UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Agricultura de Barcelona

SIMULACIÓN DE LA PRODUCCIÓN DE GIBERELINAS EN *Saccharomyces cerevisiae* MEDIANTE COBRApy

Autor: Jesús Castro Puntero

Tutores: Nuria Escudero Benito

Miguel Ángel Naranjo Ortiz

26 / 09 / 2019

Trabajo final de grado

Ingeniería de sistemas biológicos

Resum

Davant d'una població en constant creixement (amb una previsió de 8.500 milions el 2030 segons l'ONU), així com una problemàtica climàtica a nivell mundial en els propers anys, es mostra imperatiu el desenvolupament de noves tecnologies en el camp de l'agricultura. A causa de l'important paper de les fitohormones en el creixement i la maduració de la planta, l'ús de diferents hormones com les auxines, les gibberel·lines, l'etilè o les citoquinines està sent investigat.

L'ús de gibberel·lines, fitohormones capaces de promoure el creixement de tija, nombre d'inflorescències i resistència a estrès salí, es presenta com una eina amb molt potencial en el món agrícola. No obstant això, els mètodes actuals per a la seva producció, com la fermentació submergida de *Gibberella fujikuroi*, es troben en desenvolupament, fent inviables la seva utilització en cultius de forma extensiva.

En el present treball final de grau, s'ha modelitzat la utilització de *Saccharomyces cerevisiae* com a productor de gibberel·lines. Per dur a terme aquesta tasca, s'han utilitzat eines bioinformàtiques com COBRApy, un paquet implementat en Python per a l'anàlisi de rutes metabòliques de models matemàtics utilitzant per a això genomes anotats. L'ús d'aquestes eines permet estudiar *in-silico* la viabilitat de producció de gibberel·lines a *S. cerevisiae*. COBRApy permet la realització de les simulacions de rutes metabòliques i diferents estudis sobre el metabolisme de l'organisme, cosa que el fa idoni com a fase inicial d'un projecte biotecnològic.

Establertes les condicions del medi, amb una entrada de 10 mmol de glucosa com a única font de carboni, s'ha analitzat la producció màxima de biomassa, giberelinas i la combinació d'ambdues. La producció màxima de biomassa i de gibberel·lines quan s'optimitzen per separat ha estat de 9,75% i 6,73% del flux de carboni total respectivament. Per tal de millorar la producció de gibberel·lines, mitjançant l'anàlisi de comparació de distribució de fluxos es descriu una proposta de knockouts que podrien ajudar a redistribuir el flux de carboni. No obstant això, malgrat que els resultats *in-vivo* podrien diferir, el model no prediu millores per a la proposta de gens realitzada.

Paraules clau: GENRE, COBRA, *knock-out*, terpens, gibberel·lines, fitohormones

Resumen

Ante una población en constante crecimiento (con una previsión de 8.500 millones en 2030 según la ONU), así como una problemática climática a nivel mundial en los próximos años, se muestra imperativo el desarrollo de nuevas tecnologías en el campo de la agricultura. Debido al importante papel de las fitohormonas en el crecimiento y la maduración de la planta, el uso de distintas hormonas como las auxinas, las giberelinas, el etileno o las citoquininas está siendo investigado.

El uso de giberelinas, fitohormonas capaces de promover el crecimiento de tallo, número de inflorescencias y resistencia a estrés salino, se presenta como una herramienta con mucho potencial en el mundo agrícola. Sin embargo, los métodos actuales para su producción, como la fermentación sumergida de *Gibberella fujikuroi*, se encuentran en desarrollo, haciendo inviables su utilización en cultivos de forma extensiva.

En el presente trabajo final de grado, se ha modelizado la utilización de *Saccharomyces cerevisiae* como productor de giberelinas. Para llevar a cabo esta tarea, se han utilizado herramientas bioinformáticas como COBRApy, un paquete implementado en Python para el análisis de rutas metabólicas de modelos matemáticos utilizando para ello genomas anotados. El uso de estas herramientas permite estudiar *in-silico* la viabilidad de producción de giberelinas en *S. Cerevisiae.* COBRApy permite la realización de las simulaciones de rutas metabólicas y distintos estudios sobre el metabolismo del organismo, cosa que lo hace idóneo como fase inicial de un proyecto biotecnológico.

Establecidas las condiciones del medio, con una entrada de 10 mmol de glucosa como única fuente de carbono, se ha analizado la producción máxima de biomasa, giberelinas y la combinación de ambas. La producción máxima de biomasa y de giberelinas cuando se optimizan por separado ha sido de 9,75% y 6,73% del flujo de carbono total respectivamente. Con el fin de mejorar la producción de giberelinas, mediante el análisis de comparación de distribución de flujos se describe una propuesta de *knockouts* que podrían ayudar a redistribuir el flujo de carbono. No obstante, pese a que los resultados *in-vivo* podrían diferir, el modelo no predice mejoras para la propuesta de genes realizada.

Palabras clave: GENRE, COBRA, *knock-out*, terpenos, giberelinas, fitohormonas

Abstract

Faced with a population in constant growth (with a forecast of 8.5 billion in 2030 according to the UN), as well as a global climate problem in the coming years, it is imperative to develop new technologies in the field of agriculture. Due to the important role of phytohormones in the growth and maturation of the plant, the use of different hormones such as auxins, gibberellins, ethylene or cytokinins is being researched.

The use of gibberellins, phytohormones capable of promoting stem growth, number of inflorescences and resistance to saline stress, is presented as a tool with much potential in the agricultural world. However, current production methods, such as submerged fermentation of *Gibberella fujikuroi*, are under development, making it unfeasible for extensive use in crops.

The use of *Saccharomyces cerevisiae* as a producer of gibberellins has been modelled in this final grade work. To carry out this task, bioinformatic tools have been used such as COBRApy, a package implemented in Python for the analysis of metabolic pathways of mathematical models using annotated genomes. The use of these tools allow us to study *in-silico* the viability of gibberellin production in *S. Cerevisiae*. COBRApy allows simulations of metabolic pathways and different studies on the metabolism of the organism, which makes it ideal as an initial phase of a biotechnological project.

Once the conditions of the environment have been established, with an input of 10 mmol of glucose as the only source of carbon, the maximum production of biomass, gibberellins and the combination of both has been analysed. The maximum production of biomass and gibberellins when optimized separately has been 9.75% and 6.73% of the total carbon flow respectively. In order to improve the production of gibberellins, flow distribution comparison analysis describes a proposal of knockouts that could help redistribute the carbon flow. However, although the in-vivo results may differ, the model does not predict improvements to the gene proposal made.

Keywords: GENRE, COBRA, knock-out, terpenes, gibberellins, phytohormones

Índice

Índice figuras

Índice de tablas

Abreviaturas y siglas

Debido a la gran cantidad de abreviaturas derivadas del uso de bases de datos de metabolitos y reacciones, se incluye una tabla general para consultar los nombres descritos en las rutas

metabólicas adjuntas así como los acrónimos relacionados con los análisis llevados a cabo (tablas 1 y 2). Para una mejor praxis la tabla 1 se ha dividido en distintos tipos de metabolitos y *otros* acrónimos, aquellas abreviaturas y siglas que no son ni metabolitos ni enzimas. En la tabla 2 están las referencias a las enzimas. Los códigos empleados en metabolitos y enzimas pertenecen a la nomenclatura de *BiGG models* (http://bigg.ucsd.edu/).

Metabolitos relacionados con las giberelinas	
GA	Giberelina
grdp	Difosfato de geranilo
ipdp	Difosfato de isopentenilo
MVA	ácido mevalónico
MEP	metileritritol fosfato
acetil-SCoA	acetil coenzima A
HMG-CoA	3-hydroxi-3- methil-glutaril-CoA
ATP	adenosín trifosfato
dmpp	Dimetilalil difosfato
ggdp	geranilgeranil difosfato
frdp	difosfato de farnesilo
dpmev	Difosfomevalonato
frdp	Difosfato de farnesilo
ggdp	Difosfato de geraniol-geraniol
sql	Escualeno
Ssq23epx	S escualeno 2 3 epóxido
lanost	Lanosterol
Metabolitos medio de cultivo*	
h	Hidrógeno
o2	Oxígeno
so4	Sulfato
pi	Fosfato
k	Potasio
na1	Sodio
h2o	Agua
nh4	Amonio
fe2	Hierro
glc__D	D-glucosa
*Los metabolitos del medio de cultivo finalizan en "_e", lo que indica que el compartimento en el que se encuentran es el extracelular	
Otros	
GENRE	*GENome-scale metabolic Network REconstruct*
GEM	*genomic-scale models*
COBRA	*COnstraint-Based Reconstruction and Analysis*
KEGG	Kyoto Encyclopedia of Genes and Genomes
SBML	*Systems Biology Markup Language*)
KO	*knock-outs*
GPR	*gene-protein-reaction*
FBA	*Flux Balance Analysis*
BOF	*Biomass Objective Function*
v_{ub}	límite superior del flujo
v_{lb}	límite inferior del flujo
S	matriz estequiométrica
v	flujo de reacción
FVA	*Flux Variability Analysis*
SD	*Standard Minimal Medium*
YNB	*yeast nitrogen base*
pFBA	*parsimonious FBA*
MoMA	*minimization of metabolic adjustment*

Metabolitos de la función de biomasa objetivo**	
adp	Adenosín difosfato
ala__L	L-Alanina
amp	Adenosín monofosfato
arg__L	L-Arginina
asn__L	Asparagina
asp__L	L-Aspartato
atp	ATP
cmp	Carbamoíl fosfato
cys__L	L-Cisteína
damp	Desoxiadenosina monofosfato
dcmp	Desoxicitidina monofosfato
dgmp	Desoxiguanosina monofosfato
dtmp	Timidina monofosfato
gln__L	L-Glutamina
glu__L	L-Glutamato
gly	Glicina
glucógeno	Glucógeno
gmp	Guanosina monofosfato
h	Hidrógeno
h2o	Agua
his__L	L-Histidina
ile__L	L-Isoleucina
leu__L	L-Leucina
lys__L	L-lisina
met__L	L-Metionina
phe__L	L-Fenilalanina
pi	Fosfato
pro__L	L-Proline
ribflv	Riboflavina
ser__L	L-Serina
so4	Sulfato
thr__L	L-Treonina
tre	Trehalosa
trp__L	L-triptófano
tyr__L	L-Tyrosine
ump	Monofosfato de uridina
val__L	L-Valine
13BDglcn	1 3 beta D Glucan
ergst	Ergosterol
mannan	Manoproteína
pa_SC	Fosfatidato específico de levadura
pc_SC	Fosfatidilcolina específico de levadura
pe_SC	Fosfatidiletanolamina específico de levadura
ps_SC	Fosfatidilserina específico de levadura
ptd1ino_SC	Fosfatidil 1D myo inositol específico de levadura
triglyc_SC	Triglicéridos específicos de levadura
zymst	Zymosterol
**Los metabolitos de la función de biomas objetivo finalizan en "_c", lo que indica que el compartimento en el que se encuentran es el citosol	

Tabla 1 Abreviaturas y siglas - Metabolitos y otros

Enzimas		Enzimas del análisis de comparación de distribución de flujos	
CPR	NADPH citocromo p450 reductasas	CSm	Citrato sintasa
GID1	GA insensitive dwarf1	MDHm	Malato deshidrogenasa mitocondrial
SCF	SLY1, Skp1-cullin-F-box	AKGDam	Oxoglutarato deshidrogenasa lipoamida
FRTT	Farnesiltranstransferasa	AKGDbm	Oxoglutarato deshidrogenasa dihidrolipoamida S succiniltransferasa
GRTT	geraniltranstransferasa	GCC2cm_copy1	Complejo de escisión de glicina lipoamida mitocondrial
DPMVD	Difosfomevalonato descarboxilasa	SUCOASm	Formación de Succinato CoA ligasa ADP
IPDDI	Isopentenilo-difosfato D-isomerasa	ACONT	Aconitato hidratasa
DMATT	Dimetilalilranstransferasa	CITtam	Transporte de citrato mitocondrial
SQLS	Escualeno sintetasa	ICDHyr	Isocitrato deshidrogenasa (NADP)
SQLter	Transporte retículo endoplasmático escualeno	GLUt7m	Glutamato transporte uniportador mitocondrial
SQLEr	Escualeno epoxidasa en retículo endoplasmático NADP	ILETAm	Isoleucina transaminasa mitocondrial
SQ23EPXter	Transporte retículo endoplasmático S escualeno 2 3 epóxido	ILEtmi	Transporte de isoleucina de las mitocondrias al citosol
LNSTLS	Lanosterol sintetasa	3MOPtm	3 Metil 2 oxopentanoato transporte difusión mitocondrial
LANOSTt	Lanosterol transporte reversible	ILETA	Isoleucina transaminasa
EX_lanost_e	Lanosterol intercambio	ACONTm	Aconitato hidratasa
	Enzimas añadidas	NDPK2	Nucleósido-difosfato quinasa (ATP: UDP)
EntKS	Bifuncional ent-kaureno sintasa	GALUi	UTP-glucosa-1-fosfato uridililtransferasa (irreversible)
EntKO	ent-kaureno oxidasa	PGMT	Fosfoglucomutasa
EntKAO	ácido ent-kaurenoico monooxigenasa	ICDHxm	Isocitrato deshidrogenasa NAD
P450-1	citocromo P450 - 1	13GS	1 3 beta glucan sintasa
P450-2	citocromo P450 - 2	ICDHym	Isocitrato deshidrogenasa NADP
P450-3	citocromo P450 - 3	BIOMASS_SC5_not	Función objetivo de biomasa establecida en el modelo iMM904
		DOLPMMer	Dolilfosfato manosa proteína manosiltransferasa reticular endoplásmica
		DOLPMTcer	Dolilfosfato D manosiltransferasa
		DOLPt2er	Transporte reticular endoplásmico de fosfato de dolichol a través del simport de protones
		MAN1PT	Manosa 1 fosfato guanililtransferasa
		MAN6PI	Manosa-6-fosfato isomerasa
		MANNANter	Transporte de manoproteína vía difusión en el retículo plasmático
		PMANM	Fosfomannomutasa
		OAAt2m	Transporte de oxaloacetato mitocondrial
		ACLSm	Acetolactato sintasa mitocondrial
		DHAD1m	Dihidroxiácido deshidratasa 2 3 dihidroxi 3 metilbutanoato mitocondrial
		KARA1im	Acetohidroxiácido isomeroreductasa mitocondrial
		GLCS2	Glucógeno sintasa UDPGlc
		ALATA_L	L-alanina transaminasa
		ACS	Acetil-CoA sintetasa
		ALDD2y	Aldehído deshidrogenasa (acetaldehído)
		PYRDC	Piruvato descarboxilasa

Tabla 2 Abreviaturas y siglas - Enzimas

Agradecimientos

En primer lugar me gustaría agradecer a mis tutores la paciencia, la facilitación de recursos, sus ampliaciones de mis puntos de vista y su, lo que a mi parecer es, perfeccionismo estético y técnico.

Me gustaría hacer mención a Alexandra Elbakyan, creadora de Sci-Hub, persona que ha resultado ser totalmente vital para este trabajo. A Steve Preisler, por darme las primeras ideas para la realización de un trabajo así (aunque el rumbo tomado ha sido distinto al imaginado). Y a Jiří Jablonský, persona que me introdujo al método COBRA.

1. Introducción

1.1. Modelos matemáticos (GENREs)

Los modelos matemáticos que describen las reacciones metabólicas de un organismo (o de una célula) se conocen como GENREs (GENome-scale metabolic Network REconstruction) [1] o GEMs (genomic-scale models) [2]. Los GENREs permiten simular y predecir los efectos y cambios en el fenotipo cuando se presentan alteraciones genéticas o en el ambiente. Para realizar este tipo de predicciones se puede aplicar el método COBRA (COnstraint-Based Reconstruction and Analysis) [1], que permite hacer un análisis determinista a través de la programación lineal realizado en un ordenador.

1.1.1. Reconstrucción GENRE

Para poder aplicar el método COBRA, utilizado en este trabajo, es necesario un modelo virtual de las rutas metabólicas del organismo de estudio. El proceso por el cual se construyen estos modelos puede verse resumido en la figura 1 y, para llevarse a cabo, son necesarias las siguientes etapas [3]:

1. Obtención del genoma anotado partir de bases de datos como Kyoto Encyclopedia of Genes and Genomes (KEGG; https://www.genome.jp/kegg/). Un genoma anotado es aquél que tras secuenciarse, sus genes han sido ubicados y sus funciones, descritas. Conocer las enzimas presentes en el organismo a modelizar capaces de catalizar las reacciones metabólicas,permitirá la creación de un primer borrador para la red metabólica del modelo.

2. Identificación de posibles huecos e inconsistencias que deberán ser curados manualmente, para ello se consultarán datos empíricos disponibles.

3. Transformación del conjunto de reacciones metabólicas en un modelo matemático utilizando metodologías establecidas como SBML (*Systems Biology Markup Language*). La conversión a un modelo computacional puede contener distintas pseudorreacciones o reacciones agrupadas. Éstas no corresponden a reacciones enzimáticas auténticas

pero reflejan procesos celulares generales. Un ejemplo pueden ser aquellas que requieran un consumo de ATP para el mantenimiento celular, la formación de biomasa o una ruta metabólica en la cual solo se indican los reactivos y los productos totales sin indicar todas las reacciones intermedias de la misma.

4. Validar el modelo mediante métodos de simulación de sistemas biológicos como COBRA, *Bioconductor* [4] o *Graphviz* [5].

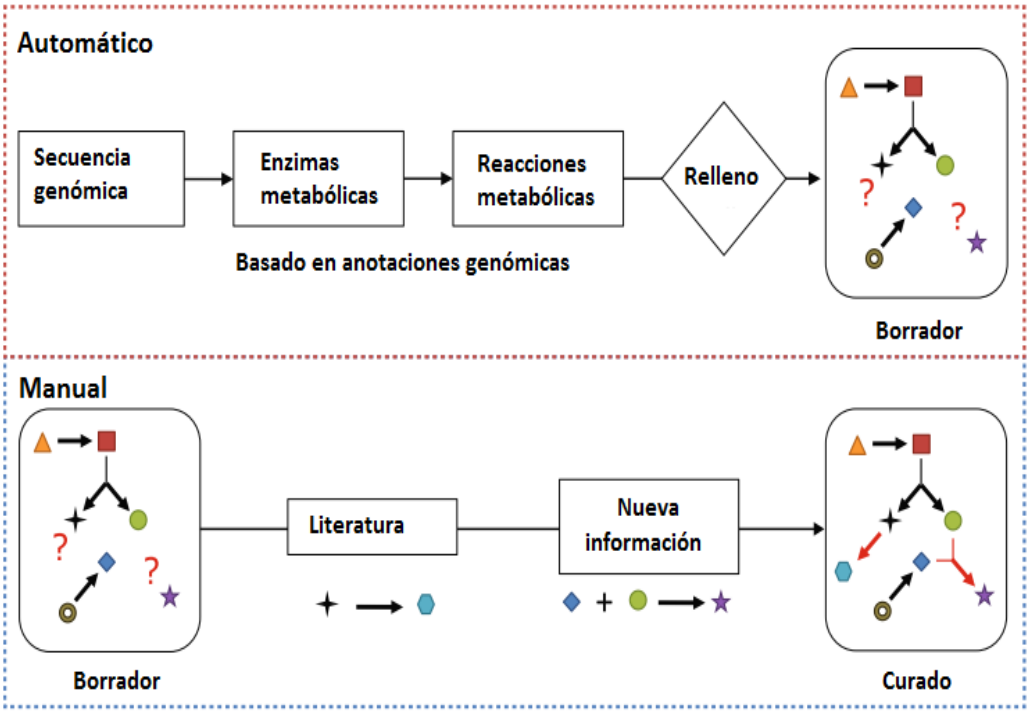

Figura 1 Etapas necesarias para la reconstrucción de un modelo metabólico de un organismo. Un primer borrador se obtiene a partir de la anotación del genoma de un organismo y, mediante programas específicos, se pueden obtener las enzimas y las reacciones metabólicas llevadas a cabo. No obstante, la aparición de huecos e incongruencias debe ser curada manualmente a partir de la bibliografía disponible. [1]

1.1.2. Modelo iMM904

Las bajas concentraciones presentes en los microorganismos productores de giberelinas terpenoides y la complicada estructura de muchos de estos compuestos limitan su idoneidad para las aplicaciones comerciales. Es por eso que se presenta como alternativa potencial el desarrollo de sistemas de síntesis biológica de terpenoides para su producción en masa. Los organismos potenciales para el desarrollo de dichas actividades son *Escherichia coli* y *Saccharomyces cerevisiae*. Debido a que *S. cerevisiae* posee la habilidad de expresar las enzimas

del citocromo P450, lo hace más interesante para la síntesis de terpenoides de origen vegetal y es por eso que será usado en este estudio [6]. Las enzimas del citocromo P450 son esenciales en la ruta biosintética de las giberelinas y su presencia en el organismo facilita y reduce las manipulaciones a realizar. Además, las células de levadura ya han sido usadas en la síntesis de monoterpenos, sesquiterpenos y diterpenos [7, 8].

Aunque se ha detallado la reconstrucción de las GENREs para comprender su estructura y elucidar posibles errores provenientes de dicha reconstrucción, en este estudio se usará un modelo de libre acceso de *Saccharomyces cerevisiae* S288C que está disponible en la base de datos de *BiGG models*. Esta es una base de datos de GEMs bioquímica, genética y genómicamente estructurados. S288C es una cepa de laboratorio ampliamente utilizada, diseñada por Robert Mortimer para estudios bioquímicos, y específicamente seleccionada para ser no floculante, lo que permite que puedan separarse con facilidad, con un conjunto mínimo de requerimientos nutricionales [9].

BiGG models usa un sistema de ID para nombrar metabolitos y reacciones compatibles con SBML que son relativamente fáciles de leer e identificar sin la ayuda de programas externos. Además, se puede indicar el compartimento de los metabolitos al final del nombre de esto, por ejemploh_e sería hidrógeno extracelular mientras que h_m sería hidrógeno mitocondrial. En este trabajo se usará la nomenclatura de *BiGG models* siempre que sea posible.

Concretamente se utilizará el modelo iMM904 (http://bigg.ucsd.edu/models/iMM904) que se basa en el genoma de *Saccharomyces cerevisiae* 288c GCF_000146045.2, conteniendo 1226 metabolitos y 1577 reacciones provenientes de 905 genes. Evidentemente, no se trata de una reconstrucción completa de las rutas metabólicas de este organismo, sino de una reconstrucción parcial de las principales rutas metabólicas de éste. Básicamente consta de todas las reacciones, metabolitos y genes implicados en el ciclo de Krebs, en el metabolismo del alcohol y en la ruta de pentosas fosfato. También tuvo mejoras respecto a su modelo predecesor iND750 como la adición de glucogenina y NADPH citocromo p450 reductasas (CPR), que se requieren en la síntesis glucógeno y para mantener la actividad catalítica en los citocromos p450, respectivamente. [9]

1.2. Giberelinas

1.2.1. Terpenos

Desde 1990 se conocen alrededor de 30.000 terpenos [10], y todos siguen el mismo principio general para formar su estructura a través de la unión de unidades de isopreno (normalmente entre los carbonos 1-4) moléculas de 5 carbonos que constituyen el esqueleto de los terpenos [11]. Dependiendo del número resultante de carbonos en los terpenos, éstos pueden ser clasificados según distintas familias (Figura 2). Según el número de carbonos distinguimos hemiterpenos (C_5), monoterpenos (C_{10}), seguidos por los sesquiterpenos (C_{15}), diterpenos (C_{20}), sesterterpenos (C_{25}), triterpenos (C_{30}), tetraterpenos (C_{40}) y politerpenos $(C_5)_n$. Terpenos y terpenoides no son términos intercambiables, aunque pueden ser encontrados y usados indistintamente en la bibliografía consultada. Los isoprenoides, son terpenos modificados que contienen al menos un grupo funcional con oxígeno. [12]

Los terpenos son biomoléculas de estructura muy diversa que pueden ser encontrados en todos los reinos de los seres vivos [13], no obstante, la mayoría han sido aislados de plantas, donde participan en diferentes procesos como la formación de fitohormonas o de la clorofila, así como en interacciones con el ambiente como defensas químicas contra herbívoros y patógenos, atractores de polinizadores y agentes alelopáticos entre otros [14]. Algunos terpenos tienen un elevado valor industrial, como los aceites esenciales, pigmentos carotenoides o goma natural. Sin embargo, debido a su gran potencial para el desarrollo de las plantas, en este estudio se centrará la atención en las giberelinas. Éstas son una clase de fitohormonas que estimulan el crecimiento y participan en distintos procesos de desarrollo durante todo el ciclo vital de la planta, incluyendo la germinación de las semillas, la elongación del tallo, la floración, la formación de frutos y la senescencia [15,16].

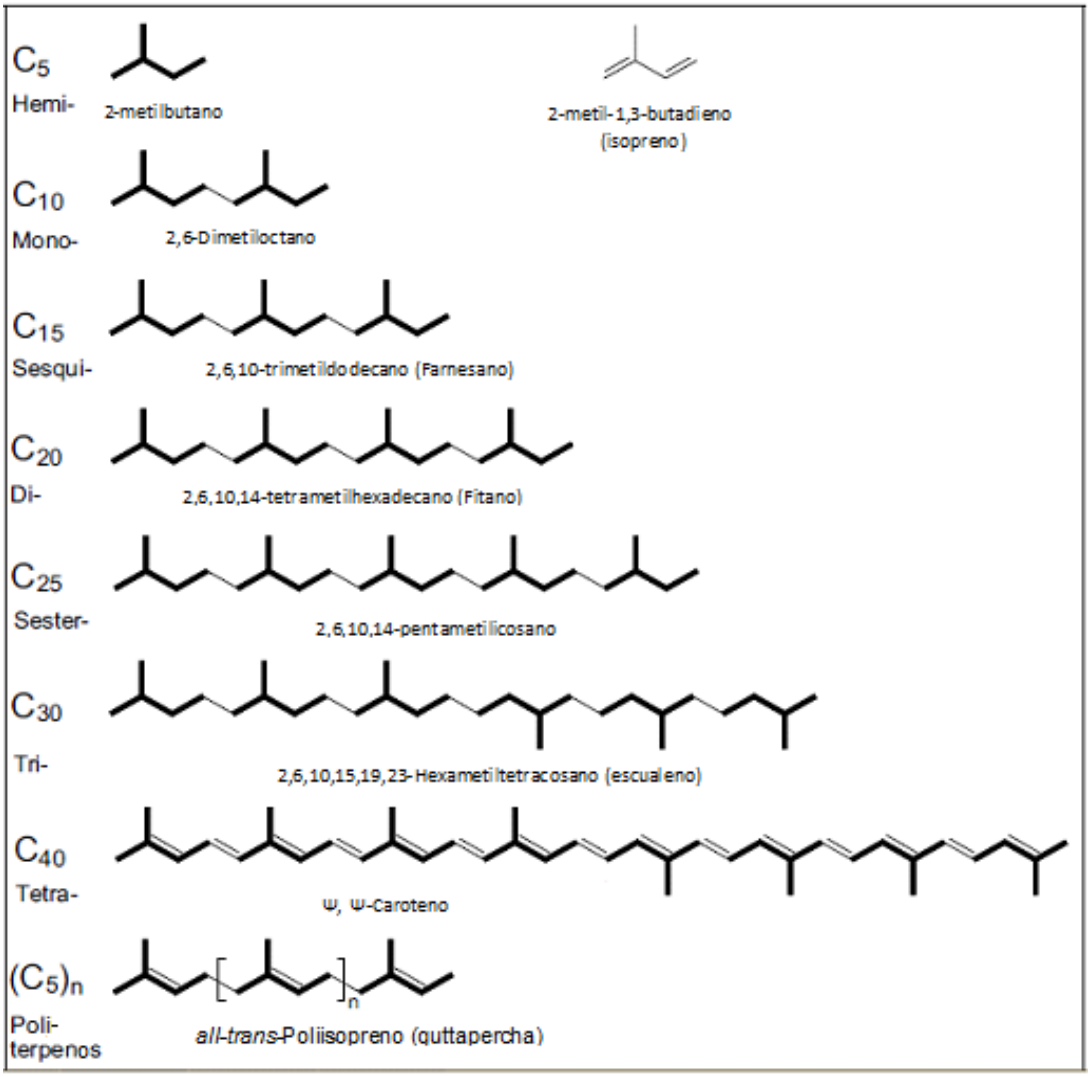

Figura 2 Familias de terpenos formadas a partir de la unidad básica (isopreno), indicada en la parte superior derecha de la figura. [17]

1.2.2. Introducción a las giberelinas

Las giberelinas (GAs) son fitohormonas implicadas en el crecimiento de las plantas. Estas biomoléculas forman una familia de diterpenoides. Hay dos tipos principales de GAs, un primer grupo que agrupa las moléculas con 20 carbonos que no presentan bioactividad conocida (C20-GAs) y otro grupo formado por GAs que han perdido un carbono y muestran una lactona (C19-GAs) donde se encuentran tanto GAs con y sin bioactividad conocida, es decir, no cumplen ninguna función en el desarrollo de la planta [18]. Las GAs bioactivas son aquellas que controlan diversos aspectos del crecimiento y el desarrollo. Estas fitohormonas fueron inicialmente

identificadas por el fitopatólogo japonés Sawada en 1917 como un metabolito secundario en el hongo *Gibberella fujikuroi* (teleomorfo de *Fusarium fujikuroi)* como agente causante de la enfermedad *bakanae* (conocida en inglés como *foolish seedling disease)* [19]. Esta enfermedad afecta al arroz, haciéndolo crecer sin control hasta que la planta muere debido a que no puede sostener su propio peso. Más tarde, pudo observarse que esta presunta toxina segregada por el hongo estimulaba el crecimiento de plantas diferentes al arroz [20]. En 1935, tras ser purificada, recibió el nombre de giberelina y su actividad biológica fue de nuevo comprobada y demostrada. Tres componentes ácidos fueron obtenidos y fueron nombrados giberelinas A1, A2 y A3. Estos componentes mostraron un efecto en el crecimiento de las plantas y en su capacidad de devolver un fenotipo normal a las plantas enanas [21]. Recientemente, se ha comprobado su producción por parte de muchos más hongos y bacterias, pese a que no se ha acabado de elucidar completamente el rol de estos metabolitos en estos microorganismos. Distintos experimentos parecen apuntar a que la producción de GAs por parte de hongos está relacionada con patogénesis, favoreciendo una mejor colonización de tejidos vegetales [15]. El rol de estas moléculas en bacterias también ha sido vinculado como un factor de virulencia. La hipótesis más aceptada es que actuarían como supresores de la ruta del ácido jasmónico en el huésped, una hormona vegetal que está relacionada con señales químicas (jasmonatos) que inducen mecanismos de defensa en plantas como respuesta a la actividad de las proteasas digestivas del patógeno. La síntesis de estas moléculas se ha descrito también en bacterias fijadoras de nitrógeno simbióticas de la familia Rhizobiaceae, aunque se sabe aún muy poco del papel que estas moléculas desempeñan en este tipo de interacciones [22].

1.2.3. Función de las giberelinas en plantas

Las giberelinas han sido reconocidas como hormonas reguladoras del crecimiento y desarrollo de plantas. El modelo actual para el funcionamiento de las GAs establece que aumentan el crecimiento de la planta mediante la eliminación de proteínas DELLA, una familia de proteínas que detienen el crecimiento de la planta. Su nombre se deriva de los aminoácidos que las componen de una pequeña región en el extremo N-terminal, ácido aspártico (D), ácido glutámico (E), leucina (L), leucina (L) y alanina (A). Estos aminoácidos son, traducidos al código *de una letra* son D-E-L-L-A.

El mecanismo de acción para la actividad de GA está mediado por la unión activatoria de éstas al receptor GID1, permitiendo su reconocimiento por parte de las proteínas DELLA. Después de la formación del complejo GID1-GA-DELLA, éste es reconocido por la caja-F (*F-box),* una de las tres proteínas componentes del complejo *Skp1-cullin-F-box* (SCF). Este complejo también incluye ubiquitina ligasa, que se encarga de la poliubiquitinación. Esto permite la degradación de la proteína DELLA a través del proteosoma 26S. [15]

Esta degradación de las proteínas DELLA resulta en cambios en la expresión génica que culmina en la elongación celular como se observa en la figura 3. Este mecanismo molecular parece estar involucrado en distintos estadios fisiológicos de las plantas incluyendo desarrollo embrionario, germinación de semillas, desarrollo de la raíz, expansión de las hojas, elongación del tallo, desarrollo de tricomas y maduración del polen [23]. El rol de las proteínas DELLA ha sido estudiado en *Arabidopsis* ([17, 24] en cebada [25], en la uva [26], en maíz [27], en arroz [28], en tomate [29] y en trigo [27]).

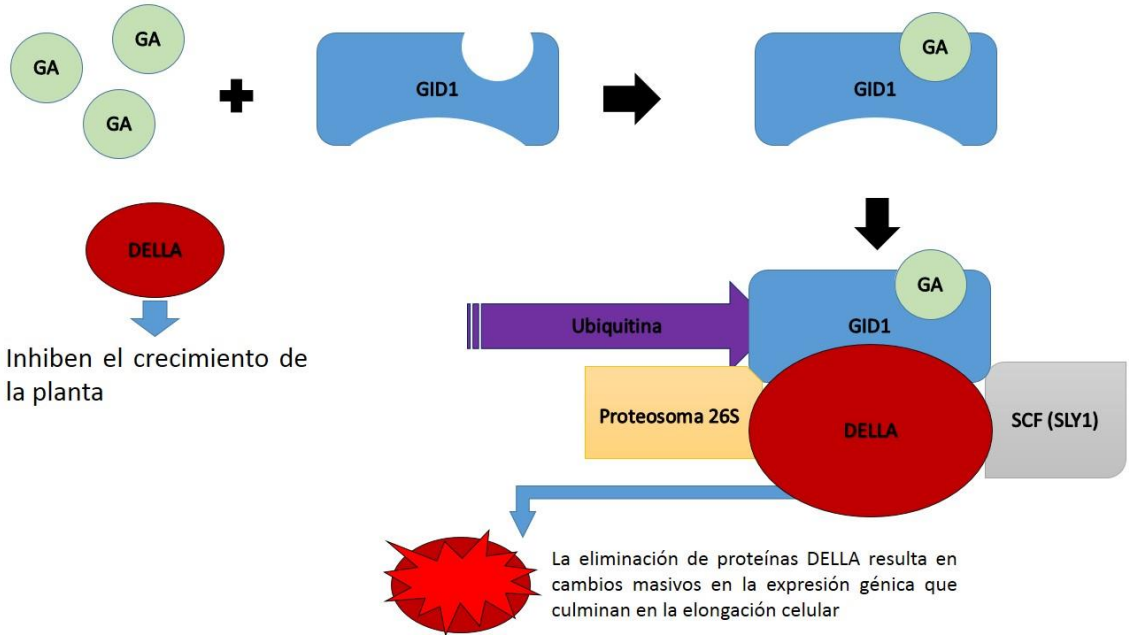

Figura 3 Modelo del mecanismo de las proteínas DELLA. Mediante la unión de las GA al receptor GID1 la ubiquitinación de éste se consigue la degradación de las proteínas DELLA. GID1 (GA insensitive dwarf1), SCF (SLY1, Skp1-cullin-F-box) [15]

1.2.4. Aplicaciones biotecnológicas de giberelinas

Los aspectos bioquímicos, moleculares y genéticos de la producción de GA todavía están bajo estudio. No obstante, destacan diferentes aplicaciones biotecnológicas de estas fitohormonas como promotor de crecimiento de plantas, mejorar la producción cantidad y calidad de los cultivos llegando incluso a permitir el cultivo de ciertas especies de vegetales en ambientes desfavorables [30]. Distintos experimentos apoyan la capacidad de las GAs para promover la resistencia a estrés salino en distintos tipos de plantas. Esta propiedad permitirá extender ciertos cultivos a ambientes que actualmente son poco propicios para su crecimiento [31].

Debido a su efecto en la promoción del crecimiento, tanto durante el período vegetativo como en la germinación, en un futuro cercano sería interesante el desarrollo de suplementos sostenibles con el medioambiente a partir de Giberelinas. GA3 ha sido utilizada en floricultura para incrementar el número de botones florales y número de inflorescencias [26]. Las GAs también han sido comúnmente usadas en la mejora de producción de frutos, aumentando el crecimiento y desarrollo después de la polinización [31]. También se ha determinado que la combinación de las giberelinas GA4+GA7 ayuda a combatir defectos y enfermedades, como el

control de *russeting* [33], un defecto que aparece en ciertas variedades de peras y manzanas que se manifiesta con manchas marrones en la piel del fruto, junto a una rugosidad anormalmente alta, disminuyendo su comercialización.

1.3. Programación lineal

La programación lineal es una técnica matemática que permite descomponer problemas complejos en un conjunto de funciones lineales. Este tipo de aproximación es particularmente útil para resolver problemas combinatoriales, cuya resolución por fuerza bruta es a menudo inviable. Esta técnica requiere de unas variables de decisión (en este caso los distintos flujos de las reacciones), una función objetivo que maximizar o minimizar y unas restricciones que limitan las variables de decisión. Es necesario que todas ellas sean funciones lineales. El algoritmo buscará un conjunto o conjuntos de parámetros que den como resultado un máximo o mínimo en la función objetivo. El uso de restricciones permite acotar el espacio de soluciones, a fin de obtener un conjunto limitado de las mismas.

En el modelo utilizado en el presente estudio, las reacciones vienen definidas como funciones lineales teniendo en cuenta los metabolitos involucrados. También se definen distintas restricciones lineales que permiten determinar la reversibilidad o irreversibilidad de las reacciones así como la entrada o salida de metabolitos. Por otro lado, la optimización de una función objetivo permite simular la maximización para el crecimiento o para la síntesis de un producto (como será el caso de las GAs), la minimización para el consumo de nutrientes del medio o la simulación del efecto de suprimir la expresión de determinados genes en el organismo [34].

Típicamente, los problemas solucionados por programación lineal, pueden ser solucionados gráficamente. No obstante debido al número de reacciones metabólicas incluidas en el modelo, que en este caso es superior a 1500, el uso de herramientas informáticas es necesaria.

2. Objetivos

En el presente trabajo de final de grado se pretende explorar la viabilidad de *Saccharomyces cerevisiae* como bioplataforma para la síntesis de giberelinas como GA3, GA1, GA4 y GA7 a escala industrial. Para ello se evaluará un modelo de esta especie mediante el método COBRA implementando COBRApy.

Para cumplir este requisito será necesario:

- Conocer el funcionamiento y la implementación del análisis de balance de flujos (FBA) para evaluar el balance estequiométrico entre la entrada de nutrientes y el producto final, el análisis de variabilidad de flujos (FVA) para observar potenciales cambios en la en la cantidad de entrada de nutrientes.
- Implementar el análisis de comparación de distribuciones de flujo (FDCA) con el fin de identificar genes potencialmente suprimibles para mejorar el rendimiento.
- Programar la ruta metabólica hasta la síntesis de giberelinas en el modelo, detallando las reacciones, los metabólitos, los genes y las enzimas involucradas.
- Comparar métodos alternativos que se estén llevando a la práctica para lograr la síntesis de giberelinas.

3. Materiales y métodos

3.1 COBRA

Aunque no se posean datos suficientemente detallados para la precisa reconstrucción de un modelo de un organismo a nivel genómico, las rutas metabólicas del mismo pueden ser simuladas en COBRA [34]. Mediante la definición de restricciones en el modelo matemático resultante, puede simularse la conservación de la masa, espontaneidad, direccionalidad y compartimentación de las reacciones, permitiendo analizar redes metabólicas. Los métodos basados en COBRA no son de solución única debido a que el usuario debe definir ciertos parámetros ambientales, algunas reacciones pueden ser suprimidas o añadidas y se trabaja con sistemas indeterminados. Por ello, distintos tipos de análisis deben ser empleados.

3.1.1 Gene-protein-reaction rule

COBRA proporciona un marco mecanicista para el análisis de modelos de sistemas moleculares y la predicción cuantitativa de los flujos de las reacciones que se producen en distintas condiciones. Estos análisis deben ser realizados sobre un modelo matemático. Por ello, se incluye en las GENREs matrices estequiométricas de las reacciones (matriz S o *S matrix*), donde se incluye las rutas metabólicas con los productos, los reactivos y sus respectivos coeficientes estequiométricos. También se incluyen los genes, las enzimas codificadas por ellos y las reacciones que permiten éstas últimas, de manera que al estudiar el efecto de mutantes nulos (*Knock Outs, KO*), se pueda saber qué reacción metabólica será anulada. Mediante los KOs se anula la expresión de un gen determinado lo que puede permitir discernir su función. Esta relación entre genes, enzimas y reacciones se conoce como GPR (*gene-protein-reaction*) y nos permite entender cómo puede afectar la eliminación de un gen a una reacción o qué conjunto

de genes deben ser eliminados para anular una reacción. En la figura 4 se muestran las diferentes opciones a nivel de síntesis de diferentes proteínas. Generalmente, se asume que cada gen codifica para una enzima que permitirá una reacción (Figura 4A). Sin embargo, existen distintos genes que codifican para enzimas que catalizan la misma reacción, o isoenzimas (Figura 4B), por lo que el KO de uno de estos genes no sería suficiente para detener la reacción. Algunas reacciones, requieren de un complejo de enzimas para poderse producir (Figura 4C). En estos casos distintos genes codifican para subunidades proteicas que catalizan una sola reacción conjuntamente, con lo que el KO de cualquiera de estos genes anulará la reacción.

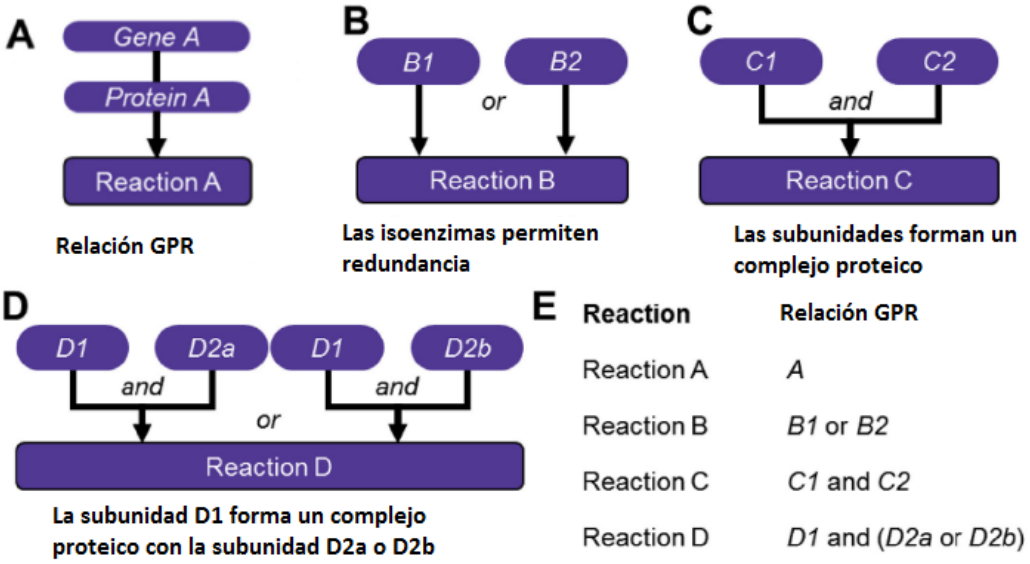

Figura 4 Relaciones GPR que describen la relación entre genotipo y fenotipo. (A) GPR que describe una reacción catalizada por una enzima codificada por un único gen. La eliminación de este gen supondría la inhabilitación de esta reacción. (B) GPR que describe una reacción que puede ser catalizada por dos isoenzimas. La reacción B será anulada en caso de la eliminación de los genes que codifican todas las isoenzimas. (C) GPR para un complejo proteico, donde dos enzimas distintas son requeridas para que se pueda llevar a cabo la reacción C. La eliminación de una de estas dos provocaría la inhabilitación de dicha reacción. (D) GPR para un complejo proteico con redundancia. En este caso la reacción D podría ser reescrita como la reacción entre dos subunidades formadas por D1 y D2a o D1 y D2b. La eliminación del gen de D1 inhabilitaría la reacción D puesto que es necesaria para la formación de las dos subunidades. (E) Tabla resumen de las reacciones y sus respectivas normas GPR escritas de manera booleana. [35]

3.1.2 Análisis de balance de flujos (FBA)

El análisis de balance de flujos FBA (*Flux Balance Analysis*) es un método de optimización basado en restricciones que permite calcular el máximo flujo posible para una reacción específica, conocida como función objetivo. Cuando esta función objetivo está enfocada a maximizar la

producción de biomasa, se conoce como función objetivo de biomasa o BOF (*Biomass Objective Function*), y se trata de una ecuación que especifica todos los precursores relevantes de biomasa en sus proporciones molares adecuadas. Sin embargo, esta ecuación es difícil de obtener de manera experimental y su obtención suele requerir extrapolar numerosos resultados previos obtenidos de organismos genéticamente cercanos y bases de datos de rutas metabólicas [3].

Para llevar a cabo el análisis FBA es necesaria una matriz estequiométrica S (matriz S) que contiene todos los metabolitos y reacciones que han sido dilucidados de la anotación genómica del organismo estudiado [2]. Esta matriz tiene una estructura *m* x *n*, conteniendo una fila por cada metabolito *m* y una columna por cada reacción *n* en el modelo. La matriz S es, habitualmente, un sistema indeterminado puesto que tiene más reacciones que metabolitos. Por ello, para obtener una solución única se usan distintas restricciones. Cada reacción, tiene un valor de flujo v_i [3]

Dada una matriz estequiométrica (S), límites superiores e inferiores (v_{ub} y v_{lb}) en los flujos de las reacciones (v) y de la reacción objetivo (v_{obj}), FBA usa la programación lineal para resolver las siguientes ecuaciones:

$$S \cdot v = \frac{dC}{dt} = 0 \qquad (1)$$

$$v_{lb} \leq v \leq v_{ub} \qquad (2)$$

La ecuación 1 expresa que la velocidad de cambio en la concentración de los metabolitos puede ser descrita como una ecuación de balance de masas donde S es la matriz estequiométrica y v indica el vector del flujo de la reacción. Usando la restricción de flujo estacionario para las concentraciones de metabolitos, la distribución de flujos debe cumplir que $S \cdot v = 0$. Para todo flujo v, existen dos restricciones, superior e inferior, tal y como puede verse expresado en la ecuación 2 [3]. Estas restricciones son las que nos permitirán simular las distintas condiciones biológicas para dar solución a un problema en concreto. Por ejemplo, mediante estas restricciones puede limitarse la entrada de glucosa en el organismo, la luz disponible o incluso el exceso de algún otro metabolito. También pueden aplicarse dichas restricciones para imponer la irreversibilidad de una reacción. Esto permite observar las rutas metabólicas y sus respectivos flujos obtenidos necesarios para llegar a la reacción planteada como flujo o función objetivo.

En la figura 5 se representa la operación definida en la ecuación 1, donde no se obtiene acumulación de ningún metabolito. También se tiene en cuenta los límites superiores e inferiores de cada reacción, lo cual permitirá adecuar el ambiente del sustrato a simular. Los límites establecidos ayudan a acotar el espacio de solución de FBA y condicionarán el resultado.

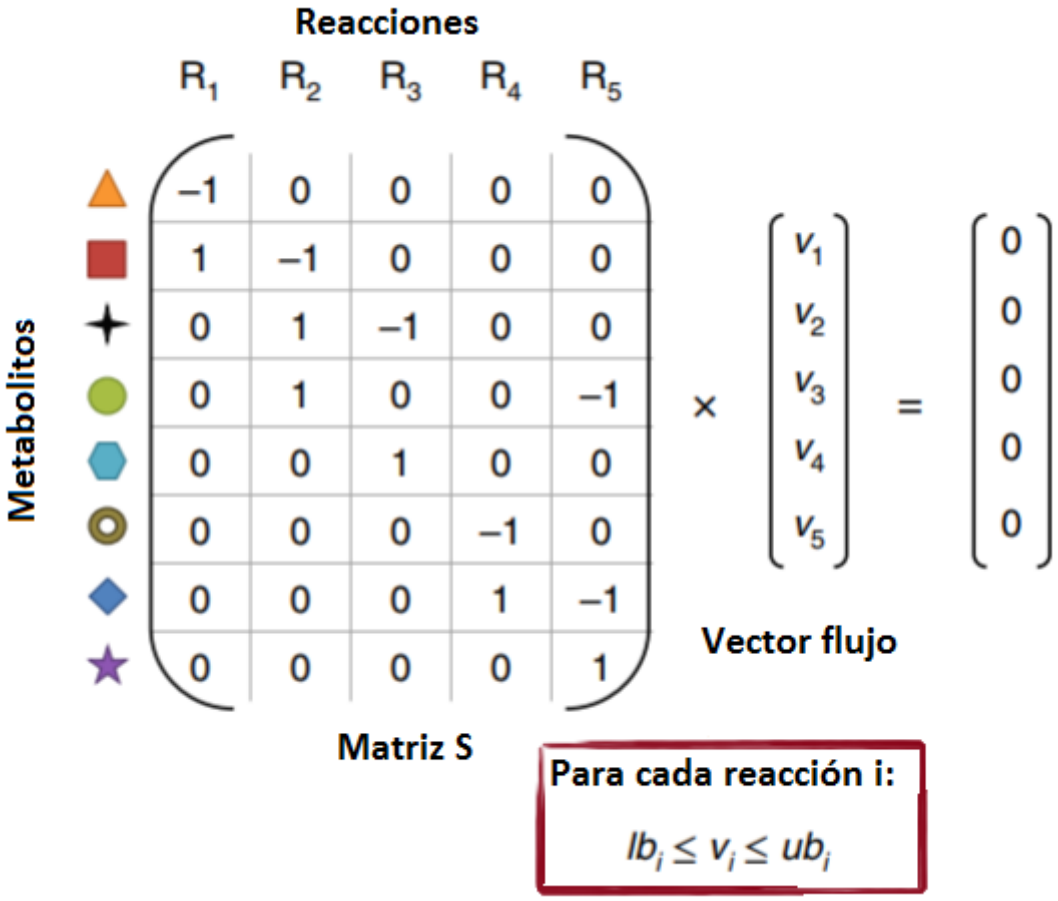

Figura 5 Esquema representativo de los componentes de la matriz S, los vectores de flujo y las restricciones usadas para obtener la solución del flujo objetivo. La matriz S contiene las reacciones metabólicas con los metabolitos implicados y sus coeficientes estequiométricos. Cada reacción está asociada a un flujo v, formando un flujo vector. Estos flujos están restringidos con un límite superior e inferior, lb y ub respectivamente. El método FBA usa la programación lineal para obtener el valor de los flujos cuando optimizamos para una reacción objetivo cuyo flujo es v_{obj}. [1]

3.1.3 Análisis de variabilidad de flujos (FVA)

Tras obtener el valor máximo teórico del flujo objetivo de la reacción de interés mediante el análisis FBA, podemos proceder a analizar la variabilidad potencial en el valor de alguno de los flujos que siga permitiendo el funcionamiento de nuestra función objetivo. Este análisis se conoce como FVA (*Flux Variability Analysis*). A diferencia del análisis FBA que solo nos devuelve el valor del flujo de la función objetivo, a partir del análisis FVA obtendremos un rango de valores de flujo posibles para cada valor de la función objetivo.

Los valores del análisis FVA pueden ser obtenidos siguiendo 2 pasos, primero aplicamos una restricción que asegure un flujo mínimo en la función objetivo, definido normalmente como un porcentaje del valor máximo obtenido en el FBA. Luego, FVA optimizará los flujos para cada reacción en la red, esto permitirá la identificación de cada valor mínimo y máximo de cada flujo de las reacciones en la red que permiten un flujo determinado de la función objetivo. [35]

3.1.4 Análisis de comparación de distribuciones de flujo (FDCA)

El análisis de comparación de distribuciones de flujo FDCA (flux distribution comparison analysis) es un método construido sobre FBA que permite identificar genes potenciales para mejorar cepas de organismos [6]. El proceso consiste en maximizar la formación de biomasa y del producto a generar (en este caso, GAs) bajo las mismas condiciones. Tras ésto, se analizan los flujos obtenidos tras el análisis FBA. Estas dos distribuciones de flujo se comparan para encontrar reacciones con diferencias significativas. Las distribuciones de flujo quedan definidas como:

$$v_{dif} = v_{GA} - v_{biomasa}$$

Donde v_{GA} es la distribución de flujos cuando GA se define como función objetivo, $v_{biomasa}$ es la distribución de flujos cuando la función para la formación de biomasa (la BOF) está establecida como función objetivo y v_{dif} es el vector que representa la diferencia entre estas dos distribuciones de flujo.

Se considerará para una determinada reacción que las diferencias son signficativas si se encuentra una reacción *i* de $v_{i,biomasa}$ con alto valor mientras que la misma reacción en $v_{i,GA}$ se mantiene 0. Esto permite identificar enzimas que podrían ser eliminadas para desviar la distribución de flujo de la biomasa hacia la formación de GA.

Para evaluar si $v_{i,biomasa}$ tiene un valor suficientemente alto, se pueden seguir distintos criterios. Generalmente, cuanto más bajo es el valor adoptado, más genes potenciales a ser descartados se obtienen. En este caso se considerarán signficativas todas aquellas reacciones con un flujo superior a un 5% del valor máximo (en valor absoluto) de la reacción de $v_{i,biomasa}$ cuando el flujo de $v_{i,GA}$ es 0.

3.1.5 Software disponible

Existen distintos paquetes de *software* que son capaces de trabajar con el método de restricciones COBRA, siendo el principal y más usado CobraToolbox, implementado en MatLab. En este trabajo se ha fijado trabajar con Python que es un lenguaje de programación interpretado el cual, al igual que MatLab, es multiparadigma. A diferencia de MatLab, Python posee una licencia de código abierto lo cual indica que es gratuito y que está basado en colaboración abierta. En el anexo 1 se adjunta la tabla donde se muestran los distintos paquetes de software que pueden ser implementados en distintos lenguajes de programación. Los paquetes de software pueden ser distribuidos como archivos comprimidos (zip,tar) o como *version-controlled repositories* (git, svn). Estas últimas permiten a muchos desarrolladores de software trabajar en un proyecto común sin necesidad de compartir una misma red [36]. La clasificación "todos" en los sistemas operativos se refiere que son funcionales en Windows, Mac y Linux. Actualmente se dispone de 3 paquetes distintos que pueden trabajar con COBRA en Python, pero debido a la escasa información sobre cómo trabaja Scrumpy y la falta de actualización realizada en él, fue rechazado para este estudio.

3.1.6 Comparación COBRApy y CBMPy

COBRApy forma parte del proyecto openCOBRA, que pretende proveer a los investigadores acceso fácil a la metodología COBRA. El proyecto openCOBRA se inició con COBRAtoolbox para MATLAB pero ahora incluye módulos basados en Python y Julia [34]. Tiene compatibilidad con Python 2.7 y 3.4 (debido a que no todas las dependencias necesarias están actualizadas a Python 3.7). Sin embargo, no está aún respaldado por Anaconda, complemento que facilita los trabajos de programación [37]. Cuenta con una instalación relativamente sencilla, puesto que la mayoría de paquetes pueden ser instalados a través del paquete instalador de Python pip, pero la necesidad de dependencias de distintos desarrolladores puede generar algún problema de compatibilidad. COBRApy cuenta con una función para comprobación de errores en la instalación, pero debido a la falta de información proveída en la identificación de errores y el bajo número de usuarios activos en la red que reporte sus errores en busca de *feedback* son posibles obstáculos para nuevos usuarios en el método.

PySCeS CBMPy es otra plataforma para el análisis de modelos basados en restricciones [38]. Está diseñada usando los principios desarrollados en el proyecto de simulación software PySCeS que son la usabilidad, la flexibilidad y la accesibilidad [39]. Al igual que COBRApy, permite el análisis FBA, FVA, análisis de red y la edición de modelos entre otros. Pese a que sólo está soportado en Python 2.7, tiene una instalación simple y a prueba de errores de compatibilidad en la instalación [40] ya que está incluído en paquetes como Python(x,y) y posee soporte en Anaconda. Además, es compatible con módulos como Spyder.

Para comprobar el correcto funcionamiento de los paquetes y obtener una primera comparación en velocidad, se ha realizado un test en Python 2.7.10 usando como consola IPython 2.4.1. Los parámetros analizados en el test son el tiempo de importación del paquete, tiempo de importación de un modelo metabólico y la optimización para el flujo objetivo del mismo. En este caso se ha usado el modelo e_coli_core.xml para *Escherichia coli* str. K-12 substr. MG1655 obtenido de http://bigg.ucsd.edu/models/e_coli_core. Para comprobar que los tiempos de importación de los paquetes COBRApy y CBMPy son parecidos y que estos no afecten a la resolución del problema lineal, se ha aplicado el código que se adjunta en el anexo 2 de este trabajo. Bajo los parámetros establecidos, COBRApy resultó más rápido que CMBPy, por lo que decidimos utilizar este paquete para futuros análisis.

3.2 Programación para la biosíntesis de giberelinas

La ruta bioquímica de la síntesis de GAs empieza desde difosfato de geranilo (grdp) vía difosfato de isopentenilo (ipdp). En plantas, la unidad básica de isoprenoide se genera por dos rutas, ácido mevalónico (MVA) en el citoplasma y metileritritol fosfato (MEP) en los plastos. En hongos, sin embargo, la única ruta biosintética disponible es la ruta MVA, que provee ipdp para la síntesis de todos los terpenoides, incluyendo las GAs. Podemos distinguir dos partes de la ruta MVA: la parte superior, donde se genera (R)-mevalonato a partir de acetil coenzima A (acetil-SCoA), y la parte inferior, donde se genera ipdp a partir de (R)-mevalonato. Una vez el primer terpeno ha sido generado, los demás son generados a partir de la adición de unidades de ipdp.

3.2.1 Vía superior de la ruta biosintética de ácido mevalónico (MVA)

La parte superior de la ruta MVA, está presente sin diferencias en eucariotas, arqueas y bacterias [41], presentando los siguientes 3 pasos metabólicos que pueden ser observados en la figura 6:

1. En una condensación de Claisen dos ésteres, en presencia de una base fuerte, reaccionan dando lugar a un β-cetoéster [42]. De manera similar a la condensación de Claisen, dos equivalentes de acetil-SCoA (funcionando como tioésters) reaccionan para formar acetoacetil-SCoA (éster de ácido acetoacético).
2. En una reacción aldólica se forman enlaces carbono-carbono gracias a la tautomería ceto-enólica del enol de una cetona y un aldehído. Siguiendo este patrón, acetoacetil-SCoA reacciona con una molécula de acetil-SCoA como carbono nucleofílico para dar HMG-CoA (3-hydroxi-3- methil-glutaril-CoA).
3. Gracias a una reducción enzimática, HMG-CoA puede ser reducido en presencia de agua a (R)-mevalonato o (R)-ácido mevalónico.

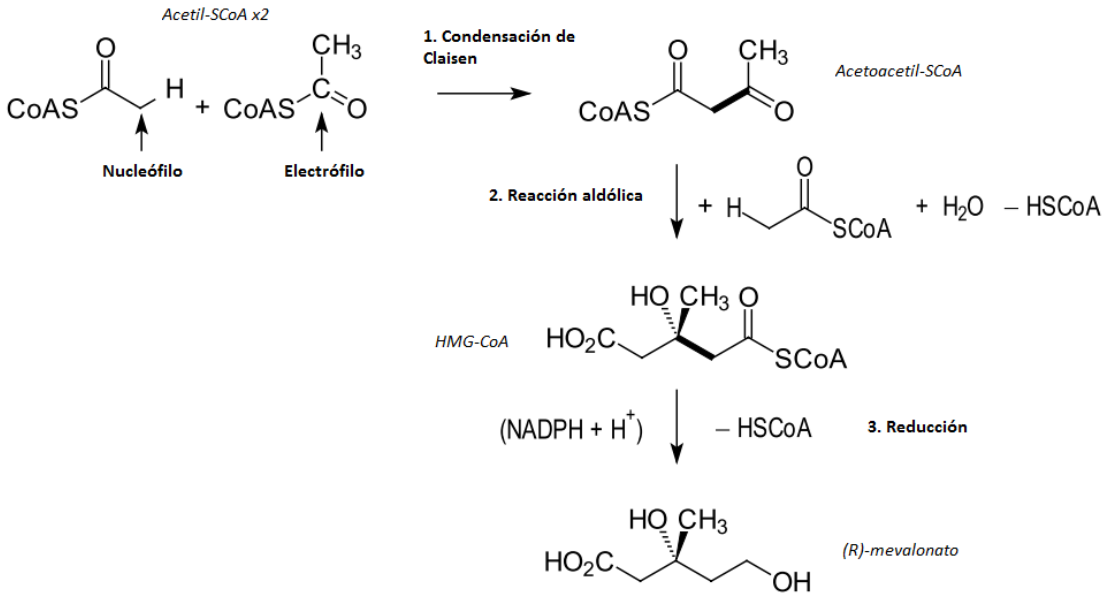

Figura 6 Vía superior de la ruta MVA, donde acetil-SCoA acaba convirtiéndose en (R)-mevalonato en 3 reacciones metabólicas. [43]

3.2.2 Vía inferior MVA

La vía inferior de MVA presenta varias variantes, pero debido a que el objeto de este trabajo se centra en eucariotas, esa será la ruta estudiada. La ruta puede resumirse en 4 pasos hasta obtener grdp y puede observarse gráficamente en la figura 7

1. La doble fosforilación del ácido mevalónico gracias al adenosín trifosfato (ATP) da lugar al ácido mevalónico difosfato.
2. Posteriormente es descarboxilado y deshidratado a isopentenil difosfato (ipdp)
3. ipdp puede isomerizarse en presencia de una isomerasa que contenga grupos SH a pirofosfato de dimetilalilo o difosfato de dimetilalilo (dmpp).
4. Una vez obtenida la unidad básica de isoprenos, los terpenos pueden ser empezados a construir. El grupo alílico electrofílico del dmpp (marcado en rojo en la figura 7) conecta con el grupo metileno nucleofílico del ipdp (marcado en azul en la figura 7) dando lugar a grdp.

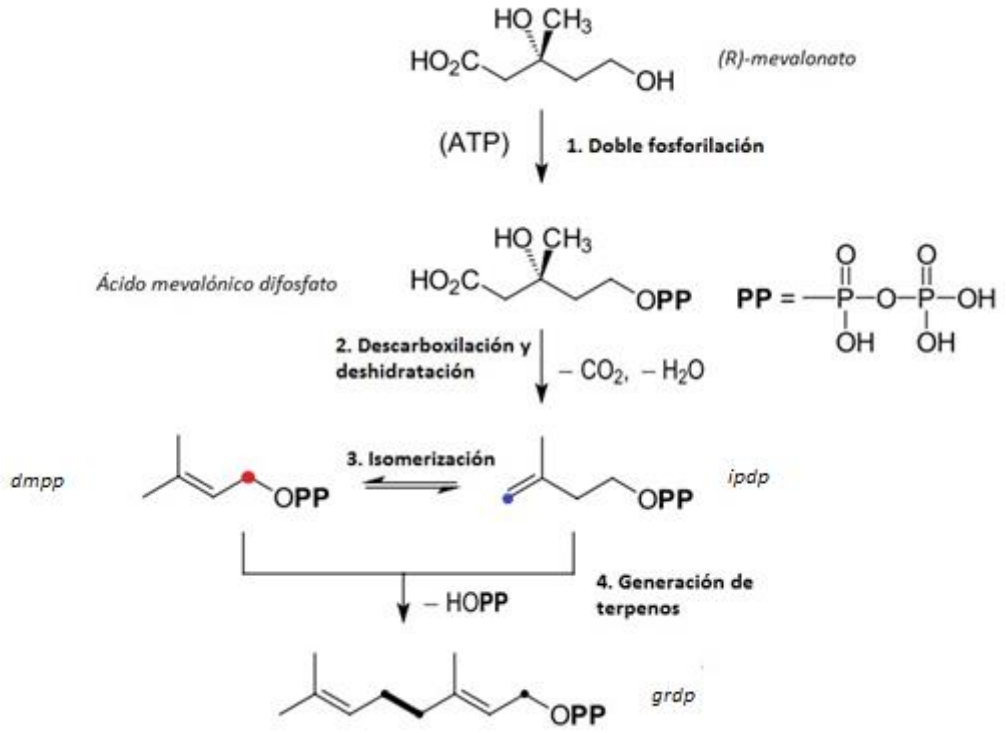

Figura 7 Vía inferior de la ruta MVA, dónde (R)-mevalonato acaba convirtiendose en grdp, un monoterpeno. A partir de la adición de ipdp a grdp se pueden generar terpenos de cadena más larga. En rojo se observa el grupo alílico electrofílico del dmpp que conecta con el el grupo metileno nucleofílico del ipdp marcado en azul. [43]

3.2.4 Reacciones de intercambio con el medio

Como se ha mencionado antes, en el modelo iMM904, el primer metabolito que encontramos para la síntesis de giberelinas, es el geranilgeranil difosfato (ggdp). En el caso de tratar de optimizar la producción de ggdp la función objetivo será la actividad de la enzima farnesiltranstransferasa (FRTT) que cataliza la conversión de difosfato de farnesilo (frdp) a ggdp. No obstante, debido a que el método COBRA no permite la acumulación de metabolitos, si un metabolito no tiene una reacción que permita eliminarlo, el programa lo entenderá como un callejón sin salida y no podrá ser optimizado. Es por eso que, cuando tratamos de optimizar, sin previa adición de ninguna otra reacción y programando en el medio cantidades ilimitadas de glucosa y oxígeno, no se obtiene ningún flujo. Estudiando más a fondo el primer problema intentamos dilucidar qué es lo que sucede viendo los metabolitos predecesores, ruta descrita en la figura 8. Para conseguir algún flujo distinto de cero en esta ruta, debemos optimizar la reacción geraniltranstransferasa (GRTT). Debido a que es una reacción intermedia, podemos observar qué es lo que ocurre con el flujo de metabolitos resultantes. El flujo de la reacción toma la ruta indicada en amarillo en la figura 8. Se puede observar que, en lugar de producir ggdp, toma la ruta para producir escualeno (sql) que entra al retículo endoplasmático generando S escualeno 2 3 epóxido (Ssq23epx) y NADP; y acaba por producir lanosterol (lanost) que es transferido al espacio extracelular. Cabe destacar, que para la producción de Ssq23epx del cual proviene el lanost, puede tomar dos reacciones, SQLEr y SQLErx, generando la primera además de Ssq23epx, NADP, y la segunda nad. Los metabolitos y reacciones involucrados en esta ruta, se ven detallados en la tabla 3.

Metabolitos BiGG	**Nombre descriptivo metabolitos**	**Reacciones BiGG**	**Descripción reacciones**
dpmev	Difosfomevalonato	DPMVD	Difosfomevalonato descarboxilasa
ipdp	Difosfato de isopentenilo	IPDDI	Isopentenilo-difosfato D-isomerasa
dmpp	Dimetilalil difosfato	DMATT	Dimetilaliltranstransferasa
grdp	Difosfato de geranilo	GRTT	Geraniltranstransferasa
frdp	Difosfato de farnesilo	FRTT	Farnesiltranstransferasa
ggdp	Difosfato de geraniol-geraniol	SQLS	Escualeno sintetasa
sql	Escualeno	SQLter	Transporte retículo endoplasmático escualeno

Ssq23epx	S escualeno 2 3 epóxido	SQLEr	Escualeno epoxidasa en retículo endoplasmático NADP
lanost	Lanosterol	SQ23EPXter	Transporte retículo endoplasmático S escualeno 2 3 epóxido
		LNSTLS	Lanosterol sintetasa
		LANOSTt	Lanosterol transporte reversible
		EX_lanost_e	Lanosterol intercambio

Tabla 3 Resumen de los metabolitos y las reacciones involucradas en el metabolismo para la producción de ggdp y lanost. La nomenclatura usada es la que podemos encontrar en la base de datos BiGG y, por lo tanto, preestablecidas en el modelo

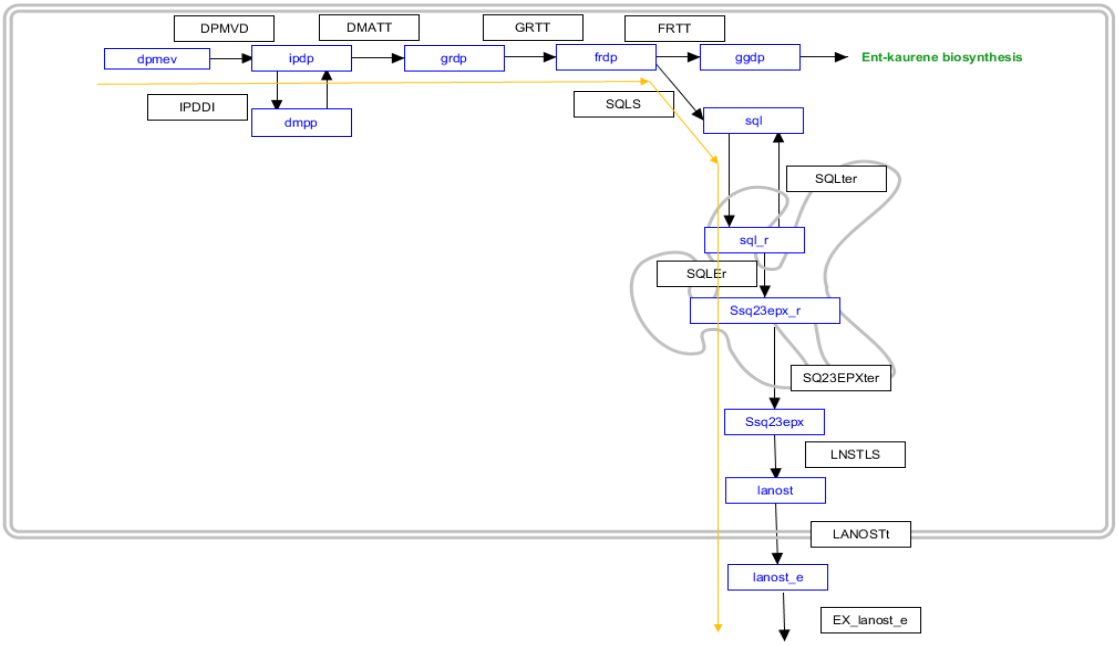

Figura 8 Metabolitos y reacciones implicadas en la producción de ggdp y la expulsión de lanost. La nomenclatura puede ser consultada en la tabla 3. En la figura puede verse representado en color amarillo el flujo de metabolitos que desarrolla el programa al optimizar la reacción GRTT. También pueden verse representados el citosol, el retículo endoplasmático y el espacio extracelular.

De esta manera, con el modelo por defecto, no puede optimizarse una reacción como FRTT ya que los metabolitos resultantes no tienen una pseudoreacción para ser eliminados. Se entiende pues, que al optimizar la reacción GRTT para la obtención de frdp, los metabolitos resultantes pasarán a formar parte de la ruta metabólica más corta para salir del sistema, a través de la reacción EX_lanost_e. De igual manera, tras la escritura de las reacciones responsables de la síntesis de GA se deben escribir las pseudorreacciones correspondientes a la eliminación de estos metabolitos con el fin de poder optimizar su producción.

3.2.3 De difosfato de geranilo a GA

En el modelo iMM904 encontramos descritas tanto la ruta de mevalonato como las reacciones necesarias para llegar a ggdp. En la figura 9 podemos observar como la formación de terpenos de cadena más larga se produce gracias a la adición de difosfatos de isopentenilo, en las reacciones DMATT, GRTT y FRTT. Una vez obtenido ggdp, será necesario añadir las reacciones correspondientes a la biosíntesis de diterpenos para poder llegar a producir GA (tabla 4). En este caso, la ruta a implementar es la usada por *Gibberella fujikuroi* m567 [44] y la información proporcionada por KEGG y *uniprot.org*.

Enzima	**Gen**	**Reacción**
Bifuncional ent-kaureno sintasa	cps/ks	EntKS
ent-kaureno oxidasa [EC:1.14.14.86]	P450-4	EntKO1
ent-kaureno oxidasa [EC:1.14.14.86]	P450-4	EntKO2
ent-kaureno oxidasa [EC:1.14.14.86]	P450-4	EntKO3
ácido ent-kaurenoico monooxigenasa [EC:1.14.14.107]	P450-1	EntKAO1
ácido ent-kaurenoico monooxigenasa [EC:1.14.14.107]	P450-1	EntKAO2
citocromo P450 – 1	P450-1	P450-1_1, P450-1_2
citocromo P450 – 2	P450-2	P450-2_1, P450-2_2, P450-2_3
citocromo P450 – 3	P450-3	P450-3_1, P450-3_2
2-oxoglutarato dioxigenasa dependiente o DES	DES	des

Tabla 4 Reacciones a añadir para completar la ruta para la biosíntesis de GA

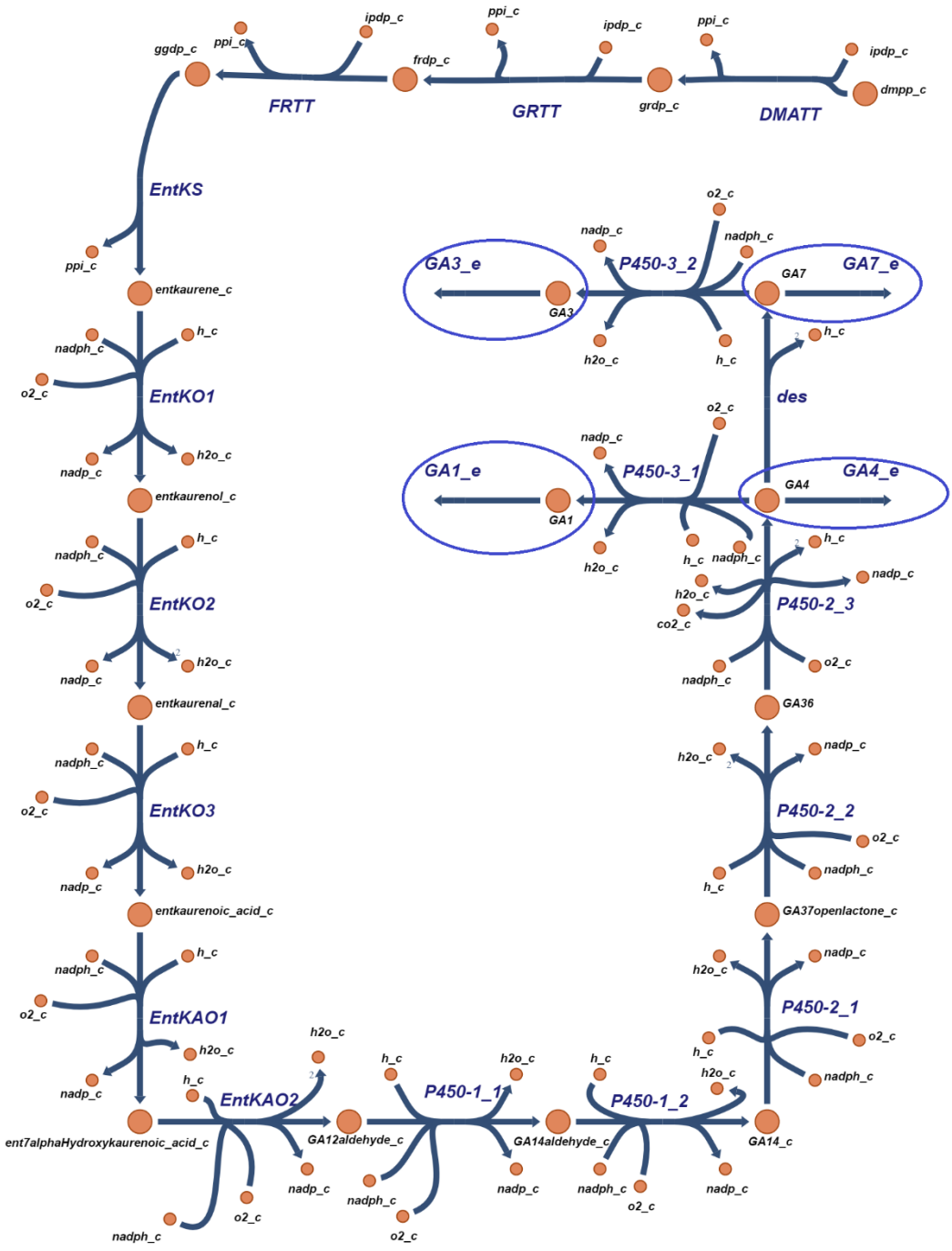

Figura 9 Ruta metabólica para la síntesis de GA1, GA4, GA7 y GA3 en Saccharomyces cerevisiae *basada en la ruta descubierta en Gibberella fujikuroi. En la figura se muestran las reacciones y los metabolitos involucrados desde la formación del primer terpeno (geranil difosfato).*

Se ha observado que los genes responsables de la biosíntesis de GA en *Gibberella fujikoroi* está organizada en un *cluster* formado por 7 genes localizados en el cromosoma 4. Utilizaremos este *cluster* como modelo para nuestros cálculos, al tratarse de la ruta de origen fúngico mejor estudiada. En este *cluster* se incluye un gen específico para la síntesis de geranilgeranil difosfato (ggs2), un gen responsable para la bifuncionalidad de la síntesis de ent-copalilo difosfato y la síntesis de kaureno (cps/ks), una desaturasa que permite producir GA7 a partir de GA4 (2-oxoglutarato dioxigenasa dependiente o DES) ([45]) y cuatro citocromos P450 monooxigenasas (P450-1,P450-2, P450-3 y P450-4) [46, 47].

La ruta bioquímica la síntesis de GA empieza desde difosfato de geranilo-geranilo via difosfato de isopentenilo, la unidad básica de construcción de todos los terpenoides e isoprenoides [48]. Como se ha descrito antes, en fungi la ruta para la biosíntesis de unidades de ipdp es la de mevalonato (MVA), que provee de ipdp para la síntesis de todos los terpenoides, incluyendo las GAs. El primer producto intermedio específico de GA, producido en dos pasos en plantas superiores y bacterias y en un paso en fungi desde ggdp, es el ent-kaureno. Este producto será oxidado en la posición C-19 vía ent-kaurenol y ent-kaurenal produciendo ácido ent-kaurenoico, que seguirá siendo oxidado para generar ácido ent-7α-hidroxikaurenoico. Una oxidación final en la posición C-6β permitirá la formación de GA12-aldehído. Hasta este punto, las rutas metabólicas entre hongos, plantas superiores y bacterias son bastante similares, siendo la única diferencia la producción de ent-kaureno desde ggdp [15]. Sin embargo, en *G*. *fujikuroi*, GA12-aldehído es primeramente 3β-hidrolizado para producir GA14-aldehído, el cuál es oxidado en la posición C-7 para formar GA14 [47, 50]. En este punto, GA14 será convertido en GA4 por oxidación, obteniendo así una giberelina de 19 carbonos: GA4. Ésta es la primera GA biológicamente activa que obtenemos, la cual es desaturada a GA7. G7 puede dar lugar a GA3 mediante hidroxilación en el carbono 13. GA1 se formará como una reacción secundaria a partir de GA4 con una hidroxilación en el carbono 13.

En el anexo 2.3 se adjunta el código empleado para la adición de los metabolitos y reacciones añadidas.

3.3 Algunos conceptos previos a los resultados

3.3.1 Medio de cultivo

Para poder llevar a cabo el análisis de balance de flujos se requiere definir un medio de cultivo y, por lo tanto, las pseudorreacciones de intercambio con el medio que va a proporcionar los metabolitos para llevar a cabo la optimización de la función objetivo.

Para el cultivo de levaduras *in-vivo* puede usarse *Standard Minimal Medium* (SD) el cual es un medio de cultivo que incluye *yeast nitrogen base* (YNB) sin aminoácidos y sin sulfato de amonio, glucosa como única fuente de carbono y agua. YNB contiene todos los nutrientes esenciales para el crecimiento de las levaduras excepto los aminoácidos, el nitrógeno y los carbohidratos, por eso su adición es requerida. Alguno de estos nutrientes contenidos en YNB se incluyen en el modelo (como sulfato, hierro, fosfato o sodio), sin embargo, debido a su baja presencia en las rutas metabólicas de dicho modelo, se supondrán ilimitados o no limitantes. Inspeccionando el cultivo por defecto en el modelo iMM904, se puede observar qué nutrientes de entrada se están considerando y cuáles son sus flujos de entrada máximos, como podemos apreciar en la tabla 5.

Metabolitos extracelulares	**Metabolitos extracelulares formato BiGG**	**Límite de entrada [mmol/h·gDW]**
Hidrógeno	h_e	ilimitado
Oxígeno	o2_e	2
Sulfato	so4_e	ilimitado
Fosfato	pi_e	ilimitado
Potasio	k_e	ilimitado
Sodio	na1_e	ilimitado
Agua	h2o_e	ilimitado
Amonio	nh4_e	ilimitado
Hierro	fe2_e	ilimitado
D-glucosa	glc__D_e	10

Tabla 5 Metabolitos en el medio de cultivo por defecto del modelo iMM904 y sus respectivos límites superiores.

3.3.2 Configurando medio aeróbico

Como se ha podido observar en la tabla 5, el oxígeno y la glucosa son los únicos componentes que no se han considerado ilimitados y que, por lo tanto, determinen la cantidad máxima de GAs que podrán ser sintetizadas. Siendo la glucosa la fuente de carbohidratos, resulta interesante poder observar su equilibrio con la cantidad de producto generado. Con respecto al oxígeno es de esperar que, trabajando en un medio aeróbico, los resultados obtenidos sean superiores a los de un medio con oxígeno limitado. La oxigenación adecuada del medio es un problema pragmático que dependerá en gran medida de las características del biorreactor. En el presente trabajo asumiremos condiciones aeróbicas constantes. Para ello es necesario modificar el límite inferior de la reacción de intercambio de oxígeno *EX_o2_e* de -2 a -9999.

3.3.3 Función objetivo de biomasa

Como se ha descrito antes, la BOF es aquella pseudorreacción que simula la producción de biomasa a partir de todas aquellas reacciones que se consideran esenciales para el crecimiento del organismo. En el caso del modelo iMM904, la ecuación se define en la ecuación 3, donde el nombre de los metabolitos están definidos según el formato de BiGG models (http://bigg.ucsd.edu/universal/reactions/BIOMASS_SC5_notrace). La información para la obtención de la pseudorreación es proveída por MetaNetX (https://www.metanetx.org/equa_info/MNXR96303)

$$
\begin{aligned}
&0.4588\ ala\_L\_c\ +\ 0.046\ amp\_c\ +\ 0.1607\ arg\_L\_c \\
&\quad +\ 0.1017\ asn\_L\_c\ +\ 0.2975\ asp\_L\_c \\
&\quad +\ 59.276\ atp\_c\ +\ 0.0447\ cmp\_c \\
&\quad +\ 0.0066\ cys\_L\_c\ +\ 0.0036\ damp\_c \\
&\quad +\ 0.0024\ dcmp\_c\ +\ 0.0024\ dgmp\_c \\
&\quad +\ 0.0036\ dtmp\_c\ +\ 0.1054\ gln\_L\_c \\
&\quad +\ 0.3018\ glu\_L\_c\ +\ 0.2904\ gly\_c \\
&\quad +\ 0.5185\ glycogen\_c\ +\ 0.046\ gmp\_c \\
&\quad +\ 59.276\ h2o\_c\ +\ 0.0663\ his\_L\_c \\
&\quad +\ 0.1927\ ile\_L\_c\ +\ 0.2964\ leu\_L\_c \\
&\quad +\ 0.2862\ lys\_L\_c\ +\ 0.0507\ met\_L\_c \\
&\quad +\ 0.1339\ phe\_L\_c\ +\ 0.1647\ pro\_L\_c \\
&\quad +\ 0.00099\ ribflv\_c\ +\ 0.1854\ ser\_L\_c \\
&\quad +\ 0.02\ so4\_c\ +\ 0.1914\ thr\_L\_c \\
&\quad +\ 0.0234\ tre\_c\ +\ 0.0284\ trp\_L\_c \\
&\quad +\ 0.102\ tyr\_L\_c\ +\ 0.0599\ ump\_c \\
&\quad +\ 0.2646\ val\_L\_c\ +\ 1.1348\ 13BDglcn\_c \\
&\quad +\ 0.0007\ ergst\_c\ +\ 0.8079\ mannan\_c\ +\ 6e \\
&\quad -\ 06\ pa\_SC\_c\ +\ 6e-05\ pc\_SC\_c\ +\ 4.5e \\
&\quad -\ 05\ pe\_SC\_c\ +\ 1.7e-05\ ps\_SC\_c\ +\ 5.3e \\
&\quad -\ 05\ ptd1ino\_SC\_c\ +\ 6.6e-05\ triglyc\_SC\_c \\
&\quad +\ 0.0015\ zymst\_c \\
&\quad \rightleftharpoons\ 59.276\ adp\_c\ +\ 58.70001\ h\_c \\
&\quad +\ 59.305\ pi\_c\ +\ \mathbf{1\ biomass}
\end{aligned}
\tag{3}
$$

4. Resultados

4.1 Resultados FBA

Una vez configurado el medio de cultivo adecuadamente, mediante la optimización del modelo usando CobraPy, se puede obtener una tabla con los resultados de flujos de entrada, de salida y de la función objetivo, siendo los "IN FLUXES" los flujos de los nutrientes de entrada, los "OUT FLUXES" los flujos para las reacciones de metabolitos expulsados al medio y "OBJECTIVES" el flujo o los flujos para las reacciones objetivo. En todos los casos siguientes, la glucosa va a ser el reactivo limitante pues es el único nutriente no configurado como ilimitado. Las unidades se expresan en mmol/h·gDW, i.e. mmol entre horas y gramos de masa seca. Esto nos permite adquirir una perspectiva estequiométrica del balance.

Realizando un primer análisis de la producción de biomasa podemos observar la cantidad máxima producida tal y como se ha expresado en la ecuación 3 con los nutrientes de entrada configurados. Esto es importante pues, en caso de querer optimizar el flujo del crecimiento y la producción de giberelinas, la BOF (*biomass objective function*) va a tener que ser limitada para poder desviar parte del flujo a las otras reacciones. Los flujos pueden observarse en la figura 10.

IN FLUXES		OUT FLUXES		OBJECTIVES	
o2_e	24.3	h2o_e	36.6	BIOMASS_SC5_...	0.975
glc__D_e	10	co2_e	25		
nh4_e	5.46	h_e	4.9		
pi_e	0.193				
so4_e	0.0754				

Figura 10 Optimización para la síntesis de biomasa. Los valores se expresan en mmol/h•gDW y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

Ignorando la función objetivo de biomasa, podemos optimizar simplemente la reacción de las giberelinas. Optimizando para GA (GA4, GA3, GA1, GA7) individualmente, podemos observar diferencias entre productividad. GA4 y GA7 reportan un flujo de 0.676 mmol/h•gDW y GA3 y GA1 reportan un flujo de 0.673 mmol/h•gDW. Dado que la diferencia de producción entre ellas es negligible (entre GA4 que, siendo la primera GA producida es la que mayor flujo produce, y GA3 que, siendo la que más reacciones requiere, produce el menor flujo hay una diferencia del

0,44%), optimizaremos simplemente para una de ellas. En el caso de la figura 11 puede observarse el flujo máximo de producción de GA3. Cabe destacar que este flujo obtenido con la optimización de una sola GA es el máximo que puede obtenerse incluso combinando la producción de múltiples GA, como puede verse en la figura 12.

IN FLUXES		OUT FLUXES		OBJECTIVES	
o2_e	20	h2o_e	51.3	GA3_e	0.673
glc__D_e	10	co2_e	19		
nh4_e	9.42	h_e	10.1		
pi_e	4.04				

Figura 11 Optimización para la síntesis de GA3. Nótese que los resultados para la producción de otras giberelinas de nera individual no comportaría cambios significativos, y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

IN FLUXES		OUT FLUXES		OBJECTIVES	
o2_e	19.6	h2o_e	51.2	GA4_e	0.2
glc__D_e	10	co2_e	18.8	GA1_e	0.0749
nh4_e	9.45	h_e	9.57	GA3_e	0.2
pi_e	4.05			GA7_e	0.2

Figura 12 Optimización para la síntesis de múltiples GA. Con el fin de poder obtener flujo en la producción de distintas GA debe imponerse un límite en éstas, de otro modo el flujo se desviaría en las reacciones más tempranas. En este caso el límite superior de estas reacciones ha sido impuesto a modo arbitrario como 0.2 en GA3_e, GA7_e y GA4_e a modo de ejemplo. La diferencia del flujo total es de un 0.3%, y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

En caso de limitar la producción de biomasa al 50% de su máximo valor y optimizar para la producción de todas las GAs configuradas observamos que solo llega a producirse GA4 pues al llegar el flujo a esta reacción es desviado inmediatamente al espacio extracelular impidiendo que llegue a las otras reacciones objetivo. Esto puede verse ejemplificado en la figura 13 donde el flujo de biomasa se ha limitado y todas las reacciones de intercambio de giberelinas con el medio han sido configuradas como objetivo.

IN FLUXES		OUT FLUXES		OBJECTIVES	
o2_e	21.7	h2o_e	43.9	GA3_e	0
glc__D_e	10	co2_e	21.7	GA4_e	0.34
nh4_e	7.49	h_e	6.88	GA7_e	0
pi_e	2.14			GA1_e	0
so4_e	0.0377			BIOMASS_SC5_...	0.487

Figura 13 Optimización para limitar la biomasa al 50% y todas las GA programadas en el modelo. Solo se obtiene flujo en GA4 mientras que en las otras permanece nulo, y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

Una vez más, se puede apreciar en la figura 14 que si mantenemos el límite del 50% en la pseudorreacción para la producción de biomasa y se añade solo una GA como objetivo, en este caso GA3, el resultado es prácticamente el mismo que el obtenido en el caso anterior.

IN FLUXES		OUT FLUXES		OBJECTIVES	
o2_e	22	h2o_e	43.9	GA3_e	0.338
glc__D_e	10	co2_e	21.8	BIOMASS_SC5_...	0.487
nh4_e	7.47	h_e	7.53		
pi_e	2.13				
so4_e	0.0377				

Figura 14 Optimización para limitar la biomasa al 50% y GA3.

En la figura 15 se aprecia mejor la producción de GA a nivel individual con distintos límites para la biomasa . Solo se ha tenido en cuenta una GA (GA3) debido a la similitud en los resultados obtenidos.

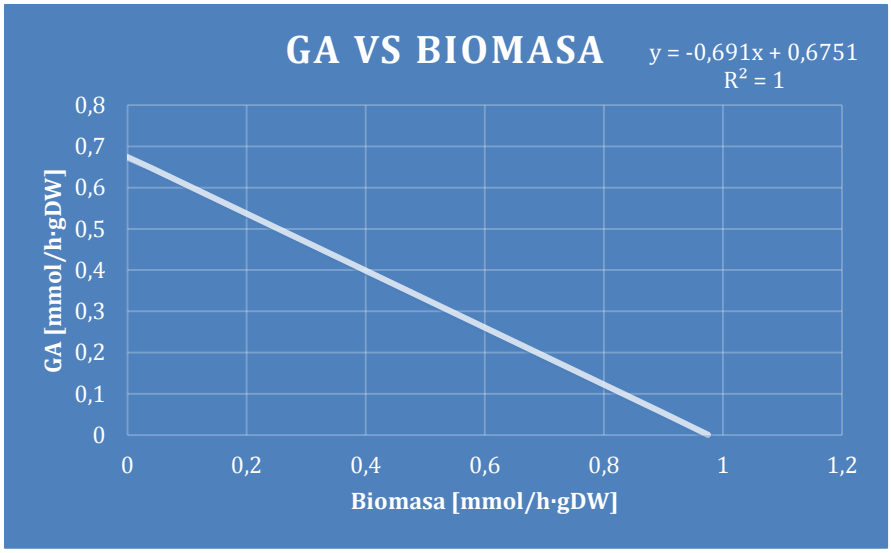

Figura 15 Producción de GA según distintos niveles de producción de biomasa. Los resultados han sido obtenidos mediante la reducción del límite superior en el flujo de la BOF

4.2 Resultados FVA

Pese a que a partir de las simulaciones de FBA siempre se va a obtener el mismo flujo usando CobraPy, los resultados no son únicos. Esto se debe a que múltiples estados de flujo pueden dar lugar al mismo óptimo. Determinando una fracción del óptimo del 95%, se pueden obtener cuánto pueden variar los flujos de los metabolitos involucrados. Para el caso de optimizar para biomasa, observamos en la figura 16 cuánto podemos reducir los nutrientes de entrada sin variar más de un 95% la función objetivo. Se observa que, debido a la posible variación de los flujos de entrada, aparecen multitud de flujos de salida potenciales que, pese a que el flujo actual de estos en el análisis FBA, podrían aparecer con la variación de las cantidades de nutrientes. Esta información se puede verificar en la figura 17, dónde el límite de entrada de oxígeno se ha limitado a -22.7 mmol/h·gDW (el límite mínimo propuesto por el método FVA) y el flujo de biomasa obtenido es superior a 0.926 mmol/h·gDW, que es el 95% del flujo máximo. También se observa, en este caso, la aparición de etanol (*etoh_e*) y formiato (*for_e*) como producto de rutas alternativas tomadas al disminuir la cantidad previamente tomada como óptima de oxígeno. Nótese que, debido a la limitación de la entrada de glucosa, que actúa como reactivo limitante, los límites superiores propuestos por el FVA no son aplicables. Cabe destacar que en los análisis FVA (figuras 16, 17, 18 y 19) la gran variedad de rutas alternativas que puede tomar el modelo para lograr la optimización de la función objetivo se refleja en los múltiples flujos de

salida potenciales que aparecen. En las figuras 16-19, sin embargo, no se muestran todos los flujos de salida posibles, ya que son prácticamente todos aquellos que pueden intercambiar un metabolito con el compartimento extracelular.

IN FLUXES			OUT FLUXES			OBJECTIVES	
id	Flux	Range	id	Flux	Range	BIOMASS_SC5_...	0.975
o2_e	24.3	[22.7, 26]	h2o_e	36.6	[34.4, 40.1]		
glc__D_e	10	[9.5, 10]	co2_e	25	[21.6, 26.7]		
nh4_e	5.46	[5.18, 8.56]	h_e	4.9	[4.14, 8.04]		
pi_e	0.193	[0.183, 0.91]	so3_e	0	[0, 2.65]		
so4_e	0.0754	[0.0716, 2.72]	glx_e	0	[0, 2.55]		
			gly_e	0	[0, 1.81]		
			urea_e	0	[0, 1.69]		

Figura 16 Análisis FVA para la producción de biomasa, y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

IN FLUXES		OUT FLUXES		OBJECTIVES	
o2_e	22.7	h2o_e	34.4	BIOMASS_SC5_...	0.927
glc__D_e	10	co2_e	24.5		
nh4_e	5.19	h_e	4.67		
pi_e	0.183	etoh_e	1.1		
so4_e	0.0717	for_e	0.00479		

Figura 17 Análisis FBA para la producción de biomasa con el oxígeno limitado según el análisis FVA para no desviarse más de un 5% de la producción óptima. Debido a la disminución de oxígeno se muestran la aparición de los nuevos metabolitos etanol (etoh_e) y formiato (for_e), y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

Al aplicar este método para la producción únicamente de GA3 obtenemos resultados similares

IN FLUXES			OUT FLUXES			OBJECTIVES	
id	Flux	Range	id	Flux	Range	GA3_e	0.673
o2_e	20	[18.7, 22]	h2o_e	51.3	[48.3, 54.2]		
glc__D_e	10	[9.5, 10]	co2_e	19	[16.7, 21]		
nh4_e	9.42	[8.95, 12.5]	h_e	10.1	[9.08, 13.1]		
pi_e	4.04	[3.83, 4.57]	urea_e	0	[0, 1.77]		
so3_e	0	[0, -2.72]	gly_e	0	[0, 1.73]		
			glx_e	0	[0, 1.69]		
			ac_e	0	[0, 1.49]		

Figura 18. Análisis FVA para la producción de GA3, y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

En cuanto al resultado de la producción combinada de GA3 con el flujo de la BOF limitado a un 50%

IN FLUXES			OUT FLUXES			OBJECTIVES	
id	Flux	Range	id	Flux	Range	GA3_e	0.339
o2_e	22	[20.3, 24.4]	h2o_e	43.9	[40.3, 47.3]	BIOMASS_SC5_...	0.487
glc__D_e	10	[9.39, 10]	co2_e	21.8	[18.4, 24.3]		
nh4_e	7.47	[6.89, 10.9]	h_e	7.53	[6.31, 10.9]		
pi_e	2.13	[1.88, 2.75]	so3_e	0	[0, 3.1]		
so4_e	0.0376	[0.0273, 3.13]	glx_e	0	[0, 2.99]		
			gly_e	0	[0, 2.17]		

Figura 19 Análisis FVA para la producción de biomasa y GA3, y se clasifican en flujos de entrada (IN FLUXES), flujos de salida (OUT FLUXES) y flujos de la reacción objetivo (OBJECTIVES).

4.2 Resultados FDCA

Tras la obtención de $v_{biomasa}$ (Figura 20), v_{GA} (realizado para la producción de GA3)(Figura 21) y, se ha procedido a calcular v_{dif} (Figura 22). Para poder discernir las diferencias signficativas, se ha encontrado el valor máximo (en valor absoluto) en las distribuciones de flujo de $v_{biomasa}$ entre aquéllas en las que v_{GA} presenta valor 0. El valor máximo obtenido ha sido de 8,52 mmol/h·gDW, por lo tanto se han tenido en cuenta como significativas todas aquellas con un valor superior a 0,426 mmol/h·gDW (en valor absoluto) (Figura 23). En la tabla 6 se pueden observar las reacciones consideradas como significativas y sus respectivos valores.

Número de las reacciones	**Identificador BiGG**	**Nombre de la enzima**	**GPR**
354	CSm	Citrato sintasa	YNR001C or YPR001W
1060	MDHm	Malato deshidrogenasa mitocondrial	YKL085W
156	AKGDam	Oxoglutarato deshidrogenasa lipoamida	YDR148C and YIL125W and YFL018C
157	AKGDbm	Oxoglutarato deshidrogenasa dihidrolipoamida S succiniltransferasa	YDR148C and YIL125W and YFL018C
809	GCC2cm_copy1	Complejo de escisión de glicina lipoamida mitocondrial	YDR019C and YMR189W and YAL044C and YFL018C
1459	SUCOASm	Succinate CoA ligase ADP formando	YGR244C and YOR142W
108	ACONT	Aconitato hidratasa	YLR304C

326	CITtam	Transporte de citrato mitocondrial	YBR291C or UNKNOWN
962	ICDHyr	Isocitrato deshidrogenasa (NADP)	YLR174W
852	GLUt7m	Glutamato transporte uniportador mitocondrial	YPR021C
972	ILETAm	Isoleucina transaminasa mitocondrial	YHR208W
978	ILEtmi	Transporte de isoleucina de las mitocondrias al citosol	-
45	3MOPtm	3 Metil 2 oxopentanoato transporte difusión mitocondrial	-
971	ILETA	Isoleucina transaminasa	YJR148W
110	ACONTm	Aconitato hidratasa	YLR304C or YJL200C
1148	NDPK2	Nucleósido-difosfato quinasa (ATP: UDP)	YKL067W
797	GALUi	UTP-glucosa-1-fosfato uridililtransferasa (irreversible)	YKL035W or YHL012W
1258	PGMT	Fosfoglucomutasa	YKL127W or YMR105C
961	ICDHxm	Isocitrato deshidrogenasa NAD	YNL037C and YOR136W
3	13GS	1 3 beta glucan sintasa	*
963	ICDHym	Isocitrato deshidrogenasa NADP	YDL066W
1577	BIOMASS_SC5_notrace	Función objetivo de biomasa establecida en el modelo iMM904	-
426	DOLPMMer	Dolilfosfato manosa proteína manosiltransferasa reticular endoplásmica	YDL093W or YOR321W or (YDL095W and YAL023C) or YJR143C or YGR199W
427	DOLPMTcer	Dolilfosfato D manosiltransferasa	YPR183W
428	DOLPt2er	Transporte reticular endoplásmico de fosfato de dolichol a través del simport de protons	-
1050	MAN1PT	Manosa 1 fosfato guanililtransferasa	YDL055C
1051	MAN6PI	Manosa-6-fosfato isomerasa	YER003C
1052	MANNANter	Transporte del retículo endoplásmico de Mannan por difusión	-
1288	PMANM	Fosfomannomutasa	YFL045C
1195	OAAt2m	Transporte de oxaloacetato mitocondrial	YKL120W
97	ACLSm	Acetolactato sintasa mitocondrial	YCL009C and YMR108W

400	DHAD1m	Dihidroxiácido deshidratasa 2 3 dihidroxi 3 metilbutanoato mitocondrial	YJR016C
1008	KARA1im	Acetohidroxiácido isomeroreductasa mitocondrial	YLR355C
827	GLCS2	Glucógeno sintasa UDPGlc	**
164	ALATA_L	L-alanina transaminasa	YDR111C
116	ACS	Acetil-CoA sintetasa	YLR153C
194	ALDD2y	Aldehído deshidrogenasa (acetaldehído NADP)	YPL061W
1376	PYRDC	Piruvato descarboxilasa	YLR134W or YGR087C or YLR044C

Tabla 6 Reacciones consideradas como significativas (reportan más de 0,426 mmol/h·gDW en v_{diff} mientras en v_{GA} reportan un flujo nulo. Se muestran ordenadas por orden de significancia.

**(YLR342W and YCR034W and YMR307W) or (YLR342W and YCR034W and YMR215W) or (YMR306W and YCR034W and YLR343W) or (YMR306W and YCR034W and YOL030W) or (YGR032W and YCR034W and YOL030W) or (YLR342W and YCR034W and YOL132W) or (YGR032W and YCR034W and YMR215W) or (YGR032W and YCR034W and YLR343W) or (YMR306W and YCR034W and YMR215W) or (YMR306W and YCR034W and YOL132W) or (YLR342W and YCR034W and YOL030W) or (YLR342W and YCR034W and YLR343W) or (YGR032W and YCR034W and YOL132W) or (YGR032W and YCR034W and YMR307W) or (YMR306W and YCR034W and YMR307W)*

***(YJL137C and YFR015C) or (YKR058W and YLR258W) or (YKR058W and YFR015C) or (YJL137C and YLR258W)*

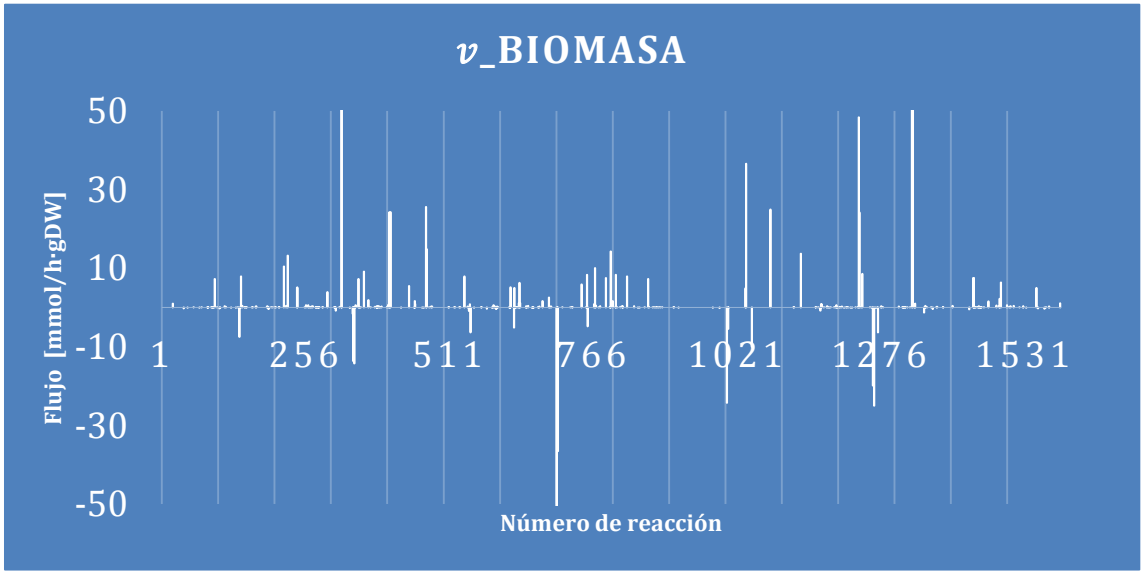

Figura 20 Distribución de flujos $v_{biomasa}$ mediante FBA.

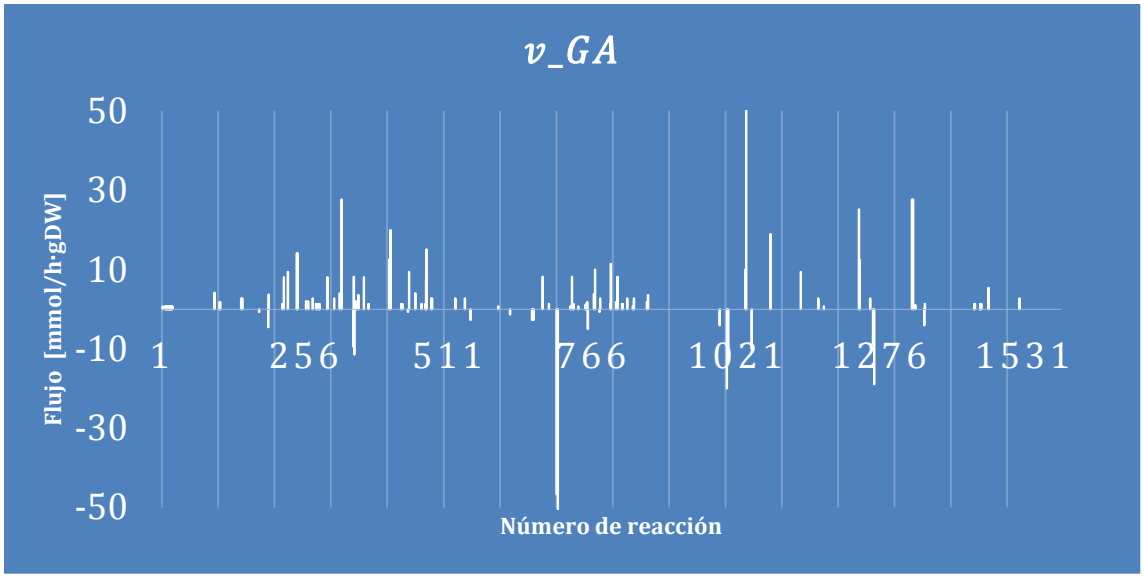

Figura 21 Distribución de flujos v_{GA} mediante FBA. Para la obtención de esta distribución GA3 fue programada como función objetivo

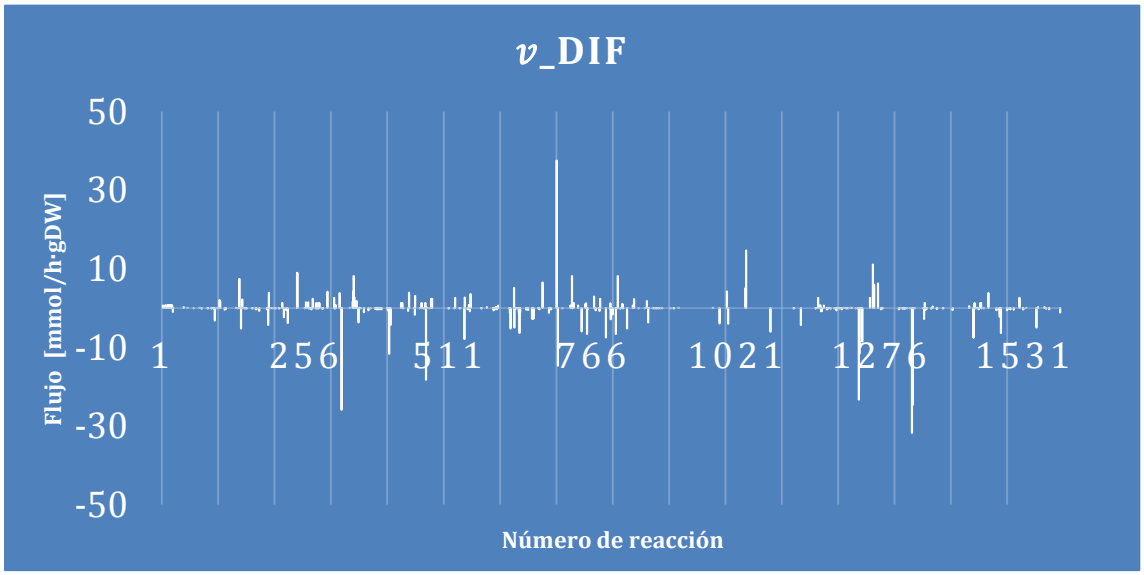

Figura 22 Distribución de flujos v_{diff} mediante FBA. El resultado de la diferencia entre la distribución v_{GA} y $v_{biomasa}$

Figura 23 Distribución de flujos v_{diff} mediante FBA. Solo aparecen las reacciones consideradas significativas, i.e. las reacciones en las que en v_{GA} tienen un valor de 0 y en $v_{biomasa}$ un valor superior a 0.426 mmol/h·gDW

Para poder identificar si hay algún gen esencial entre ellos se ha simulado *knockear* cada una de las enzima que cataliza cada reacción de forma individual y se ha procedido a optimizar el modelo teniendo en cuenta la producción de biomasa como objetivo. Obtener un valor de 0 al optimizar implica que la enzima cataliza una reacción es esencial y no puede ser *eliminada* . No obstante, es posible que el *knockeo* simultáneo de dos reacciones también produzca un *output* nulo. En este caso encontramos cuatro interacciones detalladas en la tabla 7.

(ACONT,	ACONTm)
(ILETAm,	3MOPtm)
(ILETA,	ILEtmi)
(ILETA,	ILETAm)

Tabla 7 Interacciones de knockeos esenciales. Al eliminar una pareja de reacciones de la tabla el flujo de biomasa producido se vuelve nulo

Tras marcar las enzimas “ILETAm”, “"ILEtmi" y "ACONT" como esenciales para evitar interacciones, *knockeamos* todas las otras reacciones no esenciales (tabla 8). Observamos que la cantidad máxima de biomasa producida decrece en un 18,4% (Figura 24) mientras que la producción de GA3 sigue siendo la misma (0,673 mmol GA3/gDW · h). No obstante, si limitamos la producción de biomasa a un 50% de su valor máximo en las condiciones predeterminadas, observamos que la producción de GA3 se ve afectada (Figura 25).

Reacciones esenciales	Reacciones no esenciales (propuesta de KO)
DOLPMTcer	ILETA
NDPK2	CITtam
DOLPMMer	GLUt7m
DHAD1m	ALDD2y
DOLPt2er	GALUi
CSm	AKGDam
ACLSm	GLCS2
ACS	OAAt2m
MANNANter	PYRDC
BIOMASS_SC5_notrace	3MOPtm
MAN1PT	GCC2cm_copy1
MAN6PI	AKGDbm
13GS	ICDHym
PMANM	MDHm
ICDHyr	ALATA_L
PGMT	ACONTm
KARA1im	SUCOASm
ILETAm	ICDHxm
ILEtmi	
ACONT	

Tabla 8 Reacciones esenciales y no esenciales según el análisis FDCA

```
IN FLUXES          OUT FLUXES    OBJECTIVES
-----------------  ------------  ----------------------
o2_e       26.6    h2o_e   39.7  BIOMASS_SC5_...  0.796
glc__D_e   10      co2_e   27.2
nh4_e       5.05   h_e      5.2
pi_e        0.157  4abz_e   0.6
so4_e       0.0615
```

Figura 24 Producción de biomasa tras el knockeo de todas las reacciones no esenciales especificadas en la tabla 8. Se observa que la producción de biomasa ha decrecido respecto a las condiciones predeterminadas. Se observa también la producción de 4abz_e (4-aminobenzoato)

```
IN FLUXES          OUT FLUXES        OBJECTIVES
-----------------  ----------------  ----------------------
o2_e       25.4    h2o_e     44.5    GA3_e            0.274
glc__D_e   10      co2_e     25.4    BIOMASS_SC5_...  0.487
nh4_e       6.7    h_e        6.7
pi_e        1.74   ala__L_e   0.144
so4_e       0.0376
```

Figura 25 Producción de biomasa limitada al 50% de su máximo teórico con las condiciones predeterminadas y GA3 tras el knockeo de todas las reacciones no esenciales especificadas en la tabla 8. La producción de biomasa alcanza el límite impuesto y la producción de GA3 decrece respecto las condiciones predeterminadas. También se observa la producción de L-alanina

5. Discusión

Las giberelinas pueden ser producidas mediante síntesis química [50] o por extracción de plantas [51], pero estos métodos no son económicamente viables [52]. Industrialmente, estas fitohormonas son producidas mediante fermentación sumergida (SmF) usando *Gibberella fujikuroi.*

La producción de GA3 mediante el cultivo en fermentación sumergida es controlada por regulaciones catabólicas e inhibición de sustrato [52]. Se aplican distintas técnicas con el fin de mejorar el rendimiento del proceso, como la adición de fuentes de carbono que sean lentamente consumidas [53] o alimentando el sistema lentamente con fuentes de carbono inmediatamente asimilable [54].

Sin embargo, el mayor obstáculo económico relativo en este método es la baja concentración en la que se encuentra en el producto, lo que tiene grandes costes para purificarlo. Además, el coste de separar las células por centrifugación o microfiltración llega a representar entre un 48 y un 76% del coste total de producción mediante SmF [55].

Una nueva técnica que mejora los aspectos anteriores es la fermentación en estado sólido (SSF). Ésta ha demostrado tener numerosas ventajas económicas sobre el proceso SmF en la producción de biomasa microbiana y de metabolitos [56]. La principal diferencia con SmF es que el desarrollo de microorganismos se realiza en un ambiente con baja actividad de agua en un material insoluble, que actúa como soporte físico y como fuente de nutrientes.[52]

Pese a las mejoras actualmente implantadas, la producción no supera los 0,03 g GA3 /(kg CP·h) [57] teniendo en cuenta distintos tipos de sustrato y biorreactores. Pese a que la unidad de flujos empleada en cobra (mmol/gDW·h) es una aproximación a la realidad, en este trabajo se ha obtenido un flujo de producción máximo de GA3 de 0,673 mmol GA3/ gDW·h, lo que se traduce en 233g de GA3 por cada kg de biomasa y hora. Esta comparación tiene en cuenta que en los datos obtenidos de la bibliografía consultada hacen referencia a la cantidad producida en función del peso seco del sustrato (pulpa cítrica). Además el comportamiento de levaduras y hongos filamentosos es distinto.

El uso de herramientas biotecnológicas en levaduras para la producción de GAs podría ser la solución a los problemas planteados anteriormente con un mayor rendimiento. La obtención de un producto seleccionado genéticamente y concentrado evade el mayor problema de la producción SmF. Además, el rápido crecimiento de las levaduras y su menor necesidad de aporte nutricional también ayudaría a economizar el proceso de producción.

6. Conclusiones

A partir de los análisis realizados sobre el modelo, se observa que la producción de GAs en *S. cerevisiae* es teóricamente posible y compatible con la producción de biomasa. Según el modelo, teniendo en cuenta la entrada de 10 mmol de glucosa como la única fuente de carbono, se observa que el 9,75% de este flujo puede ser convertido en biomasa cuando ésta se impone como función objetivo. Por otro lado, cuando imponemos la producción de GA3 como función objetivo, observamos que un 6,73% del flujo de carbono es redireccionado hacia su producción. Pese a que estos sean resultados máximos teóricos que pueden verse afectados en el cultivo *in-vivo* debido a la limitación de nutrientes considerados como no limitantes, la potencial toxicidad del compuesto, la aparición de otros metabolitos secundarios o la oxigenación, la aproximación utilizada es muy prometedora en comparación con los métodos actualmente en uso.

No obstante, las cepas naturales de *S. cerevisiae* están optimizadas para producir la máxima cantidad de biomasa, por lo que es necesario redireccionar el flujo de carbono hacia la producción de GA. El *knockeo* de enzimas descritas en el modelo puede ayudar a discernir entre la esencialidad de éstas para una función objetivo determinada. Además, COBRAPy permite el análisis de interacciones entre reacciones. Tras el *knockeo* de las enzimas que catalizan reacciones marcadas como no esenciales (tabla 9), se ha observado una disminución en la producción máxima de biomasa mientras que la producción máxima teórica de GAs se mantiene estable. Ésto puede ser debido a que el modelo se ve obligado a tomar otras rutas menos eficientes, lo que podría indicar una desviación hacia otras rutas en el organismo *in-vivo.* No obstante, no puede apreciarse ese efecto en el modelo con los métodos aplicados *in-silico*.

Propuesta de *KO*	
ILETA	3MOPtm
CITtam	GCC2cm_copy1
GLUt7m	AKGDbm
ALDD2y	ICDHym
GALUi	MDHm
AKGDam	ALATA_L
GLCS2	ACONTm
OAAt2m	SUCOASm
PYRDC	ICDHxm

Tabla 9 Enzinmas propuestas como KO para estudiar la relación que tiene su pérdida de función con la mejora en la producción de GAs. Estas reacciones han resultado de realizar en análisis FDCA considerando significativas todas aquellas reacciones con un flujo superior a un 5% del valor máximo (en valor absoluto).

Cabe destacar también que la eliminación de las enizmas "ILETAm", ""ILEtmi" y "ACONT", participantes en reacciones esenciales de interacción, ha sido arbitraria y se pueden contemplar otras posibilidades.

Aunque se pueda seguir trabajando en el modelo con otros métodos para analizar el efecto de los KO's en la ruta metabólica así como conseguir distintos resultados variando las restricciones ambientales, se ha demostrado el potencial COBRApy en cuanto al análisis de modelos en distintas condiciones y la posibilidad de producción de GAs en *S. cerevisiae.*

7. Bibliografía

[1] Heirendt, L., Arreckx, S., Pfau, T., Mendoza, S. N., Richelle, A., Heinken, A., … Fleming, R. M. T. (2017). Creation and analysis of biochemical constraint-based models: the COBRA Toolbox v3.0. Protocol Exchange, 8(5), 321–324. https://doi.org/10.1038/protex.2011.234

[2] Collakova, E., Yen, J. Y., & Senger, R. S. (2012). Are we ready for genome-scale modeling in plants? Plant Science, 191–192, 53–70. https://doi.org/10.1016/j.plantsci.2012.04.010

[3] Knoop, H., Zilliges, Y., Lockau, W., & Steuer, R. (2010). The Metabolic Network of Synechocystis sp. PCC 6803: Systemic Properties of Autotrophic Growth . Plant Physiology, 154(1), 410–422. https://doi.org/10.1104/pp.110.157198

[4] Bates, D. M., Ellis, B., Smith, C., Irizarry, R., Tierney, L., Huber, W., … Smyth, G. (2004). Bioconductor: open software development for computational biology and bioinformatics. Genome Biology, 5(10), R80. https://doi.org/10.1186/gb-2004-5-10-r80

[5] Ellson, J., Gansner, E., Koutsofios, L., North, S. C., & Woodhull, G. (2002). Graphviz— Open Source Graph Drawing Tools, 483–484. https://doi.org/10.1007/3-540-45848-4_57

[6] Sun, Z., Meng, H., Li, J., Wang, J., Li, Q., Wang, Y., & Zhang, Y. (2014). Identification of novel knockout targets for improving terpenoids biosynthesis in saccharomyces cerevisiae. PLoS ONE, 9(11), 9–14. https://doi.org/10.1371/journal.pone.0112615

[7] Oswald M, Fischer M, Dirninger N, Karst F (2007) Monoterpenoid biosynthesis in Saccharomyces cerevisiae. Fems Yeast Research 7: 413–421.

[8] Takahashi S, Yeo Y, Greenhagen BT, McMullin T, Song L, et al. (2007) Metabolic engineering of sesquiterpene metabolism in yeast. Biotechnology and bioengineering 97: 170–181.

[9] Mo, M. L., Palsson, B., & Herrgård, M. J. (2009). Connecting extracellular metabolomic measurements to intracellular flux states in yeast. BMC Systems Biology, 3, 1–17. https://doi.org/10.1186/1752-0509-3-37

[10] J.D. Connolly, R.A. Hill (Hrsg.), Dictionary of Terpenoids, Vol. 1: Mono- and Sesquiterpenoids, Vol. 2: Di- and higher Terpenoids, Vol. 3: Indexes, Chapman & Hall, London, New York, Tokyo, Melbourne, Madras, 1991.

[11] Eschenmoser, A. (1989). Leopold Ruzicka: from the isoprene rule to the question of the origin of life. Rad Jugosl. Akad. Znan. Umjet., 443(I), 21–67, 2 plates.

[12] https://goldbook.iupac.org/html/T/T06278.html,https://goldbook.iupac.org/html/T/T06279.html

[13] Buckingham J (ed) (1998) Dictionary of natural products on CD-ROM,vol 6.1.Chapman & Hall, London

[14] Davis, E. M., & Croteau, R. (2007). Cyclization Enzymes in the Biosynthesis of Monoterpenes, Sesquiterpenes, and Diterpenes, 209, 53–95. https://doi.org/10.1007/3-540-48146-x_2

[15] Salazar-Cerezo, S., Martínez-Montiel, N., García-Sánchez, J., Pérez-y-Terrón, R., & Martínez-Contreras, R. D. (2018). Gibberellin biosynthesis and metabolism: A convergent route for plants, fungi and bacteria. Microbiological Research, 208(January), 85–98. https://doi.org/10.1016/j.micres.2018.01.010

[16] Ribeiro, M., Citadini, V., Pace, G., Ângela, P. L., Souza, V. De, & Alves, L. D. S. (2010). gibberellin Use of in floriculture. African Journal of Biotechnology, 9(54), 54.

[17] Breitmaier, E. (2006). General Structure : The Isoprene Rule. Terpenes : Importance , General Structure , and Biosynthesis, 1–9. https://doi.org/10.1002/9783527609949.ch1

[18] Bömke, C., & Tudzynski, B. (2009). Diversity, regulation, and evolution of the gibberellin biosynthetic pathway in fungi compared to plants and bacteria. Phytochemistry, 70(15–16), 1876–1893. https://doi.org/10.1016/j.phytochem.2009.05.020

[19] Sawada K (1917) Contributions on Formosan fungi. Trans Nat Hist Soc Formosa 7:131–133

[20] Kurosawa, E., 1926. Experimental studies on the nature of the substance secreted by the "bakanae" fungus. Trans. Nat. Hist. Soc 16, 213–227

[21] Phinney, B.O., 1956. Growth response of single-gene dwarf mutants in maize to gibberellic acid. Proc. Natl. Acad. Sci. 42, 185–189.

[22] Navarro, L., Bari, R., Achard, P., Lisón, P., Nemri, A., Harberd, N. P., & Jones, J. D. G. (2008). DELLAs Control Plant Immune Responses by Modulating the Balance of Jasmonic Acid and Salicylic Acid Signaling. Current Biology, 18(9), 650–655. https://doi.org/10.1016/j.cub.2008.03.060

[23] Ogawa, M. (2003). Gibberellin Biosynthesis and Response during Arabidopsis Seed Germination. The Plant Cell Online, 15(7), 1591–1604. https://doi.org/10.1105/tpc.011650

[24] Silverstone, A.L., Ciampaglio, C.N., Sun, T.P., 1998. The Arabidopsis RGA gene encodes a transcriptional regulator repressing the gibberellin signal transduction pathway. Plant Cell 10, 155–169

[25] Chandler, P.M., Marion-Poll, A., Ellis, M., Gubler, F., 2002. Mutants at the Slender1 locus of barley cv Himalaya. Molecular and physiological characterization. Plant Physiol. 129, 181–190

[26] Boss, P.K., Thomas, M.R., 2002. Association of dwarfism and floral induction with a grape 'green revolution' mutation. Nature 416, 847–850

[27] Peng, J., Richards, D.E., Hartley, N.M., Murphy, G.P., Devos, K.M., Flintham, J.E., et al., 1999. 'Green revolution' genes encode mutant gibberellin response modulators. Nature 400, 256–261

[28] Ikeda, A., Ueguchi-Tanaka, M., Sonoda, Y., Kitano, H., Koshioka, M., Futsuhara, Y., et al., 2001. Slender rice, a constitutive gibberellin response mutant, is caused by a null mutation of the SLR1 gene, an ortholog of the height-regulating gene GAI/RGA/RHT/ D8. Plant Cell 13, 999–1010

[29] Martí, C., Orzáez, D., Ellul, P., Moreno, V., Carbonell, J., Granell, A., 2007. Silencing of DELLA induces facultative parthenocarpy in tomato fruits. Plant J. 52, 865–876

[30] B.A. Halo, A.L. Khan, M. Waqas, A. Al-Harrasi, J. Hussain, L. Ali, et al. Endophytic bacteria (Sphingomonas sp. LK11) and gibberellin can improve Solanum lycopersicum growth and oxidative stress under salinity J. Plant Interact., 10 (2015), pp. 117-125

[31] Waqas, M., Khan, A. L., Kamran, M., Hamayun, M., Kang, S. M., Kim, Y. H., & Lee, I. J. (2012). Endophytic fungi produce gibberellins and indoleacetic acid and promotes host-plant growth during stress. Molecules, 17(9), 10754–10773. https://doi.org/10.3390/molecules170910754

[32] Zhang et al., 2008 C. Zhang, U. Lee, K. Tanabe Hormonal regulation of fruit set, parthenogenesis induction and fruit expansion in Japanese pear Plant Growth Regul., 55 (2008), p. 231.

[33] Knoche et al., 2011 M. Knoche, B.P. Khanal, M. Stopar Russeting and microcracking of 'Golden Delicious' apple fruit concomitantly decline due to gibberellin A4+ 7 application J. Am. Soc. Hortic. Sci., 136 (2011), pp. 159-164

[34] Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. 2013, COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst Bio 7 : 74.

[35] Rawls, K. D., Dougherty, B. V, Blais, E. M., Stancliffe, E., Kolling, G. L., Vinnakota, K., … Papin, J. A. (2019). A simplified metabolic network reconstruction to promote understanding and development of flux balance analysis tools. Computers in Biology and Medicine, 105(December 2018), 64–71. https://doi.org/10.1016/j.compbiomed.2018.12.010

[36] Spolsky, Joel (17 de marzo de 2010). «Distributed Version Control is here to stay, baby». *Joel on Software*. Consultado el 18 de junio de 2010.

[37] Ebrahim, A., Palsson, J. A. L. B. O., & Hyduke, D. R. (2013). {COBRA}py: {CO}nstraints-{B}ased {R}econstruction and {A}nalysis for {P}ython. BMC Syst Biol., 7(74), doi: 10.1186/1752-0509-7-74. https://doi.org/10.1186/1752-0509-7-74

[38] PySCeS Constraint Based Modelling (http://cbmpy.sourceforge.net) Copyright (C) 2009-2017 Brett G. Olivier, VU University Amsterdam, Amsterdam, The Netherlands

[39] Olivier, B. G., Rohwer, J. M., & Hofmeyr, J. H. S. (2005). Modelling cellular systems with PySCeS. Bioinformatics, 21(4), 560–561. https://doi.org/10.1093/bioinformatics/bti046

[40] Olivier, B. G. (2017). CBMPy Documentation.

[41] Miziorko, H. M. (2017). Showing : View Products : Thymine-1-acetic acid Compare Products :, 505(2), 274250. https://doi.org/10.1016/j.abb.2010.09.028.ENZYMES

[42] K. Peter C. Vollhardt (1994). Química Orgánica. Barcelona: Ediciones Omega S.A. ISBN 84-282-0882-4

[43] Breitmaiteer, E. (2006). General Stbreructure : The Isoprene Rule. Terpenes : Importance , General Structure , and Biosynthesis, 1–9. https://doi.org/10.1002/9783527609949.ch1

[44] Tudzynski, B., & Hölter, K. (1998). Gibberellin Biosynthetic Pathway in Gibberella fujikuroi : Evidence for a Gene Cluster Fungal Strains and Culture Conditions. Fungal Genetics and Biology : FG & B, 170, 1087–1845.

[45] Bhattacharya, A., Kourmpetli, S., Ward, D.A., Thomas, S.G., Gong, F., Powers, S.J., et al., 2012. Characterization of the fungal gibberellin desaturase as a 2-oxoglutarate-dependent dioxygenase and its utilization for enhancing plant growth. Plant Physiol. 160, 837–845.

[46] Tudzynski, B., 2005. Gibberellin biosynthesis in fungi: genes, enzymes, evolution, and impact on biotechnology. Appl. Microbiol. Biotechnol. 66, 597–611.;

[47] Tudzynski, B., Hedden, P., Carrera, E., Gaskin, P., 2001. The P450-4 gene of Gibberella fujikuroi encodes ent-kaurene oxidase in the gibberellin biosynthesis pathway. Appl. Environ. Microbiol. 67, 3514–3522.

[48] Sponsel, V.M., Hedden, P., 2010. Gibberellin biosynthesis and inactivation. Plant Hormones. Springer, Netherlands, pp. 63–94

[49] Urrutia, O., Hedden, P., Rojas, M.C., 2001. Monooxygenases involved in GA12 and GA14 synthesis in Gibberella fujikuroi. Phytochemistry 56, 505–511

[50] Hook JM, Mander LN, Rudolf U. (1980). Journal of American Chemistry Society, 102: 6628–29

[51] Kende H. (1967). Preparation of radioactive gibberellin A1 and its metabolism in dwarf peas. Plant Physiology, 42:1612–18

[52] Rodrigues, C., Vandenberghe, L. P. D. S., De Oliveira, J., & Soccol, C. R. (2012). New perspectives of gibberellic acid production: A review. Critical Reviews in Biotechnology, 32(3), 263–273. https://doi.org/10.3109/07388551.2011.615297

[53] Darken MA, Jensen AL, Shu P. (1959). Production of gibberellic acid by fermentation. Applied Microbiology, 7:301–3

[54] Borrow A, Brown S, Jefferys EG, Kessel RJH, Lloyd EC, Lloyd PB. (1964). The kinetics of metabolism of Gibberella fujikuroi in stirred culture. Canadian Journal of Microbiology, 10:407–44

[55] Vass RC, Jefferys EG. (1979). Gibberellic Acid, In-Economic microbiology: secondary products of metabolism. vol 3, AH Rose (ed), Academic Press, London, 421–33

[56] Soccol CR, Vandenberghe LPS, Rodrigues C, Pandey A. (2006). New Perspectives for Citric Acid Production and Application. Food Technology and Biotechnology, 44:141–9

[57] De Oliveira, J., Rodrigues, C., Vandenberghe, L. P. S., Câmara, M. C., Libardi, N., & Soccol, C. R. (2017). Gibberellic Acid Production by Different Fermentation Systems Using Citric Pulp as Substrate/Support. BioMed Research International, 2017. https://doi.org/10.1155/2017/5191046

Anexos

Anexo 1 Tabla de software disponibles para la aplicación de COBRA

Nombre	Implementación	Interfaz	Desarrollo	Distribución	S.O.
COBRA Toolbox	MATLAB (y otros)	Script/narrative	Open source	git	Todos
RAVEN	MATLAB	Script	Open source	git	Todos
CellNetAnalyzer	MATLAB (y otros)	Script/GUI	Closed source	zip	Todos
FBA-SimVis	Java + MATLAB	GUI	Closed source	zip	Windows
OptFlux	Java	Script	Open source	svn	Todos
COBRA.jl	Julia	Script/narrative	Open source	git	Todos
Sybil	Pack R	Script	Open source	zip	Todos
COBRApy	Python	Script/narrative	Open source	git	Todos
CBMPy	Python	Script	Open source	zip	Todos
Scrumpy	Python	Script	Open source	tar	Todos

SurreyFBA	C++	Script/GUI	Open source	zip	Todos
FASIMU	C	Script	Open source	zip	Linux
FAME	Basado en web	GUI	Open source	zip	Todos
PathwayTools	Basado en web	GUI/script	Closed source	zip	Todos
Kbase	Basado en web	Script/narrative	Open source	git	Todos

GUI, graphical user interface; NA, not applicable. The COBRA Toolbox: https://opencobra.github.io/cobratoolbox; RAVEN: https://github.com/ SysBioChalmers/RAVEN; CellNetAnalyzer: https://www2.mpi-magdeburg.mpg.de/projects/cna/cna.html; FBA-SimVis: https://immersive-analytics. infotech.monash.edu/fbasimvis; OptFlux: http://www.optflux.org; COBRA.jl: https://opencobra.github.io/COBRA.jl; Sybil: https://rdrr.io/cran/sybil; COBRApy: http://opencobra.github.io/cobrapy; CBMPy: http://cbmpy.sourceforge.net; SurreyFBA: http://sysbio.sbs.surrey.ac.uk/sfba; FASIMU: http://www.bioinformatics.org/fasimu; FAME: http://f-a-m-e.org; Pathway Tools: http://bioinformatics.ai.sri.com/ptools; KBase: https://kbase.us.

Anexo 2. Código de Python

2.1 Comprobación tiempo de carga del paquete COBRA

Para COBRApy

```
def test_import_cobrapy():
        import time
        start = time.clock()
        import cobra
        import os
        from os.path import join
        stop =time.clock()
        print ("Total time=",stop-start)
```

Para CBMPy

```
def test_import_cbmpy():
        import time
        start = time.clock()
        import cbmpy
        stop =time.clock()
        print ("Total time=",stop-start)
```

2.2 Comprobación tiempo de carga de COBRA y el modelo de *E. coli*

COBRApy

```
def test_cobrapy_ecoli():
   import time
   start = time.clock()
   import cobra
   import os
   from os.path import join
```

```
ecoli_cobrapy=cobra.io.read_sbml_model(join("C:\wheels\cobra","e_coli_core.xml"))
ecoli_cobrapy.summary()
stop = time.clock()
print ("Total time=",stop-start)
```

Para CBMPy

```
def test_cbmpy_ecoli():
        import time
        start=time.clock()
        import cbmpy
        ecoli_cbmpy=cbmpy.CBRead.readCOBRASBML("e_coli_core.xml","C:\wheels\cobra")
        cbmpy.CBGLPK.glpk_analyzeModel(ecoli_cbmpy)
        stop = time.clock()
        print ("Total time=",stop-start)
```

2.3 Código para reacciones, metabolitos y funciones necesarias para la simulación

```
# -*- coding: cp1252 -*-
from __future__ import print_function #Para que funcione
from __future__ import absolute_import #para MOMA
import cobra    #para llamar a cobra
import os   #para poder cargar el modelo
from os.path import join    #para poder cargar el modelo
from cobra.util.solver import linear_reaction_coefficients #para cambiar la función objetivo
from cobra import Model, Reaction, Metabolite #para añadir reacciones y metabolitos
from escher import Builder
from optlang.symbolics import Zero, add
import cobra.util.solver as sutil
import logging
logging.basicConfig()
from cobra.medium import minimal_medium
from cobra.flux_analysis.parsimonious import pfba
from cobra.flux_analysis import flux_variability_analysis
import pandas as pd
from cobra.flux_analysis import (
    single_gene_deletion, single_reaction_deletion, double_gene_deletion,
```

```
   double_reaction_deletion)
pd.options.display.max_rows= 2000
model=cobra.io.read_sbml_model(join("E:\cobra", "saccharomyces.xml"))

##############################################################################################
##################################################

###Create new metabolites
entkaurene_c = Metabolite('entkaurene_c', formula='C20H32', name='Ent-kaurene', compartment='c')
ipdp_e = Metabolite("ipdp_e",formula="C5H9O7P2", name="Isopentenyl diphosphate", compartment="e")
entkaurenol_c =Metabolite('entkaurenol_c', formula='C20H32O', name='Ent-kaurenol', compartment='c')
entkaurenal_c= Metabolite('entkaurenal_c', formula='C20H30O', name='Ent-kaurenal', compartment='c')
entkaurenoic_acid_c = Metabolite('entkaurenoic_acid_c', formula='C20H3O2', name='Ent-kaurenoic acid',
compartment='c')
enthydroxy_c = Metabolite('ent7alphaHydroxykaurenoic_acid_c', formula='C20H30O3', name='ent-7alpha-
Hydroxykaur-16-en-19-oic acid', compartment='c')
GA12aldehyde= Metabolite('GA12aldehyde_c', formula='C20H28O3', name='Gibberellin A12 aldehyde',
compartment='c')
GA14aldehyde= Metabolite('GA14aldehyde_c', formula='C20H28O4', name='Gibberellin A14 aldehyde',
compartment='c')
GA14= Metabolite('GA14_c', formula='C20H28O5', name='Gibberellin A14', compartment='c')
GA37openlactone= Metabolite('GA37openlactone_c', formula='C20H28O6', name='Gibberellin A37 open lactone',
compartment='c')
GA36= Metabolite('GA36', formula='C20H26O6', name='Gibberellin A36', compartment='c')
GA4= Metabolite('GA4', formula='C20H24O5', name='Gibberellin A4', compartment='c')
GA1= Metabolite('GA1', formula='C19H24O6', name='Gibberellin A1', compartment='c')
GA7= Metabolite('GA7', formula='C19H22O5', name='Gibberellin A7', compartment='c')
GA3= Metabolite('GA3', formula='C19H22O6', name='Gibberellin A3', compartment='c')



##############################################################################################
##########################################################
#Create reaction
reaction1 = Reaction('EntKS')
reaction1.name = 'ent-kaurene synthase'
reaction1.subsystem = 'Gibberellin synthesis'
reaction1.lower_bound = 0.  # This is the default
reaction1.upper_bound = 1000.  # This is the default
```

```
reaction1.gene_reaction_rule = 'cps/ks'
model.add_reactions([reaction1])

reaction1.add_metabolites({"ggdp_c": -1.0,
                    "ppi_c": 1.0,
                    entkaurene_c: 1.0,})

reaction2 = Reaction('EntKO1')
reaction2.name = 'Bifunctional ent-kaurene synthase'
reaction2.subsystem = 'Gibberellin synthesis'
reaction2.lower_bound = 0.  # This is the default
reaction2.upper_bound = 1000.  # This is the default
reaction2.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction2])

reaction2.add_metabolites({entkaurene_c: -1.0,
                    "nadph_c": -1.0,
                    "h_c":-1.0,
                    "o2_c":-1.0,

                    entkaurenol_c: 1.0,
                    "nadp_c":1.0,
                    "h2o_c":1.0                    })


reaction3 = Reaction('EntKO2')
reaction3.name = 'Bifunctional ent-kaurene synthase'
reaction3.subsystem = 'Gibberellin synthesis'
reaction3.lower_bound = 0.  # This is the default
reaction3.upper_bound = 1000.  # This is the default
reaction3.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction3])

reaction3.add_metabolites({entkaurenol_c: -1.0,
                    "nadph_c": -1.0,
                    "h_c":-1.0,
                    "o2_c":-1.0,

                    entkaurenal_c: 1.0,
```

```
                "nadp_c":1.0,
                "h2o_c":2.0                 })


reaction4 = Reaction('EntKO3')
reaction4.name = 'Bifunctional ent-kaurene synthase'
reaction4.subsystem = 'Gibberellin synthesis'
reaction4.lower_bound = 0.  # This is the default
reaction4.upper_bound = 1000.  # This is the default
reaction4.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction4])

reaction4.add_metabolites({entkaurenal_c: -1.0,
                "nadph_c": -1.0,
                "h_c":-1.0,
                "o2_c":-1.0,

                entkaurenoic_acid_c: 1.0,
                "nadp_c":1.0,
                "h2o_c":1.0                 })

reaction5 = Reaction('EntKAO1')
reaction5.name = 'Bifunctional ent-kaurene synthase'
reaction5.subsystem = 'Gibberellin synthesis'
reaction5.lower_bound = 0.  # This is the default
reaction5.upper_bound = 1000.  # This is the default
reaction5.gene_reaction_rule = '(P450-1)'
model.add_reactions([reaction5])

reaction5.add_metabolites({entkaurenoic_acid_c: -1.0,
                "nadph_c": -1.0,
                "h_c":-1.0,
                "o2_c":-1.0,

                enthydroxy_c: 1.0,
                "nadp_c":1.0,
                "h2o_c":1.0                 })

reaction6 = Reaction('EntKAO2')
```

```
reaction6.name = 'Bifunctional ent-kaurene synthase'
reaction6.subsystem = 'Gibberellin synthesis'
reaction6.lower_bound = 0.  # This is the default
reaction6.upper_bound = 1000.  # This is the default
reaction6.gene_reaction_rule = '(P450-1)'
model.add_reactions([reaction6])

reaction6.add_metabolites({enthydroxy_c: -1.0,
                    "nadph_c": -3.0,
                    "h_c":-1.0,
                    "o2_c":-1.0,

                    GA12aldehyde: 1.0,
                    "nadp_c":1.0,
                    "h2o_c":2.0                 })

reaction7 = Reaction('P450-1_1')
reaction7.name = 'Bifunctional ent-kaurene synthase'
reaction7.subsystem = 'Gibberellin synthesis'
reaction7.lower_bound = 0.  # This is the default
reaction7.upper_bound = 1000.  # This is the default
reaction7.gene_reaction_rule = '(P450-1)'
model.add_reactions([reaction7])

reaction7.add_metabolites({GA12aldehyde: -1.0,
                    "nadph_c": -1.0,
                    "h_c":-1.0,
                    "o2_c":-1.0,

                    GA14aldehyde: 1.0,
                    "nadp_c":1.0,
                    "h2o_c":1.0                 })

reaction8 = Reaction('P450-1_2')
reaction8.name = 'Bifunctional ent-kaurene synthase'
reaction8.subsystem = 'Gibberellin synthesis'
reaction8.lower_bound = 0.  # This is the default
reaction8.upper_bound = 1000.  # This is the default
reaction8.gene_reaction_rule = '(P450-4)'
```

```
model.add_reactions([reaction8])

reaction8.add_metabolites({GA14aldehyde: -1.0,
                    "nadph_c": -1.0,
                    "h_c":-1.0,
                    "o2_c":-1.0,

                    GA14: 1.0,
                    "nadp_c":1.0,
                    "h2o_c":1.0                    })

reaction9 = Reaction('P450-2_1')
reaction9.name = 'Bifunctional ent-kaurene synthase'
reaction9.subsystem = 'Gibberellin synthesis'
reaction9.lower_bound = 0.  # This is the default
reaction9.upper_bound = 1000.  # This is the default
reaction9.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction9])

reaction9.add_metabolites({GA14: -1.0,
                    "nadph_c": -1.0,
                    "h_c":-1.0,
                    "o2_c":-1.0,

                    GA37openlactone: 1.0,
                    "nadp_c":1.0,
                    "h2o_c":1.0                    })

reaction10 = Reaction('P450-2_2')
reaction10.name = 'Bifunctional ent-kaurene synthase'
reaction10.subsystem = 'Gibberellin synthesis'
reaction10.lower_bound = 0.  # This is the default
reaction10.upper_bound = 1000.  # This is the default
reaction10.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction10])

reaction10.add_metabolites({GA37openlactone: -1.0,
                     "nadph_c": -1.0,
                    "h_c":-1.0,
```

```
                "o2_c":-1.0,

                GA36: 1.0,
                 "nadp_c":1.0,
                "h2o_c":2.0                 })

reaction11 = Reaction('P450-2_3')
reaction11.name = 'Bifunctional ent-kaurene synthase'
reaction11.subsystem = 'Gibberellin synthesis'
reaction11.lower_bound = 0.  # This is the default
reaction11.upper_bound = 1000.  # This is the default
reaction11.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction11])

reaction11.add_metabolites({GA36: -1.0,
                 "nadph_c": -1.0,
                "h_c":-1.0,
                "o2_c":-1.0,

                GA4: 1.0,
                "co2_c":1.0,
                 "nadp_c":1.0,
                 "h_c":2.0,
                "h2o_c":1.0 })


reaction12 = Reaction('P450-3_1')
reaction12.name = 'Bifunctional ent-kaurene synthase'
reaction12.subsystem = 'Gibberellin synthesis'
reaction12.lower_bound = 0.  # This is the default
reaction12.upper_bound = 1000.  # This is the default
reaction12.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction12])

reaction12.add_metabolites({GA4: -1.0,
                 "nadph_c": -1.0,
                "h_c":-1.0,
                "o2_c":-1.0,
```

```
                GA1: 1.0,
                 "nadp_c":1.0,
                "h2o_c":1.0 })

reaction13 = Reaction('des')
reaction13.name = 'Bifunctional ent-kaurene synthase'
reaction13.subsystem = 'Gibberellin synthesis'
reaction13.lower_bound = 0.  # This is the default
reaction13.upper_bound = 1000.  # This is the default
reaction13.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction13])

reaction13.add_metabolites({GA4: -1.0,

                GA7: 1.0,
                 "h_c":2.0})

reaction14 = Reaction('P450-3_2')
reaction14.name = 'Bifunctional ent-kaurene synthase'
reaction14.subsystem = 'Gibberellin synthesis'
reaction14.lower_bound = 0.  # This is the default
reaction14.upper_bound = 1000.  # This is the default
reaction14.gene_reaction_rule = '(P450-4)'
model.add_reactions([reaction14])

reaction14.add_metabolites({GA7: -1.0,
                 "nadph_c": -1.0,
                "h_c":-1.0,
                "o2_c":-1.0,

                GA3: 1.0,
                 "nadp_c":1.0,
                "h2o_c":1.0 })



GA1_e = Reaction('GA1_e')
GA1_e = Reaction('GA1_e')
```

```
GA1_e.name = 'FE'
GA1_e.subsystem = 'Gibberellin synthesis'
GA1_e.lower_bound = 0 # This is the default
GA1_e.upper_bound = 99999  # This is the default
##GA1_e.gene_reaction_rule =
model.add_reactions([GA1_e])

GA1_e.add_metabolites({GA1 :-1.0,})

GA7_e = Reaction('GA7_e')
GA7_e = Reaction('GA7_e')
GA7_e.name = 'FE'
GA7_e.subsystem = 'Gibberellin synthesis'
GA7_e.lower_bound = 0 # This is the default
GA7_e.upper_bound = 99999  # This is the default
##GA7_e.gene_reaction_rule =
model.add_reactions([GA7_e])

GA7_e.add_metabolites({GA7 :-1.0,})

GA4_e = Reaction('GA4_e')
GA4_e = Reaction('GA4_e')
GA4_e.name = 'FE'
GA4_e.subsystem = 'Gibberellin synthesis'
GA4_e.lower_bound = 0 # This is the default
GA4_e.upper_bound = 99999  # This is the default
##GA4_e.gene_reaction_rule =
model.add_reactions([GA4_e])

GA4_e.add_metabolites({GA4 :-1.0,})


GA3_e = Reaction('GA3_e')
GA3_e = Reaction('GA3_e')
GA3_e.name = 'FE'
GA3_e.subsystem = 'Gibberellin synthesis'
GA3_e.lower_bound = 0 # This is the default
GA3_e.upper_bound = 99999  # This is the default
##GA3_e.gene_reaction_rule =
```

```
model.add_reactions([GA3_e])

GA3_e.add_metabolites({GA3 :-1.0,})
##############################################################################################
##########################################################################

#obtener el nombre con sus restricciones para ver cuales estan "activadas"

def checkin():
   i=0
   for i in range(len(model.reactions)):
      reaction = model.reactions[i]
      if str(reaction.compartments) == "set(['e'])" and reaction.lower_bound != 0:
         print(reaction.name)
         print(reaction.bounds)

def changein():
   i=0
   for i in range(len(model.reactions)):
      reaction = model.reactions[i]
      if str(reaction.compartments) == "set(['e'])":
         reaction.lower_bound = -9999
         print(reaction.name)
         print(reaction.bounds)

def change(reaction,value):
   aka = model.reactions.get_by_id(reaction)
   aka.lower_bound=-value

def obj(reac):
   reaction = model.reactions.get_by_id(str(reac))
   model.objective=reaction
   model.optimize()
   model.summary()

def cflux(reac):
    reaction = model.reactions.get_by_id(str(reac))
    print(reaction.flux)
```

```
def prod(metabolite):
    prod = Reaction('metprod')
    prod.name = 'Metabolite production'
    prod.subsystem = 'Gibberellin synthesis'
    prod.lower_bound = -99999 # This is the default
    prod.upper_bound = 99999  # This is the default
    ##prod.gene_reaction_rule =
    model.add_reactions([prod])

    prod.add_metabolites({metabolite :-1.0,})
    model.objective=prod
    model.optimize()
    print(model.summary())

def KO():
    cobra.manipulation.delete_model_genes(model,
["YDR226W","YJR159W","YFR053C","YGL253W","YER081W","YIL074C",])
    model.optimize()
    print(model.summary())

def escher():
    b = Builder(model=model)
    b.display_in_browser()

def out_fun1():
    solutionpfba = pfba(model)
    file = open("biomass.txt","w")
    file.write(str(solutionpfba.fluxes))
    file.close()

def out_fun2():
    model.objective= GA3_e
    solutionpfba = pfba(model)
    file = open("GA3.txt","w")
    file.write(str(solutionpfba.fluxes))
    file.close()

def out_funfba():
```

```
    solutionfba=model.optimize()
    file = open("FBA.txt","w")
    file.write(str(solutionfba.fluxes))
    file.close()

def out_funmoma():
    sol=model.optimize()
    solutionmoma=cobra.flux_analysis.moma(model, solution=sol)
    file = open("biomassmoma.txt","w")
    file.write(str(solutionmoma.fluxes))
    file.close()

def out_funmomaGA():
    model.objective= GA3_e
    sol=model.optimize()
    solutionmoma=cobra.flux_analysis.moma(model, solution=sol)
    file = open("GA3moma.txt","w")
    file.write(str(solutionmoma.fluxes))
    file.close()

def find_genes():
    i=0
    reactions = ["ACt2r",
                 "ACtr",
                 "AGTi",
                 "G6PI3",
                 "GCC2cm_copy1",
                 "GLUK",
                 "ICDHxm",
                 "ICL",
                 "NDPK3",
                 "ORNTA",
                 "SUCCtm",
                 "TALA"]
    for i in range (0,len(reactions)):
        reactioni=model.reactions.get_by_id(reactions[i])
        print(reactioni.name,",", reactioni.gene_reaction_rule)
        normal=reactioni.upper_bound
        reactioni.upper_bound=0
```

```
##      print(reactioni.id, model.optimize(), model.objective.value)
##      reactioni.upper_bound=normal


def find_genes_moma():
  i=0
  reactions = ["AASAD1",
          "ACALDtm",
          "ADHAPR_SC",
          "AGTi",
          "ALCD2x_copy1",
          "ALCD2irm",
          "ASPTA",
          "DDPA",
          "DHORDfum",
          "ETOHtm",
          "FACOAL181",
          "FAS160COA",
          "FAS181",
          "FDH",
          "G6PI3",
          "GALUi",
          "GAT2_SC",
          "GCC2cm_copy1",
          "GK2",
          "GLCS2",
          "GLUK",
          "HEX7",
          "HSDy",
          "ICL",
          "LNS14DM",
          "MDH",
          "MTHFD2i",
          "NDPK3",
          "NDPK8",
          "ORNTA",
          "PPND",
          "SBTD_D2",
          "SBTR",
          "SUCCtm"]
```

```
    for i in range (0,len(reactions)):
        reactioni=model.reactions.get_by_id(reactions[i])
##          print(reactioni.name,",", reactioni.gene_reaction_rule)
        normal=reactioni.upper_bound
        reactioni.upper_bound=0
        print(reactioni.id, model.optimize(), model.objective.value)
##          reactioni.upper_bound=normal
        print(reactioni.id, model.optimize(), model.objective.value)

def genesFDCA():
    i=0
    reactions = ["CSm",
                 "MDHm",
                 "AKGDam",
                 "AKGDbm",
                 "GCC2cm_copy1",
                 "SUCOASm",
                 "ACONT",
                 "CITtam",
                 "ICDHyr",
                 "GLUt7m",
                 "ILETAm",
                 "ILEtmi",
                 "3MOPtm",
                 "ILETA",
                 "ACONTm",
                 "NDPK2",
                 "GALUi",
                 "PGMT",
                 "ICDHxm",
                 "13GS",
                 "ICDHym",
                 "BIOMASS_SC5_notrace",
                 "DOLPMMer",
                 "DOLPMTcer",
                 "DOLPt2er",
```

```
            "MAN1PT",
            "MAN6PI",
            "MANNANter",
            "PMANM",
            "OAAt2m",
            "ACLSm",
            "DHAD1m",
            "KARA1im",
            "GLCS2",
            "ALATA_L",
            "ACS",
            "ALDD2y",
            "PYRDC"]

    for i in range (0,len(reactions)):
        reactioni=model.reactions.get_by_id(reactions[i])
##      print(reactioni.name,",", reactioni.gene_reaction_rule)
        normal=reactioni.upper_bound
        reactioni.upper_bound=0
        print(reactioni.id, model.optimize(), model.objective.value)
##      reactioni.upper_bound=normal

def genesFDCA_ne():
    i=0
    reactions = ["ILETA",
##              "ILETAm",
            "CITtam",
            "GLUt7m",
            "ALDD2y",
            "GALUi",
            "AKGDam",
            "GLCS2",
            "OAAt2m",
            "PYRDC",
            "3MOPtm",
##              "ILEtmi",
            "GCC2cm_copy1",
            "AKGDbm",
            "ICDHym",
```

```
##            "ACONT",
           "MDHm",
           "ALATA_L",
           "ACONTm",
           "SUCOASm",
           "ICDHxm"]



  for i in range (0,len(reactions)):
     reactioni=model.reactions.get_by_id(reactions[i])
##      print(reactioni.name,",", reactioni.gene_reaction_rule)
     normal=reactioni.upper_bound
     reactioni.upper_bound=0
     print(reactioni.id, model.optimize(), model.objective.value)
##      reactioni.upper_bound=normal

def deletions():
  reactions = ["MDHm",
           "AKGDam",
           "AKGDbm",
           "GCC2cm_copy1",
           "SUCOASm",
           "ACONT",
           "CITtam",
           "GLUt7m",
           "ILETAm",
           "ILEtmi",
           "3MOPtm",
           "ILETA",
           "ACONTm",
           "GALUi",
           "PGMT",
           "ICDHxm",
           "ICDHym",
           "DOLPt2er",
           "MAN1PT",
           "MAN6PI",
           "MANNANter",
```

```
        "PMANM",
        "OAAt2m",
        "GLCS2",
        "ALATA_L",
        "ALDD2y",
        "PYRDC"]
  double_reaction_deletion(
    model, model.reactions(reactions), return_frame=True).round(4)


def moma(model, solution=None, linear=True):
  """
  Compute a single solution based on (linear) MOMA.

  Compute a new flux distribution that is at a minimal distance to a
  previous reference solution. Minimization of metabolic adjustment (MOMA) is
  generally used to assess the impact
  of knock-outs. Thus the typical usage is to provide a wildtype flux
  distribution as reference and a model in knock-out state.

  Parameters
  ----------
  model : cobra.Model
    The model state to compute a MOMA-based solution for.
  solution : cobra.Solution, optional
    A (wildtype) reference solution.
  linear : bool, optional
    Whether to use the linear MOMA formulation or not (default True).

  Returns
  -------
  cobra.Solution
    A flux distribution that is at a minimal distance compared to the
    reference solution.

  See Also
  --------
  add_moma : add MOMA constraints and objective
```

```
    """
    with model:
        add_moma(model=model, solution=solution, linear=linear)
        solution = model.optimize()
    return solution



def add_moma(model, solution=None, linear=True):
    r"""Add constraints and objective representing for MOMA.

    This adds variables and constraints for the minimization of metabolic
    adjustment (MOMA) to the model.

    Parameters
    ----------
    model : cobra.Model
        The model to add MOMA constraints and objective to.
    solution : cobra.Solution, optional
        A previous solution to use as a reference. If no solution is given,
        one will be computed using pFBA.
    linear : bool, optional
        Whether to use the linear MOMA formulation or not (default True).

    Notes
    -----
    In the original MOMA [1]_ specification one looks for the flux distribution
    of the deletion (v^d) closest to the fluxes without the deletion (v).
    In math this means:

    minimize \sum_i (v^d_i - v_i)^2
    s.t. Sv^d = 0
         lb_i <= v^d_i <= ub_i

    Here, we use a variable transformation v^t := v^d_i - v_i. Substituting
    and using the fact that Sv = 0 gives:

    minimize \sum_i (v^t_i)^2
    s.t. Sv^d = 0
```

```
    v^t = v^d_i - v_i
    lb_i <= v^d_i <= ub_i

So basically we just re-center the flux space at the old solution and then
find the flux distribution closest to the new zero (center). This is the
same strategy as used in cameo.

In the case of linear MOMA [2]_, we instead minimize \sum_i abs(v^t_i). The
linear MOMA is typically significantly faster. Also quadratic MOMA tends
to give flux distributions in which all fluxes deviate from the reference
fluxes a little bit whereas linear MOMA tends to give flux distributions
where the majority of fluxes are the same reference with few fluxes
deviating a lot (typical effect of L2 norm vs L1 norm).

The former objective function is saved in the optlang solver interface as
``"moma_old_objective"`` and this can be used to immediately extract the
value of the former objective after MOMA optimization.

See Also
--------
pfba : parsimonious FBA

References
----------
.. [1] Segrè, Daniel, Dennis Vitkup, and George M. Church. “Analysis of
    Optimality in Natural and Perturbed Metabolic Networks.”
    Proceedings of the National Academy of Sciences 99, no. 23
    (November 12, 2002): 15112. https://doi.org/10.1073/pnas.232349399.
.. [2] Becker, Scott A, Adam M Feist, Monica L Mo, Gregory Hannum,
    Bernhard Ø Palsson, and Markus J Herrgard. “Quantitative
    Prediction of Cellular Metabolism with Constraint-Based Models:
    The COBRA Toolbox.” Nature Protocols 2 (March 29, 2007): 727.
"""
if 'moma_old_objective' in model.solver.variables:
  raise ValueError('model is already adjusted for MOMA')

# Fall back to default QP solver if current one has no QP capability
if not linear:
  model.solver = sutil.choose_solver(model, qp=True)
```

```
    if solution is None:
        solution = pfba(model)
    prob = model.problem
    v = prob.Variable("moma_old_objective")
    c = prob.Constraint(model.solver.objective.expression - v,
                        lb=0.0, ub=0.0, name="moma_old_objective_constraint")
    to_add = [v, c]
    model.objective = prob.Objective(Zero, direction="min", sloppy=True)
    obj_vars = []
    for r in model.reactions:
        flux = solution.fluxes[r.id]
        if linear:
            components = sutil.add_absolute_expression(
                model, r.flux_expression, name="moma_dist_" + r.id,
                difference=flux, add=False)
            to_add.extend(components)
            obj_vars.append(components.variable)
        else:
            dist = prob.Variable("moma_dist_" + r.id)
            const = prob.Constraint(r.flux_expression - dist, lb=flux, ub=flux,
                                    name="moma_constraint_" + r.id)
            to_add.extend([dist, const])
            obj_vars.append(dist ** 2)
    model.add_cons_vars(to_add)
    if linear:
        model.objective.set_linear_coefficients({v: 1.0 for v in obj_vars})
    else:
        model.objective = prob.Objective(
            add(obj_vars), direction="min", sloppy=True)
################################################################################################
SQLS=model.reactions.get_by_id("SQLS")
FRTT=model.reactions.get_by_id("FRTT")
GRTT=model.reactions.get_by_id("GRTT")
nh4=model.reactions.get_by_id("EX_nh4_e")
pi=model.reactions.get_by_id("EX_pi_e")
so4=model.reactions.get_by_id("EX_so4_e")
so3=model.reactions.get_by_id("EX_so3_e")
##pfba_solution = cobra.flux_analysis.pfba(model)
```

```
glc=model.reactions.get_by_id("EX_glc__D_e")
##glc.lower_bound=-1000

o2=model.reactions.get_by_id("EX_o2_e")
o2.lower_bound=-9999
biomass=model.reactions.get_by_id("BIOMASS_SC5_notrace")
##model.objective = [biomass, GA3_e]
##biomass.objective_coefficient=2
def optimizeGA3():
  with model:
    model.objective = [biomass, GA3_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA3_e_flux = GA3_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA3_e.flux)
      GA3_e.upper_bound = GA3_e.flux - max_GA3_e_flux/50

def optimizeGA1():
  with model:
    model.objective = [biomass, GA1_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA1_e_flux = GA1_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA1_e.flux)
      GA1_e.upper_bound = GA1_e.flux - max_GA1_e_flux/50

def optimizeGA4():
  with model:
    model.objective = [biomass, GA4_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA4_e_flux = GA4_e.flux
    for i in range (0,51):
      model.optimize()
```

```
        print(biomass.flux, GA4_e.flux)
        GA4_e.upper_bound = GA4_e.flux - max_GA4_e_flux/50

def optimizeGA7():
  with model:
    model.objective = [biomass, GA7_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA7_e_flux = GA7_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA7_e.flux)
      GA7_e.upper_bound = GA7_e.flux - max_GA7_e_flux/50

def optimizeGAs():
  with model:
    model.objective = [biomass, GA3_e, GA1_e, GA4_e, GA7_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA7_e_flux = GA7_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA3_e.flux, GA1_e.flux, GA4_e.flux, GA7_e.flux)
      biomass.upper_bound = biomass.flux - max_biomass_flux/50

def optimizebiomassGA3():
  with model:
    model.objective = [biomass, GA3_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA3_e_flux = GA3_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA3_e.flux)
      biomass.upper_bound = biomass.flux - max_biomass_flux/50

def optimizebiomassGA7():
  with model:
    model.objective = [biomass, GA7_e]
```

```
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA7_e_flux = GA7_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA7_e.flux)
      biomass.upper_bound = biomass.flux - max_biomass_flux/50


def optimizebiomassGA1():
  with model:
    model.objective = [biomass, GA1_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA1_e_flux = GA1_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA1_e.flux)
      biomass.upper_bound = biomass.flux - max_biomass_flux/50

def optimizebiomassGA4():
  with model:
    model.objective = [biomass, GA4_e]
    model.optimize()
    max_biomass_flux = biomass.flux
    max_GA4_e_flux = GA4_e.flux
    for i in range (0,51):
      model.optimize()
      print(biomass.flux, GA4_e.flux)
      biomass.upper_bound = biomass.flux - max_biomass_flux/50


##############################################################################################
#########################################################

##opt = model.optimize()
##solution = model.summary()
```

Anexo 3. Out fluxes de FVA

3.1 FVA usando la función objetivo de biomasa como función objetivo

OUT FLUXES

id	Flux	Range			
h2o_e	36.6	[34.4, 40.1	orn_e	0	[0, 0.532]
co2_e	25	[21.6, 26.7	pro__L_e	0	[0, 0.532]
h_e	4.9	[4.14, 8.04	ptrc_e	0	[0, 0.532]
so3_e	0	[0, 2.65]	cys__L_e	0	[0, 0.521]
glx_e	0	[0, 2.55]	hxan_e	0	[0, 0.52]
gly_e	0	[0, 1.81]	thym_e	0	[0, 0.48]
urea_e	0	[0, 1.69]	gua_e	0	[0, 0.467]
ac_e	0	[0, 1.61]	sbt__D_e	0	[0, 0.466]
acald_e	0	[0, 1.33]	3mbald_e	0	[0, 0.457]
pyr_e	0	[0, 1.33]	gam6p_e	0	[0, 0.445]
oaa_e	0	[0, 1.15]	3c3hmp_e	0	[0, 0.431]
ser__L_e	0	[0, 1.14]	iamoh_e	0	[0, 0.431]
etoh_e	0	[0, 1.13]	leu__L_e	0	[0, 0.43]
for_e	0	[0, 1.12]	2mbald_e	0	[0, 0.417]
ala__L_e	0	[0, 1.12]	3mop_e	0	[0, 0.417]
asp__L_e	0	[0, 0.994]	arg__L_e	0	[0, 0.405]
fum_e	0	[0, 0.994]	lys__L_e	0	[0, 0.398]
mal__L_e	0	[0, 0.994]	2mbtoh_e	0	[0, 0.395]
lac__D_e	0	[0, 0.889]	ile__L_e	0	[0, 0.392]
lac__L_e	0	[0, 0.889]	his__L_e	0	[0, 0.384]
succ_e	0	[0, 0.878]	ibutac_e	0	[0, 0.376]
glyc_e	0	[0, 0.795]	4abz_e	0	[0, 0.355]
asn__L_e	0	[0, 0.735]	met__L_e	0	[0, 0.333]
akg_e	0	[0, 0.719]	iamac_e	0	[0, 0.314]
thr__L_e	0	[0, 0.663]	2mbac_e	0	[0, 0.295]
4abut_e	0	[0, 0.656]	pacald_e	0	[0, 0.289]
cit_e	0	[0, 0.656]	tyr__L_e	0	[0, 0.289]
glu__L_e	0	[0, 0.656]	2phetoh_e	0	[0, 0.279]
2hb_e	0	[0, 0.609]	phe__L_e	0	[0, 0.279]
btd_RR_e	0	[0, 0.609]	spmd_e	0	[0, 0.248]
gln__L_e	0	[0, 0.609]	pap_e	0	[0, 0.243]
2mppal_e	0	[0, 0.598]	dttp_e	0	[0, 0.242]
aces_e	0	[0, 0.573]	pnto__R_e	0	[0, 0.226]
xan_e	0	[0, 0.559]	pheac_e	0	[0, 0.225]
ibutoh_e	0	[0, 0.554]	id3acald_e	0	[0, 0.224]
val__L_e	0	[0, 0.552]	ind3eth_e	0	[0, 0.217]
lanost_e	0	[0, 0.0648]	trp__L_e	0	[0, 0.217]
zymst_e	0	[0, 0.0596]	g3pc_e	0	[0, 0.186]
epist_e	0	[0, 0.0568]	Nbfortyr_e	0	[0, 0.133]
fecost_e	0	[0, 0.0568]	ttdca_e	0	[0, 0.124]
ergst_e	0	[0, 0.0563]			

3.2 FVA usando la producción de GA3 como función objetivo

OUT FLUXES

id	Flux	Range
h2o_e	51.3	[48.3, 54.2]
co2_e	19	[16.7, 21]
h_e	10.1	[9.08, 13.1]
so3_e	0	[0, 2.72]
urea_e	0	[0, 1.77]
gly_e	0	[0, 1.73]
glx_e	0	[0, 1.69]
ac_e	0	[0, 1.49]
for_e	0	[0, 1.22]
acald_e	0	[0, 1.2]
pyr_e	0	[0, 1.2]
oaa_e	0	[0, 1.09]
etoh_e	0	[0, 1.08]
ser__L_e	0	[0, 1.01]
ala__L_e	0	[0, 0.996]
fum_e	0	[0, 0.996]
mal__L_e	0	[0, 0.996]
asp__L_e	0	[0, 0.924]
lac__D_e	0	[0, 0.924]
succ_e	0	[0, 0.915]
lac__L_e	0	[0, 0.861]
glyc_e	0	[0, 0.854]
asn__L_e	0	[0, 0.758]
thr__L_e	0	[0, 0.668]
akg_e	0	[0, 0.66]
2hb_e	0	[0, 0.631]
4abut_e	0	[0, 0.594]
cit_e	0	[0, 0.594]
glu__L_e	0	[0, 0.594]
xan_e	0	[0, 0.577]
aces_e	0	[0, 0.571]
btd_RR_e	0	[0, 0.568]
gln__L_e	0	[0, 0.568]
hxan_e	0	[0, 0.549]
2mppal_e	0	[0, 0.543]
ibutoh_e	0	[0, 0.518]
cys__L_e	0	[0, 0.505]
lanost_e	0	[0, 0.0577]
zymst_e	0	[0, 0.0534]
epist_e	0	[0, 0.0511]
fecost_e	0	[0, 0.0511]
ergst_e	0	[0, 0.0506]
cys__L_e	0	[0, 0.505]
gua_e	0	[0, 0.503]
val__L_e	0	[0, 0.498]
pro__L_e	0	[0, 0.496]
orn_e	0	[0, 0.486]
ptrc_e	0	[0, 0.486]
thym_e	0	[0, 0.479]
gam6p_e	0	[0, 0.462]
sbt__D_e	0	[0, 0.46]
2mbald_e	0	[0, 0.403]
3mop_e	0	[0, 0.403]
3mbald_e	0	[0, 0.397]
arg__L_e	0	[0, 0.395]
2mbtoh_e	0	[0, 0.389]
his__L_e	0	[0, 0.388]
3c3hmp_e	0	[0, 0.383]
iamoh_e	0	[0, 0.383]
ile__L_e	0	[0, 0.378]
leu__L_e	0	[0, 0.372]
4abz_e	0	[0, 0.365]
ibutac_e	0	[0, 0.363]
lys__L_e	0	[0, 0.353]
met__L_e	0	[0, 0.331]
2mbac_e	0	[0, 0.294]
iamac_e	0	[0, 0.291]
pacald_e	0	[0, 0.286]
tyr__L_e	0	[0, 0.286]
2phetoh_e	0	[0, 0.279]
phe__L_e	0	[0, 0.273]
pap_e	0	[0, 0.25]
spmd_e	0	[0, 0.25]
dttp_e	0	[0, 0.245]
id3acald_e	0	[0, 0.228]
pheac_e	0	[0, 0.227]
pnto__R_e	0	[0, 0.224]
ind3eth_e	0	[0, 0.223]
trp__L_e	0	[0, 0.219]
g3pc_e	0	[0, 0.154]
Nbfortyr_e	0	[0, 0.129]
nadp_e	0	[0, 0.123]
ttdca_e	0	[0, 0.123]

3.3 FVA usando la producción de GA3 y la biomasa como funciones objetivo. La biomasa está limitada a un 50% de su producción máxima.

OUT FLUXES

id	Flux	Range			
h2o_e	43.9	[40.3, 47.3]	pro__L_e	0	[0, 0.627]
co2_e	21.8	[18.4, 24.3]	orn_e	0	[0, 0.623]
h_e	7.53	[6.31, 10.9]	ptrc_e	0	[0, 0.623]
so3_e	0	[0, 3.1]	cys__L_e	0	[0, 0.604]
glx_e	0	[0, 2.99]	gua_e	0	[0, 0.572]
gly_e	0	[0, 2.17]	thym_e	0	[0, 0.566]
ac_e	0	[0, 2.05]	sbt__D_e	0	[0, 0.561]
urea_e	0	[0, 2.01]	gam6p_e	0	[0, 0.546]
acald_e	0	[0, 1.6]	3mbald_e	0	[0, 0.541]
pyr_e	0	[0, 1.6]	3c3hmp_e	0	[0, 0.512]
oaa_e	0	[0, 1.39]	iamoh_e	0	[0, 0.512]
for_e	0	[0, 1.39]	leu__L_e	0	[0, 0.503]
etoh_e	0	[0, 1.37]	2mbald_e	0	[0, 0.489]
ser__L_e	0	[0, 1.33]	3mop_e	0	[0, 0.489]
ala__L_e	0	[0, 1.31]	arg__L_e	0	[0, 0.476]
fum_e	0	[0, 1.22]	lys__L_e	0	[0, 0.468]
mal__L_e	0	[0, 1.22]	2mbtoh_e	0	[0, 0.465]
asp__L_e	0	[0, 1.17]	his__L_e	0	[0, 0.461]
lac__D_e	0	[0, 1.09]	ile__L_e	0	[0, 0.458]
succ_e	0	[0, 1.08]	ibutac_e	0	[0, 0.457]
lac__L_e	0	[0, 1.06]	4abz_e	0	[0, 0.43]
glyc_e	0	[0, 0.981]	met__L_e	0	[0, 0.389]
akg_e	0	[0, 0.889]	iamac_e	0	[0, 0.381]
asn__L_e	0	[0, 0.882]	2mbac_e	0	[0, 0.354]
4abut_e	0	[0, 0.793]	pacald_e	0	[0, 0.35]
cit_e	0	[0, 0.793]	tyr__L_e	0	[0, 0.35]
glu__L_e	0	[0, 0.793]	2phetoh_e	0	[0, 0.338]
thr__L_e	0	[0, 0.78]	phe__L_e	0	[0, 0.334]
btd_RR_e	0	[0, 0.738]	spmd_e	0	[0, 0.296]
gln__L_e	0	[0, 0.738]	pap_e	0	[0, 0.289]
2hb_e	0	[0, 0.721]	dttp_e	0	[0, 0.289]
aces_e	0	[0, 0.713]	pheac_e	0	[0, 0.275]
2mppal_e	0	[0, 0.709]	id3acald_e	0	[0, 0.272]
xan_e	0	[0, 0.68]	pnto__R_e	0	[0, 0.268]
ibutoh_e	0	[0, 0.66]	ind3eth_e	0	[0, 0.265]
val__L_e	0	[0, 0.646]	trp__L_e	0	[0, 0.263]
hxan_e	0	[0, 0.635]	g3pc_e	0	[0, 0.209]
zymst_e	0	[0, 0.0698]	Nbfortyr_e	0	[0, 0.161]
epist_e	0	[0, 0.0664]	ttdca_e	0	[0, 0.149]
fecost_e	0	[0, 0.0664]	nadp_e	0	[0, 0.144]
ergst_e	0	[0, 0.0658]	lanost_e	0	[0, 0.0763]