynamic, genome-scale flux model of Lactococcus lactis to increase specific recombinant protein expression. Metabolic engineering. 11**,** 367-381.
Park, J. H., Lee, K. H., Kim, T. Y., Lee, S. Y., 2007. Metabolic engineering of Escherichia coli for the production of L-valine based on transcriptome analysis and in silico gene knockout simulation. Proceedings of the national academy of sciences. 104**,** 7797-7802.
Patil, K. R., Rocha, I., Förster, J., Nielsen, J., 2005. Evolutionary programming as a platform for in silico metabolic engineering. BMC bioinformatics. 6**,** 1.
Phithakrotchanakoon, C., Champreda, V., Aiba, S.-i., Pootanakit, K., Tanapongpipat, S., 2013. Engineered Escherichia coli for Short-Chain-Length Medium-Chain-Length Polyhydroxyalkanoate Copolymer Biosynthesis from Glycerol and Dodecanoate. Bioscience, biotechnology, and biochemistry. 77**,** 1262-1268.
Ranganathan, S., Suthers, P. F., Maranas, C. D., 2010. OptForce: an optimization procedure for identifying all genetic manipulations leading to targeted overproductions. PLoS computational biology. 6**,** e1000744.
Schellenberger, J., Que, R., Fleming, R. M. T., Thiele, I., Orth, J. D., Feist, A. M., Zielinski, D. C., Bordbar, A., Lewis, N. E., Rahmanian, S., Kang, J., Hyduke, D. R., Palsson, B. O., 2011. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. Nature protocols. 6**,** 1290-1307.

Tepper, N., Shlomi, T., 2010. Predicting metabolic engineering knockout strategies for chemical production: accounting for competing pathways. Bioinformatics. 26**,** 536-543.
Wang, Q., Zhang, X., Li, F., Hou, Y., Liu, X., Zhang, X., 2011. Identification of a UDP-glucose pyrophosphorylase from cotton (Gossypium hirsutum L.) involved in cellulose biosynthesis in Arabidopsis thaliana. Plant Cell Rep. 30**,** 1303-1312.
Xu, Z., Zheng, P., Sun, J., Ma, Y., 2013. ReacKnock: identifying reaction deletion strategies for microbial strain optimization based on genome-scale metabolic network. PloS one. 8**,** e72150.
Yang, L., Cluett, W. R., Mahadevan, R., 2011. EMILiO: a fast algorithm for genome-scale strain design. Metabolic engineering. 13**,** 272-281.
Yen, J. Y., Nazem-Bokaee, H., Freedman, B. G., Athamneh, A. I. M., Senger, R. S., 2013. Deriving metabolic engineering strategies from genome-scale modeling with flux ratio constraints. Biotechnology journal. 8**,** 581-594.
Yen, J. Y., Tanniche, I., Fisher, A., Gillaspy, G., Bevan, D., Senger, R., 2015. Designing metabolic engineering strategies with genome-scale metabolic flux modeling. Advances in genomics and genetics. 7**,** 149-160.
Yim, H., Haselbeck, R., Niu, W., Pujol-Baxley, C., Burgard, A., Boldt, J., Khandurina, J., Trawick, J. D., Osterhout, R. E., Stephen, R., 2011. Metabolic engineering of Escherichia coli for direct production of 1, 4-butanediol. Nature chemical biology. 7**,** 445-452.
Zhang, G., Qi, J., Xu, J., Niu, X., Zhang, Y., Tao, A., Zhang, L., Fang, P., Lin, L., 2013. Overexpression of UDP-glucose pyrophosphorylase gene could increase cellulose content in Jute (Corchorus capsularis L.). Biochemical and biophysical research communications. 442**,** 153-158.

CHAPTER 4

MODEL-GUIDED ANALYSIS OF SNRK1.1 OVEREXPRESSION IN ARABIDOPSIS PREDICTS SIGNIFICANT CHANGES IN STARCH METABOLISM OVER PLANT DEVELOPMENT

Jiun Y. Yen [1,2], Sarah P. Williams [2], Ryan S. Senger [1,3], Glenda E. Gillaspy [2]

[1] Department of Biological Systems Engineering, Virginia Tech, Blacksburg, VA, USA

[2] Department of Biochemistry, Virginia Tech, Blacksburg, VA, USA

[3] Department of Chemical Engineering, Virginia Tech, Blacksburg, VA, USA

ABSTRACT

The understanding of enzymatic functions in plant signaling pathways is extremely challenging and convoluted by interactions between metabolic and signaling responses. One important enzyme in both signaling and metabolism, the sucrose non-fermenting related kinase 1 (SnRK1), has been shown to play a pivotal role in plant stress and energy signaling. Previous studies in *Arabidopsis thaliana* showed evidence that SnRK1 regulates global transcriptional responses to hypoxic stress and carbon starvation. SnRK1 overexpression has also been shown to delay plant developmental transitions, increase biomass, and reduce starch accumulation. Further understanding of the SnRK1 regulatory pathway may enable accurate metabolic engineering of plant energy metabolism to improve biomass yield. To address the complexity of SnRK1 metabolic regulation, we modified and deployed a constraint-based **ge**nome-scale **m**etabolic model (GEM) of Arabidopsis to help guide our study of SnRK1 overexpression plants. Quantitative validation of model predictions with experimental data showed high accuracy in predicting growth and starch turnover in both wild-type (WT) and SnRK1 overexpressors during development. Results suggest changes in plant development and starch are independent responses to SnRK1 overexpression, which supports a previous speculation on simultaneous SnRK1 regulation of plant development and starch metabolism. This study demonstrates the utility of flux-based modeling in studying signaling pathways and it introduces a novel modeling framework to enable prediction of non-steady state metabolite accumulation in a diurnal cycle.

INTRODUCTION

In the absence of mobility, plants have evolved a diverse set of sensors to enable rapid response to unpredictable changes in their surrounding environment (Hasegawa et al., 2000; Sheen, 2014; Yamaguchi-Shinozaki and Shinozaki, 2006). Environmental stress can often compromise photosynthesis and respiration leading to a significant loss in cellular energy. Sucrose non-Fermenting Related Kinase 1 (SnRK1), a plant ortholog of mammalian AMP kinase, is encoded by a family of genes, and has been proposed to be a critical sensor of energy level in plants (Polge and Thomas, 2007). Previous studies in Arabidopsis showed evidence of SnRK1 global regulation of plant metabolism under stress and carbon starvation (Baena-González et al., 2007). Transcriptomic analysis showed ectopic expression of SnRK1 represses transcription of genes involved in anabolic pathways and activates those involved in catabolic pathways and autophagy. In addition, this subset of SnRK1-regulated genes was also regulated when plants are grown under low sugar, low CO_2, or an extended night period (Baena-González et al., 2007). The SnRK1-regulated metabolic pathways include the biosynthesis of cell wall, protein, lipid, and sugars (Baena-González et al., 2007).

In the Arabidopsis genome, there are three genes in the SnRK1 gene family, which includes the functional SnRK1.1 and SnRK1.2, and the unexpressed pseudogene SnRK1.3 (Baena-González et al., 2007; Hrabak et al., 2003). SnRK1.1 and SnRK1.2 double-knockout plants have greatly stunted growth and are infertile (Baena-González et al., 2007). Overexpression of SnRK1.1 can delay developmental transition and reduced rosette size in early development and increased rosette size in the post-flowering stage (Baena-González et al., 2007; Gazzarrini and Tsai, 2014; Williams et al., 2014). In contrast, overexpression of SnRK1.2 induces early flowering and increased rosette size in early development (Williams et al., 2014).

The gene expression pattern of SnRK1.2 in seedlings is spatially restricted to roots, and hydathodes and vascular tissue within leaves; whereas, SnRK1.1 gene expression is more uniform in all tissue types (Williams et al., 2014).

To overcome the inability to photosynthesize at night, most plants convert a portion of their photosynthate into starch during the day, and then remobilize starch in the night for growth and maintenance (Gibon et al., 2009). Gene expression analysis showed that SnRK1 is involved in activating starch degradation during energy deprivation (Baena-González et al., 2007). SnRK1 loss-of-function plants showed elevated end-of-day (EOD) and end-of-night (EON) starch levels, which implicates reduced starch turnover in the night (Baena-González et al., 2007). In addition, SnRK1.1 overexpression in Arabidopsis resulted in reduced starch accumulation when overexpression plants were supplemented with high glucose (Jossier et al., 2009). Interestingly, overexpression of SnRK1 in a sink tissue, potato tuber, increased tuber starch content by up to 30% (McKibbin et al., 2006). This disparity is likely due to differences in tissue type (i.e. source vs. sink tissue), nonetheless, it is generally agreed that SnRK1 has a significant role in regulating starch metabolism and plant development (Baena-González et al., 2007; Gazzarrini and Tsai, 2014; Jossier et al., 2009; Williams et al., 2014).

It has been shown that starch can impact plant developmental transitions (Matsoukas et al., 2013; Yu et al., 2000). Specifically, it was found that the total starch level does not regulate developmental transition; but rather, the reduction of starch turnover rate at night can significantly delay juvenile-to-adult phase transition during the vegetative stage, which can lead to delayed vegetative-to-reproductive stage transition (Matsoukas et al., 2013). Previous data seem to indicate that the developmental delay observed in SnRK1 overexpression plants is not a

consequence of altered starch metabolism and that SnRK1.1 may regulate starch and developmental transition simultaneously, however, concrete results have yet to be presented.

Many challenges faced in sugar signaling research come from the difficulties in dissecting the convoluted mixture of metabolic and regulatory events that are further complicated on a temporal scale (Sheen, 2014). In light of recent advances in bioinformatics and computational biology, multiple plant researchers have sought alternative analysis and computational methods, including the utilization of constraint-based **ge**nome-scale metabolic **m**odels (GEMs) to address metabolic complexity (Cheung et al., 2013; Grafahrend-Belau et al., 2013; Poolman et al., 2013; Töpfer et al., 2013). A GEM is a network of biochemical reactions defined solely by stoichiometry; kinetics and regulatory features are not required but can be added where information is known (Poolman et al., 2009; Schilling et al., 1999). Flux balance analysis (FBA) is commonly used with a GEM to calculate metabolic flux distributions given a growth objective and a pseudo-steady state. The pseudo-steady state assumption holds when the rate of metabolic reaction is significantly higher than the rate of organism development, such as in filling seeds or developing fruit (Allen et al., 2009; Caspeta et al., 2012; Colombié et al., 2015). Under the pseudo-steady state assumption, flux of its formation is equal to the flux of that metabolite's consumption; thus, metabolic network flux distribution (within the bounds of constraints) is calculated traditionally with linear programming using Equation 1, where S_{ij} is the stoichiometric coefficient of metabolite i in reaction j, v_j is the calculated flux of reaction j, and X_i is the concentration of metabolite (Caspeta et al., 2012; Förster et al., 2003; Park et al., 2009).

$$\frac{dX_i}{dt} = S_{ij} \cdot v_j = 0, v_{j,min} \leq v_j \leq v_{j,max} \qquad \text{(Equation 1)}$$

The utility of flux-based modeling in studying plant metabolism has been a subject of great interest (Collakova et al., 2012). To show that it can be used for plant studies, a comprehensive analysis that compared experimentally measured fluxes with predicted fluxes of central carbon metabolism showed that flux-based modeling with GEMs can accurately predict steady-state flux in Arabidopsis cells *in vitro* (Williams et al., 2010). Although utilization of GEMs to accurately predict fluxes of plants *in vivo* is challenged by the lack of a long-term steady-state condition, a previous study on the effect of light intensity on rice metabolism revealed which metabolic interactions can be described solely with mass-balancing (Poolman et al., 2013). This study showed that flux-based modeling can be used to determine whether a metabolic change involves regulatory control, based on the accuracy of the prediction. Another study on the metabolic changes in developing tomato fruit demonstrated that GEMs and pseudo-steady state flux-based modeling can also be used to predict metabolic flux "snap-shots" of developmental stages (Colombié et al., 2015).

The objective of this study is to determine if flux-based modeling can be used to guide experimental research to determine the metabolic roles of enzymes in plant signaling pathways. As the proof-of-concept, we investigated whether starch turnover is regulated differently in WT and plants ectopically expressing a SnRK1.1 gene fused to an HA epitope tag (SnRK1.1:HA plant). Because overexpression of SnRK1.1:HA has distinct phenotypic consequences at different developmental stages, we examined starch turnover at two different stages of plant development. To do so, we modified an existing Arabidopsis GEM (AraGEM) (de Oliveira Dal'Molin et al., 2010) using experimentally measured biomass data and a novel modeling framework to predict quantitative changes of biomass and starch over a 24-hour diurnal growth cycle. Predicted growth and starch turnover rates were validated with experimental results, and

showed agreement with computational predictions. Results from this model-guided analysis of starch metabolism suggest that the delayed developmental transition in SnRK1.1 overexpression plants may not be associated with starch. Our analysis also supports previous speculation that SnRK1.1 plays a role in regulating starch and plant development simultaneously.

MATERIALS AND METHODS

Plant growth conditions

Arabidopsis thaliana ecotype Landsberg erecta (Ler-0) plants were used for all experiments. Plants were grown in a controlled growth chamber at 22°C and 55% relative humidity under 16 hours of light, provided with fluorescent lamps (140 μE). Soil-grown plants were maintained on Sunshine Mix #1 and watered with Miracle-Gro Liquid Houseplant Food (8-7-6: 8% total nitrogen, 7% available phosphate, P2O5, 6% soluble potash, K2O, 0.1% Iron, Fe; Scotts Miracle-Gro Products, Inc.).

Tissue collection

All pre-flowering plant tissues for protein, biomass, cellulose, and metabolite analyses were harvested from 14 day-old plants. All tissues of other ages were green mature leaves harvested from rosette. For starch analyses, liquid nitrogen was poured directly onto rosette and samples were harvested while frozen.

All samples except EOD and EON samples were harvested at 1 PM. EOD samples were harvested 30 minutes before the dark cycle began. EON samples were harvested 30 minutes prior to the light cycle.

Quantification of cell wall

Rosette leaves and stem tissues were harvested and dried via lyophilization and then pulverized with 3 mm steel beads. Samples were washed using 70% ethanol and centrifuged for

10 minutes at top speed on a table-top microcentrifuge. For cellulose, the pellet was resuspended in 1 mL of Updegraff reagent (acetic acid:nitric acid:water, 8:1:2 [v/v]) before boiling for 30 minutes on a heating block at 98ºC. The remaining crystalline cellulose was pelleted by centrifugation and dissolved in 1 mL of 67% (v/v) sulfuric acid. Crystalline cellulose amount was quantified colorimetrically at 620 nm in a spectrophotometer using the anthrone reagent (Updegraff, 1969).

Lignin and glycosyl levels were analyzed by the Complex Carbohydrate Research Center (CCRC) at University of Georgia. For cell wall glycosyl analysis, the ethanol insoluble pellet was de-starched with amylase and amyloglucosidase as described in starch quantification. Remaining pellet was sent to CCRC and analyzed using previously described method (Santander et al., 2013). For lignin analysis, dried plant tissue was sent directly to CCRC and pyrolyzed using single-shot pyrolysis at 500°C to produce volatile compounds, which were analyzed with a beam mass spectrometer (Extrel Core Mass Spectrometers).

Quantification of biomass and relative leaf expansion rate

Dry weight was measured after 2 days of lyophilization at -50ºC. Leaf surface area (A) was measured by first flattening the leaves between two glass slides and photographing the entire slide with the leaf surface positioned parallel to the camera lens. The photograph was processed in ImageJ to select only the leaf or the glass slide. The pixel count under the leaf selection was normalized to the pixel count under the glass slide selection. A was calculated by multiplying the normalized pixel count by the known surface area of the glass slide. Relative leaf expansion rate (RER) was calculated as the ratio of the change of A from t_0 to t_1 to the A at t_0 (Tardieu et al., 1999). Mathematical formulation is shown in Equation 2 below.

$$RER = \frac{A_1 - A_0}{A_0} \qquad \text{(Equation 2)}$$

Day-RER was calculated from *A* at EON to EOD of the same day, and night-RER was calculated from *A* at EOD to EON of the next day.

Quantification of amino acids and total lipid

A previously published extraction and quantification method for amino acids and lipids was used with the following modifications (Collakova et al., 2013). In brief, amino acids and protein were extracted by homogenizing lyophilized plant powder with 200 μL of chloroform and 200 μL of 10 mM HCl. Norvaline (10 μL of 5 mM) was used as the internal standard. Non-polar extract containing fatty acids and lipid was transferred, dried, and weighted for total lipid content. Protein in the polar extract was hydrolyzed by overnight incubation in vapor of 6N HCl at 110°C. Hydrolyzed extracts and amino acid standards of known concentrations were derivatized with Waters AccQ-Tag^{TM} Ultra Kit and analyzed on an H-class Acquity UPLC-FLD equipped with a 10-cm Waters AccQ-TagTM Ultra C18 (1.7 μm x 2.1 mm) column (Waters, Milford, MA, USA) using Waters 10.2-minute method for free amino acid analysis (Collakova et al., 2013).

Gas-exchange measurement

Net CO_2 assimilation rate was measured on pre-flowering and post-flowering plants grown under previously described conditions using a LI-6400 infrared gas analyzer (Li-Cor). Whole pre-flowering plant was placed in a Li-Cor 6400-17 whole plant Arabidopsis chamber. A single intact post-flowering mature leaf was placed in the original chamber (2 cm × 3 cm). After acclimation to 140 μE and CO_2 concentration of 400 μmol/L for 10 minutes, the rates of CO_2 assimilation were measured.

Starch Quantification

Starch was measured in tissues harvested at mid-day, EON, and EON. Analysis was performed as previously described with slight modification (Smith and Zeeman, 2006). Briefly, 1 – 4 mg of lyophilized plant sample was pulverized via bead-beating. Soluble sugars were removed using 80% ethanol. Starch was hydrolyzed by incubating with 6 U of α-amyloglucosidase and 1 U of amylase. Glucose was quantified with HPLC. The transitory starch pool was calculated as the difference between starch levels at EOD and EON.

Construction of SnRK1.1:HA genome-scale models

Four GEMs were built using AraGEM as base model for WT and SnRK1.1:HA plants at pre-flowering and post-flowering stages. Experimentally measured concentrations of biomass compounds except starch in μmol·mgDW^{-1} were used to construct the biomass equation (reaction 47) of each model. Starch was structured as a separate pool to allow for changes in concentration independent from the other biomass compounds. Because the glycosyl profile of hemicellulose and pectin compositions was a relative quantitation, concentrations of glycosyls were determined using literature data. Hemicellulose and pectin makes up 66% of the cell wall (Zablackis et al., 1995). We found the cell wall of WT Arabidopsis is 63% of dry weight, thus the glycosyl concentrations in each model could then be estimated by multiplying the relative glycosyl levels by 416 μg (total glycosyl)·mg DW^{-1}.

Mathematical formulation of the modified biomass equation and additional biomass constraints are summarized as follows:

$$For\ biomass\ compound\ X: d(\dot{X}) = (S \cdot v) - (x_1' B_1' - x_0' B_0') = 0$$

$$B_0 = B_0' + my_0$$

$$B_1 = B_1' + my_1$$

Where $\dot{X}$ is μmol of compound X per plant changed from t_0 to t_1 (i.e. 1 PM to 1:30 PM). x' is the adjusted concentration of compound X, which is normalized to a starch-less plant dry weight, B'. The stoichiometric relationship of compound X with all other reactions is described by the stoichiometric matrix S, and flux vector v carries a unit of μmol·plant^{-1}·0.5 h^{-1} instead of the conventional μmol·mgDW^{-1}·h^{-1}. B_0 is an input constraint, such as the dry weight measured at 1 PM of a 14 day-old plant. B_1 is the biomass to predict, such as the dry weight at 1:30 PM of the same age plant. Molar quantity of starch is denoted as y, and the molecular weight of starch is m. The known starch level in μmol·plant^{-1} at 1 PM of day 14 is y_0, and the level to predict at 1:30 PM is y_1. Change of starch (dy) is constrained as follow.

$$dy = y_1 - y_0$$

$$change\ of\ starch\ in\ \mu mol\ per\ plant = \left(\sum s \cdot v\right) - dy = 0$$

The stoichiometric matrix of the AraGEM model was modified to include the above equations. Models and MATLAB implementations are included in the Supplemental Information (Appendix C).

Predicting the transitory starch pool

As illustrated in Figure 4-1, the transitory starch pool ($dS_{EOD \rightarrow EON}$) is calculated as the maximum amount of starch that could be synthesized in the day and utilized in the night while returning to the initial level after 24 hours. Thus, the objective of the model simulation is maximizing $dS_{EOD \rightarrow EON}$ while minimizing the difference between initial and final starch levels of the simulation ($dS_{t0 \rightarrow t24}$). To simulate growth and the change of starch level over a 24-hour period starting from 1 PM, models were subjected to three simulation stages: (A) 9 hours of photosynthesis (1 PM to 10 PM), (B) 8 hours of respiration (10 PM to 6 AM), and (C) 7 hours of photosynthesis (6 AM to 1 PM). The rates of starch accumulation in stage A (r_A) and C (r_C) were

assumed to be similar *in vivo*, thus the difference between r_A and r_C is also minimized in the objective of the model simulation. Net CO_2 assimilation rates in $\mu mol \cdot cm^{-2} \cdot h^{-1}$ multiplied by the rosette surface area and 0.5-hour time-steps were used to constrain the CO_2 uptake rates of the models during photosynthesis. Experimentally measured relative leaf expansion rate was used to estimate total CO_2 assimilation over time during the photosynthetic stages. During the respiration stage, CO_2 exchange was constrained to export and starch was constrained to import as a carbon source. Metabolic flux distribution over each 0.5-hour time-step was predicted using FBA with the objective of maximizing the non-starch biomass (B_1') (i.e. growth).

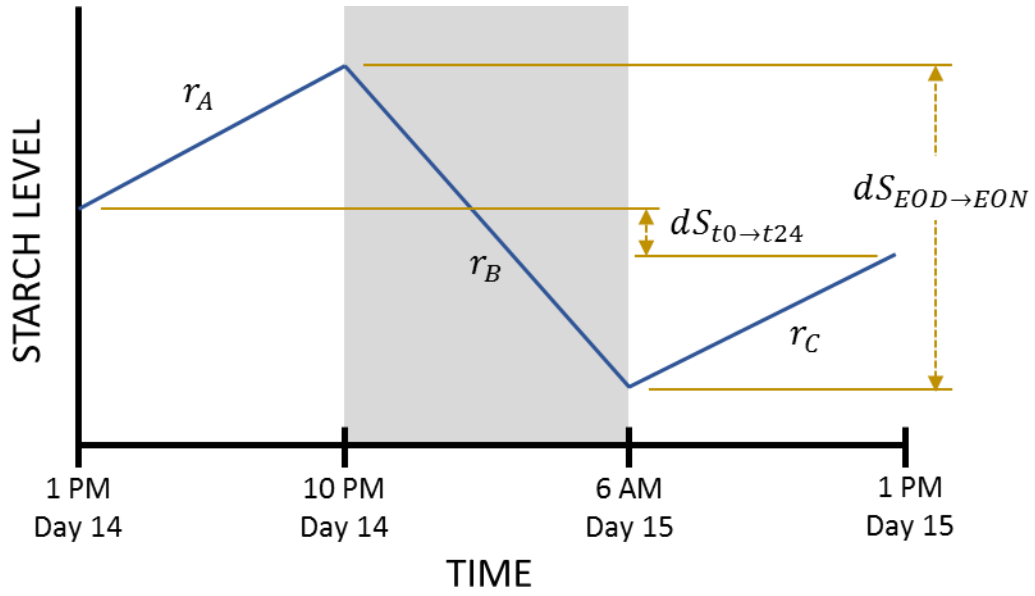

Figure 4-1. Model simulation of starch change in a 14 day-old plant over a 24-hour period. Simulation began with a 9-hour light period starting at 1 PM when plant dry weight and the biomass components were experimentally measured. It was followed by an 8-hour dark period starting at 10 PM, then ended with a 7-hour light period of the next day starting at 6 AM. The objective of the simulation is to (i) maximize the transitory starch pool ($dS_{EOD\rightarrow EON}$), (ii) minimize the difference between initial and final starch levels ($dS_{t0\rightarrow t24}$), and (iii) minimize the difference between starch accumulation rates of day 14 (r_A) and day 15 (r_C). The starch turnover rate (r_B) is also predicted.

Analysis of predicted metabolic flux distribution

To compare the predicted metabolic fluxes of WT and SnRK1.1:HA models, the average flux of each simulation stage was calculated and compared. Because there are two light simulation stages (A and C), the predicted fluxes in both light stages were averaged together. The number of reactions in a pathway that were increased or decreased in SnRK1.1:HA models compared to WT models were recorded. This frequency was used to rank pathways with the most flux changes to the least flux changes. This comparison was performed to identify pathways with highest frequency of flux increase or decrease in SnRK1.1:HA model as compared to WT model during the day and the night simulations at the pre-flowering and the post-flowering stages. Four single-developmental stage comparisons were performed for (1) increased in the pre-flowering stage, (2) decreased in the pre-flowering stage, (3) increased in the post-flowering stage, and (4) decreased in the post-flowering stage. Two cross-developmental stage comparisons were performed to identify pathways with highest frequency of flux increase or decrease from the pre-flowering stage to the post-flowering stage. In addition, two cross-developmental stage comparisons were performed to identify pathways with the highest frequency of flux increase in the pre-flowering stage and decreased in the post-flowering stage or decreased in the pre-flowering stage and increased in the post-flowering stage.

RESULTS

SnRK1.1:HA plants differ in biomass and developmental age

Previous studies on SnRK1.1:HA plants showed reduced rosette size in the pre-flowering stage, a delayed transition to flowering, and increased rosette size in the post-flowering stage (Williams et al., 2014). Representative differences in size and developmental transition are shown in Figure 4-2A. Analysis of 14 day-old rosette dry weight supports our previous observation that SnRK1.1:HA plants show reduced leaf biomass compared to WT (Figure 4-2B). To compare differences in post-flowering plants, we used well-established senescence gene markers, SAG12 and SAG21 to identify ages of WT and SnRK1.1:HA that are developmentally matched in terms of senescence (Watanabe et al., 2013). We selected a senescence stage in which plants exhibited mostly green rosette leaves and fully developed stems, which is accompanied by low SAG12 expression and robust expression of SAG21 (Watanabe et al., 2013). As shown in Figure 4-2D, a 42 day-old SnRK1.1:HA plant has the same SAG 12 and 21 profile as a 35 day-old WT plant. Rosette dry weights of post-flowering SnRK1.1:HA plants were significantly greater than that of WT, which indicates an increase in vegetative biomass in SnRK1.1:HA plants (Figure 4-2C).

Analysis of the relative surface expansion rate (RER) of 14 day-old leaves showed that day-RER was 89% in WT and 83% in SnRK1.1:HA, however, by 21 days day-RER was reduced to 59% in WT and 54% in SnRK1.1:HA (Figure 4-2E). Reduction in RER ratio was due to decrease in day-RER and concurrent increase in night-RER. This analysis also revealed that 21 day-old SnRK1.1:HA plants began to show greater rosette growth than WT.

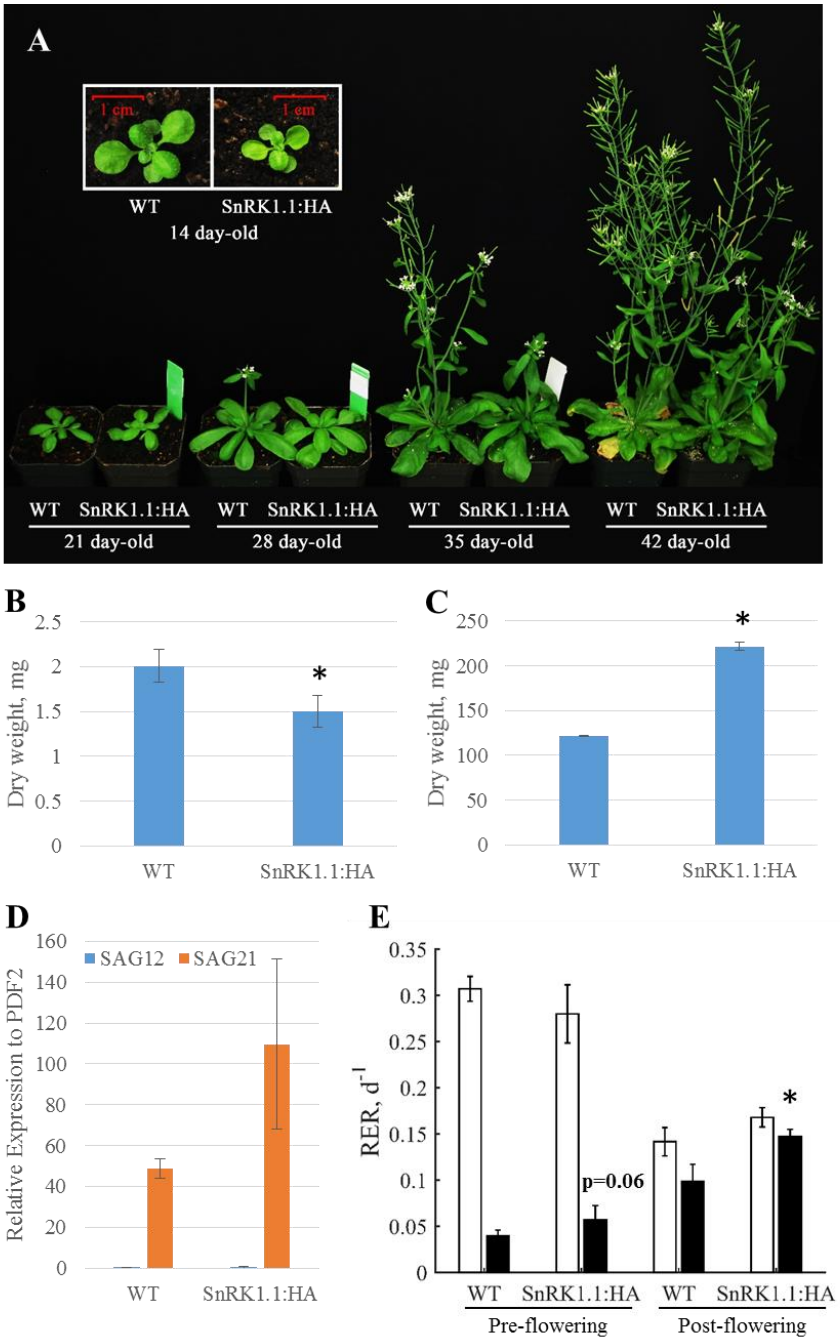

Figure 4-2. Differences in growth and development of WT and SnRK1.1:HA plants. (A) Phenotypic appearances of WT and SnRK1.1:HA plants. Dry weight of WT and SnRK1.1:HA in pre-flowering (B) and post-flowering (C) stages were measured for whole rosettes (n = 4). Senescence stage (D) of mature leaves of 35 day-old WT and 42 day-old SnRK1.1:HA indicated by SAG12 (blue) and SAG21 (red) gene expression markers (n = 2). The RER of WT and SnRK1.1:HA (E) in the day (white) and the night (black) in pre-flowering and post-flowering stages (n = 5 for pre-flowering, n = 8 for post-flowering). Values are shown as mean±SE, and asterisk indicates $p < 0.05$.

Biomass composition and photosynthetic rate are significantly altered in SnRK1.1:HA plants

To construct genome-scale models, specific to our WT and SnRK1.1:HA plants, the biomass equations of the models were parameterized to experimental measurements. We focused on the primary components of plant biomass, including cell wall, amino acids/protein, lipid, and starch. Cell wall component analysis includes absolute quantification of cellulose and lignin and relative quantification of hemicellulose. As shown in Figure 4-3A, cellulose was significantly reduced by 42% and 41% in SnRK1.1:HA plants in pre-flowering and post-flowering stages, respectively. Analysis of lignin shows that levels of syringyl and guaiacyl were decreased by 20% and 23%, respectively, in SnRK1.1:HA in the post-flowering stage (Figure 4-3B). Analysis of the relative levels of glycosyls in hemicellulose and pectin showed no significant differences between WT and SnRK1.1:HA at both developmental stages (Figure 4-3C). Our results show that SnRK1.1:HA overexpression alters cellulose and lignin content.

Analysis of amino acid, which contains free amino acids and hydrolyzed protein, showed similar relative levels in both genotypes at both developmental stages, which suggests no change in amino acid ratios (Figure 4-4A). Absolute levels of all quantified amino acids (Figure 4-4B) in post-flowering SnRK1.1:HA plants were reduced by an average of 38.4% (± 5.8) compare to levels in post-flowering WT plants. Measurement of starch at mid-day showed a 16% increase in SnRK1.1:HA plants in the pre-flowering stage compared to WT (Figure 4-5A). In the post-flowering stage, starch concentration in SnRK1.1:HA was 47% lower than in WT. This indicates that overexpression of SnRK1.1 has a different impact on starch levels at different developmental stages. Total lipid was quantified by measuring the dry weight of the non-polar extract, which could contain trace amounts of chlorophyll. As shown in Figure 4-5B, lipid levels of WT and SnRK.1:HA were not different in the pre-flowering stage; however, the post-

flowering SnRK1.1:HA lipid level was significantly lower as compared to post-flowering WT. To model plant growth during photosynthesis, experimentally measured net CO_2 assimilation rate was used to constrain the maximum CO_2 uptake during the day period as described in Methods. As shown in Figure 4-5C, net CO_2 assimilation rate of SnRK1.1:HA was 26% lower than that of WT in the post-flowering stage.

These biomass compositional data were used to replace the biomass equation of the original AraGEM model. Out of the 32 metabolites we measured, absolute concentrations of 23 metabolites, including cellulose, 3 lignin monomers, total lipid, 17 amino acids, and starch were directly incorporated into the biomass equation. The concentrations of the 9 glycosyls derived from hemicellulose and pectin were estimated by normalizing their relative levels to 41.6% (g/g) of the dry weight before incorporating into the biomass equation. This generates four GEMs for WT and SnRK1.1:HA at pre-flowering and post-flowering developmental stages.

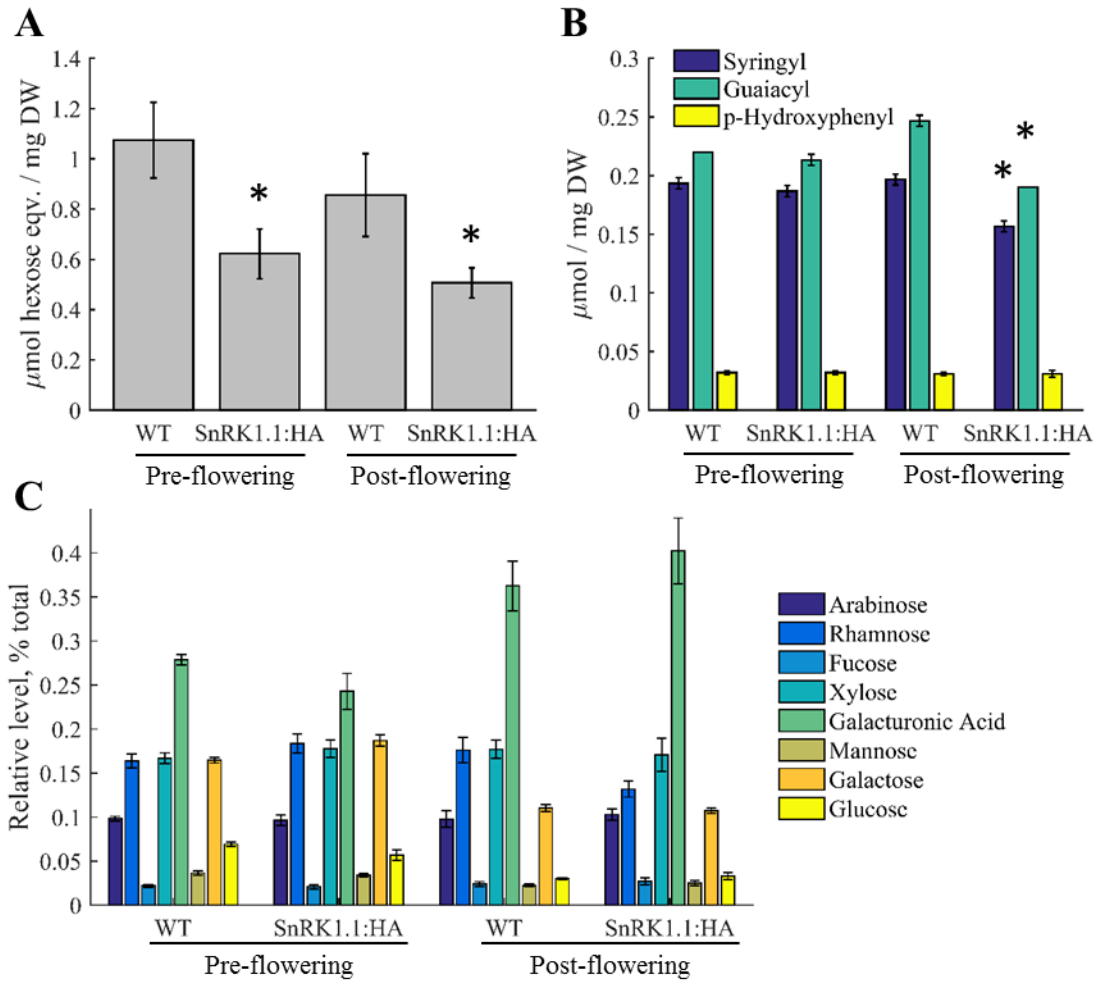

Figure 4-3. Cell wall compositions of WT and SnRK1.1:HA over plant development. Measurements were taken with whole plants in the pre-flowering stage and mature leaves in the post-flowering stage. Cellulose (A) and lignin compositions (B) in pre-flowering and post-flowering stages are shown in absolute quantities. Glycosyl composition of hemicellulose and pectin (C) at both stages are shown in relative quantity to the sum. Values are shown as mean±SE for n = 3, and asterisk indicates $p < 0.05$.

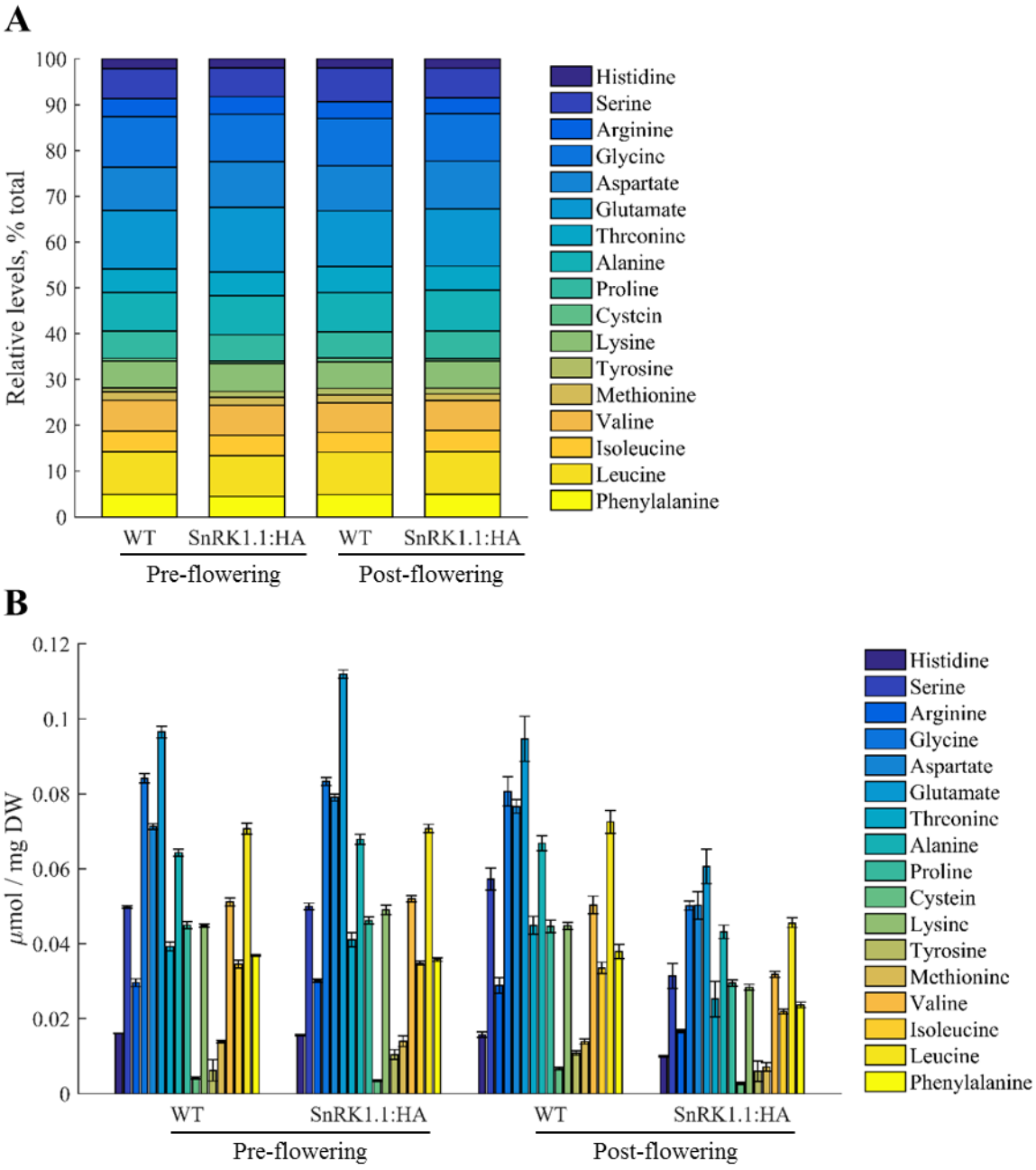

Figure 4-4. Amino acids profile of WT and SnRK1.1. Composition analysis shows relative levels (A) and absolute levels (B) of 17 amino acids from hydrolyzed protein and amino acid extracts of WT and SnRK1.1:HA in pre-flowering and post-flowering stages (n = 3).

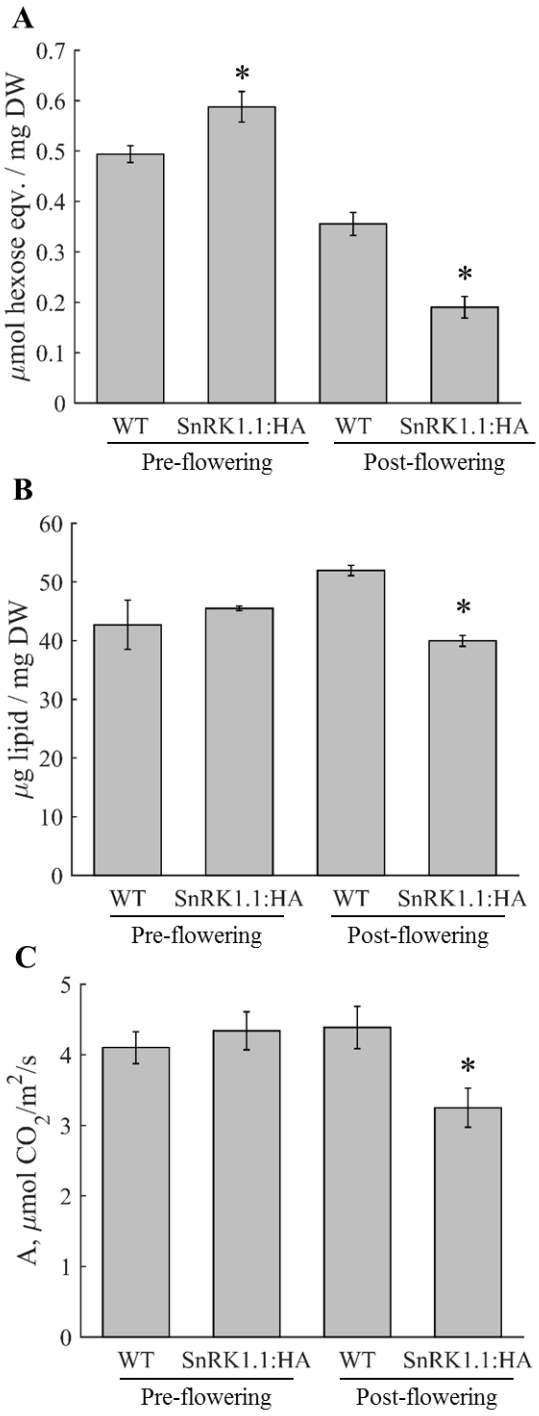

Figure 4-5. Changes in starch, lipid, and net CO_2 assimilation rate in SnRK1.1:HA plants. Total starch (A, total lipid (B), and net CO_2 assimilation rates (C) were quantified with whole plants in pre-flowering stage and mature leaves in post-flowering stage (n = 3 for starch and lipid, n = 4 for gas exchange). Values are shown as mean±SE and asterisk indicates $p < 0.05$.

Model simulation of growth and required transitory starch pool

As emphasized previously, starch is a critical carbon storage molecule that is accumulated in the day and consumed in the night for growth. To validate the WT and the SnRK1.1:HA models, we examined their accuracy in predicting the accumulation and depletion of starch at pre-flowering and post-flowering stages. Each model was parameterized and constrained with only the dry weight, day and night relative leaf expansion rates, biomass composition, and CO_2 assimilation rate described previously. Growth and starch concentration was simulated over a 24-hour period from the time of sample harvest. All samples were harvested 7 hours into a 16-hour-day regime, thus the first stage of the simulation was 9 hours of the remaining day, followed by 8 hours of night, then 7 hours of a light period over the next day. During the simulation, rates of starch accumulation and turnover were assumed to be linear as shown in previous experimental studies (Gibon et al., 2004; Graf et al., 2010).

As shown in Figure 4-6A, simulation of 14 day-old WT and SnRK1.1:HA plants predicts that within 24 hours WT will increase biomass from 2.01 to 2.99 mg DW (0.98 mg DW increase) and SnRK1.1:HA will increase from 1.51 to 2.39 mg DW (0.88 mg DW increase). Experimental values are very similar to these predicted values, with 15 day-old WT at 3.08 mg DW (1.07 mg DW increase) and SnRK1.1:HA at 2.35 mg DW (0.84 mg DW increase). Thus our models can predict the reduction in growth rate in SnRK1.1:HA overexpressors as seen previously (Williams et al., 2014). Model simulations also predicts that most of the increase in dry weight during the day was due to the increase of starch, and non-starch biomass was mostly accumulated in the night. SnRK1.1:HA plants were predicted to have 21% higher EOD and 56% higher EON starch levels as well as 15% greater transitory starch level as compared to WT in the pre-flowering stage (Figure 4-6B). The increase in transitory starch level indicates faster starch turnover at

night. To validate these predictions, EOD and EON starch levels were quantified and transitory starch was determined as the difference between EOD and EON levels. As shown in (Figure 4-6C), experimentally measured EOD and EON starch levels in pre-flowering SnRK1.1:HA plants were significantly higher than in pre-flowering WT by 22% and 50%, respectively. Experimentally quantified transitory starch level was also 12% higher in SnRK1.1:HA plants (Figure 4-6D). These results suggest that our model predictions on starch mobilization in pre-flowering plants are accurate.

As shown in Figure 4-7A, simulation of post-flowering leaves predicted that SnRK1.1:HA plants could increase biomass at 126.6 mg DW per day, which was much faster than WT plants that increased by 62.7 mg DW per day. This prediction is in agreement with previous studies, showing that post-flowering SnRK1.1:HA continue to increase rosette biomass when WT plant rosette growth has halted (Williams et al., 2014). Model simulation of starch concentration in the post-flowering stage predicted that WT plants accumulate and turnover 89% more starch as compared to SnRK1.1:HA plants (Figure 4-7B). Our models also predicted that starch could be depleted by the end of the night growth period in the post-flowering stage of both types of plants. Experimental measurements showed that WT accumulated 59% more starch by the end of the day and 62% more starch by the end of the night as compared to SnRK1.1:HA plants (Figure 4-7C). Starch was reduced to 94 μmol/mg DW in WT and 58 μmol/mg DW in SnRK1.1:HA plants at the end of the night, which was significantly lower than the EON starch levels in the pre-flowering stage. Experimental measurement of the transitory starch pool showed that WT level was 58% higher as compared to SnRK1.1:HA plants. Although, there are differences between the predicted values and the experimental values in post-flowering plants; nonetheless, the trends support model predictions strongly.

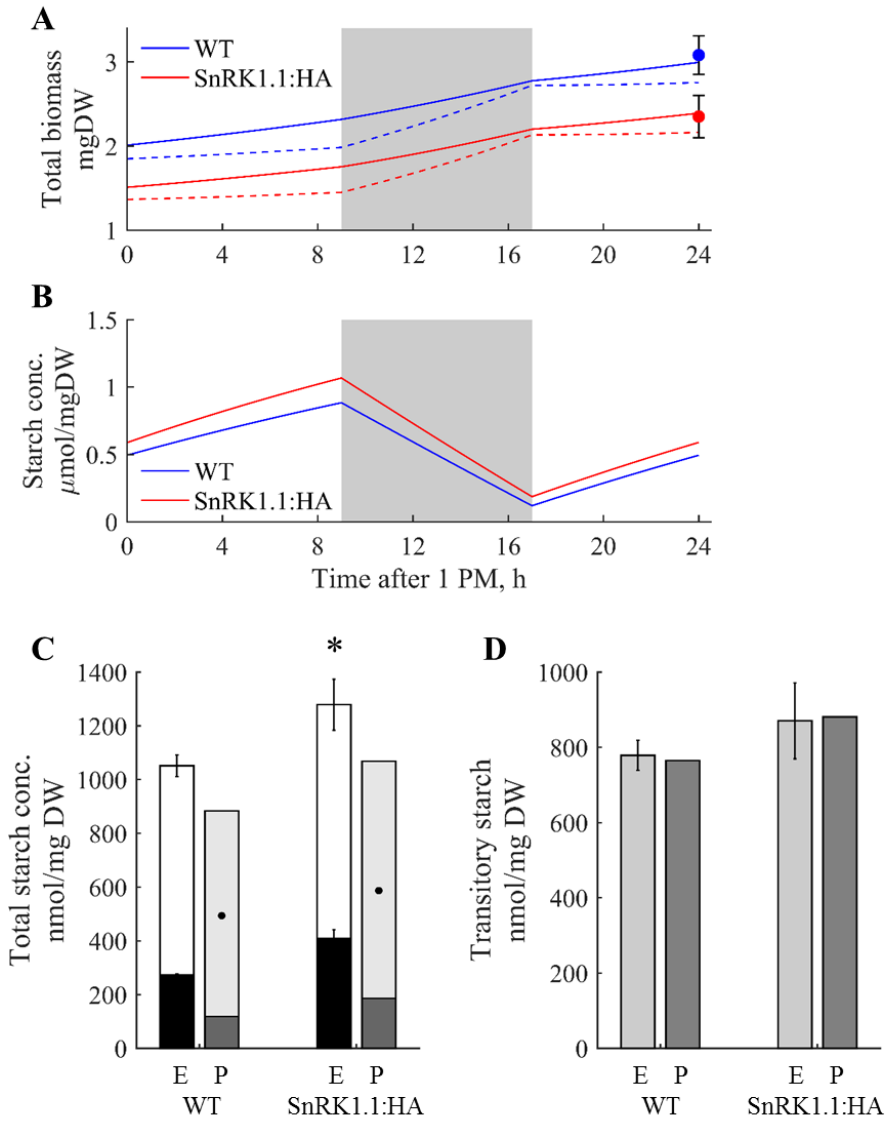

Figure 4-6. Predicted values for growth and starch turnover in pre-flowering plants compared to experimental values. (A) Predicted growth over the 24-hour diurnal cycle for WT (blue) and SnKR1.1:HA (red) starting at 14 days are shown as total predicted dry mass (solid lines) and predicted non-starch dry mass (dashed lines), and compared to experimentally measured total dry mass at 15 day-old (blue and red dots). (B) Predicted changes in starch concentration over the 24-hour diurnal cycle. Experimental values are indicated by E and predicted values are indicated by P in C and D. (C) Predicted EOD (light grey) and EON (dark grey) starch concentrations simulated from the initial starch levels (dots) compared to experimentally measured EOD (white) and EON (black) starch concentrations. (D) Predicted transitory starch pools (dark grey) compared to experimentally measured transitory starch pools (light grey). Experimental data are shown as mean ± SE (n = 4 for biomass, n = 3 for starch), and asterisk indicates $p < 0.05$.

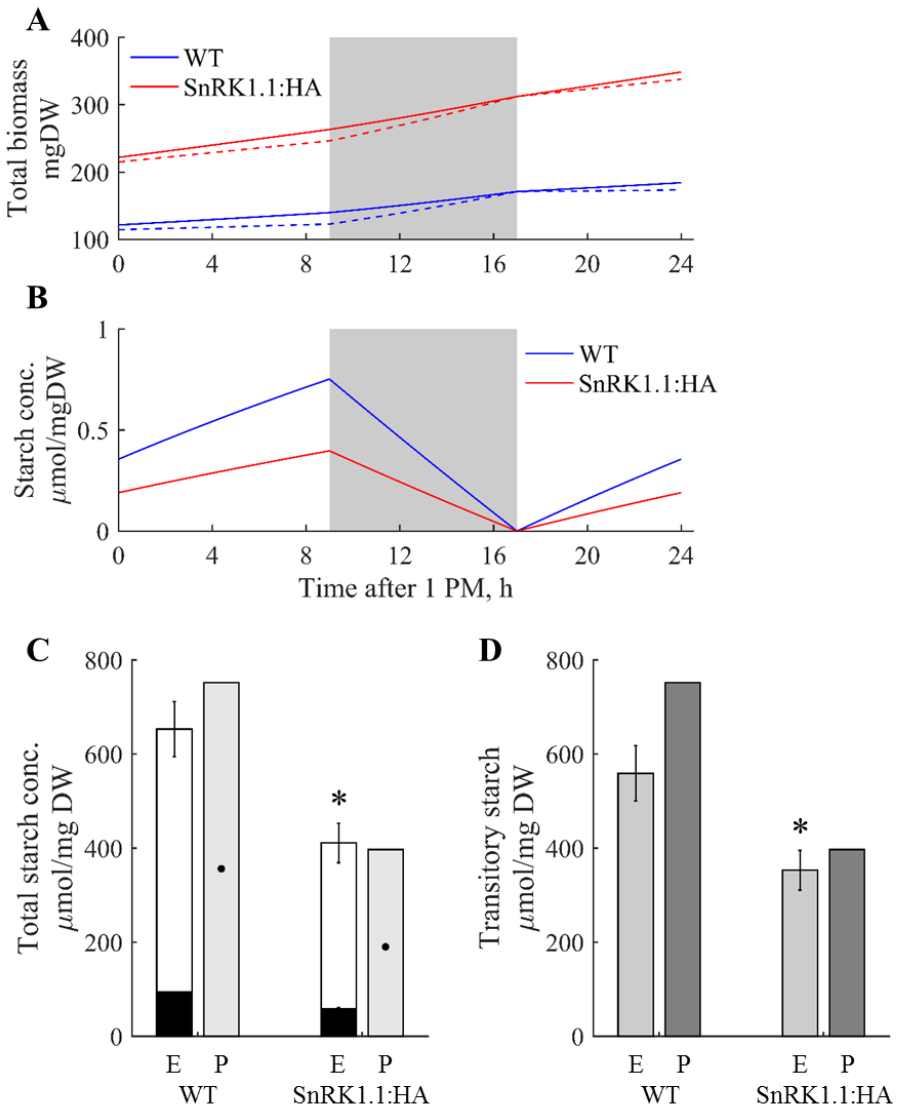

Figure 4-7. Predicted values for growth and starch turnover of post-flowering plants compared to experimental values. (A) Predicted growth over the 24-hour diurnal cycle for WT (blue) and SnKR1.1:HA (red) starting at 35 days for WT and 42 days for SnRK1.1:HA are shown as total predicted dry mass (solid lines) and predicted non-starch dry mass (dashed lines). (B) Predicted changes in starch concentration over the 24-hour diurnal cycle. Experimental values are indicated by E and predicted values are indicated by P in C and D. (C) Predicted EOD (light grey) and EON (dark grey) starch concentrations simulated from the initial starch levels (dots) compared to experimentally measured EOD (white) and EON (black) starch concentrations. (D) Predicted transitory starch pools (dark grey) compared to experimentally measured transitory starch pools (light grey). Experimental data are shown as mean ± SE (n = 4 for biomass, n = 3 for starch), and asterisk indicates $p < 0.05$.

Refinement of model predictions using experimental data

The differences between predicted and experimental starch values lead us to refine our models using the starch experimental data. To refine our models, we focused on two types of adjustments: 1) the initial starch level and 2) the simulation objective. As described previously, the mid-day starch level was used as the initial starch level of the simulation. Since starch turnover has been shown to be close to linear, the mean value between the experimentally observed EOD and EON starch levels should be used as the new initial starch level. Simulations of the pre-flowering stage were repeated under the same parameters except the adjustment of the initial starch values to 0.662 μmol/mg DW for WT and 0.843 μmol/mg DW for SnRK1.1:HA. As shown in Figure 4-8C, predicted total starch levels of the adjusted pre-flowering models closely match the experimental observations. Although predicted total starch levels were changed significantly, the predicted growth remained almost identical because the predicted transitory starch pool, the primary factor for non-starch biomass accumulation, was not altered significantly (Figure 4-8A, D).

For post-flowering simulations, the simulation objective was altered in addition to adjusting the initial starch value. The objective of maximizing starch accumulation and utilization generated more accurate predictions in pre-flowering models than in post-flowering models. To better capture the post-flowering metabolism, the simulation objective was changed to minimize the difference between predicted and observed transitory starch pool (Figure 4-8F). This enabled simulation of growth as well as EOD and EON starch levels more accurately. This adjustment increased the predicted non-starch growth in WT, as illustrated by the steeper blue dashed line in Figure 4-8B compared to Figure 4-7A. The predicted EOD and EON starch levels of the adjusted model matched the experimental data (Figure 4-8E).

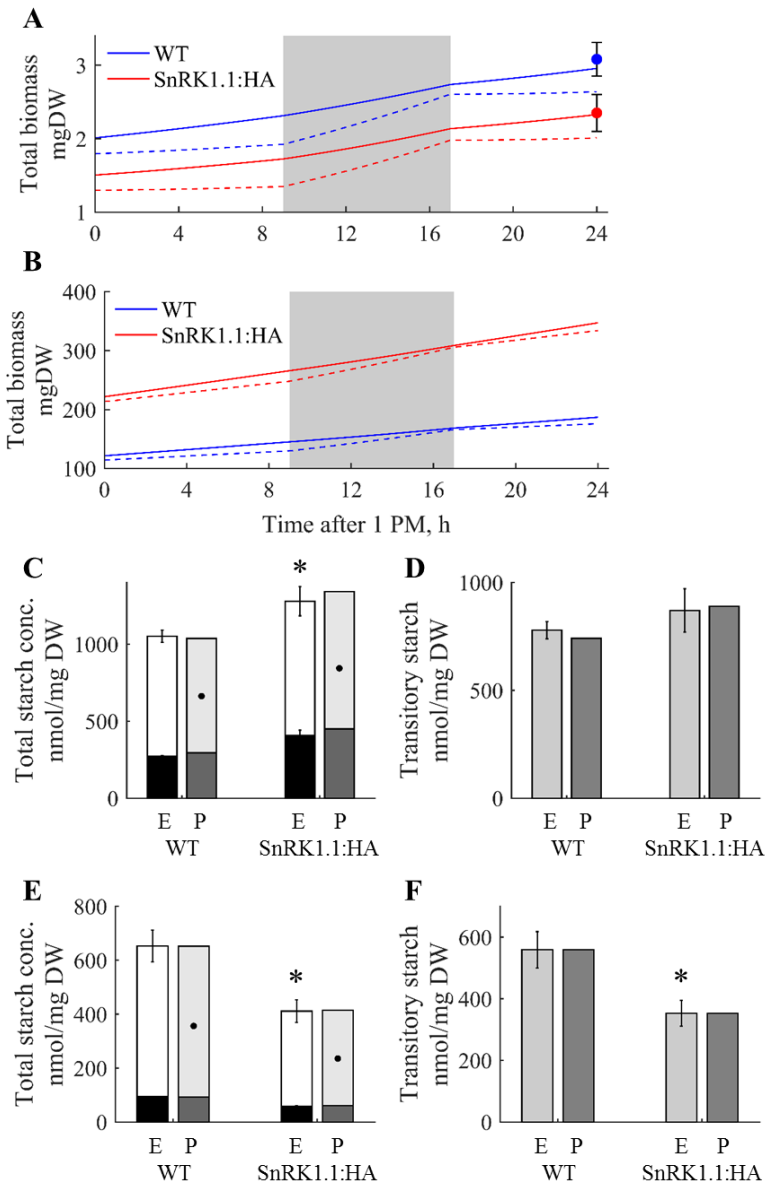

Figure 4-8. Predicted values for growth and starch levels after adjustments to experimental values. Predicted pre-flowering (A) and post-flowering (B) growth over the 24-hour diurnal cycle for WT (blue) and SnKR1.1:HA (red) are shown as total predicted dry mass (solid lines) and predicted non-starch dry mass (dashed lines). Predicted pre-flowering growth is compared to experimentally measured total dry mass at 15 day-old (blue and red dots). Experimental values are indicated by E and predicted values are indicated by P in C, D, E, and F. (C) Predicted pre-flowering stage EOD (light grey) and EON (dark grey) starch concentrations simulated with refined models from the initial starch levels (dots) compared to experimentally measured EOD (white) and EON (black) starch concentrations. (D) Predicted pre-flowering stage transitory starch pools (dark grey) simulated with refined models compared to experimentally measured transitory starch pools (light grey). Same for post-flowering stage (E and F). Experimental data are the same as in Figure 4-6 and Figure 4-7 shown as mean ± SE (n = 4 for biomass, n = 3 for starch), and asterisk indicates $p < 0.05$.

Starch accumulation and mobilization over plant development

After refining the models to given experimental data, we examined how the models can be used to predict starch levels across multiple stages of development. A linear assumption between pre-flowering and post-flowering stages was used to interpolate (1) the biomass composition, (2) the initial starch level of each stage, and (3) the starch accumulation and turnover rates predicted with the refined model. These parameters were interpolated for 24-hour simulation of 21 and 28 day-old WT, and 21, 28, and 35 day-old SnRK1.1:HA. As shown in Figure 4-9A, the changes of predicted total starch level at EOD and EON are close to but not completely linear. This analysis predicted that the increase in EOD starch level in SnRK1.1:HA diminished when both plants were around 35 days-old. To validate this prediction, EOD and EON starch levels were measured at the same ages as for the model simulation. As shown in Figure 4-9B, there are no statistically significant differences between WT and SnRK1.1:HA EOD starch levels after 14 day-old; however, the trend continued to show higher level of EOD and EON starch until the plants were 35 day-old. The data suggests that SnRK1.1:HA EOD starch becomes lower than WT EOD starch between 28 and 35 day-old. In addition, there are no significant differences in the transitory starch levels when WT and SnRK1.1:HA plants are compared by chronological age.

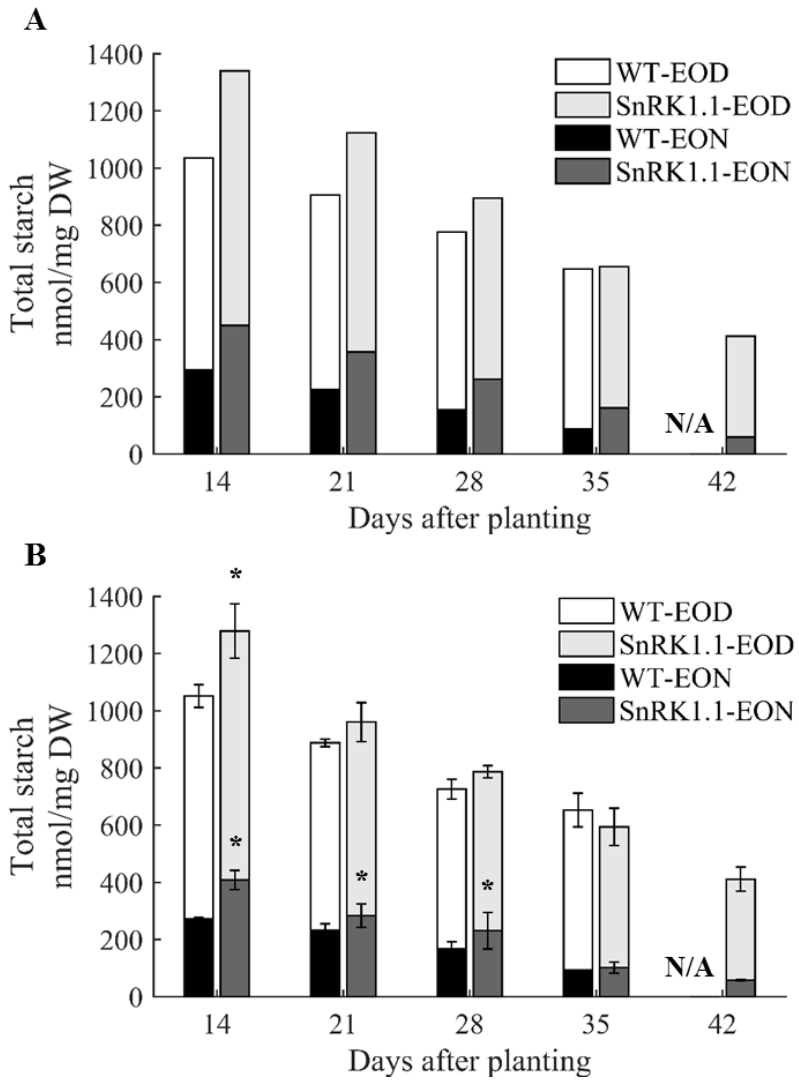

Figure 4-9. Predicted and experimental EOD and EON starch levels over plant development. (A) Predicted EOD (light grey) and EON (dark grey) starch levels for plant ages between the designated pre-flowering and post-flowering stages. (B) Experimentally measured EOD (white) and EON (black) starch levels for plant ages between the designated pre-flowering and post-flowering stages (n = 3). Experimental data of pre-flowering and post-flowering stages are the same as in Figure 4-6 and Figure 4-7. Data are shown as mean ± SE, and asterisk indicates $p < 0.05$.

Predicted metabolic differences between WT and SnRK1.1:HA plants

The adjusted models were used to predict metabolic changes induced by overexpression of SnRK1.1:HA at pre-flowering and post-flowering stages. The analysis included only enzyme catalyzing reactions filtered through criteria described in the Methods, leaving a total of 211 reactions for analysis. These reactions were associated with 52 of 137 pathways in the AraGEM model. Analyses were performed by comparing the reaction fluxes of SnRK1.1:HA to WT GEMs predicted with FBA. Metabolic pathways were ranked by the number of reactions in each pathway that had altered predicted fluxes. A pathway could be in both the increased flux list and the decreased flux list (i.e. starch and sucrose metabolism in Table 4-1) if some of its reactions were predicted to have increased fluxes and a similar number of its reactions were predicted to have decreased fluxes.

As shown in Table 4-1, more reactions in sugar metabolism and amino-acid metabolism were predicted to altered by SnRK1.1:HA overexpression in pre-flowering plants. Surprisingly, fatty-acid biosynthesis was predicted to increase in pre-flowering SnRK1.1:HA plants even though experimental measurements showed no significant differences between total lipid levels in the pre-flowering stage plants. Further investigation revealed that 31 of the 72 reactions associated with fatty-acid biosynthesis in AraGEM were predicted to have higher fluxes in pre-flowering SnRK1.1:HA plants as compared to WT plants at night. Another group of reactions predicted to carry higher fluxes at night are associated with hemicellulose biosynthesis. In contrast, reactions in cellulose biosynthesis were predicted to carry lower fluxes in pre-flowering SnRK1.1:HA plants than in pre-flowering WT plants.

In the post-flowering stage, reactions involved in lignin and hemicellulose biosynthesis were predicted to have higher fluxes in SnRK1.1:HA plants during the day as compared to WT

plants (Table 4-2). Interestingly, no reaction fluxes were predicted to be significantly increased by SnRK1.1:HA overexpression during the night. Of the 211 reactions examined, 106 reactions were predicted to have lower fluxes in SnRK1.1:HA plants during the day and 192 reactions were predicted to have lower fluxes during the night. As expected, cellulose synthase was predicted to be lower in both day and night cycles. In contrast to the predicted increase in fatty acid biosynthesis in pre-flowering night cycle, post-flowering SnRK1.1:HA was predicted to have less fatty acid biosynthesis at night compared to WT.

The predicted flux changes in fatty acid biosynthesis during pre-flowering stage versus post-flowering stage provide an avenue for investigating the consistency of predicted metabolic patterns over plant development. As shown in Table 4-3, no reaction in SnRK1.1:HA was predicted to have higher flux than WT throughout development. Only 6 reactions were predicted to have lower fluxes than WT across development, which were primarily reactions associated with cellulose biosynthesis. Similarly, reactions that were predicted to have alternating patterns were examined. As shown in Table 4-4, more pathways were predicted to switch from having higher fluxes in pre-flowering stage to having lower fluxes in post-flowering stage. A total of 197 reactions were predicted to exhibit this pattern, of which, 13 reactions were in the day cycle and 185 reactions were in the night cycle. Fatty acid biosynthesis is on the top of the night cycle list, which also includes hemicellulose biosynthesis, amino acid biosynthesis, and lignin biosynthesis. Starch biosynthesis is on the day cycle list of reactions that have higher fluxes than WT in pre-flowering stage and lower fluxes in post-flowering stage. The only reactions that were predicted to exhibit the opposite pattern of lower fluxes in pre-flowering stage and higher fluxes in post-flowering stage were 2 hemicellulose biosynthesis reactions. However, there were more

hemicellulose associated reactions that were predicted to have higher fluxes in pre-flowering stage and lower fluxes in post-flowering stage.

Table 4-1. Predicted metabolic changes by SnRK1.1:HA overexpression in the pre-flowering developmental stage.

Pathways with increased flux in SnRK1.1:HA		**Pathways with decreased flux in SnRK1.1:HA**	
In the day	**In the night**	**In the day**	**In the night**
Pentose phosphate pathway	Fatty acid biosynthesis	Cell wall biosynthesis	Starch and sucrose metabolism
Carbon fixation	Cell wall biosynthesis	Nucleotide sugar metabolism	Cell wall biosynthesis
Starch and sucrose metabolism	Hemicellulose biosynthesis	Starch and sucrose metabolism	Galactose metabolism
Glycolysis / Gluconeogenesis	Aminoacyl-tRNA biosynthesis	Galactose metabolism	Aminoacyl-tRNA biosynthesis
ATP and NADPH generated from light (overall reaction in chloroplast)	Valine, leucine and isoleucine biosynthesis	Hemicellulose biosynthesis	Cysteine metabolism
Fructose and mannose metabolism	Citrate cycle (TCA cycle)	Aminoacyl-tRNA biosynthesis	Glycine, serine and threonine metabolism
Galactose metabolism	Glycine, serine and threonine metabolism	Cysteine metabolism	Glycolysis / Gluconeogenesis
Pentose and glucuronate interconversions	Histidine metabolism	Glycine, serine and threonine metabolism	Methionine metabolism
Streptomycin biosynthesis	Alanine and aspartate metabolism	Glycolysis / Gluconeogenesis	Nucleotide sugars metabolism
	Nitrogen metabolism	Methionine metabolism	Oxidative phosphorylation (overall in mitochondria)

Table 4-2. Predicted metabolic changes by SnRK1.1:HA overexpression in the post-flowering developmental stage.

Pathways with increased flux in SnRK1.1:HA		**Pathways with decreased flux in SnRK1.1:HA**	
In the day	**In the night**	**In the day**	**In the night**
Cell wall biosynthesis	N/A	Aminoacyl-tRNA biosynthesis	Fatty acid biosynthesis
Hemicellulose biosynthesis		Valine, leucine and isoleucine biosynthesis	Cell wall biosynthesis
Fructose and mannose metabolism		Glycine, serine and threonine metabolism	Aminoacyl-tRNA biosynthesis
Nucleotide sugar metabolism		Histidine metabolism	Hemicellulose biosynthesis
Carbon fixation		Alanine and aspartate metabolism	Valine, leucine and isoleucine biosynthesis
Citrate cycle (TCA cycle)		Arginine and proline metabolism	Glycine, serine and threonine metabolism
Coumarine and phenylpropanoid biosynthesis		Lysine biosynthesis	Citrate cycle (TCA cycle)
Galactose metabolism		Carbon fixation	Histidine metabolism
Glyoxylate and dicarboxylate metabolism		Citrate cycle (TCA cycle)	Nucleotide sugars metabolism
Pentose and glucuronate interconversions		Glutamate metabolism	Alanine and aspartate metabolism

Table 4-3. Predicted metabolic changes that are consistent during development.

Pathways always with increased flux in SnRK1.1:HA		**Pathways always with decreased flux in SnRK1.1:HA**	
In the day	**In the night**	**In the day**	**In the night**
N/A	N/A	Aminoacyl-tRNA biosynthesis	Starch and sucrose metabolism
		Cell wall biosynthesis	Cell wall biosynthesis
		Cysteine metabolism	Galactose metabolism
		Glycine, serine and threonine metabolism	Aminoacyl-tRNA biosynthesis
		Methionine metabolism	Cysteine metabolism
		Starch and sucrose metabolism	Glycine, serine and threonine metabolism
			Glycolysis / Gluconeogenesis
			Methionine metabolism
			Nucleotide sugar metabolism
			Oxidative phosphorylation (overall in mitochondria)

Table 4-4. Predicted metabolic changes that vary during development.

Increased flux in pre-flower, decreased flux in post-flower		**Decreased flux in pre-flower, increased flux in post-flower**	
In the day	**In the night**	**In the day**	**In the night**
Pentose phosphate pathway	Fatty acid biosynthesis	Cell wall biosynthesis	N/A
Carbon fixation	Cell wall biosynthesis	Hemicellulose biosynthesis	
Starch and sucrose metabolism	Hemicellulose biosynthesis	Nucleotide sugar metabolism	
Glycolysis / Gluconeogenesis	Aminoacyl-tRNA biosynthesis		
ATP and NADPH generated from light (overall reaction in chloroplast)	Valine, leucine and isoleucine biosynthesis		
Fructose and mannose metabolism	Citrate cycle (TCA cycle)		
Galactose metabolism	Glycine, serine and threonine metabolism		
Pentose and glucuronate interconversions	Histidine metabolism		
Streptomycin biosynthesis	Alanine and aspartate metabolism		
	Nitrogen metabolism		

DISCUSSION

Four GEMs were constructed to predict growth and starch metabolism in WT and SnRK1.1:HA during pre-flowering and post-flowering developmental stages. Experimental validation of the model predictions revealed that the models can quantitatively predict growth and starch accumulation and turnover with high accuracy. Further refinement of the models enabled accurate prediction of the starch changes across multiple stages of plant development and analysis of metabolic flux changes.

Non-starch biomass is accumulated at night in pre-flowering Arabidopsis

Simulation of pre-flowering growth was performed with the assumption that plants maximize the transitory starch level. Under this assumption, very little mass was predicted to accumulate during day growth because most of the photoassimilates were predicted to partition into starch. As shown in Figure 4-6D, the predicted transitory starch levels closely matched experimentally measured levels, which suggests that the predicted biomass accumulation pattern for pre-flowering plants may be valid. The presented model disagrees with a previous study that specifically modeled diurnal leaf growth (Weraduwage et al., 2015). The previous study concluded that, according to their model simulation, day time growth in terms of both size and mass is greater than night time growth (Weraduwage et al., 2015). The disagreement may be due to the differences between the fundamental model frameworks. The Weraduwage et al. model was constructed based on empirical relationships between photosynthesis and growth; whereas, flux-based modeling with AraGEM solely depends on mass balancing to satisfy the biomass equation (de Oliveira Dal'Molin et al., 2010; Weraduwage et al., 2015). Flux-based modeling allowed us to calculate the theoretical maximum of a system. In this case, the greatest amount of growth and starch biosynthesis given a known CO_2 assimilation rate and leaf RER. This means

that our model finds no other way to distribute day and night growth if the experimentally measured transitory starch level must be satisfied.

The role of SnRK1.1 in starch metabolism and plant development

Starch has been shown to play a significant role in regulating flowering time (Matsoukas et al., 2013; Yu et al., 2000). Leaf starch deficiency has been shown to delay floral initiation (Yu et al., 2000). It has been proposed that the reduction of starch turnover rate is a primary cause of delayed developmental transition (Matsoukas et al., 2013). Our experimental data shows that although SnRK1.1:HA plants have delayed flowering and development, they maintain a significantly higher starch level than WT in the pre-flowering stage. A previous study on the effect of SnRK1.1 overexpression on flowering time showed that WT flowers 19 days after planting and SnRK1.1:HA flowers 22 days after planting (Williams et al., 2014). Our analysis of 21 day-old WT and SnRK1.1:HA showed no significant difference in the transitory starch level (Figure 4-9B), which indicated no significant difference in starch turnover rate before transitioning to reproductive stage. Our results suggest that the delayed developmental transition in SnRK1.1:HA plants may not be associated with starch. This further supports a previous conclusion of SnRK1.1 involvement in regulating starch and developmental transition (Baena-González et al., 2007; Gazzarrini and Tsai, 2014). A previous study found that double-knockout of SnRK1.1 and SnRK1.2 resulted in stunted plant growth, significantly increased starch accumulation, and reduced starch turnover as compared to WT of the same chronological age (Baena-González et al., 2007). In contrast, our data shows that SnRK1.1:HA overexpression does not increase starch turnover or reduce starch accumulation, and instead shows an increase of starch accumulation at a younger stage (Figure 4-6C and D). When WT and SnRK1.1:HA plants were compared by chronological age, the transitory starch levels measured at 14, 21, 28,

and 35 days showed no significant differences (Figure 4-9B). In addition, the accurate predictions of starch metabolism in WT and SnRK1.1:HA plants using flux-based modeling suggests that SnRK1.1 starch metabolism may be governed by mass balancing, as opposed to regulatory control. Based on our results, we hypothesize that SnRK1.1 may serve to prevent the inhibition of starch turnover (Figure 4-10). This would explain the reduced starch turnover in SnRK1 double-knockout plants as well as the lack of difference in starch turnover in our SnRK1.1:HA plants (Baena-González et al., 2007). It also explains the reduction of starch in SnRK1 overexpression plants under glucose supplemented growth (Jossier et al., 2009). In this case, high glucose reduced SnRK1 phosphorylation leading to SnRK1 inactivation, which reduced starch turnover and elevated starch content (Jossier et al., 2009; Rubenstein et al., 2008). Overexpression plants are less sensitive to glucose, thus they have greater starch turnover and lower starch accumulation than WT (Jossier et al., 2009).

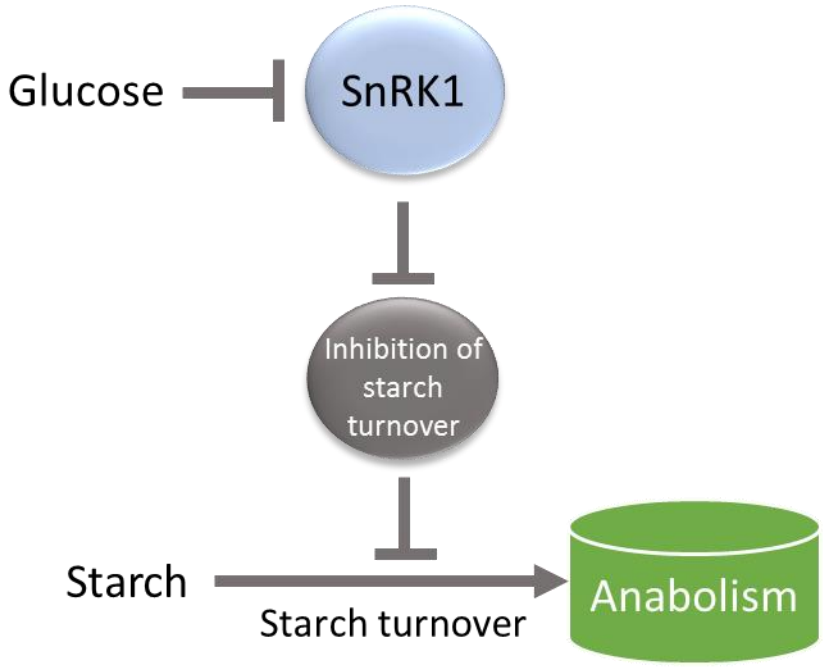

Figure 4-10. Proposed model for the role of SnRK1 in starch metabolism. Gene knockout or glucose inhibition of SnRK1 reduces starch turnover, which increases starch accumulation (Baena-González et al., 2007; Jossier et al., 2009). SnRK1 overexpression plants under high glucose have reduced sensitivity to glucose due to elevated enzyme abundance, which increases starch turnover and lower starch accumulation compares to WT grown under high glucose (Jossier et al., 2009). SnRK1 overexpression plants do not significantly reduce starch under normal growth condition due to indirect regulation starch turnover as shown in this work and previous studies (Baena-González et al., 2007; Jossier et al., 2009).

Modeling non-steady state metabolic activities

It is well accepted that plant metabolism is highly dynamic; thus, tremendous efforts from many research groups have been invested in the attempt to develop robust models. Methods to perform dynamic FBA (dFBA) were initially introduced to model metabolic reprogramming in *E. coli* during diauxic growth (Mahadevan et al., 2002). Since then dFBA has been utilized in studying the effects of alternating growth nutrients and environmental conditions as well as generating metabolic engineering strategies to overproduce commodity chemicals and protein (Anesiadis et al., 2008; Hjersted et al., 2007; Lequeux et al., 2010; Luo et al., 2009; Oddone et al., 2009). Reformulation of dFBA has also been examined by integrating other optimization methods, such as minimization of metabolic adjustments (MOMA) and regulatory on/off minimization (ROOM) to capture the biological objective of minimizing energetic cost (Kleessen and Nikoloski, 2012; Segre et al., 2002; Shlomi et al., 2005).

Despite these advances, constraint-based modeling still depends on the pseudo-steady state assumption, which a developing plant is not under. A previous study on barley implemented a multiscale metabolic modeling (MMM) approach, which enabled modeling of source-sink interactions on a spatiotemporal resolution (Grafahrend-Belau et al., 2013). The MMM approach recognized the limitation of FBA, and restrict the FBA model to solving only spatial distribution of metabolic flux (Grafahrend-Belau et al., 2013). It then recruited a dynamic functional plant model to solve temporal metabolic adjustments (Grafahrend-Belau et al., 2013). The resulting model captured metabolic activities; however, quantitative results were not validated experimentally. A recent study took a different approach to model the development of tomato fruit (Colombié et al., 2015). Temporal resolution was achieved by using 9 steady-state tomato models each composed of a biomass composition measured at a distinct time points. This

approach is justifiable because of the relatively slow metabolic changes in tomato fruit on the scale of fruit development (Colombié et al., 2015). In contrast, metabolism of leaf tissues during day and night can be very different. The present study demonstrates a novel approach to simulate the dynamics of non-steady state plant metabolism during diurnal cycle across development. To the authors' knowledge, this is the first time plant growth and starch concentrations have been predicted accurately through genome-scale modeling.

Refinement of models enabled more accurate predictions

The ability of our models to accurately predict trends in absolute growth and starch level changes indicates that they may be useful in predicting other metabolic activities. Experimental measurements of starch can, in turn, be used to correct inaccuracy in the predictions and improve model performance. This reiterative process has traditionally been used in model development process (Grafahrend-Belau et al., 2013; Weraduwage et al., 2015).

The sources of disagreement in our model guided predictions and experimental data on starch could result from 1) subtle differences between biological samples used for parameterization and validation and 2) the inaccuracy of assuming maximum transitory starch. As shown in Figure 4-6C, the predicted EOD and EON starch levels in pre-flowering plants are lower than the experimental values. This may be attributed to lower mid-day starch levels in the biological samples used for model parameterization compared to samples used for validation despite the rigorous experimental controls. When the initial starch level of the simulation was adjusted to a new mid-day starch level estimated with the experimental measured EOD and EON starch concentrations, the predicted EOD and EON starch concentrations became much closer to the experimental values (Figure 4-8C). The differences between experimental and predicted starch values are more dramatic for post-flowering plants. This suggests the assumptions used in

model simulation, in which the plant maximizes the transitory starch pool, works better when modeling pre-flowering plants. This means that in addition to adjusting the initial starch value, the simulation objective must be changed, however, regulation of starch turnover in plants has been shown to be complex (Fernie et al., 2002; Stitt and Zeeman, 2012). Rather than imposing a more complex assumption for the simulation, the objective of the simulation was simply changed to replicate the experimental transitory starch level. Because of this new objective, it is not surprising that the new predictions on starch levels in post-flowering plants are exactly the same as the experimental values. Although these refinements are artificial, they are necessary for predicting metabolic fluxes and starch levels across multiple plant ages.

Model predictions suggest greater metabolic changes at night

The primary goal of this study is to see whether flux-based modeling can be used to help reveal the role of SnRK1.1 in plant metabolism. Through simulations and experimental validations, we were able to propose a novel hypothesis on the role of SnRK1.1 in starch metabolism. It would be interesting to examine if SnRK1.1:HA overexpression resulted in any other significant metabolic changes. The predicted SnRK1.1:HA metabolic flux distribution at each developmental stage (pre-flowering and post-flowering) was compared with predicted WT metabolic flux distribution. Pathways with highest frequency of changes were compiled into lists. Table 4-1 to Table 4-4 shows the truncated lists of most altered pathways at both stages of development and diurnal cycles. The assumption is that pathways with the most predicted changes are most effected by SnRK1.1:HA overexpression. The rationale behind this assumption is that predicted fluxes are different to compensate for the differences in the biomass composition and starch levels, which are different because of SnRK1.1:HA overexpression.

It is not surprising to see that the most altered pathways are predicted to be involved in the synthesis of biomass components given the assumption for this analysis. It is interesting that most of the predicted changes occur at night. This is primarily due to the prediction that more non-starch growth occurs during the night. Plant studies are queried rarely during the night unless there are specific objectives because research is mostly performed during the day. If the model is accurate about night growth, it would be critical to expand more research into studying night time metabolic changes of plants. A quick search on Google Scholar for "Arabidopsis" would retrieve over 1 million results, however, querying for "Arabidopsis night" only retrieves 61,000 results. Previous studies have already shown that diurnal transition can have significant impact on the metabolism at both metabolomics and transcriptomics levels (Gibon et al., 2006). Extended night can also elevate the levels of many amino acids (Gibon et al., 2006). Extended night has also been shown to activate SnRK1 and initiates significant gene expression reprogramming (Baena-González et al., 2007). Further analysis of night time metabolism may reveal new roles for many previously characterized genes.

CONCLUSION

Novel GEMs were built to model the metabolism of Arabidopsis overexpressing SnRK1.1:HA in effort to further understand the metabolic role of SnRK1.1 over growth and development. Biomass compositions including cell wall, lipids, amino acids, and starch were measured in SnRK1.1:HA and WT at pre-flowering and post-flowering developmental stages. These data are used to re-parameterize the AraGEM model, and construct SnRK1.1:HA and WT GEMs. The original AraGEM framework was expanded to allow for 24-hour simulation of growth and starch accumulation and turnover in day and night cycles. Simulation predicted previously undermined dynamics between biomass and starch. Experimental validation revealed

that the quantitative levels of growth and starch were predicted with unprecedented accuracy. Data from experimental validation were used to refine the models for a second round of prediction. The refined models were used to guide further analysis of starch metabolism in SnRK1.1:HA and help build a novel hypothesis on the role of SnRK1.1 in regulating starch, which is SnRK1.1 prevents the inhibition of starch turnover. Analysis of model predictions on the metabolism also stressed a previously overlooked value in night time plant metabolism. This work demonstrated novel techniques and workflow to use GEM to guide analysis plant metabolism.

REFERENCES

Allen, D. K., Libourel, I. G. L., Shachar-Hill, Y., 2009. Metabolic flux analysis in plants: coping with complexity. Plant, cell & environment. 32**,** 1241-1257.

Anesiadis, N., Cluett, W. R., Mahadevan, R., 2008. Dynamic metabolic engineering for increasing bioprocess productivity. Metabolic engineering. 10**,** 255-266.

Baena-González, E., Rolland, F., Thevelein, J. M., Sheen, J., 2007. A central integrator of transcription networks in plant stress and energy signalling. Nature. 448**,** 938-942.

Caspeta, L., Shoaie, S., Agren, R., Nookaew, I., Nielsen, J., 2012. Genome-scale metabolic reconstructions of Pichia stipitis and Pichia pastoris and in silico evaluation of their potentials. BMC systems biology. 6**,** 24.

Cheung, C. Y. M., Williams, T. C. R., Poolman, M. G., Fell, D. A., Ratcliffe, R. G., Sweetlove, L. J., 2013. A method for accounting for maintenance costs in flux balance analysis improves the prediction of plant cell metabolic phenotypes under stress conditions. The plant journal. 75**,** 1050-1061.

Collakova, E., Aghamirzaie, D., Fang, Y., Klumas, C., Tabataba, F., Kakumanu, A., Myers, E., Heath, L. S., Grene, R., 2013. Metabolic and transcriptional reprogramming in developing soybean (Glycine max) embryos. Metabolites. 3**,** 347-372.

Collakova, E., Yen, J. Y., Senger, R. S., 2012. Are we ready for genome-scale modeling in plants? Plant science. 191–192**,** 53-70.

Colombié, S., Nazaret, C., Bénard, C., Biais, B., Mengin, V., Solé, M., Fouillen, L., Dieuaide-Noubhani, M., Mazat, J. P., Beauvoit, B., 2015. Modelling central metabolic fluxes by constraint-based optimization reveals metabolic reprogramming of developing Solanum lycopersicum (tomato) fruit. The plant journal. 81**,** 24-39.

de Oliveira Dal'Molin, C. G., Quek, L. E., Palfreyman, R. W., Brumbley, S. M., Nielsen, L. K., 2010. AraGEM, a genome-scale reconstruction of the primary metabolic network in Arabidopsis. Plant physiology. 152**,** 579-89.

Fernie, A. R., Willmitzer, L., Trethewey, R. N., 2002. Sucrose to starch: a transition in molecular plant physiology. Trends in plant science. 7**,** 35-41.

Förster, J., Famili, I., Fu, P., Palsson, B. Ø., Nielsen, J., 2003. Genome-scale reconstruction of the Saccharomyces cerevisiae metabolic network. Genome research. 13**,** 244-253.
Gazzarrini, S., Tsai, A. Y.-L., 2014. Trehalose-6-phosphate and SnRK1 kinases in plant development and signaling: the emerging picture. Frontiers in plant science. 5**,** 119.
Gibon, Y., Blaesing, O. E., Palacios, N., Pankovic, D., Hendriks, J. H. M., Fisahn, J., Hoehne, M., Günter, M., Stitt, M., 2004. Adjustment of diurnal starch turnover to short days: Depletion of sugar during the night leads to a temporary inhibition of carbohydrate utilisation, accumulation of sugars and post-translational activation of ADPglucose pyrophosphorylase in the following light period. The plant journal. 39.
Gibon, Y., PYL, E. T., Sulpice, R., Lunn, J. E., Hoehne, M., Guenther, M., Stitt, M., 2009. Adjustment of growth, starch turnover, protein content and central metabolism to a decrease of the carbon supply when Arabidopsis is grown in very short photoperiods. Plant, cell & environment. 32**,** 859-874.
Gibon, Y., Usadel, B., Blaesing, O. E., Kamlage, B., Hoehne, M., Trethewey, R., Stitt, M., 2006. Integration of metabolite with transcript and enzyme activity profiling during diurnal cycles in Arabidopsis rosettes. Genome biology. 7**,** R76.
Graf, A., Schlereth, A., Stitt, M., Smith, A. M., 2010. Circadian control of carbohydrate availability for growth in Arabidopsis plants at night. Proceedings of the national academy of sciences. 107**,** 9458-9463.
Grafahrend-Belau, E., Junker, A., Eschenröder, A., Müller, J., Schreiber, F., Junker, B. H., 2013. Multiscale metabolic modeling: dynamic flux balance analysis on a whole-plant scale. Plant physiology. 163**,** 637-647.
Hasegawa, P. M., Bressan, R. A., Zhu, J.-K., Bohnert, H. J., 2000. Plant cellular and molecular responses to high salinity. Annual review of plant biology. 51**,** 463-499.
Hjersted, J. L., Henson, M. A., Mahadevan, R., 2007. Genome-scale analysis of Saccharomyces cerevisiae metabolism and ethanol production in fed-batch culture. Biotechnology and bioengineering. 97**,** 1190-1204.
Hrabak, E. M., Chan, C. W. M., Gribskov, M., Harper, J. F., Choi, J. H., Halford, N., Kudla, J., Luan, S., Nimmo, H. G., Sussman, M. R., Thomas, M., Walker-Simmons, K., Zhu, J.-K., Harmon, A. C., 2003. The Arabidopsis CDPK-SnRK Superfamily of Protein Kinases. Plant physiology. 132**,** 666-680.
Jossier, M., Bouly, J. P., Meimoun, P., Arjmand, A., Lessard, P., Hawley, S., Grahame Hardie, D., Thomas, M., 2009. SnRK1 (SNF1-related kinase 1) has a central role in sugar and ABA signalling in Arabidopsis thaliana. The plant journal. 59**,** 316-328.
Kleessen, S., Nikoloski, Z., 2012. Dynamic regulatory on/off minimization for biological systems under internal temporal perturbations. BMC systems biology. 6**,** 1.
Lequeux, G., Beauprez, J., Maertens, J., Van Horen, E., Soetaert, W., Vandamme, E., Vanrolleghem, P. A., 2010. Dynamic metabolic flux analysis demonstrated on cultures where the limiting substrate is changed from carbon to nitrogen and vice versa. BioMed research international. 2010.
Luo, R., Wei, H., Ye, L., Wang, K., Chen, F., Luo, L., Liu, L., Li, Y., Crabbe, M. J. C., Jin, L., 2009. Photosynthetic metabolism of C3 plants shows highly cooperative regulation under changing environments: A systems biological analysis. Proceedings of the national academy of sciences. 106**,** 847-852.
Mahadevan, R., Edwards, J. S., Doyle, F. J., 2002. Dynamic flux balance analysis of diauxic growth in Escherichia coli. Biophysical journal. 83**,** 1331-1340.

Matsoukas, I. G., Massiah, A. J., Thomas, B., 2013. Starch metabolism and antiflorigenic signals modulate the juvenile-to-adult phase transition in Arabidopsis. Plant, cell & environment. 36**,** 1802-1811.
McKibbin, R. S., Muttucumaru, N., Paul, M. J., Powers, S. J., Burrell, M. M., Coates, S., Purcell, P. C., Tiessen, A., Geigenberger, P., Halford, N. G., 2006. Production of high-starch, low-glucose potatoes through over-expression of the metabolic regulator SnRK1. Plant biotechnology journal. 4**,** 409-418.
Oddone, G. M., Mills, D. A., Block, D. E., 2009. A dynamic, genome-scale flux model of Lactococcus lactis to increase specific recombinant protein expression. Metabolic engineering. 11**,** 367-381.
Park, J. M., Kim, T. Y., Lee, S. Y., 2009. Constraints-based genome-scale metabolic simulation for systems metabolic engineering. Biotechnology advances. 27.
Polge, C., Thomas, M., 2007. SNF1/AMPK/SnRK1 kinases, global regulators at the heart of energy control? Trends in plant science. 12**,** 20-28.
Poolman, M. G., Kundu, S., Shaw, R., Fell, D. A., 2013. Responses to light intensity in a genome-scale model of rice metabolism. Plant physiology. 162**,** 1060-1072.
Poolman, M. G., Miguet, L., Sweetlove, L. J., Fell, D. A., 2009. A genome-scale metabolic model of Arabidopsis and some of its properties. Plant physiology. 151**,** 1570-1581.
Rubenstein, E. M., McCartney, R. R., Zhang, C., Shokat, K. M., Shirra, M. K., Arndt, K. M., Schmidt, M. C., 2008. Access denied: Snf1 activation loop phosphorylation is controlled by availability of the phosphorylated threonine 210 to the PP1 phosphatase. The Journal of biological chemistry. 283**,** 222-230.
Santander, J., Martin, T., Loh, A., Pohlenz, C., Gatlin III, D. M., Curtiss III, R., 2013. Mechanisms of intrinsic resistance to antimicrobial peptides of Edwardsiella ictaluri and its influence on fish gut inflammation and virulence. Microbiology. 159**,** 1471-1486.
Schilling, C. H., Edwards, J. S., Palsson, B. O., 1999. Toward metabolic phenomics: analysis of genomic data using flux balances. Biotechnology progress. 15**,** 288-295.
Segre, D., Vitkup, D., Church, G. M., 2002. Analysis of optimality in natural and perturbed metabolic networks. Proceedings of the national academy of sciences. 99**,** 15112-15117.
Sheen, J., 2014. Master regulators in plant glucose signaling networks. Journal of plant biology. 57**,** 67-79.
Shlomi, T., Berkman, O., Ruppin, E., 2005. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. Proceedings of the national academy of sciences. 102**,** 7695-7700.
Smith, A. M., Zeeman, S. C., 2006. Quantification of starch in plant tissues. Nature protocols. 1**,** 1342-5.
Stitt, M., Zeeman, S. C., 2012. Starch turnover: pathways, regulation and role in growth. Current opinion in plant biology. 15**,** 282-292.
Tardieu, F., Granier, C., Muller, B., 1999. Modelling leaf expansion in a fluctuating environment: are changes in specific leaf area a consequence of changes in expansion rate? New phytologist. 143**,** 33-43.
Töpfer, N., Caldana, C., Grimbs, S., Willmitzer, L., Fernie, A. R., Nikoloski, Z., 2013. Integration of genome-scale modeling and transcript profiling reveals metabolic pathways underlying light and temperature acclimation in Arabidopsis. The plant cell. 25**,** 1197-1211.

Updegraff, D. M., 1969. Semimicro determination of cellulose inbiological materials. Analytical biochemistry. 32**,** 420-424.

Watanabe, M., Balazadeh, S., Tohge, T., Erban, A., Giavalisco, P., Kopka, J., Mueller-Roeber, B., Fernie, A. R., Hoefgen, R., 2013. Comprehensive dissection of spatiotemporal metabolic shifts in primary, secondary, and lipid metabolism during developmental senescence in Arabidopsis. Plant physiology. 162**,** 1290-310.

Weraduwage, S. M., Chen, J., Anozie, F. C., Morales, A., Weise, S. E., Sharkey, T. D., 2015. The relationship between leaf area growth and biomass accumulation in Arabidopsis thaliana. Frontiers in plant science. 6**,** 167.

Williams, S. P., Rangarajan, P., Donahue, J. L., Hess, J. E., Gillaspy, G. E., 2014. Regulation of Sucrose non-Fermenting Related Kinase 1 genes in Arabidopsis thaliana. Frontiers in plant science. 5.

Williams, T. C. R., Poolman, M. G., Howden, A. J. M., Schwarzlander, M., Fell, D. A., Ratcliffe, R. G., Sweetlove, L. J., 2010. A Genome-Scale Metabolic Model Accurately Predicts Fluxes in Central Carbon Metabolism under Stress Conditions. Plant physiology. 154**,** 311-323.

Yamaguchi-Shinozaki, K., Shinozaki, K., 2006. Transcriptional regulatory networks in cellular responses and tolerance to dehydration and cold stresses. Annual review of plant biology. 57**,** 781-803.

Yu, T.-S., Lue, W.-L., Wang, S.-M., Chen, J., 2000. Mutation of Arabidopsis plastid phosphoglucose isomerase affects leaf starch synthesis and floral initiation. Plant physiology. 123**,** 319-326.

Zablackis, E., Huang, J., Muller, B., Darvill, A. G., Albersheim, P., 1995. Characterization of the cell-wall polysaccharides of Arabidopsis thaliana leaves. Plant physiology. 107**,** 1129-1138.

CHAPTER 5

CONCLUSIONS

Computational modeling enables us to efficiently organize and apply knowledge in effort to explore the unknown. Rapid independent advances in both computational and experimental biology have created an atmosphere to communicate computational results and gaps in knowledge to experimental researchers. In return, this collaboration can be used to update the knowledgebase and identify new gaps and/or areas for exploration. The research presented in this dissertation recognized the pitfalls of many current predictive algorithms for metabolic engineering, which include overwhelming predictions, unrealistic scoring methods, lack of quantitative predictions, long run times, and the absence of proper experimental validation. Here, the NR-Opt toolbox is presented and contains novel predictive algorithms for metabolic engineering strategy design. In addition, this research has developed a novel flux-based modeling framework involving GEMs that includes an analytical pipeline to enable model-guided discovery. The utility of this modeling framework was validated when quantitative predictions on starch changes in WT and transgenic plants agree with experimental results.

The work done in this research provided a well curated list of current predictive algorithms for designing metabolic engineering strategies. A novel algorithm was introduced in the NR-Opt toolbox to overcome the limitations of current algorithms. Finally, a sophisticated modeling framework was developed to model non-steady state metabolic changes and guide sugar-signaling studies in plants. These advances can significantly improve the efficiency of metabolic engineering workflow and enable accurate genome-scale metabolic analysis of plant signaling pathways.

FUTURE DIRECTIONS

It is obvious that further advancements in computational biology will enable a large array of benefits, which conservatively includes precision medicine, effective gene therapy, accurate metabolic engineering, and optimized farming practice. For this reason, there are significant focuses on combining multiple modeling frameworks, such as kinetic models, metabolic models, signaling and regulatory models, heuristic models, and statistical models, to integrate as much current biology as possible to help guide future research. The greatest challenge is the need for high quality curation and cross-validation, which can best be accomplished with one or both of two ways: 1) crowd sourcing through interaction of the scientific community and the public, and 2) developing sophisticated machine learning methods that replaces human intervention. Needless to say that not only is the first option more attainable, the first option can lead to the realization of the latter. It is necessary to develop an efficient framework to facilitate crowd sourcing. It is also imperative to communicate the necessity and generate incentive for crowd sourcing. Current technology industry is ahead of the scientific community in crowd sourcing data, such as health, diet, and daily activities, by creating user-friendly applications and wearable technologies. These data are arguably being applied in the least creative and sophisticate analysis relative to the developments in academia. For this reason, the most critical next step is to learn from the industrial approaches and utilize their methods to benefit scientific agenda. This can include developing user-friendly applications that stream-line data integration into biological modeling platforms and improving communication of public benefits in utilizing methods that are beyond common comprehension. Open-source communities, such as Blender 3D modeling software, generates exciting products that attracts talents who seeks personal satisfaction over financial benefits. If the same can be done in the computational biology community, then the

advancement will accelerate at unprecedented speed, and bringing all the previously discussed benefits much closer to realization.

It would be interesting to further explore the utility of the novel flux-based modeling framework introduced in this research in studying animal metabolism. It may be possible to model non-steady dietary and fat storage in mammalian organisms, such as mice and human. Publicly available data of net body biomass composition can be used to construct the biomass equation. Different dietary habits can be used as constraints to model changes in stored fat. Levels of maintenance energy can be a function of daily exercise. This study may reveal previously overlooked relationship between body types and rate of weight gain/loss.

APPENDICES

A. ALGORITHM FOR THE NODE-REWARD OPTIMIZATION TOOLBOX

THE NR-OPT ALGORITHM

Both NR-Knock and NR-Ox search for a metabolic engineering strategy that can satisfy the termination criteria (i.e. 95% maximum BPCY). The search is accomplished with a modified steepest ascent hill climbing search algorithm that allows for delayed ascension. Here, NR-Knock is used to explain the algorithm. NR-Knock first takes in the WT organism GEM, which is a strategy with no elimination of reaction (i.e. no gene KO). NR-Knock assigns the user-defined initial reward point (p_0) to the WT strategy reward point (p_{WT}) (i.e., these values are equal initially). In the example shown in Figure A-1, p_0 is 0. There are two rules on the reward points: (i) any strategy with $p < 0$ is eliminated from "expansion" and (ii) no strategy can hold more than the user-defined maximum reward points (p_{max}). In the example shown in Figure A-1, p_{max} is 1. NR-Knock performs FBA with the WT model to calculate the minimum BPCY under maximal growth. In the example in Figure A-1, the minimum BPCY of the WT strategy is 0. Because no other strategy is examined, WT strategy with a BPCY of 0 is currently the "best" solution. The WT model is then used to calculate the maximum BPCY, thus allowing the user to define the BPCY criteria to terminate NR-Knock search (i.e. 95% of maximum BPCY). Because this is a steepest ascent hill climbing algorithm, for a strategy to receive an additional reward point, its minimum BPCY must be higher than the current best BPCY. One point is deducted from a strategy that fails to do so.

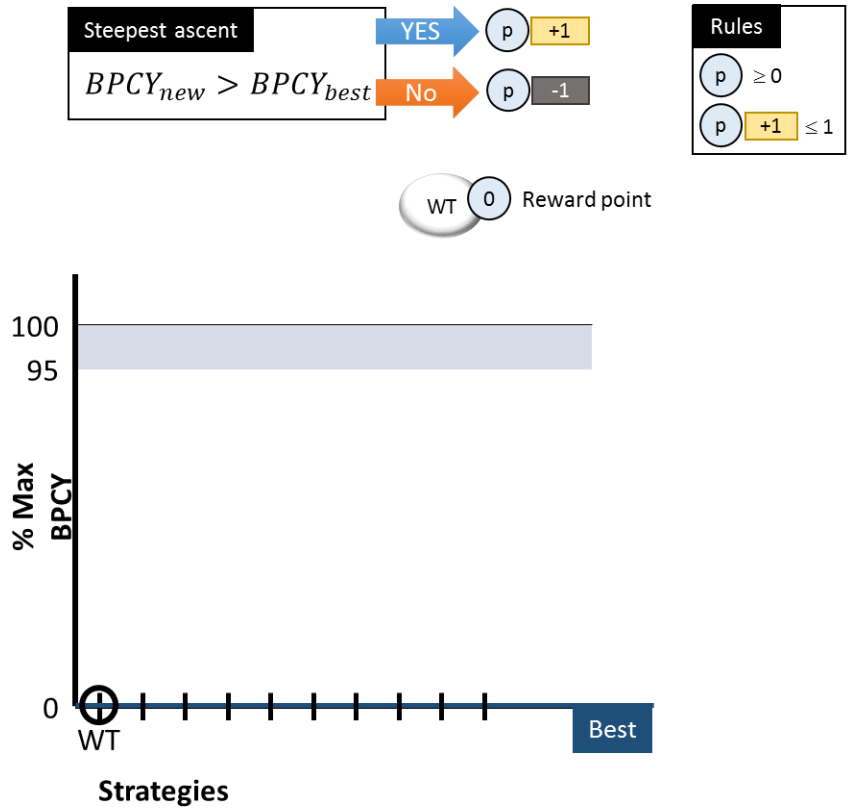

Figure A-1. Initialization of NR-Knock search.

The WT strategy "expands" into single-gene KO strategies, which test for elimination of a single reaction in the WT GEM. In the example shown in Figure A-2, there are 4 possible single-KO strategies to test. Each strategy inherits the reward point of the WT strategy. FBA is performed for each strategy to determine their minimum BPCY. In the example shown in Figure A-2, all single-KO strategies have higher BPCY than the current best. NR-Knock now performs 2 steps: (i) award 1 point to each strategy and (ii) sort the strategies from the highest BPCY.

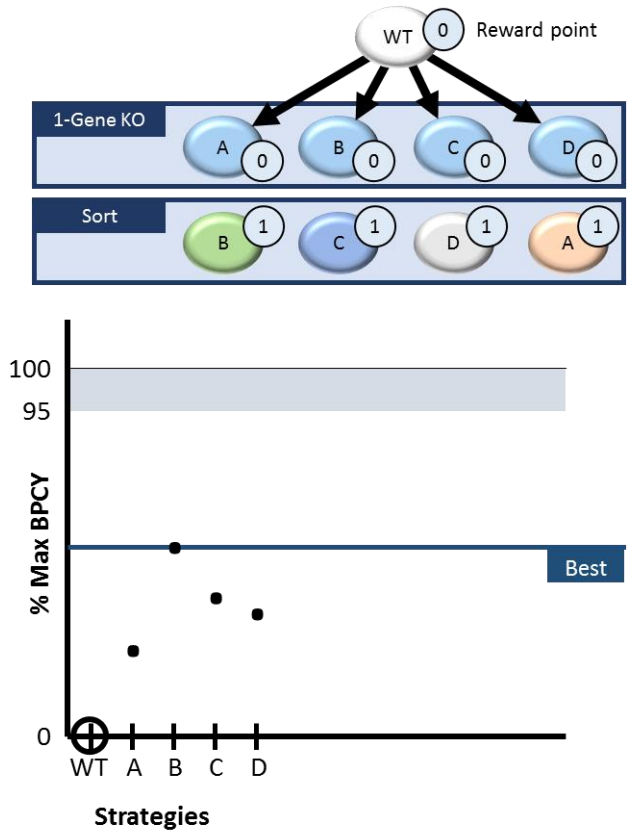

Figure A-2. Evaluation of single-KO strategies.

The current best strategy expands into double-KO strategies that inherits their parent's reward point value. In the example shown in Figure A-3, there are 3 double-KO strategies that can be derived from the single-KO of B. FBA is performed on each of these double-KO strategies. In the example shown in Figure A-3, none of the double-KO strategies have a BPCY that exceeds the current best BPCY; thus, 1 point is deducted from each. Because the reward points of the double-KO strategies have not fallen below 0, they are sorted by BPCY (highest to lowest) and allowed to expand into triple-KO strategies.

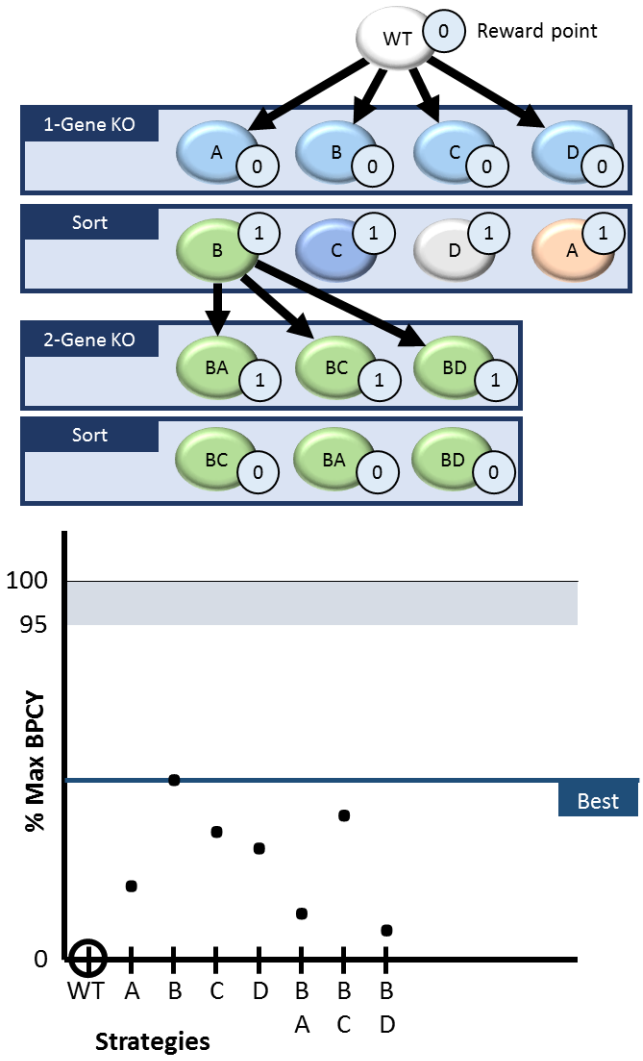

Figure A-3. Evaluation of double-KO strategies.

The best double-KO strategy is expanded into triple-KO strategies and evaluated with FBA. In the example shown in Figure A-4, none of the triple-KO strategies have BPCY higher than the best, thus 1 point is deducted. These triple-KO strategies are eliminated from further expansion because their reward points are below 0. Thus, the next best double-KO strategy is expanded into triple-KO strategies for evaluation.

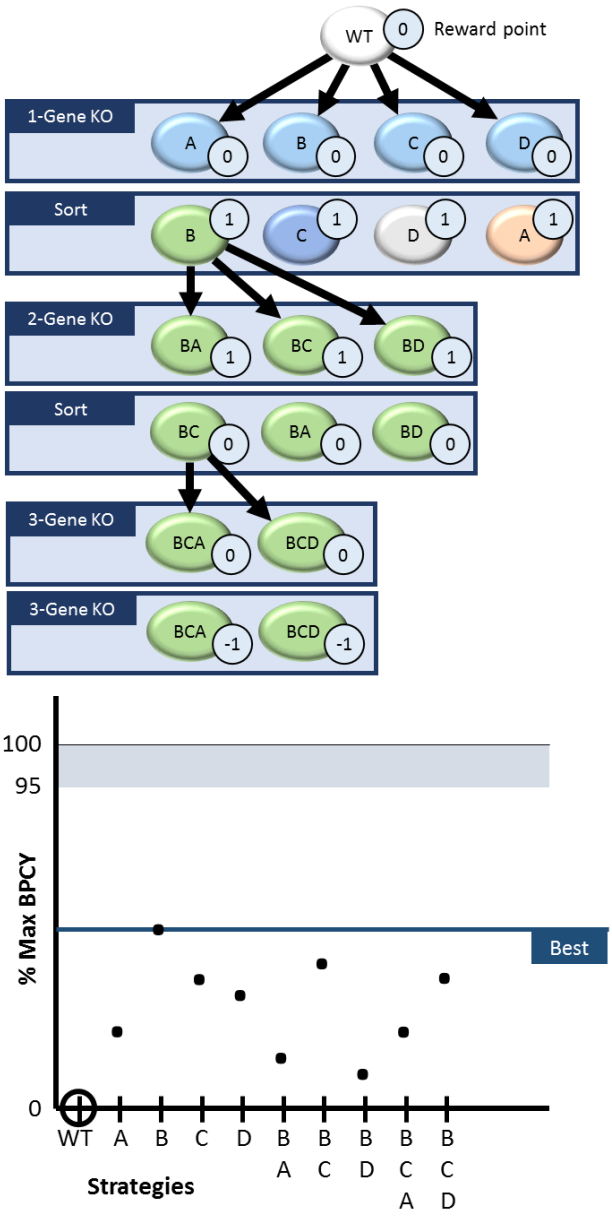

Figure A-4. Evaluation of triple-KO strategies.

In the example shown in Figure A-5, the next best double-KO strategy (BA) is expanded into only 1 triple-KO strategy (BAD) because the algorithm does not evaluate repeats. FBA is performed on the triple-KO strategy (BAD), and reveals a BPCY that satisfies the termination criteria. NR-Knock terminates and designates triple-KO of BAD as the "best" strategy. Any strategy that had at some point significantly improved BPCY (i.e. single-KO of A, B, C, or D)

are alternative strategies. The user has the option of manipulating the BPCY acceptance value (e.g., 95% of maximum BPCY) so that multiple designs of a desired BPCY level are returned.

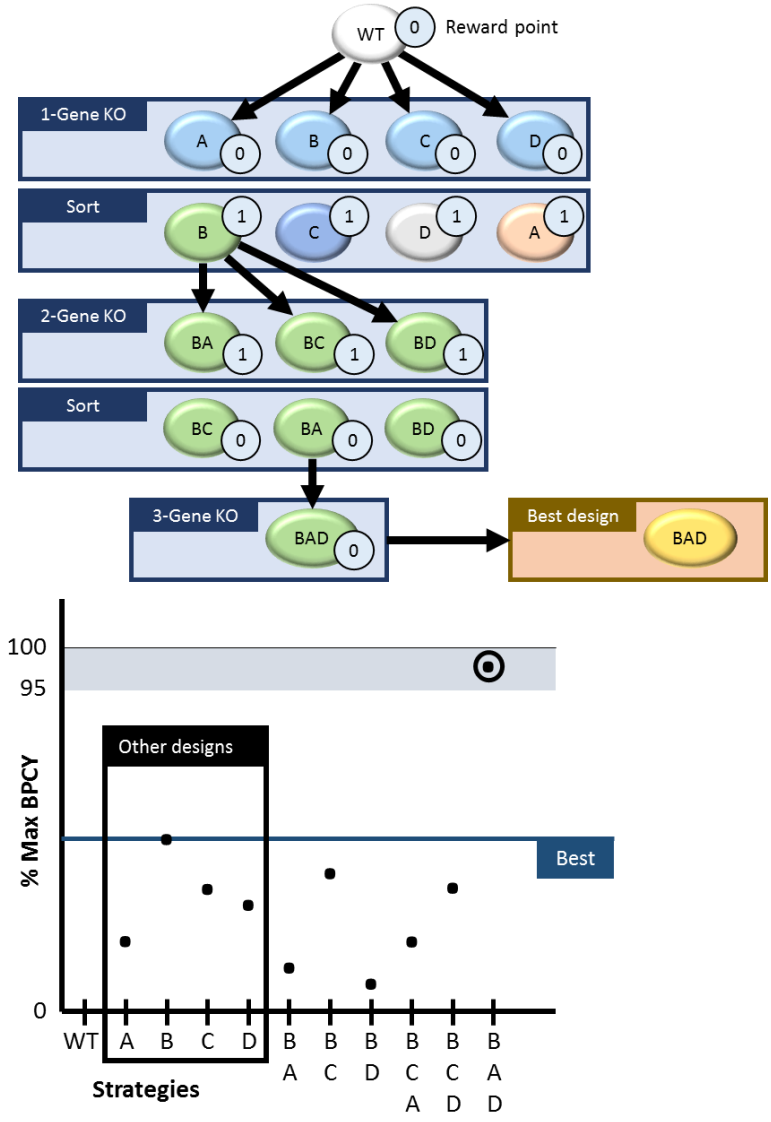

Figure A-5. Identification of the best metabolic engineering strategy.

NR-Ox follows uses the same algorithm as described for NR-Knock. The primary difference is that instead of performing FBA after elimination of reactions, NR-Ox perform FBA after increasing or decreasing flux ratio of a reaction|node pair to a user-defined level. As described in Methods, the flux ratio (r_i) of a reaction|node pair i is the ratio of a metabolite partitioned as a reactant to a reaction (the value is between 0 and 1). If, for example, user defines

flux ratio high and low coefficients as 0.9 ($coef_{hi}$) and 0.01 ($coef_{lo}$) to evaluate OX and KD strategies. These coefficients are used as described in Equation 1 and Equation 2 to calculate the OX and KD flux ratios.

$$r_{i,OX} = r_i + (1 - r_i) \times coef_{hi} \qquad \text{(Equation 1)}$$

$$r_{i,KD} = r_i \times coef_{lo} \qquad \text{(Equation 2)}$$

If a reaction/node pair has a flux ratio of 0.4, NR-Ox performs FBA after raising its flux ratio to 0.95 ($0.4 + (1 - 0.4) \times 0.9$) to assess the OX strategy, then performs FBA after reducing its flux ratio to 0.004 (0.4×0.01) to assess the KD strategy.

B. MATLAB CODES FOR THE NODE-REWARD OPTIMIZATION TOOLBOX

There are three sections: (i) the core codes and their dependencies, (ii) the driver, and (iii) an example setup of the driver to design strategies for BDO production in *E. coli*. For both NR-Knock and NR-Ox, a folder containing the design strategies will be created. The only relevant file is a text file named "scoreEvolution," which contains all the concise strategies. For NR-Knock, each line is a design in the comma delimited format: [reaction ID 1],[reaction ID 2],[reaction ID 3],…,[BPCY score]. Similarly, the NR-Ox output is in the format: [compound ID 1 | reaction ID 1 | flux ratio assigned],[compound ID 2 | reaction ID 2 | flux ratio assigned],…,[BPCY score]. Be cautious with metric units for BPCY score. In the programs, BPCY score is calculated as $score_{BPCY} = v_{target} \times v_{biomass\ eq}$.

1) NR-OPT CORE CODE AND DEPENDENCIES

Function (NR-Knock): nodeRewardKnockP

```
%% Predicting knock-out strategies using a reward algorithm
% Essentially OptKnock with an additional reduction by only searching
% through node-associated reactions
% **Utilizes parallel computing**
% Author: Jiun Yen
% Date: 2016.11.27
% Version: 2016.12.27
%
% This is an exhaustive search with a reward system to reduce search space
% All members of the RID array starts with p point(s) (allowing p
% additional failed attempt); if successful, the set receive an additional
% point.
% This search is depth-first
%
%
% Input:
%   param.
%       m -model
%       target_rid - RID of the target reaction to maximize yield
%       bio_rid - RID of the biomass equation (growth)
%       BPCYacceptance - biomass-product coupled yield (Choon et al. 2013)
%                      - percent of max theoretical BPCY to accept solution
%                      and terminate run
%       Nbests - number of best strategies to keep
%       searchLethals - whether to look for RIDs which KO is lethal
%       excluded - RIDs to exclude
```

```
%       v_cutoff - flux cutoff for zero
%       hi_cutoff - flux cutoff for high values (recommend: 100)
%       dv_cutoff - minimum degree of improvement each iteration
%       p0 - point(s) to start with
%       pmax - max number of points
%       komax - max number of KOs
%       forceBest - whether new score must be the best known score
%       savedir - folder to save results in
%       fn - file name header
%       mskoption - option for mosek optimizer

function [strategies, param] = nodeRewardKnockP(param)

time0 = tic;

%% compute test RIDs
param.m.c(:) = 0;
param.m.c(param.bio_rid) = 1;

% first check viability
v0 = m_linprogP(param.m, 1, 0, param.mskoption);
if ~v0(param.bio_rid)
    % terminate if no growth
    fprintf('No growth from initial model. Exit search.\n');
    return
end

% perform FVA to find all solution space
fprintf('Performing FVA\n');
[vmax, vmin] = fvaP(param.m, param.mskoption);
fprintf('Completed FVA\n');
v0 = abs(vmax) + abs(vmin);
v0(abs(v0) < param.v_cutoff) = 0;
v0(abs(v0) > param.hi_cutoff) = 0;

% find maximum theoretical BPCY
maxbpcy = findMaxBPCY(param);

% compute flux ratios to find node-associated reactions
fr = genFR(param.m, v0);

% identify all gene-coding reactions
g_rids = geneCodingRxns(param.m);
param.g_rids = g_rids;

% determine RIDs to test with
rids = intersect(fr.rids, g_rids);

% search for lethals if necessary
if param.searchLethals

    Nrxns = size(param.m.S,2);
    m = param.m;
    mskoption = param.mskoption;
    v_cutoff = param.v_cutoff;
```

```
    lethalrids = false(Nrxns,1);
    parfor ii = 1:Nrxns
        [~, bV] = m_linprogP(constraintKO(m, ii), 1, 0, mskoption);
        if bV < v_cutoff
            lethalrids(ii) = true;
        end
    end
    param.lethalrids = find(lethalrids);
    param.excludeRids = union(param.excludeRids, param.lethalrids);

    clear Nrxns m mskoption v_cutoff lethalrids
end

% remove excluded RIDs
rids = setdiff(rids, param.excludeRids);

% print to file
csvwrite([param.savedir param.fn 'all_RIDs.csv'], rids);

%% initialize primary output variables
strategies.best.scores = zeros(param.Nbests, 1);
strategies.best.rids = cell(param.Nbests, 1);

%% initialize eval param
evalParam.max = 1;
evalParam.target = param.target_rid;
evalParam.modifier = [];
evalParam.v_cutoff = param.v_cutoff;
evalParam.mskoption = param.mskoption;

%% perform tests
% The inner search is performed with parallel CPUs. Because of this,
% instead of using stacks, an array of definitely size is used to search
% each time

% constant variables
dv_cutoff = param.dv_cutoff;
roundN = abs(ceil(log(dv_cutoff)/log(10))) + 1;
fn = [param.savedir param.fn 'scoreEvolution.csv'];
f = fopen(fn, 'w');
fprintf(f, 'RID sets,Scores\n');
fclose(f);
BPCYacceptance = param.BPCYacceptance;

% stack to store KO strategies (not a real stack, just an array, because
Matlab...)
ridStack = [];
scoreStack = [];
pointStack = [];
scoreHistory = [];

% initialize bpcyRate
bpcyRate = BPCYacceptance - 1;
```

```
start = true;
while bpcyRate < BPCYacceptance && (~isempty(ridStack) || start)

    % pop the stacks
    if start

        start = false;
        rid0 = [];
        [~, bV] = m_linprogP(param.m, 1, 0, param.mskoption);
        m = param.m;
        m.lb(param.bio_rid) = bV;
        m.c(:) = 0;
        m.c(param.target_rid) = 1;
        [~, tV] = m_linprogP(m, 0, 0, param.mskoption);
        if BPCYacceptance > 0
            score0 = bV * tV;
        else
            score0 = tV;
        end
        parentScore = score0;
        point0 = param.p0;
        ncol = 1;
        m = param.m;

    else

        rid0 = ridStack{1};
        ridStack(1) = [];
        if param.forceBest
            % force new score to be better than all current known score
            score0 = max(scoreHistory);
        else
            % new score just needs to be better than parent's
            score0 = scoreStack(1);
        end
        parentScore = scoreStack(1);
        scoreStack(1) = [];
        point0 = pointStack(1);
        pointStack(1) = [];
        ncol = length(rid0) + 1;
        m = constraintKO(param.m, rid0);

    end

    % precalculate points if strategy fails
    pointFail = point0 - 1;

    % determine if maximum KO is reached
    isMaxKO = length(rid0) >= param.komax-1;

    % determine reward value
    if point0 >= param.pmax
        reward = 0;
    else
        reward = 1;
```

```
    end

    % determine RIDs to test
    v0 = m_linprogP(m, 1, 1, param.mskoption);
    v0(abs(v0) < param.v_cutoff) = 0;
    ridx = intersect(rids, find(v0));
    ridx = setdiff(ridx, rid0);
    ridc = parallel.pool.Constant(ridx);
    Nridx1 = length(ridx);

    % set file print
    fc = parallel.pool.Constant(@() fopen(fn, 'At'), @fclose);

    %% solve LP problems with parallel cores
    toExpand = true(Nridx1, 1);
    points = false(Nridx1, 1);
    scores = zeros(Nridx1, 1);
    orig_scores = zeros(Nridx1, 1); % this is to keep record to determine top
strategies

    fprintf('Evaluate strategies\n');
    parfor ii = 1:Nridx1

        testRids = [rid0 ridc.Value(ii)];
        mes = mesEvalP(m, addMes(initMes, ridc.Value(ii)), evalParam);

        if mes.objV > 0

            % scoring method
            if BPCYacceptance > 0
                % score by Biomass-Product Coupled Yield (Choon et al.
                % 2013)
                score = mes.score * mes.objV;
            else
                % conventional scoring by product yield
                score = mes.score;
            end
            tmpScore = round(score, roundN);

            % keep recrod of scores to rank
            if tmpScore > 0 && (tmpScore - parentScore) > dv_cutoff
                orig_scores(ii) = score;
            end

            % determine better scores and assign points
            if tmpScore > 0 && isempty(find(scoreHistory == tmpScore, 1))
                % only if this is a new score
                if (tmpScore - score0) > dv_cutoff

                    % this RID set has better score than score0
                    % 1 point will be awarded, unless reaches pmax
                    points(ii) = true;
                    scores(ii) = tmpScore;

                    str = sprintf('%u,', testRids);
```

```
                    fprintf(fc.Value, [str '%4.4f\n'], score);

                else

                    if pointFail < 0
                        % failed and not enough points
                        % this RID set lineage ends here
                        toExpand(ii) = false;
                    else
                        % this RID set has enough points to be expanded
                        % but 1 point will be deducted for failing
                        scores(ii) = score0;
                    end

                end
            elseif pointFail < 0
                toExpand(ii) = false;
            end

            if isMaxKO
                toExpand(ii) = false;
            end

        else
            toExpand(ii) = false;
        end

    end

    clear fc

    %% push the RID-sets-to-expand into stacks
    n = sum(toExpand);
    if n > 0
        % sort results
        [~,i] = sortrows(scores, -1);
        ridx2 = ridx(i);
        scores = scores(i);
        toExpand = toExpand(i);
        points = points(i);

        % push to stacks
        tmp = [mat2cell([repmat(rid0, n, 1) ridx2(toExpand)], ones(n,1),
ncol); ridStack];
        ridStack = tmp;
        tmp = [scores(toExpand); scoreStack];
        scoreStack = tmp;
        tmp = [zeros(n,1); pointStack];
        tmp(points(toExpand)) = point0 + reward;
        tmp(~points(toExpand)) = point0 - 1;
        pointStack = tmp;

        % adding to scoreHistory
        tmp = union(scoreHistory, scores);
        scoreHistory = tmp;
```

```
        clear tmp

        % determine bpcy rate if necessary
        if BPCYacceptance > 0
            bpcyRate = scores(1) / maxbpcy;
        end
    end

    %% update best strategies
    [~, rank] = sortrows(orig_scores, -1);
    j = 1;
    while j <= param.Nbests && j <= Nridx1
        if orig_scores(rank(j)) > 0
            betterThan = strategies.best.scores < orig_scores(rank(j));
            if sum(betterThan) > 0
                tmp = [strategies.best.scores(~betterThan);
orig_scores(rank(j)); strategies.best.scores(betterThan(1:end-1))];
                strategies.best.scores = tmp;
                ridtmp = {[rid0 ridx(rank(j))]};
                tmp = [strategies.best.rids(~betterThan); ridtmp;
strategies.best.rids(betterThan(1:end-1))];
                strategies.best.rids = tmp;
            end
        end
        j = j + 1;
    end

end

strategies.time = toc(time0);
strategiesToRemove = strategies.best.scores == 0;
strategies.best.scores(strategiesToRemove) = [];
strategies.best.rids(strategiesToRemove) = [];

%% print elapsed time
f = fopen(fn, 'At');
fprintf(f, 'Elapsed time: %4.1f minutes\n', strategies.time/60);
fclose(f);

%% print best strategies
f = fopen([param.savedir param.fn 'bestScores.csv'], 'w');
fprintf(f, 'Srategies,Scores\n');
for i = 1:length(strategies.best.scores)
    fprintf(f, '%u,', strategies.best.rids{i});
    fprintf(f, '%4.4f\n', strategies.best.scores(i));
end
fclose(f);
```

Function (NR-Ox): nodeRewardOxP

```
%% Predicting engineering strategies using a reward algorithm
% **Utilizes parallel computing**
```

```
% Author: Jiun Yen
% Date: 2016.11.27
% Version: 2017.2.4
%
% This is an exhaustive search with a reward system to reduce search space
% All members of the RID array starts with p point(s) (allowing p
% additional failed attempt); if successful, the set receive an additional
% point.
% This search is depth-first
%
%
% Input:
%   param.
%       m -model
%       bio_rid - RID of the biomass equation (growth)
%       target_rid - RID of the target reaction to maximize yield
%       excludeRids - RIDs to exclude from mod
%       BPCYacceptance - biomass-product coupled yield (Choon et al. 2013)
%                      - percent of max theoretical BPCY to accept solution
%                      and terminate run
%       Nbests - number of best strategies to keep
%       v_cutoff - flux cutoff for zero
%       dv_cutoff - minimum degree of improvement each iteration
%       ratioHiCutoff - high cutoff for flux ratio
%       ratioLoCutoff - low cutoff for flux ratio
%       hiCoeff - coefficient to calc increase of flux ratio
%           r_hi = r + (1-r)*hiCoeff
%       loCoeff - coefficient to calc decrease of flux ratio
%           r_lo = r * loCoeff
%       p0 - point(s) to start with
%       pmax - max number of points
%       maxMod - max number of modifications
%       forceBest - whether new score must be the best known score
%       savedir - folder to save results in
%       fn - file name header
%       mskoption - option for mosek optimizer

function [strategies, param] = nodeRewardOxP(param)

time0 = tic;
strategies = [];

%% initialize nodeEval parameters
nodeParam0.m = decomposeS(param.m);
nodeParam0.m.c(:) = 0;
nodeParam0.m.c(param.bio_rid) = 1;
nodeParam0.bio_rid = param.bio_rid;
nodeParam0.target_rid = param.target_rid;
nodeParam0.scoreByBPCY = logical(param.BPCYacceptance);
nodeParam0.max = 1;
nodeParam0.v_cutoff = param.v_cutoff;
nodeParam0.mskoption = param.mskoption;

% calc dhiCoeff
param.dhiCoeff = 1 - param.hiCoeff;
```

```
% find maximum theoretical BPCY
maxbpcy = findMaxBPCY(param);

% constants
dv_cutoff = param.dv_cutoff;
roundN = abs(ceil(log(dv_cutoff)/log(10))) + 1;
fn = [param.savedir param.fn 'scoreEvolution.csv'];
f = fopen(fn, 'w');
fprintf(f, 'NID sets,Scores\n');
fclose(f);
BPCYacceptance = param.BPCYacceptance;
Nnodes0 = size(nodeParam0.m.S,1);

%% initialize primary output variables
strategies.best.scores = zeros(param.Nbests, 1);
strategies.best.nids = cell(param.Nbests, 1);
strategies.best.rids = cell(param.Nbests, 1);
strategies.best.ratios = cell(param.Nbests, 1);

%% Perform tests

nidStack = [];
ridStack = [];
ratioStack = [];
scoreStack = [];
pointStack = [];
scoreHistory = [];
bpcyRate = 0;

start = true;
% while ~isempty(nidStack) || start
while bpcyRate < BPCYacceptance && (~isempty(nidStack) || start)

    % pop the stacks
    if start

        start = false;
        nodeMod0.nids = [];
        nodeMod0.rids = [];
        nodeMod0.rs = [];
        m0 = nodeParam0.m;
        [~, bV] = m_linprogP(m0, 1, 0, param.mskoption);
        obj = find(m0.c);
        m0.lb(obj) = bV;
        m0.ub(obj) = bV;
        m0.c(:) = 0;
        m0.c(param.target_rid) = 1;
        [~, tV] = m_linprogP(m0, 0, 0, param.mskoption);
        if BPCYacceptance > 0
            score0 = bV * tV;
        else
            score0 = tV;
        end
        parentScore = score0;
        point0 = param.p0;
```

```
    ncol = 1;

else

    nodeMod0.nids = nidStack{1};
    nodeMod0.rids = ridStack{1};
    nodeMod0.rs = ratioStack{1};
    nidStack(1) = [];
    ridStack(1) = [];
    ratioStack(1) = [];
    if param.forceBest
        % force new score to be better than all current known score
        score0 = max(scoreHistory);
    else
        % new score just needs to be better than parent's
        score0 = scoreStack(1);
    end
    parentScore = scoreStack(1);
    scoreStack(1) = [];
    point0 = pointStack(1);
    pointStack(1) = [];
    ncol = length(nodeMod0.nids) + 1;

end

% precalculate points if strategy fails
pointFail = point0 - 1;

% install current nodeMod
nodeParam = installFR(nodeParam0, nodeMod0);

% determine reward value
isMaxKO = length(nodeMod0.nids) >= param.maxMod-1;
if point0 >= param.pmax
    reward = 0;
else
    reward = 1;
end

% solve current model to get FR
m = nodeParam.m;
v = m_linprogP(m, 1, 0, param.mskoption);
m.lb(param.bio_rid) = v(param.bio_rid);
m.c(:) = 0;
m.c(param.target_rid) = 1;
v = m_linprogP(m, 0, 0, param.mskoption);
v(abs(v) < nodeParam.v_cutoff) = 0;
v(param.target_rid) = 0;
fr = genFR(nodeParam.m, v);

% determine new nodes to test
nids = find(fr.isNode);
nids(nids > Nnodes0) = [];
nidx = setdiff(nids, nodeMod0.nids);
nidc = parallel.pool.Constant(nidx);
```

```
Nnidx = length(nidx);

% populate all test cases
r = full(fr.r);
r2 = r;
r2(nodeMod0.nids,:) = [];   % remove nodes that are already in the set
Ntests = length(find(r2))*2;
clear r2
nids1 = zeros(Ntests, 1);
rids1 = zeros(Ntests, 1);
ratios1 = zeros(Ntests, 1);
k = 0;
for i = 1:Nnidx
    ridstmp = setdiff(find(r(nidx(i),:)), param.excludeRids);
    for j = 1:length(ridstmp)
        rtmp = r(nidx(i),ridstmp(j));

        if rtmp < param.ratioHiCutoff
            % OX
            k = k + 1;
            nids1(k) = nidx(i);
            rids1(k) = ridstmp(j);
            ratios1(k) = param.hiCoeff + param.dhiCoeff * rtmp;
        end

        if rtmp > param.ratioLoCutoff
            % KD
            k = k + 1;
            nids1(k) = nidx(i);
            rids1(k) = ridstmp(j);
            ratios1(k) = param.loCoeff * rtmp;
        end

    end
end
Ntests = k;
k = k + 1;
nids1(k:end) = [];
rids1(k:end) = [];
ratios1(k:end) = [];

% set file print
fc = parallel.pool.Constant(@() fopen(fn, 'At'), @fclose);

%% solve LP problems with parallel cores
toExpand = true(Ntests, 1);
points = false(Ntests, 1);
scores = zeros(Ntests, 1);
orig_scores = zeros(Ntests, 1);

fprintf('Evaluating %u tests\n', Ntests);
fprintf('%u,', nodeMod0.nids);
fprintf('\n');
fprintf('%u,', nodeMod0.rids);
fprintf('\n');
```

```
    fprintf('%4.4f,', nodeMod0.rs);
    fprintf('\n');

    parfor ii = 1:Ntests

        % Evaluate node
        nodeMod = nodeMod0;
        nodeMod.nids = [nodeMod0.nids nids1(ii)];
        nodeMod.rids = [nodeMod0.rids rids1(ii)];
        nodeMod.rs = [nodeMod0.rs ratios1(ii)];
        nodeParam1 = installFR(nodeParam, nodeMod);
        [~, bV] = m_linprogP(nodeParam1.m, 1, 0, nodeParam1.mskoption);
        nodeParam1.m.lb(nodeParam1.bio_rid) = bV;
        nodeParam1.m.c(:) = 0;
        nodeParam1.m.c(nodeParam1.target_rid) = 1;
        [~, tV] = m_linprogP(nodeParam1.m, 0, 0, nodeParam1.mskoption);

        % Evalute score and assign reward
        if bV > 0

            if BPCYacceptance > 0
                score = bV * tV;
            else
                score = tV;
            end
            tmpScore = round(score, roundN);

            % keep recrod of scores to rank
            if tmpScore > 0 && (tmpScore - parentScore) > dv_cutoff
                orig_scores(ii) = score;
            end

            if tmpScore > 0 && isempty(find(scoreHistory == tmpScore, 1))
                % only if this is a new score
                if (tmpScore - score0) > dv_cutoff

                    % this RID set has better score than score0
                    % 1 point will be awarded, unless reaches pmax
                    points(ii) = true;
                    scores(ii) = tmpScore;

                    tmp = reshape([nodeMod.nids; nodeMod.rids;
nodeMod.rs],length(nodeMod.nids)*3,1);
                    str = sprintf('%u|%u|%4.4f,', tmp);
                    fprintf(fc.Value, [str '%4.4f\n'], score);

                else

                    if pointFail < 0
                        % failed and not enough points
                        % this RID set lineage ends here
                        toExpand(ii) = false;
                    else
                        % this RID set has enough points to be expanded
                        % but 1 point will be deducted for failing
```

```
                    scores(ii) = score0;
                end

            end

        elseif pointFail < 0
            toExpand(ii) = false;
        end

        if isMaxKO
            toExpand(ii) = false;
        end

    else
        toExpand(ii) = false;
    end

end

clear fc

%% push the RID-sets-to-expand into stacks
n = sum(toExpand);
fprintf('%u to expand\n', n);
if n > 0
    % sort results
    [~,i] = sortrows(scores, -1);
    nids2 = nids1(i);
    rids2 = rids1(i);
    ratios2 = ratios1(i);
    scores = scores(i);
    toExpand = toExpand(i);
    points = points(i);

    % push to stacks
    tmp = [mat2cell([repmat(nodeMod0.nids, n, 1) nids2(toExpand)],
ones(n,1), ncol); nidStack];
    nidStack = tmp;
    tmp = [mat2cell([repmat(nodeMod0.rids, n, 1) rids2(toExpand)],
ones(n,1), ncol); ridStack];
    ridStack = tmp;
    tmp = [mat2cell([repmat(nodeMod0.rs, n, 1) ratios2(toExpand)],
ones(n,1), ncol); ratioStack];
    ratioStack = tmp;
    tmp = [scores(toExpand); scoreStack];
    scoreStack = tmp;
    tmp = [zeros(n,1); pointStack];
    tmp(points(toExpand)) = point0 + reward;
    tmp(~points(toExpand)) = point0 - 1;
    pointStack = tmp;

    % adding to scoreHistory
    tmp = union(scoreHistory, scores);
    scoreHistory = tmp;
```

```
        clear tmp

        % determine bpcy rate if necessary
        if BPCYacceptance > 0
            bpcyRate = max(scores) / maxbpcy;
            fprintf('BPCY: %u\n', max(scores));
            fprintf('BPCYrate: %u\n', bpcyRate);
        end

    end

    %% update best strategies
    [~, rank] = sortrows(orig_scores, -1);
    j = 1;
    while j <= param.Nbests && j <= Ntests
        if orig_scores(rank(j)) > 0
            betterThan = strategies.best.scores < orig_scores(rank(j));
            if sum(betterThan) > 0
                tmp = [strategies.best.scores(~betterThan);
orig_scores(rank(j)); strategies.best.scores(betterThan(1:end-1))];
                strategies.best.scores = tmp;
                tmp = [strategies.best.nids(~betterThan); {[nodeMod0.nids
nids1(rank(j))]}; strategies.best.nids(betterThan(1:end-1))];
                strategies.best.nids = tmp;
                tmp = [strategies.best.rids(~betterThan); {[nodeMod0.rids
rids1(rank(j))]}; strategies.best.rids(betterThan(1:end-1))];
                strategies.best.rids = tmp;
                tmp = [strategies.best.ratios(~betterThan); {[nodeMod0.rs
ratios1(rank(j))]}; strategies.best.ratios(betterThan(1:end-1))];
                strategies.best.ratios = tmp;
            end
        end
        j = j + 1;
    end

    %% print best strategies
    strategiesToRemove = strategies.best.scores == 0;
    strategies.best.scores(strategiesToRemove) = [];
    strategies.best.nids(strategiesToRemove) = [];
    strategies.best.rids(strategiesToRemove) = [];
    strategies.best.ratios(strategiesToRemove) = [];
    f = fopen([param.savedir param.fn 'bestScores.csv'], 'w');
    fprintf(f, 'Srategies,Scores\n');
    for i = 1:length(strategies.best.scores)
        tmp = reshape([strategies.best.nids{i}; strategies.best.rids{i};
strategies.best.ratios{i}],length(strategies.best.nids{i})*3,1);
        fprintf(f, '%u|%u|%4.4f,', tmp);
        fprintf(f, '%4.4f\n', strategies.best.scores(i));
    end
    fclose(f);

end

strategies.time = toc(time0);
```

```
%% print elapsed time
f = fopen(fn, 'At');
fprintf(f, 'Elapsed time: %4.1f minutes\n', strategies.time/60);
fclose(f);
```

Function: addMes

```
%% Expand metabolic engineering strategy - use with initMes()
% Author: Jiun Yen
% Date: 2016.11.12
% Version: 2016.11.12

function mes = addMes(mes, rids, ubs, lbs, score, objV)

if ~isempty(intersect(mes.rids, rids))
    return
end

mes.rids = [mes.rids rids];

if nargin < 3
    mes.ubs = [mes.ubs zeros(1, length(rids))];
    mes.lbs = [mes.lbs zeros(1, length(rids))];
else
    mes.ubs = [mes.ubs ubs];
    mes.lbs = [mes.lbs lbs];
end

if nargin < 5
    mes.score = 0;
    mes.evaluated = false;
else
    mes.score = score;
    mes.evaluated = true;
end

if nargin < 6
    mes.objV = 0;
else
    mes.objV = objV;
end
```

Function: constraintKO

```
%% Constrain flux(s) of specified reaction(s) to zero

function m = constraintKO(m, rid)
m.lb(rid) = 0;
m.ub(rid) = 0;
```

Function: decomposeS

```
%% Break down original COBRA model so there are only positive v

function m = decomposeS(m)

Nrxns = size(m.S,2);

% S matrix decomposition
m.S = sparse([full(m.S) -full(m.S)]);

% boundary decomposition
ub1 = zeros(Nrxns,1);
ub2 = zeros(Nrxns,1);
lb1 = zeros(Nrxns,1);
lb2 = zeros(Nrxns,1);
ub1(m.ub > 0) = m.ub(m.ub > 0);
lb1(m.lb > 0) = m.lb(m.lb > 0);
ub2(m.lb < 0) = abs(m.lb(m.lb < 0));
lb2(m.ub < 0) = abs(m.ub(m.ub < 0));
m.ub = [ub1;ub2];
m.lb = [lb1;lb2];

% reassign obj function
obj = find(m.c);
m.c = zeros(Nrxns * 2, 1);
m.c(obj) = 1;
```

Function: findMaxBPCY

```
%% Calculate the maximum theoretical BPCY

function [maxbpcy, bV, tV] = findMaxBPCY(param)

tV = fminsearch(@bpcy, 0, [], param);
m = param.m;
m.lb(param.target_rid) = tV;
[~, bV] = m_linprogP(m, 1, 0, param.mskoption);
maxbpcy = tV * bV;

function s = bpcy(t0, param)

m = param.m;
m.lb(param.target_rid) = t0;
[~, b] = m_linprogP(m, 1, 0, param.mskoption);
s = 1000 - b * t0;
```

Function: fvaP

```
% Flux variability analysis (for the parallel driver)
```

```
% Author: Jiun Yen
% Version: 2016.12.18
% Description: Analysis of flux variability as described in Appendix A of
%   Ranganathan et al. The upper and lower flux range is determined through
%   iterative optimization of each reaction in model m.
% Input:
%   m - a Cobra model
% Output:
%   vmax,vmin - matrix of upper and lower flux ranges of all reactions in m

function [vmax,vmin,t] = fvaP(m, option, set)
tic

% final optimal obj solution
s = m_linprogP(m, 1, 0, option);
obj = find(m.c);
m.ub(obj) = floor(s(obj)*10^6)/10^6;
m.lb(obj) = m.ub(obj);

% identify set
Nrxns = size(m.S,2);
m.c(:) = 0;
if nargin < 3 || isempty(set)
    set = 1:Nrxns;
    set(obj) = [];
    Nset = Nrxns - 1;
else
    Nset = length(set);
end

% initialize output
vmax = zeros(Nrxns,1);
vmin = zeros(Nrxns,1);

% assign obj solution
vmax(obj) = s(obj);
vmin(obj) = s(obj);

% perform FVA in parallel
vmax_sub = zeros(Nset,1);
vmin_sub = zeros(Nset,1);
parfor i = 1:Nset

    m1 = m;

    m1.c(set(i)) = 1;

    m1.ub(set(i)) = 1000;
    m1.lb(set(i)) = -1000;

    % find max
    s = m_linprogP(m1,1,0,option);
    vmax_sub(i) = s(set(i));

    % find min
```

```
    s = m_linprogP(m1,0,0,option);
    vmin_sub(i) = s(set(i));

end

% assign solutions
vmax(set) = vmax_sub;
vmin(set) = vmin_sub;

t = toc;
```

Function: genFR

```
% Search for nodes with flux solution (v), and generate flux ratios
% v should be cleaned-up (free of noisy values that are essentially zeros)

function fr = genFR(m, v)

fr.S = m.S;
fr.v = v;

[Nmets, Nrxns] = size(m.S);

% multiply S by v and avoid summation
r_mat = m.S .* repmat(v', Nmets, 1);

% identify consumption fluxes (negative fluxes)
negs = r_mat < 0;

% remove production fluxes (positive fluxes)
r_mat = r_mat .* negs;

% find nodes, mets with 2 or more competing fluxes (more than 1)
isNode = full(sum(negs, 2) > 1);

% remove non-nodes (set to 0)
r_mat = r_mat .* repmat(isNode, 1, Nrxns);

% calculate total flux producing each met (node)
v_sums = sum(r_mat, 2);

% cheat: to prevent NaN, since non-mets will have 0 fluxes, 0/1 = 0
v_sums(v_sums == 0) = 1;

% finalize flux ratio matrices by dividing each node by its total flux
r_mat = r_mat ./ repmat(v_sums, 1, Nrxns);

% find rxns that are in node
rids = find(sum(logical(r_mat),1));

fr.isNode = isNode;
```

```
fr.r = r_mat;
fr.rids = rids;
```

Function: geneCodingRxns

```
%% Return only reactions associated to a gene
function rids = geneCodingRxns(m)
rids = find(sum(full(logical(m.rxnGeneMat)),2));
```

Function: initMes

```
%% Special set data structure for metabolic engineering strategies (mes)
% Author: Jiun Yen
% Date: 2016.11.12
% Version: 2016.11.12
% Structure:
%   rids: rids in this set
%   ubs: flux upper bound assigned to the rids (same size vector as set)
%   lbs: flux lower bound assigned to the rids (same size vector as set)
%   score: score of this set
%   objV: flux of the objective function

function mes = initMes()

mes.rids = [];
mes.ubs = [];
mes.lbs = [];
mes.score = 0;
mes.objV = 0;
mes.evaluated = false;
```

Function: installFR

```
%% Install flux ratio into COBRA model
% *** MUST perform decomposeS first!!***
% nodeMods.
%   nids - node IDs
%   rids - reaction IDs
%   rs - ratios for RIDs

function node = installFR(node, nodeMods)

m = node.m;
Nmets = size(m.S,1);

for n = 1:length(nodeMods.nids)

    rid0 = nodeMods.rids(n);
    rids = setdiff(find(m.S(nodeMods.nids(n),:)), rid0);
```

```
    m.S(Nmets + n, rid0) = nodeMods.rs(n) - 2;
    m.S(Nmets + n, rids) = nodeMods.rs(n);

end

node.m = m;
```

Function: m_linprogP

```
% m_linprog for parallel programming

function [sol, objx] = m_linprogP(m, opt, minGlobal,option,beq)

Nrxns = size(m.S,2);
sol = zeros(Nrxns,1);
obj = logical(m.c);
objx = 0;
tol = 1e-10;

if nargin < 5
    beq = zeros(size(m.S,1),1);
end
if nargin < 4
    option = mskoptimset('');
    option = mskoptimset(option,'Simplex','primal');
end

if minGlobal

    % initialize variables
    x0 = zeros(2*Nrxns,1);
    Aeq = [full(m.S) -full(m.S)];

    % boundary decomposition
    ub1 = zeros(Nrxns,1);
    ub2 = zeros(Nrxns,1);
    lb1 = zeros(Nrxns,1);
    lb2 = zeros(Nrxns,1);
    ub1(m.ub > 0) = m.ub(m.ub > 0);
    lb1(m.lb > 0) = m.lb(m.lb > 0);
    ub2(m.lb < 0) = abs(m.lb(m.lb < 0));
    lb2(m.ub < 0) = abs(m.ub(m.ub < 0));
    ub = [ub1;ub2];
    lb = [lb1;lb2];

    % max or min obj func
    if opt > 0
        f = [-m.c;m.c];
    else
        f = [m.c;-m.c];
    end
    flogical = logical(f);
```

```
    % find solution satisfy objective function
    sol0 = linprog(f,[],[],Aeq,beq,lb,ub,x0,option);

    % continue only if optimal satisfy objective
    if sum(sol0(flogical) < lb(flogical)) ~= 0 || sum(sol0(flogical) >
ub(flogical)) ~= 0
        return
    end

    % reverse objective for global minimization
    lb(flogical) = sol0(flogical);
    ub(flogical) = sol0(flogical);
    f = double(~f);

    % find solution for global minimization
    sol1 = linprog(f,[],[],Aeq,beq,lb,ub,x0,option);

    % recompose original solution array
    if sum(sol1 < lb-tol) ~= 0 || sum(sol1 > ub+tol) ~= 0
        return
    end
    sol = sol1(1:Nrxns) - sol1(Nrxns+1:end);
    objx = sol(obj);

else

    % initialize variables
    x0 = zeros(Nrxns,1);
    Aeq = full(m.S);
    f = m.c;
    lb = m.lb;
    ub = m.ub;

    % max or min obj func
    if opt > 0
        f = -f;
    end

    % find solution satisfy objective function
    soltmp = linprog(f,[],[],Aeq,beq,lb,ub,x0,option);
    if sum(soltmp < lb-tol) ~= 0 || sum(soltmp > ub+tol) ~= 0
        return
    end
    sol = soltmp;
    objx = sol(obj);

end
```

Function: addPathway

```
%% Add pathway (multiple reactions/compounds) to m

function m = addPathway(m, cid_add, rid_add, S_add)
```

```
[Nmets0, Nrxns0] = size(m.S);

for i = 1:length(cid_add.cids)
    m.mets(cid_add.cids(i)) = cid_add.mets(i);
    m.metNames(cid_add.cids(i)) = cid_add.metNames(i);
end

Nrxns = length(rid_add.rids);
for i = 1:Nrxns
    m.rxns(rid_add.rids(i)) = rid_add.rxns(i);
    m.rxnNames(rid_add.rids(i)) = rid_add.rxnNames(i);
    m.subSystems{rid_add.rids(i)} = '';
end

m.lb(end+1:end+Nrxns) = 0;
m.ub(end+1:end+Nrxns) = 1000;
m.c(end+1:end+Nrxns) = 0;

for i = 1:length(S_add.s)
    m.S(S_add.cids(i),S_add.rids(i)) = S_add.s(i);
end
```

Function: mesEvalP

```
%% Evaluate MES by performing FBA (parallel computing version)
% assign score as flux of objective function unless otherwise specified in
% parameter (param)
% Author: Jiun Yen
% Date: 2016.11.12
% Version: 2016.11.12
% Input:
%   m: model (COBRA format)
%   mes: MES structure built by initMes()
%   param: parameters
%       max - BOOLEAN, to max (1) or min (0) objective function
%       target - rid of flux used to determine the score (default: obj)
%       modifier - equation to calculate score (default: [])
%       v_cutoff - cutoff for zero
%       mskoption - option for mosek optimizer

function mes = mesEvalP(m, mes, param)

m.ub(mes.rids) = mes.ubs;
m.lb(mes.rids) = mes.lbs;

if isfield(param, 'minGlobal') && param.minGlobal
    [mes.v, objV] = m_linprogP(m, 1, 1, param.mskoption);

    if abs(objV) > param.v_cutoff
        mes.objV = objV;

        if isempty(param.modifier)
```

```
            score = mes.v(param.target);
        else
            score = param.modifier(mes.v(param.target));
        end

        if abs(score) > param.v_cutoff
            mes.score = score;
        end
    end
else
    [~, objV] = m_linprogP(m, param.max, 0, param.mskoption);

    if abs(objV) > param.v_cutoff
        mes.objV = objV;

        if ~param.target || param.target == find(m.c)
            if isempty(param.modifier)
                score = objV;
            else
                score = param.modifier(objV);
            end
        else
            obj = find(m.c);
            m.c(:) = 0;
            m.c(param.target) = 1;
            m.ub(obj) = objV;
            m.lb(obj) = objV;
            [~, objV] = m_linprogP(m, 0, 0, param.mskoption);
            if isempty(param.modifier)
                score = objV;
            else
                score = param.modifier(objV);
            end
        end

        if abs(score) > param.v_cutoff
            mes.score = score;
        end
    end
end

mes.evaluated = true;
```

2) DRIVER FILE

Driver for NR-Knock

```
%% Driver - Node-Reward Knock
% All numbered sections (1-5) are required

%% (1) path to core codes and MESEK solver on HPC

%% (2) Load model and biomass compositions and other criteria
```

```
%% (3) Specify parameters

% Target compound for engineering
% leave cid empty if rid (the exchange reaction of target cpd) is known
target_cid = [];
target_rid = [];

% flux cutoff - below this is zero
v_cutoff = 0.000000001;

%% (4) Additional model modifications

%% Automated steps - REQUIRES MODEL EXIST AS m

% Assesss target compound
% this create a new reaction for target compound if target_rid is empty
if isempty(target_rid) || logical(target_rid) || ~isBioComp(m, target_cid,
config.bio_rid, 1)
    [m, target_rid] = addExchangeRxn(m, target_cid, 0, 1000, 0);
end

% keep only basic model components
m1.S = m.S;
m1.ub = m.ub;
m1.lb = m.lb;
m1.c = m.c;
m1.rxnGeneMat = m.rxnGeneMat;

% setup Mosek solver option
option = mskoptimset('');
option = mskoptimset(option,'Simplex','primal');

%% (5) Setup primary argument - param

% RIDs to exclude (i.e. exchange, transport)
excludeRids = [];

% setup param
param.m = m1;
param.target_rid = target_rid;
param.bio_rid = [];             % ID of the biomass equation
param.BPCYacceptance = 0.98;
param.Nbests = 40;
param.searchLethals = false;
param.excludeRids = excludeRids;
param.v_cutoff = v_cutoff;
param.hi_cutoff = 150;
param.dv_cutoff = 0.01;
param.p0 = 0;
param.pmax = 1;
param.komax = 4;
param.forceBest = true;
param.savedir = 'NRKnockP';     % folder name
param.fn = '/NRKnockP_out_';    % output header
param.mskoption = option;
```

```
%% Perform NR-Knock
% make folder if does not exist
mkdir(param.savedir);
clearvars -except param

% run nodeRewardKnockP
[strategies, param_out] = nodeRewardKnockP(param);
clearvars -except param param_out strategies

% save workspace
save([param.savedir param.fn 'results']);
```

Driver for NR-Ox

```
%% Driver - Node-Reward Ox
% All numbered sections (1-5) are required

%% (1) path to core codes and MESEK solver on HPC

%% (2) Load model and biomass compositions and other criteria

%% (3) Specify parameters

% Target compound for engineering
% leave cid empty if rid (the exchange reaction of target cpd) is known
target_cid = [];
target_rid = [];

% flux cutoff - below this is zero
v_cutoff = 0.000000001;

%% (4) Additional model modifications

%% Automated steps - REQUIRES MODEL EXIST AS m

% Assesss target compound
% this create a new reaction for target compound if target_rid is empty
if isempty(target_rid) || logical(target_rid) || ~isBioComp(m, target_cid,
config.bio_rid, 1)
    [m, target_rid] = addExchangeRxn(m, target_cid, 0, 1000, 0);
end

% keep only basic model components
m1.S = m.S;
m1.ub = m.ub;
m1.lb = m.lb;
m1.c = m.c;
m1.rxnGeneMat = m.rxnGeneMat;

% setup Mosek solver option
option = mskoptimset('');
```

```
option = mskoptimset(option,'Simplex','primal');

%% (5) Setup primary argument - param

% RIDs to exclude (i.e. exchange, transport)
excludeRids = [];

% setup param
param.m = m1;
param.bio_rid = [];              % ID of the biomass equation
param.target_rid = target_rid;
param.excludeRids = excludeRids;
param.BPCYacceptance = 0.98;
param.Nbests = 40;
param.v_cutoff = v_cutoff;
param.dv_cutoff = 0.01;
param.ratioHiCutoff = 0.8;
param.ratioLoCutoff = 0.1;
param.hiCoeff = 0.9;
param.loCoeff = 0.01;
param.p0 = 0;
param.pmax = 1;
param.maxMod = 4;
param.forceBest = true;
param.savedir = 'NROxP';         % folder name
param.fn = '/NROxP_out_';        % output header
param.mskoption = option;

%% Perform NR-Knock
% make folder if does not exist
mkdir(param.savedir);
clearvars -except param

% run nodeRewardOxP
[strategies, param_out] = nodeRewardOxP(param);
clearvars -except param param_out strategies

% save workspace
save([param.savedir param.fn 'results']);
```

3) EXAMPLE – DESIGNS FOR OVERPRODUCTION OF BDO IN E. COLI

NR-Knock example

```
%% Driver - Node-Reward Knock
% Example: Predict metabolic engineering strategies to overproduce
% 1,4-butanediol in E. coli
% All numbered sections (1-5) are required

%% (1) path to core codes and MESEK solver on HPC
addpath('/home/qksilver/dragonstooth/codes/core')
addpath('/home/qksilver/dragonstooth/mosek/7/toolbox/r2013a/')
```

```
%% (2) Load model and biomass compositions and other criteria
load('20161103_Ec_iAF1260f1_C1000R2To01_N50000_bComps.mat')
load('Ec_iAF1260_flux1_exchanges.mat')
load('Ec_iAF1260_flux1_lethal_KOs.mat')
clearvars -except starti Nsamples bComps config lethal_rids exchange_rids
Vopts m0

%% (3) Specify parameters

% Target compound for engineering
% leave cid empty if rid (the exchange reaction of target cpd) is known
target_cid = 1672;  % 1,4-butanediol
target_rid = [];

% flux cutoff - below this is zero
v_cutoff = 0.000000001;

%% (4) Additional model modifications
% Add Lee 2011 BDO biosynthetic pathway
load('Lee_BDO_pathway.mat')
m = addPathway(m0,cid_add,rid_add,S_add);
clear cid_add rid_add S_add

% Anaerobic condition (O2 exchange -> rid 933)
m.lb(933) = 0;

% Glucose uptake rate
m.lb(849) = -20;

%% Automated steps - REQUIRES MODEL EXIST AS m

% Assesss target compound
% this create a new reaction for target compound if target_rid is empty
if isempty(target_rid) || logical(target_rid) || ~isBioComp(m, target_cid,
config.bio_rid, 1)
    [m, target_rid] = addExchangeRxn(m, target_cid, 0, 1000, 0);
end

% keep only basic model components
m1.S = m.S;
m1.ub = m.ub;
m1.lb = m.lb;
m1.c = m.c;
m1.rxnGeneMat = m.rxnGeneMat;

% setup Mosek solver option
option = mskoptimset('');
option = mskoptimset(option,'Simplex','primal');

%% (5) Setup primary argument - param

% RIDs to exclude (i.e. exchange, transport)
excludeRids = union(lethal_rids, exchange_rids);
```

```
% setup param
param.m = m1;
param.target_rid = target_rid;
param.bio_rid = 1005;           % ID of the biomass equation
param.BPCYacceptance = 0.98;
param.Nbests = 40;
param.searchLethals = false;
param.excludeRids = excludeRids;
param.v_cutoff = v_cutoff;
param.hi_cutoff = 150;
param.dv_cutoff = 0.01;
param.p0 = 0;
param.pmax = 1;
param.komax = 4;
param.forceBest = true;
param.savedir = 'NRKnockP';     % folder name
param.fn = '/NRKnockP_out_';    % output header
param.mskoption = option;

%% Perform NR-Knock
% make folder if does not exist
mkdir(param.savedir);
clearvars -except param

% run nodeRewardKnockP
[strategies, param_out] = nodeRewardKnockP(param);
clearvars -except param param_out strategies

% save workspace
save([param.savedir param.fn 'results']);
```

NR-Ox example

```
%% Driver - Node-Reward Ox
% Example: Predict metabolic engineering strategies to overproduce
% 1,4-butanediol in E. coli
% All numbered sections (1-5) are required

%% (1) path to core codes and MESEK solver on HPC
addpath('/home/qksilver/dragonstooth/codes/core')
addpath('/home/qksilver/dragonstooth/mosek/7/toolbox/r2013a/')

%% (2) Load model and biomass compositions and other criteria
load('20161103_Ec_iAF1260f1_C1000R2To01_N50000_bComps.mat')
load('Ec_iAF1260_flux1_exchanges.mat')
load('Ec_iAF1260_flux1_lethal_KOs.mat')
clearvars -except starti Nsamples bComps config lethal_rids exchange_rids
Vopts m0

%% (3) Specify parameters

% Target compound for engineering
% leave cid empty if rid (the exchange reaction of target cpd) is known
```

```
target_cid = 1672;  % 1,4-butanediol
target_rid = [];

% flux cutoff - below this is zero
v_cutoff = 0.000000001;

%% (4) Additional model modifications
% Add Lee 2011 BDO biosynthetic pathway
load('Lee_BDO_pathway.mat')
m = addPathway(m0,cid_add,rid_add,S_add);
clear cid_add rid_add S_add

% Anaerobic condition (O2 exchange -> rid 933)
m.lb(933) = 0;

% Glucose uptake rate
m.lb(849) = -20;

%% Automated steps - REQUIRES MODEL EXIST AS m

% Assesss target compound
% this create a new reaction for target compound if target_rid is empty
if isempty(target_rid) || logical(target_rid) || ~isBioComp(m, target_cid,
config.bio_rid, 1)
    [m, target_rid] = addExchangeRxn(m, target_cid, 0, 1000, 0);
end

% keep only basic model components
m1.S = m.S;
m1.ub = m.ub;
m1.lb = m.lb;
m1.c = m.c;
m1.rxnGeneMat = m.rxnGeneMat;

% setup Mosek solver option
option = mskoptimset('');
option = mskoptimset(option,'Simplex','primal');

%% (5) Setup primary argument - param

% RIDs to exclude (i.e. exchange, transport)
excludeRids = union(lethal_rids, exchange_rids);
excludeRids = union(excludeRids, setdiff(1:2382, geneCodingRxns(m1)));

% setup param
param.m = m1;
param.bio_rid = 1005;           % ID of the biomass equation
param.target_rid = target_rid;
param.excludeRids = excludeRids;
param.BPCYacceptance = 0.95;
param.Nbests = 40;
param.v_cutoff = v_cutoff;
param.dv_cutoff = 0.01;
param.ratioHiCutoff = 0.8;
param.ratioLoCutoff = 0.1;
```

```
param.hiCoeff = 0.9;
param.loCoeff = 0.01;
param.p0 = 0;
param.pmax = 1;
param.maxMod = 4;
param.forceBest = true;
param.savedir = 'NROxP';          % folder name
param.fn = '/NROxP_out_';         % output header
param.mskoption = option;

%% Perform NR-Knock
% make folder if does not exist
mkdir(param.savedir);
clearvars -except param

% run nodeRewardOxP
[strategies, param_out] = nodeRewardOxP(param);
clearvars -except param param_out strategies

% save workspace
save([param.savedir param.fn 'results']);
```

C. MATLAB CODES FOR TO SIMULATE GROWTH AND STARCH METABOLISM

There are two sections: 1) core codes and their dependencies and 2) the drivers. Drivers are setup to solve and simulate growth and starch as described in Materials and Methods.

CORE CODES AND DEPENDENCIES

Function: solveGrowth

```
%% Solve for growth from t0 to t1
% Author: Jiun Yen
% Date: 2016.12.11
% Version: 2016.12.11
%
% Dependencies: stackGEM(), updateBiomassEq()
% Input
%   param
%       m - COBRA model
%       obj - objective of optimization (default to B1 if empty)
%       opt - max(1) or min(0) obj
%       B0p - rid of biomass equation
%       B0V - initial biomass (mgDW)
%       Nrxns - number of rxns in model
%       Nmets - number of mets in model
%       rids - reactions to constrain to v for model from t0->t1
%       ubs - flux constraints for rids
%       lbs - flux constraints for rids
%       bcomp - biomass composition compound information struct
%       biomass0
%           sid - SIDs of mets in stoichiometric matrix of m
%           data - data on biomass composition at t0
%       biomass1
%           sid - SIDs of mets in stoichiometric matrix of m
%           data - data on biomass composition at t1
%       ymets - SIDs of independent biomass compounds (i.e. starch - 1477)
%       yubs - ub constraints for dymol
%       ylbs - lb constraints for dymol
%       minGlobal - to minimize global flux (yes - 1, no - 0)

function [s, info] = solveGrowth(param)

%% Define constants
B0p = param.B0p;
B1p = param.Nrxns + 1;
B0 = B1p + 1;
B1 = B0 + 1;
Nymets = length(param.ymets);
ymol0 = linspace(B1+1,B1+Nymets,Nymets);
tmp = max(ymol0);
ymol1 = linspace(tmp+1,tmp+Nymets,Nymets);
tmp = max(ymol1);
dymol = linspace(tmp+1,tmp+Nymets,Nymets);
```

```
%% Set up model
% Generate new model
m = param.m;
m.ub(param.rids) = param.ubs;
m.lb(param.rids) = param.lbs;

% extract quantitative data and molecular weights on independent biomass
compounds
ymetData = zeros(Nymets, 1);
ymw = zeros(Nymets, 1);
for i = 1:Nymets
    ymetData(i) = param.biomass0.data(param.biomass0.sids == param.ymets(i));
    ymw(i) = param.bcomp.mw(param.bcomp.sids == param.ymets(i));
end

% Remove independent biomass compounds from biomass0 and biomass1
[~,tmp] = intersect(param.biomass0.sids, param.ymets);
param.biomass0.data(tmp) = 0;
[~,tmp] = intersect(param.biomass1.sids, param.ymets);
param.biomass1.data(tmp) = 0;

% Update biomass equation with biomass composition data at t0 and t1
m = updateBiomassEq(m, B0p, [], param.biomass0.data, false,
param.biomass0.sids);
m = updateBiomassEq(m, B1p, [], -param.biomass1.data, false,
param.biomass1.sids);

% add biomass-ymets relationships
tmp = param.Nmets + 1;
m.S(tmp, B0) = 1;
m.S(tmp, B0p) = -1;
m.S(tmp, ymol0) = -ymw;
m = updateMet(m, tmp, 'B0-ymass_t1');
m = updateRxn(m, B0, 'Biomass_t0', 0, 1000, 0);
m = updateRxn(m, B0p, 'Biomass_noYmets_t0', 0, 1000, 0);
tmp = param.Nmets + 2;
m.S(tmp, B1) = 1;
m.S(tmp, B1p) = -1;
m.S(tmp, ymol1) = -ymw;
m = updateMet(m, tmp, 'B1-ymass_t1');
m = updateRxn(m, B1, 'Biomass_t1', 0, 1000, 0);
m = updateRxn(m, B1p, 'Biomass_noYmets_t1', 0, 1000, 0);

% associate ymets
for i = 1:Nymets
    tmp = tmp + 1;
    m.S(tmp, ymol0(i)) = -1;
    m.S(tmp, ymol1(i)) = 1;
    m.S(tmp, dymol(i)) = -1;
    m = updateMet(m, tmp, ['change_of_' m.metNames{param.ymets(i)}]);
    m = updateRxn(m, ymol0(i), [m.metNames{param.ymets(i)} '_t0'], 0, 1000,
0);
    m = updateRxn(m, ymol1(i), [m.metNames{param.ymets(i)} '_t1'], 0, 1000,
0);
```

```
    m = updateRxn(m, dymol(i), ['change_of_' m.metNames{param.ymets(i)}],
param.ylbs(i), param.yubs(i), 0);
    m.S(param.ymets(i),dymol(i)) = -1;
end

% Constraint biomass and ymet quantity at t0
m.lb(B0) = param.B0V;
m.ub(B0) = m.lb(B0);
m.lb(B1) = m.lb(B0);
m.lb(ymol0) = ymetData .* param.B0V;
m.ub(ymol0) = m.lb(ymol0);

%% Solve with FBA
m.c(:) = 0;
if ~isempty(param.obj)
    m.c(param.obj) = 1;
    s = m_linprog(m, param.opt, 0);
    m.ub(param.obj) = s(param.obj);
    m.lb(param.obj) = m.ub(param.obj);
    m.c(param.obj) = 0;
end
m.c(B1p) = 1;
s = m_linprog(m, 1, param.minGlobal);

%% construct info
info.m = m;
info.const.B0 = B0;
info.const.B1 = B1;
info.const.B0p = B0p;
info.const.B1p = B1p;
info.const.ymol0 = ymol0;
info.const.ymol1 = ymol1;
info.const.dymol = dymol;

%% Nested function to update compound name
function m = updateMet(m, sid, name)
m.mets(sid) = {name};
m.metNames(sid) = {name};

%% Nested function to update reaction name, set bounds, and c
function m = updateRxn(m, rid, name, lb, ub, c)
m.rxns(rid) = {name};
m.rxnNames(rid) = {name};
m.lb(rid) = lb;
m.ub(rid) = ub;
m.c(rid) = c;
```

Function: sim24hgrowth

```
function [sols, info] = sim24hgrowth(param, timeParam)

% h = waitbar(0, 'initializing');

% solve first step, t0 -> t1
```

```
[s, info] = solveGrowth(param);
sols = zeros(length(s), timeParam.steps-1);
sols(:,1) = s;
% waitbar(1/timeParam.steps, h, 'solving');

for t = 2:timeParam.steps-1
    param.B0V = s(info.const.B1);

    % update starch level
    param.biomass0.data(25) = s(info.const.ymol1) / s(info.const.B1);
    param.biomass1 = param.biomass0;

    %% do this in multiple stages of a 24-hour day
    % Stage 1: Starting from 1 PM (t0 -> t1), there is light until 10 PM (9h)
    % Stage 2: Light's off from 10 PM to 6 AM (8h)
    % Stage 3: Light's back on from 6 AM to 1 PM (7h)
    tmp = rem(timeParam.range(t),24);
    if tmp ~= 0 && tmp < timeParam.eod
        % Stage 1
        % update CO2 and light uptake
        param.rids = [48 63];
        param.ubs = param.ubs .* param.beta_d;
        tmp_ubs = param.ubs;
        param.ylbs = param.ylb_a * s(info.const.B1p);

        param.opt = 1;

    elseif tmp ~= 0 && tmp < timeParam.eon
        % Stage 2
        % update growth condition and starch utilization
        param.rids = [48 63 4];
        param.ubs = [0 0 1000];
        param.lbs = [-1000 0 -1000];
        param.ylbs = param.ylb_b * s(info.const.B1p);

        param.opt = 0;

        % update leaf size
        tmp_ubs = tmp_ubs .* param.beta_n;

    else
        % Stage 3
        param.rids = [48 63];
        tmp_ubs = tmp_ubs .* param.beta_d;
        param.ubs = tmp_ubs;
        param.lbs = [0 0];
        param.ylbs = param.ylb_c * s(info.const.B1p);

        param.opt = 1;
    end

    [s, info] = solveGrowth(param);
    if abs(s(info.const.B0p)) < 0.0000001
        break;
    end
```

```
    if tmp < timeParam.eon && abs(s(info.const.ymol1)) < 0.0000001
        break;
    end
    sols(:,t) = s;

%     waitbar(t/timeParam.steps, h, 'solving');
end

sols = sols ./ param.scale;

% close(h)
```

Function: m_linprog

```
function sol = m_linprog(m, opt, minGlobal,beq)

if nargin < 4
    beq = zeros(size(m.S,1),1);
end

% option = optimset('Algorithm','interior-point');
option = mskoptimset('');
option = mskoptimset(option,'Simplex','primal');

Nrxns = size(m.S,2);
sol = zeros(Nrxns,1);
tol = 1e-9;

if minGlobal
    % initialize variables
    x0 = zeros(2*Nrxns,1);
    Aeq = [full(m.S) -full(m.S)];

    % boundary decomposition
    ub1 = zeros(Nrxns,1);
    ub2 = zeros(Nrxns,1);
    lb1 = zeros(Nrxns,1);
    lb2 = zeros(Nrxns,1);
    ub1(m.ub > 0) = m.ub(m.ub > 0);
    lb1(m.lb > 0) = m.lb(m.lb > 0);
    ub2(m.lb < 0) = abs(m.lb(m.lb < 0));
    lb2(m.ub < 0) = abs(m.ub(m.ub < 0));
    ub = [ub1;ub2];
    lb = [lb1;lb2];

    % max or min obj func
    if opt > 0
        f = [-m.c;m.c];
    else
        f = [m.c;-m.c];
    end
    flogical = logical(f);

    % find solution satisfy objective function
```

```
    sol0 = linprog(f,[],[],Aeq,beq,lb,ub,x0,option);

    % continue only if optimal satisfy objective
    if sum(sol0(flogical) < lb(flogical)) ~= 0 || sum(sol0(flogical) >
ub(flogical)) ~= 0
        return
    end

    % reverse objective for global minimization
    lb(flogical) = sol0(flogical);
    ub(flogical) = sol0(flogical);
    f = double(~f);

    % find solution for global minimization
    sol1 = linprog(f,[],[],Aeq,beq,lb,ub,x0,option);

    % recompose original solution array
    sol = sol1(1:Nrxns) - sol1(Nrxns+1:end);
    if sum(sol1 < lb-tol) ~= 0 || sum(sol1 > ub+tol) ~= 0
        sol = zeros(Nrxns,1);
%         fprintf('Solution is unreliable - out of bounds.\n');
    end
else
    % initialize variables
    x0 = zeros(Nrxns,1);
    Aeq = full(m.S);
    f = m.c;
    lb = m.lb;
    ub = m.ub;

    % max or min obj func
    if opt > 0
        f = -f;
    end

    % find solution satisfy objective function
    sol = linprog(f,[],[],Aeq,beq,lb,ub,x0,option);
    if sum(sol < lb-tol) ~= 0 || sum(sol > ub+tol) ~= 0
        sol = zeros(Nrxns,1);
%         fprintf('Solution is unreliable - out of bounds.\n');
    end
end
```

Function: getdStarch

```
function score = getdStarch(vars0, param, timeParam)

% vars = fminsearch(@findMaxTstarch, vars0, [], param, timeParam);
%
% % assign variables
% param.ylbs = vars(1) * (param.B0V -
param.bcomp.mw(36)*param.biomass0.data(25)*param.B0V);
% param.ylb_a = vars(1);
% param.ylb_b = vars(2);
```

```
% param.ylb_c = vars(3);
%
% % simulate
% [sols, info] = sim24hgrowth(param, timeParam);
%
% % generate output
% totalGrowth = sols(info.const.B0,:);
% totalGrowth(end+1) = sols(info.const.B1,end);
% starch_umol = sols(info.const.ymol0,:);
% starch_umol(end+1) = sols(info.const.ymol1,end);
% starch_conc = starch_umol ./ totalGrowth;
% dStarch = abs(starch_conc(end) - param.biomass0.data(25));
%
% function tStarch = findMaxTstarch(vars0, param, timeParam)

% assign variables
param.ylbs = vars0(1) * (param.B0V -
param.bcomp.mw(36)*param.biomass0.data(25)*param.B0V);
param.ylb_a = vars0(1);
param.ylb_b = vars0(2);
param.ylb_c = vars0(3);

% simulate
[sols, info] = sim24hgrowth(param, timeParam);

%% Plots
totalGrowth = sols(info.const.B0,:);
totalGrowth(end+1) = sols(info.const.B1,end);
noStarchGrowth = sols(info.const.B0p,:);
noStarchGrowth(end+1) = sols(info.const.B1p,end);
starch_umol = sols(info.const.ymol0,:);
starch_umol(end+1) = sols(info.const.ymol1,end);
starch_conc = starch_umol ./ totalGrowth;
% dStarch_umol_B0p_dt =
sols(info.const.dymol,:)./sols(info.const.B0p,:)/timeParam.dt;
tStarch = abs(starch_conc(timeParam.eodi) - starch_conc(timeParam.eoni));
dStarch = abs(starch_conc(end) - param.biomass0.data(25));

clf('reset')
subplot(2,1,1)
hold on
plot(timeParam.range, totalGrowth,'.')
plot(timeParam.range, noStarchGrowth,'.')
ylabel('Total biomass, mgDW')
subplot(2,1,2)
plot(timeParam.range, starch_conc,'.')
ylabel('Starch conc., \mumol/mgDW')
xlabel('time after 1 PM, h')
% subplot(3,1,3)
% plot(timeParam.range(1:end-1), dStarch_umol_B0p_dt,'.')
clc
fprintf('EOD starch: %4.3f umol/mgDW\n', starch_conc(timeParam.eodi));
fprintf('EON starch: %4.3f umol/mgDW\n', starch_conc(timeParam.eoni));
fprintf('transitory starch: %4.3f umol/mgDW\n', tStarch);
fprintf('t0 starch: %4.3f umol/mgDW\n', starch_conc(1));
fprintf('t1 starch: %4.3f umol/mgDW\n', starch_conc(end));
```

```
fprintf('t0 Biomass: %4.3f mg\n', totalGrowth(1));
fprintf('t1 Biomass: %4.3f mg\n', totalGrowth(end));
pause(0.00001)

% generate output
score = -(tStarch - 10 * dStarch - 20 * abs(vars0(1) - vars0(3)));
% score = abs(tStarch - 0.559) + dStarch + abs(vars0(1) - vars0(3));
```

Function: updateBiomassEq

```
%% Assign new concentrations to compounds in the biomass equation

function m = updateBiomassEq(m,r,bcomp,data,isKids,ids)

if isKids
    for i = 1:length(ids)
        m.S(bcomp.sids(bcomp.kids == ids(i)),r) = data(i);
    end
else
    m.S(ids,r) = data;
end
```

Function: calcMassRatio

```
% determine how much of the total biomass is covered with data

function massRatio = calcMassRatio(bcomp,data,kids,sids)

massRatio = 0;
if nargin < 4 || isempty(sids)
    for i = 1:length(kids)
        massRatio = massRatio + bcomp.mw(bcomp.kids == kids(i))*data(i);
    end
else
    for i = 1:length(sids)
        massRatio = massRatio + bcomp.mw(bcomp.sids == sids(i))*data(i);
    end
end
```

DRIVER FILES

Driver to solve and simulate results in Figure 6 and Figure 7

```
%% Driver to Ler solveGrowth
% for pre-flowering
% only for pre-flowering stage because of the use of RGR for CO2 exchange

clear
clc
```

```
load('20161214_LerHA_model_constraints.mat')
load('rgr_means.mat')

%% modify glycosyl data - scale up to xx%
targetMass = 0.416;   % 416 ug/mgDW
metids = false(32,1);
mws = zeros(32,1);
for i = 1:32
    j = bcomp.sids == sids(i);
    n = bcomp.group(j);
    if strcmp(n, 'Hemicellulose') || strcmp(n, 'Pectin')
        metids(i) = true;
        mws(i) = bcomp.mw(j);
    end
end
mws = mws(metids);
totalMass = mws' * data(metids,:);
massRatio = targetMass ./ totalMass;
data(metids,:) = data(metids,:) .* repmat(massRatio,sum(metids),1);

clear targetMass metids mws i j n totalMass massRatio

%% scale
scale = 0.1;

%% Which data set (Ler pre, HA pre, Ler post, HA post)

% test Ler-pre
id = 1;
    sampleParam.a = 0.06;
    sampleParam.b = -0.08;
    sampleParam.c = 0.06;
    sampleParam.rgr_d = rgrs(id,1);
    sampleParam.rgr_n = rgrs(id,2);
    sampleParam.starch0 = 0.662;
    sampleParam.B0 = 2.01;
    sampleParam.obj = [];

% test HA-pre
% id = 2;
%     sampleParam.a = 0.06;
%     sampleParam.b = -0.08;
%     sampleParam.c = 0.06;
%     sampleParam.rgr_d = rgrs(id,1);
%     sampleParam.rgr_n = rgrs(id,2);
%     sampleParam.starch0 = 0.843;
%     sampleParam.B0 = 1.51;
%     sampleParam.obj = [];

% test Ler-post
% id = 3;
%     sampleParam.a = 0.05;
%     sampleParam.b = -0.08;
%     sampleParam.c = 0.05;
%     sampleParam.rgr_d = rgrs(id,1);
```

```
%     sampleParam.rgr_n = rgrs(id,2);
%     sampleParam.starch0 = 0.3735;
%     sampleParam.B0 = 121.45;
%     sampleParam.obj = [];

% test HA-post
% id = 4;
%     sampleParam.a = 0.04;
%     sampleParam.b = -0.05;
%     sampleParam.c = 0.04;
%     sampleParam.rgr_d = rgrs(id,1);
%     sampleParam.rgr_n = rgrs(id,2);
%     sampleParam.starch0 = 0.2346;
%     sampleParam.B0 = 221.73;
%     sampleParam.obj = [];

toSolve = 1;

c_ratio = calcMassRatio(bcomp,data(:,id),[],sids);

%% Set up time parameters
% hour
timeParam.frame0 = 0;
timeParam.frame1 = 24;
timeParam.stepsPerHr = 2;
timeParam.eod = 9;
timeParam.eon = 17;
timeParam.dframe = timeParam.frame1 - timeParam.frame0;
timeParam.steps = timeParam.dframe * timeParam.stepsPerHr + 1;
timeParam.dt = timeParam.dframe / (timeParam.steps-1);
timeParam.range = linspace(timeParam.frame0, timeParam.frame1,
timeParam.steps);
timeParam.eodi = timeParam.eod * timeParam.stepsPerHr + 1;
timeParam.eoni = timeParam.eon * timeParam.stepsPerHr + 1;

%% rebuild only basic model
m0.rxns = Ler.rxns;
m0.rxnNames = Ler.rxnNames;
m0.mets = Ler.mets;
m0.metNames = Ler.metNames;
m0.S = Ler.S;
m0.lb = Ler.lb;
m0.ub = Ler.ub;
m0.c = Ler.c;

%% calculate photon and CO2 uptake as umol/plant/time interval
% values at t0
hv_t0 = light * 3600 / m2a_ratio(id) * timeParam.dt * scale * rosDWMean(id);
co2_t0 = co2Mean(id) * 3600 / m2a_ratio(id) * timeParam.dt * scale *
rosDWMean(id) * 1;

%% Define necessary coefficients
beta_d = (sampleParam.rgr_d + 1)^(1/(16*timeParam.stepsPerHr));
beta_n = (sampleParam.rgr_n + 1)^(1/(8*timeParam.stepsPerHr));
ylb_a = sampleParam.a*timeParam.dt;
```

```
ylb_b = sampleParam.b*timeParam.dt;
ylb_c = sampleParam.c*timeParam.dt;

%% Set up param
% at t0
param.m = m0;
param.obj = sampleParam.obj;
param.opt = 1;
param.B0p = 47;
param.B0V = sampleParam.B0 * scale;
[param.Nmets, param.Nrxns] = size(param.m.S);
param.rids = [48 63];
param.ubs = [co2_t0 hv_t0];
param.lbs = [0 0];
param.bcomp = bcomp;
param.biomass0.sids = sids;
param.biomass0.data = data(:,id);
tmp = setdiff(1:32, 25);
param.biomass0.data(tmp) = data(tmp, id) * param.B0V / (param.B0V -
param.bcomp.mw(36)*data(25,id)*param.B0V);
param.biomass1 = param.biomass0;
param.biomass0.data(25) = sampleParam.starch0;
param.biomass1 = param.biomass0;
param.ymets = 1477;     % starch_biomass
param.yubs = 1000;
param.ylbs = ylb_a * (param.B0V -
param.bcomp.mw(36)*param.biomass0.data(25)*param.B0V);
param.minGlobal = 0;
param.scale = scale;
param.beta_d = beta_d;
param.beta_n = beta_n;
param.ylb_a = ylb_a;
param.ylb_b = ylb_b;
param.ylb_c = ylb_c;

%% Solve growth over time frame

if toSolve
    vars0 = [param.ylb_a param.ylb_b param.ylb_c];
    vars = fminsearch(@getdStarch, vars0, [], param, timeParam);
    sampleParam.a = vars(1)/timeParam.dt;
    sampleParam.b = vars(2)/timeParam.dt;
    sampleParam.c = vars(3)/timeParam.dt;
    while sum(abs(vars - vars0) > 0.0001) > 0
        vars0 = vars;
        vars = fminsearch(@getdStarch, vars0, [], param, timeParam);
        sampleParam.a = vars(1)/timeParam.dt;
        sampleParam.b = vars(2)/timeParam.dt;
        sampleParam.c = vars(3)/timeParam.dt;
    end
else
    [sols, info] = sim24hgrowth(param, timeParam);
end

clearvars -except toSolve param timeParam rosDWMean sols info c_ratio
sampleParam
```

```
%% Plots
if ~toSolve
    totalGrowth = sols(info.const.B0,:);
    totalGrowth(end+1) = sols(info.const.B1,end);
    noStarchGrowth = sols(info.const.B0p,:);
    noStarchGrowth(end+1) = sols(info.const.B1p,end);
    starch_umol = sols(info.const.ymol0,:);
    starch_umol(end+1) = sols(info.const.ymol1,end);
    starch_conc = starch_umol ./ totalGrowth;

    figure
    set(0,'DefaultAxesFontName', 'Times New Roman')
    set(0,'DefaultAxesFontSize', 12)
    subplot(2,1,1)
    hold on
    rectangle('position',[timeParam.eod 1 8
2],'edgecolor','none','facecolor',[0.8 0.8 0.8])
    plot(timeParam.range, totalGrowth,'-','linewidth',1)
    plot(timeParam.range, noStarchGrowth,'-','linewidth',1)
    xlim([0 24])
    ylabel([{'Total biomass'} {'mgDW'}])
    h = legend([{'Total DW'},{'Non-starch
DW'}],'location','northwest','fontsize',12);
    pos = get(h,'position');
    pos(1) = pos(1) + 0.025;
    pos(2) = pos(2) + 0.025;
    set(h,'position',pos);
    legend boxoff
    subplot(2,1,2)
    hold on
    rectangle('position',[timeParam.eod 0 8
1.5],'edgecolor','none','facecolor',[0.8 0.8 0.8])
    plot(timeParam.range, starch_conc,'-','linewidth',1)
    xlim([0 24])
    ylabel([{'Starch conc.'} {'\mumol/mgDW'}])
    xlabel('time after 1 PM, h')
    fig = gcf;
    fig.Position = [0 0 500 400];
    fig.PaperUnits = 'inches';
    fig.PaperPosition = [0 0 5 4];
    fprintf('max starch: %4.3f umol/mgDW\n', max(starch_conc));
    fprintf('min starch: %4.3f umol/mgDW\n', min(starch_conc));
    fprintf('transitory starch: %4.3f umol/mgDW\n', (max(starch_conc) -
min(starch_conc)));
    fprintf('t0 starch: %4.3f umol/mgDW\n', starch_conc(1));
    fprintf('t1 starch: %4.3f umol/mgDW\n', starch_conc(end));
    fprintf('t0 Biomass: %4.3f mg\n', totalGrowth(1));
    fprintf('t1 Biomass: %4.3f mg\n', totalGrowth(end));
end
```

Driver to simulate results in Figure 8

```
%% Driver to simulate growth and starch changes over 24 hours
```

```
% for pre-flowering and post-flowering after refinement

clear
clc
load('20161214_LerHA_model_constraints.mat')
load('rgr_means.mat')

%% modify glycosyl data - scale up to xx%
targetMass = 0.416;   % 416 ug/mgDW
metids = false(32,1);
mws = zeros(32,1);
for i = 1:32
    j = bcomp.sids == sids(i);
    n = bcomp.group(j);
    if strcmp(n, 'Hemicellulose') || strcmp(n, 'Pectin')
        metids(i) = true;
        mws(i) = bcomp.mw(j);
    end
end
mws = mws(metids);
totalMass = mws' * data(metids,:);
massRatio = targetMass ./ totalMass;
data(metids,:) = data(metids,:) .* repmat(massRatio,sum(metids),1);

clear targetMass metids mws i j n totalMass massRatio

%% scale
scale = 0.1;

%% Which data set (Ler pre, HA pre, Ler post, HA post)

% test Ler-pre
id = 1;
    sampleParam.a = 0.0653;
    sampleParam.b = -0.091;
    sampleParam.c = 0.064;
    sampleParam.rgr_d = rgrs(id,1);
    sampleParam.rgr_n = rgrs(id,2);
    sampleParam.starch0 = 0.662;
    sampleParam.B0 = 2.01;
    sampleParam.obj = [];

% test HA-pre
% id = 2;
%     sampleParam.a = 0.0897;
%     sampleParam.b = -0.106;
%     sampleParam.c = 0.0731;
%     sampleParam.rgr_d = rgrs(id,1);
%     sampleParam.rgr_n = rgrs(id,2);
%     sampleParam.starch0 = 0.843;
%     sampleParam.B0 = 1.505;
%     sampleParam.obj = [];

% test Ler-post
% id = 3;
```

```
%     sampleParam.a = 0.046;
%     sampleParam.b = -0.069;
%     sampleParam.c = 0.0451;
%     sampleParam.rgr_d = rgrs(id,1);
%     sampleParam.rgr_n = rgrs(id,2);
%     sampleParam.starch0 = 0.3735;
%     sampleParam.B0 = 121.45;
%     sampleParam.obj = [];

% test HA-post
% id = 4;
%     sampleParam.a = 0.0287;
%     sampleParam.b = -0.0425;
%     sampleParam.c = 0.0286;
%     sampleParam.rgr_d = rgrs(id,1);
%     sampleParam.rgr_n = rgrs(id,2);
%     sampleParam.starch0 = 0.2346;
%     sampleParam.B0 = 221.73;
%     sampleParam.obj = [];

toSolve = 1;

c_ratio = calcMassRatio(bcomp,data(:,id),[],sids);

%% Set up time parameters
% hour
timeParam.frame0 = 0;
timeParam.frame1 = 24;
timeParam.stepsPerHr = 2;
timeParam.eod = 9;
timeParam.eon = 17;
timeParam.dframe = timeParam.frame1 - timeParam.frame0;
timeParam.steps = timeParam.dframe * timeParam.stepsPerHr + 1;
timeParam.dt = timeParam.dframe / timeParam.steps;
timeParam.range = linspace(timeParam.frame0, timeParam.frame1,
timeParam.steps);
timeParam.eodi = timeParam.eod * timeParam.stepsPerHr + 1;
timeParam.eoni = timeParam.eon * timeParam.stepsPerHr + 1;

%% rebuild only basic model
m0.rxns = Ler.rxns;
m0.rxnNames = Ler.rxnNames;
m0.mets = Ler.mets;
m0.metNames = Ler.metNames;
m0.S = Ler.S;
m0.lb = Ler.lb;
m0.ub = Ler.ub;
m0.c = Ler.c;

%% calculate photon and CO2 uptake as umol/plant/time interval
% values at t0
hv_t0 = light * 3600 / m2a_ratio(id) * timeParam.dt * scale * rosDWMean(id);
co2_t0 = co2Mean(id) * 3600 / m2a_ratio(id) * timeParam.dt * scale *
rosDWMean(id) * 1;
```

```
%% Define necessary coefficients
beta_d = (sampleParam.rgr_d + 1)^(1/(16*timeParam.stepsPerHr));
beta_n = (sampleParam.rgr_n + 1)^(1/(8*timeParam.stepsPerHr));
ylb_a = sampleParam.a*timeParam.dt;
ylb_b = sampleParam.b*timeParam.dt;
ylb_c = sampleParam.c*timeParam.dt;

%% Set up param
% at t0
param.m = m0;
param.obj = sampleParam.obj;
param.opt = 1;
param.B0p = 47;
param.B0V = sampleParam.B0 * scale;
[param.Nmets, param.Nrxns] = size(param.m.S);
param.rids = [48 63];
param.ubs = [co2_t0 hv_t0];
param.lbs = [0 0];
param.bcomp = bcomp;
param.biomass0.sids = sids;
param.biomass0.data = data(:,id);
tmp = setdiff(1:32, 25);
param.biomass0.data(tmp) = data(tmp, id) * param.B0V / (param.B0V -
param.bcomp.mw(36)*data(25,id)*param.B0V);
param.biomass1 = param.biomass0;
param.biomass0.data(25) = sampleParam.starch0;
param.biomass1 = param.biomass0;
param.ymets = 1477;     % starch_biomass
param.yubs = 1000;
param.ylbs = ylb_a * (param.B0V -
param.bcomp.mw(36)*param.biomass0.data(25)*param.B0V);
param.minGlobal = 1;
param.scale = scale;
param.beta_d = beta_d;
param.beta_n = beta_n;
param.ylb_a = ylb_a;
param.ylb_b = ylb_b;
param.ylb_c = ylb_c;

%% Solve growth over time frame

if toSolve
    vars0 = [param.ylb_a param.ylb_b param.ylb_c];
    vars = fminsearch(@getdStarch, vars0, [], param, timeParam);
    sampleParam.a = vars(1)/timeParam.dt;
    sampleParam.b = vars(2)/timeParam.dt;
    sampleParam.c = vars(3)/timeParam.dt;
    while sum(abs(vars - vars0) > 0.0001) > 0
        vars0 = vars;
        vars = fminsearch(@getdStarch, vars0, [], param, timeParam);
        sampleParam.a = vars(1)/timeParam.dt;
        sampleParam.b = vars(2)/timeParam.dt;
        sampleParam.c = vars(3)/timeParam.dt;
    end
else
    [sols, info] = sim24hgrowth(param, timeParam);
```

```
end

clearvars -except toSolve param timeParam rosDWMean sols info c_ratio
sampleParam

%% Plots
if ~toSolve
    totalGrowth = sols(info.const.B0,:);
    totalGrowth(end+1) = sols(info.const.B1,end);
    noStarchGrowth = sols(info.const.B0p,:);
    noStarchGrowth(end+1) = sols(info.const.B1p,end);
    starch_umol = sols(info.const.ymol0,:);
    starch_umol(end+1) = sols(info.const.ymol1,end);
    starch_conc = starch_umol ./ totalGrowth;

    figure
    set(0,'DefaultAxesFontName', 'Times New Roman')
    set(0,'DefaultAxesFontSize', 12)
    subplot(2,1,1)
    hold on
    rectangle('position',[timeParam.eod 1 8
2],'edgecolor','none','facecolor',[0.8 0.8 0.8])
    plot(timeParam.range, totalGrowth,'-','linewidth',1)
    plot(timeParam.range, noStarchGrowth,'-','linewidth',1)
    xlim([0 24])
    ylabel([{'Total biomass'} {'mgDW'}])
    h = legend([{'Total DW'},{'Non-starch
DW'}],'location','northwest','fontsize',12);
    pos = get(h,'position');
    pos(1) = pos(1) + 0.025;
    pos(2) = pos(2) + 0.025;
    set(h,'position',pos);
    legend boxoff
    subplot(2,1,2)
    hold on
    rectangle('position',[timeParam.eod 0 8
1.5],'edgecolor','none','facecolor',[0.8 0.8 0.8])
    plot(timeParam.range, starch_conc,'-','linewidth',1)
    xlim([0 24])
    ylabel([{'Starch conc.'} {'\mumol/mgDW'}])
    xlabel('time after 1 PM, h')
    fig = gcf;
    fig.Position = [0 0 500 400];
    fig.PaperUnits = 'inches';
    fig.PaperPosition = [0 0 5 4];
    fprintf('max starch: %4.3f umol/mgDW\n', max(starch_conc));
    fprintf('min starch: %4.3f umol/mgDW\n', min(starch_conc));
    fprintf('transitory starch: %4.3f umol/mgDW\n', (max(starch_conc) -
min(starch_conc)));
    fprintf('t0 starch: %4.3f umol/mgDW\n', starch_conc(1));
    fprintf('t1 starch: %4.3f umol/mgDW\n', starch_conc(end));
    fprintf('t0 Biomass: %4.3f mg\n', totalGrowth(1));
    fprintf('t1 Biomass: %4.3f mg\n', totalGrowth(end));
end
```

Driver to simulate results in Figure 9

```
%% Driver to simulate growth and starch changes over 24 hours
% for pre-flowering and post-flowering after refinement
% solving and simulating for stages in-between pre- and post-flowering

clear
clc
load('20161214_LerHA_model_constraints.mat')
load('rgr_means.mat')

%% modify glycosyl data - scale up to xx%
targetMass = 0.416;   % 416 ug/mgDW
metids = false(32,1);
mws = zeros(32,1);
for i = 1:32
    j = bcomp.sids == sids(i);
    n = bcomp.group(j);
    if strcmp(n, 'Hemicellulose') || strcmp(n, 'Pectin')
        metids(i) = true;
        mws(i) = bcomp.mw(j);
    end
end
mws = mws(metids);
totalMass = mws' * data(metids,:);
massRatio = targetMass ./ totalMass;
data(metids,:) = data(metids,:) .* repmat(massRatio,sum(metids),1);

clear targetMass metids mws i j n totalMass massRatio

%% scale
scale = 0.1;

%% Which data set (Ler pre, HA pre, Ler post, HA post)

t0 = 14;
t1 = 42;

tx = 35;
coef = (tx - t0)/(t1 - t0);

allstarch0 = [0.4941 0.5879 0.3556 0.1901];
allB0 = rosDWMean;

% test
id = 2;
    co2x = (co2Mean(id+2) - co2Mean(id)) * coef + co2Mean(id);
    m2a_ratiox = (m2a_ratio(id+2) - m2a_ratio(id)) * coef + m2a_ratio(id);
    B0x = (allB0(id+2) - allB0(id)) * coef + allB0(id);
    sampleParam.a = 0.0403;
    sampleParam.b = -0.0662;
    sampleParam.c = 0.0448;
    sampleParam.rgr_d = rgrs(id,1);
    sampleParam.rgr_n = rgrs(id,2);
```

```
    sampleParam.starch0 = (allstarch0(id+2) - allstarch0(id)) * coef +
allstarch0(id);
    sampleParam.B0 = B0x;
    sampleParam.obj = [];

toSolve = 0;

datax = (data(:,id+2) - data(:,id)) * coef + data(:,id);
c_ratio = calcMassRatio(bcomp,datax,[],sids);

%% Set up time parameters
% hour
timeParam.frame0 = 0;
timeParam.frame1 = 24;
timeParam.stepsPerHr = 2;
timeParam.eod = 9;
timeParam.eon = 17;
timeParam.dframe = timeParam.frame1 - timeParam.frame0;
timeParam.steps = timeParam.dframe * timeParam.stepsPerHr + 1;
timeParam.dt = timeParam.dframe / timeParam.steps;
timeParam.range = linspace(timeParam.frame0, timeParam.frame1,
timeParam.steps);
timeParam.eodi = timeParam.eod * timeParam.stepsPerHr + 1;
timeParam.eoni = timeParam.eon * timeParam.stepsPerHr + 1;

%% rebuild only basic model
m0.rxns = Ler.rxns;
m0.rxnNames = Ler.rxnNames;
m0.mets = Ler.mets;
m0.metNames = Ler.metNames;
m0.S = Ler.S;
m0.lb = Ler.lb;
m0.ub = Ler.ub;
m0.c = Ler.c;

%% calculate photon and CO2 uptake as umol/plant/time interval
% values at t0
hv_t0 = light * 3600 / m2a_ratiox * timeParam.dt * scale * B0x;
co2_t0 = co2x * 3600 / m2a_ratiox * timeParam.dt * scale * B0x * 1;

%% Define necessary coefficients
beta_d = (sampleParam.rgr_d + 1)^(1/(16*timeParam.stepsPerHr));
beta_n = (sampleParam.rgr_n + 1)^(1/(8*timeParam.stepsPerHr));
ylb_a = sampleParam.a*timeParam.dt;
ylb_b = sampleParam.b*timeParam.dt;
ylb_c = sampleParam.c*timeParam.dt;

%% Set up param
% at t0
param.m = m0;
param.obj = sampleParam.obj;
param.opt = 1;
param.B0p = 47;
param.B0V = sampleParam.B0 * scale;
[param.Nmets, param.Nrxns] = size(param.m.S);
```

```
param.rids = [48 63];
param.ubs = [co2_t0 hv_t0];
param.lbs = [0 0];
param.bcomp = bcomp;
param.biomass0.sids = sids;
param.biomass0.data = datax;
tmp = setdiff(1:32, 25);
param.biomass0.data(tmp) = datax(tmp) * param.B0V / (param.B0V -
param.bcomp.mw(36)*datax(25)*param.B0V);
param.biomass1 = param.biomass0;
param.biomass0.data(25) = sampleParam.starch0;
param.biomass1 = param.biomass0;
param.ymets = 1477;     % starch_biomass
param.yubs = 1000;
param.ylbs = ylb_a * (param.B0V -
param.bcomp.mw(36)*param.biomass0.data(25)*param.B0V);
param.minGlobal = 0;
param.scale = scale;
param.beta_d = beta_d;
param.beta_n = beta_n;
param.ylb_a = ylb_a;
param.ylb_b = ylb_b;
param.ylb_c = ylb_c;

%% Solve growth over time frame

if toSolve
    vars0 = [param.ylb_a param.ylb_b param.ylb_c];
    vars = fminsearch(@getdStarch, vars0, [], param, timeParam);
    sampleParam.a = vars(1)/timeParam.dt;
    sampleParam.b = vars(2)/timeParam.dt;
    sampleParam.c = vars(3)/timeParam.dt;
    while sum(abs(vars - vars0) > 0.0001) > 0
        vars0 = vars;
        vars = fminsearch(@getdStarch, vars0, [], param, timeParam);
        sampleParam.a = vars(1)/timeParam.dt;
        sampleParam.b = vars(2)/timeParam.dt;
        sampleParam.c = vars(3)/timeParam.dt;
    end
else
    [sols, info] = sim24hgrowth(param, timeParam);
end

clearvars -except toSolve param timeParam rosDWMean sols info c_ratio
sampleParam

%% Plots
if ~toSolve
    totalGrowth = sols(info.const.B0,:);
    totalGrowth(end+1) = sols(info.const.B1,end);
    noStarchGrowth = sols(info.const.B0p,:);
    noStarchGrowth(end+1) = sols(info.const.B1p,end);
    starch_umol = sols(info.const.ymol0,:);
    starch_umol(end+1) = sols(info.const.ymol1,end);
    starch_conc = starch_umol ./ totalGrowth;
```

```
    figure
    set(0,'DefaultAxesFontName', 'Times New Roman')
    set(0,'DefaultAxesFontSize', 12)
    subplot(2,1,1)
    hold on
    rectangle('position',[timeParam.eod 1 8
2],'edgecolor','none','facecolor',[0.8 0.8 0.8])
    plot(timeParam.range, totalGrowth,'-','linewidth',1)
    plot(timeParam.range, noStarchGrowth,'-','linewidth',1)
    xlim([0 24])
    ylabel([{'Total biomass'} {'mgDW'}])
    h = legend([{'Total DW'},{'Non-starch
DW'}],'location','northwest','fontsize',12);
    pos = get(h,'position');
    pos(1) = pos(1) + 0.025;
    pos(2) = pos(2) + 0.025;
    set(h,'position',pos);
    legend boxoff
    subplot(2,1,2)
    hold on
    rectangle('position',[timeParam.eod 0 8
1.5],'edgecolor','none','facecolor',[0.8 0.8 0.8])
    plot(timeParam.range, starch_conc,'-','linewidth',1)
    xlim([0 24])
    ylabel([{'Starch conc.'} {'\mumol/mgDW'}])
    xlabel('time after 1 PM, h')
    fig = gcf;
    fig.Position = [0 0 500 400];
    fig.PaperUnits = 'inches';
    fig.PaperPosition = [0 0 5 4];
    fprintf('max starch: %4.3f umol/mgDW\n', max(starch_conc));
    fprintf('min starch: %4.3f umol/mgDW\n', min(starch_conc));
    fprintf('transitory starch: %4.3f umol/mgDW\n', (max(starch_conc) -
min(starch_conc)));
    fprintf('t0 starch: %4.3f umol/mgDW\n', starch_conc(1));
    fprintf('t1 starch: %4.3f umol/mgDW\n', starch_conc(end));
    fprintf('t0 Biomass: %4.3f mg\n', totalGrowth(1));
    fprintf('t1 Biomass: %4.3f mg\n', totalGrowth(end));
end
```