Package ‘GlobalFit’

August 12, 2016

Type Package
Title Bi-Level Optimization of Metabolic Network Models
Version 1.2
Date 2016-08-12
Author Daniel Hartleb
Maintainer Daniel Hartleb <daniel.hartleb@hhu.de>
Description Initial metabolic networks often inaccurately predict in-silico growth or non-growth if compared to in-vivo data. This package refines metabolic network models by making networks changes (i.e., removing, adding, changing reversibility of reactions; adding and removing biomass metabolites) and simultaneously matching sets of experimental growth and non-growth data (e.g., KO-mutants, mutants grown under different media conditions,...)
License GPL-3
Depends R (>= 2.12.0), sybil (>= 1.1.7), methods
Suggests cplexAPI (>= 1.2.1)
NeedsCompilation no
Repository CRAN
Date/Publication 2016-08-12 12:36:23

R topics documented:

GlobalFit-package .................................................. 2
bilevel_optimize .................................................. 6
example_net1 ..................................................... 14
example_net2 ..................................................... 15
example_net3 ..................................................... 15
sysBiolAlg_FastGlobalFit-class ..................................... 15
sysBiolAlg_FastGlobalFit_MULT_BIOM-class ......................... 16
sysBiolAlg_GlobalFit-class ......................................... 17

Index 19
GlobalFit-package  Bi-Level Optimization of Metabolic Network Models

Description

Initial metabolic networks often inaccurately predict in-silico growth or non-growth if compared to in-vivo data. This package refines metabolic network models by making networks changes (i.e., removing, adding, changing reversibility of reactions; adding and removing biomass metabolites) and simultaneously matching sets of experimental growth and non-growth data (e.g., KO-mutants, mutants grown under different media conditions,...). Three versions of GlobalFit are provided; an old (algorithm=2), a newer (faster) version (algorithm=1) and a third version (algorithm=3) which can be used to remove thermodynamically infeasible cycles.

Details

Package: GlobalFit
Type: Package
Version: 1.2
Date: 2016-08-12
License: GPL-3

Author(s)

Daniel Hartleb
daniel.hartleb@hhu.de

Examples

```r
## Not run:
library(sybil)
library(GlobalFit)
library("cplexAPI")
SYBIL_SETTINGS("SOLVER", "cplexAPI")
SYBIL_SETTINGS("SOLVER", "sybilGUROBI")

### EXAMPLE1: RECONCILIATION OF TWO FALSE PREDICTIONS

data(example_net1)

# names of reactions, which are not allowed to be removed
not_delete_for=c(find_id(example_net1), "Biomass")
not_delete_back=c(find_id(example_net1), "Biomass")
```
#set biomass object function
obj_coef(example_net1)[which(react_id(example_net1)=="Biomass")]=1

# create list of influxes
influxes=list()
influxes[[1]]=list(exRea="A[e]_import",value=-10)

# set influxes
lowbnd(example_net1)[pos=which(react_id(example_net1)=="A[e]_import")]=-10

# growth cases
on=list()
on[[1]]=list(on=influxes,name="LIVE!",ko_react=c("AtoB"),forced=TRUE,viability_threshold=1,
gene_copy_number=1)
on[[2]]=list(on=influxes,name="LIVE!",ko_react=c("AtoB"),forced=TRUE,viability_threshold=1,
gene_copy_number=1)
#non-growth cases
off=list()
off[[1]]=list(on=influxes,name="DIE!",ko_react=c("AtoR"),forced=FALSE,viability_threshold=1,
gene_copy_number=1)
off[[2]]=list(on=influxes,name="DIE!",ko_react=c("AtoR"),forced=FALSE,viability_threshold=1,
gene_copy_number=1)

#optional parameter list for solver, in this example for cplex
p_list=list(CPX_PARAM_THREADS=as.integer(1),CPX_PARAM_EPRHS=as.double(1e-9),
CPX_PARAM_NETPRHS=as.double(1e-11),CPX_PARAM_EPINT=as.double(1e-09),
CPX_PARAM_TILIM=1e5,CPX_PARAM_PARALLELMODE=CPX_PARALLEL_OPPORTUNISTIC)

# create list of reactions, that are allowed to be reversed
reverse_reaction_list=list()
reverse_reaction_list[[1]]=list(reaction="AtoC",pen=1)
reverse_reaction_list[[2]]=list(reaction="TtoB",pen=1)

# create list of additional reactions
additional_reactions_list=list()
additional_reactions_list[[1]]=list(id="KtoB",name="KtoB reaction",eq="(2.1) K[e] => B[e]",pen=7)
additional_reactions_list[[2]]=list(id="TtoR",name="TtoR reaction",eq="T[e] <= R[e] + Q[e]",pen=3)
additional_reactions_list[[3]]=list(id="CtoB",name="CtoQ reaction",eq="C[e] => B[e]",pen=5)

# create list of additional biomass metabolites
additional_biomass_mets=list()
additional_biomass_mets[[1]]=list(met="Q[e]",pen=0.1,factor=-1)
additional_biomass_mets[[2]]=list(met="P[e]",pen=0.1,factor=-1)
additional_biomass_mets[[3]]=list(met="B[e]",pen=0.1,factor=-1)

# create list of biomass metabolites, that are allowed to be removed
remove_biomass_mets=list()
remove_biomass_mets[[1]]=list(met="S[e]",pen=40.1)
remove_biomass_mets[[2]]=list(met="T[e]",pen=0.1)
remove_biomass_mets[[3]]=list(met="Z[e]",pen=0.1)

# set penalties for removing reactions (network contains nine reactions, so we have to set 9 values)
remove_penalties_hin=c(1,2,5,3,4,5,6,7,8,9)
remove_penalties_back=(1,2,5,3,4,6,7,8,9)

opt_net=bilevel_optimize(network=example_net1,\x0non=on,off=off,algorithm=1,\x0nadditional_reactions=additional_reactions_list,\x0nnot_delete_for=not_delete_for,\x0nnot_delete_back=not_delete_back,\x0nminimize=FALSE,\x0nsimple=FALSE,\x0nverboseMode=1,\x0nparam_list=p_list,\x0ncancel_case_penalty=NULL,\x0nuse_indicator_constraints=FALSE,\x0nstat_file=NULL,\x0nreact_file=NULL,\x0nremove_penalties_hin=remove_penalties_hin,\x0nremove_penalties_back=remove_penalties_back,\x0nreverse_reaction_list=reverse_reaction_list,\x0nalternatives=0,\x0nMaxPenalty=NULL,\x0nadditional_biomass_metabolites=additional_biomass_mets,\x0nremove_biomass_metabolites=remove_biomass_mets,\x0nvariable_lower_bound=NULL,\x0nforced_modifications=0)

#EXAMPLE2: NETWORK CONTAINS THERMODYNAMIC INFEASIBLE CYCLES
# (CYC1 AND CYC2 CAN CARRY FLUX WITHOUT ANY INFLUX);
# WE USE GLOBALFIT AND DIFFERENT BIOMASS OBJECTIVE FUNCTIONS
# FOR THE GROWTH AND NON-GROWTH CASE

data(example_net2)

#set wild type biomass object function
obj_coef(example_net2)[which(react_id(example_net2)=="Biomass")]=1

#create 2 lists of influxes (one list is empty)
influxes=list()
influxes[[1]]=list(exRea="T[e].import",value=-10)
influxes2=list()

#set influxes for wild type
lowbnd(example_net2)[pos=which(react_id(example_net2)=="T[e].import")]=-10

#growth cases with wild type biomass
on=list()
on[[1]]=list(on=influxes,name="LIVE!",ko_react=c(),forced=TRUE,\x0nviability_threshold=1,\ngene_copy_number=1,biomass="Biomass")

#non-growth cases with different biomass ("CYC1","CYC2")
off=list()
off[[1]]=list(on=influxes,name="DIE!",ko_react=c(),forced=FALSE,\nviability_threshold=1,\ngene_copy_number=1,biomass=c("CYC1","CYC2")

# no alternative modifications allowed
reverse_reaction_list=list()
additional_reactions_list=list()
additional_biomass_mets=list()
remove_biomass_mets=list()

# names of reactions, which are not allowed to be removed, including the cycle reactions
not_delete_for=c(\x0nreact_id(FindExchReact(example_net2)),"Biomass","CYC1","CYC2")
not_delete_back=c(\x0nreact_id(FindExchReact(example_net2)),"Biomass","CYC1","CYC2")
opt_net=bilevel_optimize(network=example_net2,on=on,off=off,algorithm=3,
additional_reactions=additional_reactions_list,not_delete_for=not_delete_for,
not_delete_back=not_delete_back,minimize=False,simple=False,verboseMode=1,
param_list=p_list,cancel_case_penalty=NULL,use_indicator_constraints=False,
stat_file=NULL,react_file=NULL,reverse_reaction_list=reverse_reaction_list,
alternatives=0,MaxPenalty=NULL,additional_biomass_metabolites=additional_biomass_mets,
remove_biomass_metabolites=remove_biomass_mets,
variable_lower_bound=NULL,forced_modifications=0)

#EXAMPLE3: NON-GROWTH CASE CAN ONLY BE RESOLVED BY CHANGING THE LOWER BOUND OF AN INFLUX
#(ONLY WORKS WITH THE SLOWER IMPLEMENTATION OF GLOBALFIT; ALGORITHM=2).
#THIS CAN BE USED TO FIND SUITABLE QUALITATIVE MEDIA COMPOSITIONS.
#NOTE IN THIS SIMPLE EXAMPLE THE VIABILITY THRESHOLD
#OF THE NON-GROWTH CASE IS HIGHER THAN THE GROWTH CASE

data(example_net3)

# names of reactions, which are not allowed to be removed
not_delete_for=c(react_id(findExchReact(example_net3)),"Biomass")
not_delete_back=c(react_id(findExchReact(example_net3)),"Biomass")

#set wild type biomass object function
obj_coef(example_net3)[which(react_id(example_net3)=="Biomass")]=1

#create 2 lists of influxes (one list is empty)
influxes=list()
influxes[[1]]=list(exRea="T[e]_import",value=-100)

influxes2=list()

#set influxes for wild type
lowbnd(example_net3)[pos=which(react_id(example_net3)=="T[e]_import")]=-100

#growth cases with wild type biomass
on=list()
on[[1]]=list(on=influxes,name="LIVE!",ko.react=c(),forced=TRUE,viability_threshold=1,
gene_copy_number=1)

#non-growth cases with different biomass(A[e]_import)
off=list()
off[[1]]=list(on=influxes,name="DIE!",ko.react=c(),forced=FALSE,viability_threshold=2,
gene_copy_number=1)
## bilevel_optimize

### Description

pre und post-processing of optimization. Adds reaction to network, creates output, creates optimization object, interprets solution, applies modification to network.

### Usage

```python
bilevel_optimize(network, on = c(), off = c(), algorithm = 1,
additional_reactions = NULL, minimize = TRUE, simple = FALSE,
verboseMode = 1, cancel_case_penalty = NULL, not_delete_for = c(),
not_delete_back = c(), param_list = NULL, use_indicator_constraints = FALSE,
remove_penalties_hin = c(), remove_penalties_back = c(), stat_file = NULL,
react_file = NULL, reverse_reaction_list = NULL, MaxPenalty = NULL,
alternatives = 0, bio_stoich = 1e-05, additional_biomass_metabolites = NULL,
remove_biomass_metabolites = NULL, variable_lower_bound = NULL,
forced_modifications = 0)
```
Arguments

network metabolic network model of type modelorg
on each entry must contain: on: list of influxes
name: character, name of growth case
ko_react: vector of reaction, which are knocked out (i.e., lower and upper bound = 0)
forced: logical, FALSE (growth case can be ignored with according penalty) or TRUE (case cannot be ignored)
viability_threshold: numerical > 0 threshold, which is considered as growth
gene_copy_number: integer > 0, multiplies the penalty for ignoring this growth case
biomass: specifies the biomass objective function of this growth case

Example:
on=list()
on[[1]]=list(on=influxes,name="LIVE!",ko_react=c(),forced=TRUE,
viability_threshold=1,gene_copy_number=1,biomass="Biomass")
default: NULL

off list of non-growth cases each entry must contain: on: list of influxes
name: character, name of non-growth case
ko_react: vector of reactions, which are knocked out (i.e., lower and upper bound = 0)
forced: logical, FALSE (growth case can be ignored with according penalty) or TRUE (case cannot be ignored)
viability_threshold: numerical > 0 threshold, which is considered as growth
gene_copy_number: integer > 0, multiplies the penalty for ignoring this non-growth case
biomass: specifies the biomass objective function of this growth case (or vector of reactions if algorithm 3 is chosen)

Example:
off=list()"off="[[1]]=list(on=influxes,name="DIE!",ko_react=c(),forced=FALSE,
viability_threshold=1,gene_copy_number=1,biomass="A[e]_import")
default: NULL

algorithm integer, specifies which version of GlobalFit should be used:
1: fast version
2: old version, but allows to use variable_lower_bound
3: fast version, used for removing thermodynamically infeasible cycles
default: 1

additional_reactions list containing additional reaction. Each entry must contain the following attributes.
id: character, id of reaction
name: character, name of reaction
eq: character, equation of reaction.
pen: numeric >0, penalty for adding this reaction to the network

Example:
additional_reactions_list=list()
additional_reactions_list[[1]]=list(id="KtoB",name="KtoB reaction",eq="(2.1) K[e] => B[e]",pen=7)
additional_reactions_list[[2]]=list(id="TtoR",name="TtoR reaction",eq="T[e] <=> R[e]",pen=3)
additional_reactions_list[[3]]=list(id="TtoQ",name="TtoQ reaction",eq="T[e] => Q[e]",pen=5)

default: NULL

\ minimize logical, specifies if blocked reaction should be removed from network before optimizing. May decrease solving time, but it takes time to calculate blocked reactions
default: FALSE

\ simple logical, if run in simple mode (TRUE) only the number of contradicting cases are minimized, network changes are not penalized
default: FALSE

\ verboseMode numeric, should output be printed (1 => yes; !1 => no)
default: 1

cancel_case_penalty numerical>0, penalty for ignoring (case is violating viability threshold) a single growth case
default: NULL, penalty is higher than all network changes combined

\ not_delete_for vector of reaction names; forward reactions that are not allowed to be removed (e.g., biomass objective function, exchange reactions)
default: NULL

\ not_delete_back vector of reaction names; backward reactions that are not allowed to be removed (e.g., biomass objective function, exchange reactions)
default: NULL

\ param_list list of specific parameters, that will be passed on to the solver
default: NULL

\ use_indicator_constraints logical, indicator constraints may prevent trickle flow, only usable if cplex (cplexAPI) is used as solver
default: FALSE

\ remove_penalties_hin Vector of penalties for removing each forward reaction. Must have the same length as the number of reactions in the network.
If not specified the penalty for removing each reaction will be set to 1.
default: NULL

remove_penalties_back
Vector of penalties for removing each forward reaction. Must have the same
length as the number of reactions in the network.
If not specified the penalty for removing each reaction will be set to 1.
default: NULL

stat_file path for stat file
default: NULL

react_file path of react file, contains all network modifications (subset of stat file)
default: NULL

reverse_reaction_list
list containing reaction, that are allowed to be reversed and according penalty.
The following attributes must be defined for each entry:
  reaction, character: name of reaction
  pen, numeric>0, penalty for reversing reaction

Example:
reverse_reaction_list=list()
reverse_reaction_list[[1]]=list(reaction="KtoT",pen=1)
reverse_reaction_list[[2]]=list(reaction="TtoB",pen=1)

default: NULL

MaxPenalty integer >=0, amount of alternative solution that should be calculated
default: 0

alternatives integer >=0, amount of alternative solution that should be calculated
default: 0

bio_stoich numeric>0, stoichiometric coefficient for additional biomass metabolites
default: 1e-5

additional_biomass_metabolites
list of additional biomass metabolites
met specifies the metabolite, which can be added to the biomass objective func-
tion (note metabolites, which are already in the biomass objective function can
not be added).
pen specifies the corresponding penalty for adding this metabolite.
factor: -1 or 1; -1 metabolite can be added as substrate; 1 metabolite can be
added as product
Example:
additional_biomass_mets=list()
additional_biomass_mets[[1]]=list(met="K[e]",pen=0.1,factor=1)

default: NULL
remove_biomass_metabolites
  list of metabolites that can be removed from the biomass objective function
  met specifies the metabolite, which can be removed from the biomass objective
  function
  pen specifies the corresponding penalty

  Example:
  remove_biomass_mets=list()
  remove_biomass_mets[[1]]=list(met="Z[e]",pen=0.01)

  default: NULL

variable_lower_bound
  list containing reactions, which lower-bound can be optimized.
  This can be used to calculate a qualitatively optimized media formulation
  Can only be used if algorithm=2

  reaction specifies the name of the reaction
  min specifies the minimal possible value
  max specifies the maximal possible value
  penalty specifies the penalty for changing the lowerbound

  Example:
  varying_lower_bound_list=list()
  varying_lower_bound_list[[1]]=list(reaction="T[e]_import",min=-20,max=-0,penalty=0.1)

  default: NULL

forced_modifications
  integer>=0, number of minimal modification; may reduce computation time if
  set >=1
  default: 0

Value
  optimized metabolic network model of type modelorg

Author(s)
  Daniel Hartleb

References
  Hartleb D, Jarre F, Lercher MJ. Improved Metabolic Models for E. coli and Mycoplasma genitalium
  from GlobalFit, an Algorithm That Simultaneously Matches Growth and Non-Growth Data Sets.
Examples

```r
## Not run:
library(sybil)
library(GlobalFit)
library("cplexAPI")
SYBILL_SETTINGS("SOLVER", "cplexAPI")
#SYBILL_SETTINGS("SOLVER", "sybilGUROBI")

##EXAMPLE1: RECONCILIATION OF TWO FALSE PREDICTIONS

data(example_net1)

# names of reactions, which are not allowed to be removed
not_delete_for=c(react_id(findReact(example_net1)), "Biomass")
not_delete_back=c(react_id(findReact(example_net1)), "Biomass")

#set biomass object function
obj_coef(example_net1)[which(react_id(example_net1)="Biomass")]=1

#create list of influxes
influxes=list()
influxes[[1]]=list(exRea="A[e].import",value=-10)

#set influxes
lowbnd(example_net1)[pos=which(react_id(example_net1)="A[e].import")]=10

#growth cases
on=list()
on[[1]]=list(on=influxes,name="LIVE!",ko_react=c("AtoB"),forced=TRUE,viability_threshold=1,
gene_copy_number=1)
on[[2]]=list(on=influxes,name="LIVE!",ko_react=c("AtoB"),forced=TRUE,viability_threshold=1,
gene_copy_number=1)

#non-growth cases
off=list()
off[[1]]=list(on=influxes,name="DIE!",ko_react=c("AtoR"),forced=FALSE,viability_threshold=1,
gene_copy_number=1)
off[[2]]=list(on=influxes,name="DIE!",ko_react=c("AtoR"),forced=FALSE,viability_threshold=1,
gene_copy_number=1)

#optional parameter list for solver, in this example for cplex
p_list=list(CPX_PARAM_THREADS=as.integer(1),CPX_PARAM_EPRHS=as.double(1e-9),
CPX_PARAM_NETEPRHS=as.double(1e-11),CPX_PARAM_EPINT=as.double(1e-09),
CPX_PARAM_TILIM=1e5,CPX_PARAM_PARALLELMODE=CPX_PARALLEL_OPPORTUNISTIC)

#create list of reactions, that are allowed to be reversed
reverse_reaction_list=list()
reverse_reaction_list[[1]]=list(react="AtoC",pen=1)
reverse_reaction_list[[2]]=list(react="TtoB",pen=1)

#create list of additional reactions
additional_reactions_list=list()
```
additional_reactions_list[[1]]=list(id="KtoB",name="KtoB reaction",eq="(2.1) K[e] \Rightarrow B[e]",pen=7)
additional_reactions_list[[2]]=list(id="TtoR",name="TtoR reaction",eq="T[e] \Leftrightarrow R[e] + Q[e]",pen=3)
additional_reactions_list[[3]]=list(id="CtoB",name="CtoQ reaction",eq="C[e] \Rightarrow B[e]",pen=5)

# create list of additional biomass metabolites
additional_biomass_mets=list()
additional_biomass_mets[[1]]=list(met="Q[e]",pen=0.1,factor=-1)
additional_biomass_mets[[2]]=list(met="R[e]",pen=0.1,factor=-1)
additional_biomass_mets[[3]]=list(met="B[e]",pen=0.1,factor=-1)

# create list of biomass metabolites, that are allowed to be removed
remove_biomass_mets=list()
remove_biomass_mets[[1]]=list(met="S[e]",pen=0.1)
remove_biomass_mets[[2]]=list(met="T[e]",pen=0.1)
remove_biomass_mets[[3]]=list(met="Z[e]",pen=0.1)

# set penalties for removing reactions (network contains nine reactions, so we have to set 9 values)
remove_penalties_hin=c(1,2,5,3,4,5,6,7,8,9)
remove_penalties_back=c(1,2,5,3,4,5,6,7,8,9)

opt_net=bilevel_optimize(network=example_net1,on=on_off=off,algorithm=1,
additional_reactions=additional_reactions_list,not_delete_for=not_delete_for,
not_delete_back=not_delete_back,remove_penalties_hin=remove_penalties_hin,
remove_penalties_back=remove_penalties_back,reverse_reaction_list=reverse_reaction_list,
alternatives=0,MaxPenalty=NULL,additional_biomass_metabolites=additional_biomass_mets,
remove_biomass_metabolites=remove_biomass_mets,variable_lower_bound=NULL,forced_modifications=0)

# EXAMPLE2: NETWORK CONTAINS THERMODYNAMIC INFEASIBLE CYCLES
# (CYC1 AND CYC2 CAN CARRY FLUX WITHOUT ANY INFLUX);
# WE USE GLOBALFIT AND DIFFERENT BIOMASS OBJECTIVE FUNCTIONS
# FOR THE GROWTH AND NON-GROWTH CASE

data(example_net2)

# set wild type biomass object function
obj_coef(example_net2)[which(react_id(example_net2)=="Biomass")]=1

# create 2 lists of influxes (one list is empty)
influxes=list()
influxes[[1]]=list(exRea="T[e]_import",value=-10)
influxes2=list()

# set influxes for wild type
lowbnd(example_net2)[pos=which(react_id(example_net2)=="T[e]_import")]=-10

# growth cases with wild type biomass
on=list()
on[[1]]=list(on=influxes, name="LIVE!", ko_react=c(), forced=TRUE, viability_threshold=1, gene_copy_number=1, biomass="Biomass")

# non-growth cases with different biomass ("CYC1","CYC2")
off=list()
off[[1]]=list(on=influxes, name="DIE!", ko_react=c(), forced=FALSE, viability_threshold=1, gene_copy_number=1, biomass=c("CYC1","CYC2")

# no alternative modifications allowed
reverse_reaction_list=list()
additional_reactions_list=list()
additional_biomass_mets=list()
remove_biomass_mets=list()

# names of reactions, which are not allowed to be removed, including the cycle reactions
not_delete_for=c(react_id(findExchReact(example_net2)),"Biomass","CYC1","CYC2")
not_delete_back=c(react_id(findExchReact(example_net2)),"Biomass","CYC1","CYC2")

opt_net=bilevel_optimize(network=example_net2, on=on, off=off, algorithm=3,
additional_reactions=additional_reactions_list, not_delete_for=not_delete_for,
not_delete_back=not_delete_back, minimize=FALSE, simple=FALSE, verboseMode=1,
param_list=p_list, cancel_case_penalty=NULL, use_indicator_constraints=FALSE,
stat_file=NULL, react_file=NULL, reverse_reaction_list=reverse_reaction_list,
alternatives=0, MaxPenalty=NULL, additional_biomass_metabolites=additional_biomass_mets,
remove_biomass_metabolites=remove_biomass_mets,
variable_lower_bound=NULL, forced_modifications=0)

#EXAMPLE3: NON-GROWTH CASE CAN ONLY BE RESOLVED BY CHANGING THE LOWER BOUND OF AN INFLUX
#(ONLY WORKS WITH THE SLOWER IMPLEMENTATION OF GLOBALFIT; ALGORITHM=2).
#THIS CAN BE USED TO FIND SUITABLE QUALITATIVE MEDIA COMPOSITIONS.
#NOTE IN THIS SIMPLE EXAMPLE THE VIABILITY THRESHOLD
#OF THE NON-GROWTH CASE IS HIGHER THAN THE GROWTH CASE

data(example_net3)

# names of reactions, which are not allowed to be removed
not_delete_for=c(react_id(findExchReact(example_net3)),"Biomass")
not_delete_back=c(react_id(findExchReact(example_net3)),"Biomass")

#set wild type biomass object function
obj_coef(example_net3)[which(react_id(example_net3)=="Biomass")]=1

#create 2 lists of influxes (one list is empty)
influxes=list()
influxes[[1]]=list(exRea="T[e].import",value=-100)

influxes2=list()

#set influxes for wild type
lowbnd(example_net3)[pos=which(react_id(example_net3)=="T[e].import")]=-100

growth cases with wild type biomass
on=list()
on[[1]]=list(on=influxes,name="LIVE!",ko_react=c(),forced=TRUE,viability_threshold=1,
gene_copy_number=1)

#non-growth cases with different biomass(A[e].import)
off=list()
off[[1]]=list(on=influxes,name="DIE!",ko_react=c(),forced=FALSE,viability_threshold=2,
gene_copy_number=1)

## set varying_lower_bound; T[e].import is allowed to vary between 0 and -20.
# Because the viability threshold of the non-growth case is 2
# and the viability threshold of the growth case is 1;
# the optimized value should be between -2 and -1

varying_lower_bound_list=list() varyin

# no alternative modifications allowed
reverse_reaction_list=list() additional_reactions_list=list() additional_biomass_mets=list() remove_biomass_mets=list()

opt_net=bilevel_optimize(network=example_net3,on=on,off=off,algorithm=2, additional_reactions=additional_reactions_list,not_delete_for=not_delete_for, not_delete_back=not_delete_back,minimize=FALSE,simple=FALSE,verboseMode=1, param_list=p_list,cancel_case_penalty=NULL,use_indicator_constraints=FALSE, stat_file=NULL,react_file=NULL,reverse_reaction_list=reverse_reaction_list, alternatives=0,MaxPenalty=NULL,additional_biomass_metabolites=additional_biomass_mets, remove_biomass_metabolites=remove_biomass_mets,variable_lower_bound=varying_lower_bound_list, forced_modifications=0)

## End(Not run)

---

example_net1  

Example metabolic network

---

Description

Example metabolic network model of type modelorg with 9 reactions and 8 metabolites.
Example metabolic network

**Description**

Example metabolic network model of type modelorg with 9 reactions and 6 metabolites.

Example metabolic network

**Description**

Example metabolic network model of type modelorg with 3 reactions and 2 metabolites.

sysBiolAlg_FastGlobalFit-class

**Class** "sysBiolAlg_FastGlobalFit"

**Description**

Class, that builds the (fast) bilevel optimization object.

**Objects from the Class**

Objects can be created by calls of the form `new("sysBiolAlg_FastGlobalFit", model, LPvariant, useNames, cnames, ...

**Slots**

wu: Object of class "numeric" ~~
wl: Object of class "numeric" ~~
fnc: Object of class "integer" ~~
fnr: Object of class "integer" ~~
problem: Object of class "optObj" ~~
algorithm: Object of class "character" ~~
.nr: Object of class "integer" ~~
.nc: Object of class "integer" ~~
fldind: Object of class "integer" ~~
alg_par: Object of class "list" ~~
## sysBiolAlg_FastGlobalFit_MULT_BIOM-class

### Description
Class, that builds the bilevel optimization object of algortihm 3.

### Objects from the Class
Objects can be created by calls of the form `new("sysBiolAlg_FastGlobalFit_MULT_BIOM", model, LPvariant, useNames, cnames, rnames, pname, scaling, penalties_hin, ... bio_stoich, use_indicator_constraints, variable_lower_bound, forced_alterations, writeprobtofilename, NNNI)`.

### Slots
- `wu`: Object of class "numeric"
- `wl`: Object of class "numeric"
- `fnc`: Object of class "integer"
- `fnr`: Object of class "integer"
- `problem`: Object of class "optObj"
- `algorithm`: Object of class "character"
- `nr`: Object of class "integer"
- `nc`: Object of class "integer"
- `fldind`: Object of class "integer"
- `alg_par`: Object of class "list"

### Extends
Class "sysBiolAlg", directly.
Methods

initialize signature(.Object = "sysBiolAlg_FastGlobalFit_MULT_BIOM"): ...

Author(s)

Daniel Hartleb

Examples

showClass("sysBiolAlg_GlobalFit")

Class "sysBiolAlg_GlobalFit"

Description

Class, that builds the bilevel optimization object of algorithm 2.

Objects from the Class

Objects can be created by calls of the form new("sysBiolAlg_GlobalFit", model, LPvariant, useNames, cnames, rnames, ...

Slots

wu: Object of class "numeric" ~~
wl: Object of class "numeric" ~~
fnc: Object of class "integer" ~~
fnr: Object of class "integer" ~~
problem: Object of class "optObj" ~~
algorithm: Object of class "character" ~~
nr: Object of class "integer" ~~
nc: Object of class "integer" ~~
fldind: Object of class "integer" ~~
alg_par: Object of class "list" ~~

Extends

Class "sysBiolAlg", directly.

Methods

initialize signature(.Object = "sysBiolAlg_GlobalFit"): ...
Author(s)
Daniel Hartleb

Examples
showClass("sysBiolAlg_GlobalFit")
Index

*Topic classes
  sysBiolAlg_FastGlobalFit-class, 15
  sysBiolAlg_FastGlobalFit_MULT_BIOM-class, 16
  sysBiolAlg_GlobalFit-class, 17

*Topic datasets
  example_net1, 14
  example_net2, 15
  example_net3, 15
  bilevel_optimize, 6
  example_net1, 14
  example_net2, 15
  example_net3, 15

GlobalFit (GlobalFit-package), 2
GlobalFit-package, 2
initialize, sysBiolAlg_FastGlobalFit-method
(sysBiolAlg_FastGlobalFit-class), 15
initialize, sysBiolAlg_FastGlobalFit_MULT_BIOM-method
(sysBiolAlg_FastGlobalFit_MULT_BIOM-class), 16
initialize, sysBiolAlg_GlobalFit-method
(sysBiolAlg_GlobalFit-class), 17

sysBiolAlg, 16, 17
sysBiolAlg_FastGlobalFit-class, 15
sysBiolAlg_FastGlobalFit_MULT_BIOM-class, 16
sysBiolAlg_GlobalFit-class, 17