Package 'LogicOpt'

May 7, 2016

Type Package

Title Truth Table Logic Optimizer

Date 2016-05-06

Description Access to powerful logic minimization algorithms and data structures that operate on a sum-of-products truth table. The core algorithms are built on Espresso Version 2.3 developed at UC Berkeley for digital logic synthesis purposes. Enhancements have been made to integrate within the R framework and support additional logic optimization use cases such as those needed by Qualitative Comparative Analysis (QCA) and Genetic Programming. There are no expressed or implied warranties.

Version 1.0.0

Author William Stiehl [aut, cre, cph]

Maintainer William Stiehl <wwstiehl@gmail.com>

Depends R (>= 3.2.3)

License GPL (>= 2)

NeedsCompilation yes

LazyData true

RoxygenNote 5.0.1

Suggests testthat

Repository CRAN

Date/Publication 2016-05-07 01:27:50

R topics documented:

`LogicOpt-package` *LogicOpt: A Package for Logic Optimization of Truth Tables using Espresso*

Description

The package **LogicOpt** provides access to powerful logic minimization algorithms that operate on a sum-of-products truth table. The core algorithms are built on Espresso Version 2.3 developed at UC Berkeley for digital logic synthesis purposes. The espresso C code has been extended extensively to integrate within the R framework and support additional logic optimization use cases that operate on R data frame tables. The primary interface to the package is through the `logicopt` function.

New algorithms have been developed that leverage the espresso core routines and data structures to support logic optimization use cases used by Qualitative Comparative Analysis (QCA). See the "primes", "multi-min", and "multi-full" options for the "mode" paramater in function `logicopt`. These modes along with the paramater find_dc=TRUE provide features similar to those provided by `QCAGUI::eqmcc` and `QCApro::eQMC` but the espresso-based routines here are able to handle much larger functions and run in less runtime. Integration to QCA is through the `QCAGUI::truthTable` or `QCApro::truthTable` functions provided in QCA packages **QCAGUI** and **QCApro** respectively (other QCA packages may also work but have not been tested). These functions provide the necessary functionality to convert a raw QCA dataset into a QCA truth table (see documentation in those packages for more details). The function `QCAtt2LOtt` then provides the last step to get to an expresso compatible truth table. See dataset `l.represent.1` for an example flow from a raw QCA dataset to final optimized QCA results.

This package also contains a Genetic Programming use case where a maze navigation program for a robot is optimized. See the paper referenced below as well as the example truth table `l.robot1` for more details. The `logicopt` option mode="echo" was developed to support the abiilty to view the input truth table prior to optimization.

Details

Package:	LogicOpt
Type:	Package
Version:	0.1.0
Date:	2016-04-15
License:	GPL (>= 2)

Author(s)

Authors:
William Stiehl
<wwstiehl@gmail.com>

Maintainer:
William Stiehl

References

Brayton, Robert King; Hachtel, Gary D.; McMullen, Curtis T.; Sangiovanni-Vincentelli, Alberto L.. (1984), Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, ISBN 0-89838-164-9

Rudell, Richard L. (1986-06-05), "Multiple-Valued Logic Minimization for PLA Synthesis" Memorandum No. UCB/ERL M86-65 (Berkeley) http://www.eecs.berkeley.edu/Pubs/TechRpts/1986/ERL-86-65.pdf

Dusa, Adrian (2016). QCAGUI: Modern functions for Qualitative Comparative Analysis. R Package URL: http://cran.r-project.org/package=QCAGUI

Thiem, Alrik. 2016. Professional Functionality for Performing and Evaluating Qualitative Comparative Analysis. R Package Version 1.1-0. http://www.alrik-thiem.net/software

Keane, A.J. 2015. "Genetic Programming, Logic Design and Case-Based Reasoning for Obstacle Avoidance." Learning and Intelligent Optimization: 9th International Conference; pp.104-118

`l.partybans.0` *Logicopt truth table created from "partybans.csv" dataset*

Description

l.partybans.0 is an logicopt compatible truth table generated from the QCA dataset "partybans.csv" where output "PB" is 0.

Usage

```
data(l.partybans.0)
```

Format

R data frame table

Source

compass.org website

Examples

```
## Not run:
# Read raw QCA dataset from csv file
inpath <- system.file("extdata/raw_qca/partybans.csv", package="LogicOpt")
partybans <- read.csv(inpath,row.names=1,na="")

# Load QCA package
library(QCAGUI)

# Create the QCA truth table
q.partybans.0 <- truthTable(partybans, conditions = c("C","F","T","R","V"), outcome = "PB{0}")

# Create the logicopt  truth table
l.partybans.0 <- QCAtt2LOtt(q.partybans.0)

## End(Not run)

# Load up logicopt truth table
data(l.partybans.0)

# Optimize and print logicopt truth table
partybans0 <- logicopt(l.partybans.0,5,1,find_dc=TRUE,mode="multi-min")
print_multi_tt(partybans0,eqn=TRUE,n_in=5,n_out=1,QCA=TRUE)
```

---

l.partybans.1 *Logicopt truth table created from "partybans.csv" dataset*

Description

l.partybans.1 is an logicopt compatible truth table generated from the QCA dataset "partybans.csv" where output "PB" is 1.

Usage

```
data(l.partybans.1)
```

Format

R data frame table

Source

compass.org website

Examples

```
## Not run:
# Read raw QCA dataset from csv file
inpath <- system.file("extdata/raw_qca/partybans.csv", package="LogicOpt")
partybans <- read.csv(inpath,row.names=1,na="")

# Load QCA package
library(QCAGUI)

# Create the QCA truth table
q.partybans.1 <- truthTable(partybans, conditions = c("C","F","T","R","V"), outcome = "PB{1}")

# Create the logicopt truth table
l.partybans.1 <- QCAtt2LOtt(q.partybans.1)

## End(Not run)

# Load up logicopt truth table
data(l.partybans.1)

# Optimize logicopt truth table and print results
partybans1 <- logicopt(l.partybans.1,5,1,find_dc=TRUE,mode="multi-min")
print_multi_tt(partybans1,eqn=TRUE,n_in=5,n_out=1,QCA=TRUE)
```

---

l.represent.0 *Logicopt truth table created from "represent.csv" dataset*

Description

l.represent.0 is an logicopt compatible truth table generated from the QCA dataset "represent.csv" where output "WNP" is 0.

Usage

```
data(l.represent.0)
```

Format

R data frame table

Source

compass.org website and various QCA packages

Examples

```
## Not run:
# Read raw QCA dataset from csv file
inpath <- system.file("extdata/raw_qca/represent.csv", package="LogicOpt")
represent <- read.csv(inpath,row.names=1,na="")

# Need to load a QCA package that contains truthTable function: (pick one)
# library(QCAGUI)
# library(QCApro)

# Create the QCA truth table
q.represent.0 <- truthTable(represent, outcome = "WNP{0}")

# Create the logicopt truth table
l.represent.0 <- QCAtt2LOtt(q.represent.0)

## End(Not run)

# Load up truth table
data(l.represent.0)

# Optimize logicopt truth table  and print results
represent0 <- logicopt(l.represent.0,5,1,find_dc=TRUE,mode="multi-min")
print_multi_tt(represent0,eqn=TRUE,n_in=5,n_out=1,QCA=TRUE)
```

---

l.represent.1 *Logicopt truth table created from "represent.csv" dataset*

Description

l.represent.1 is an logicopt compatible truth table generated from the QCA dataset "represent.csv" where output "WNP" is 1.

Usage

```
data(l.represent.1)
```

Format

R data frame table

Source

compass.org website and various QCA packages

Examples

```
## Not run:
# Read raw QCA dataset from csv file
inpath <- system.file("extdata/raw_qca/represent.csv", package="LogicOpt")
represent <- read.csv(inpath,row.names=1,na="")

# Need to load a QCA package that contains truthTable function: (pick one)
# library(QCAGUI)
# library(QCApro)

# Create the QCA truth table
q.represent.1 <- truthTable(represent, outcome = "WNP{1}")

# Create the logicopt truth table
l.represent.1 <- QCAtt2LOtt(q.represent.1)

## End(Not run)

# Load up truth table
data(l.represent.1)

# Optimize logicopt truth table  and print results
represent1 <- logicopt(l.represent.1,5,1,find_dc=TRUE,mode="multi-min")
print_multi_tt(represent1,eqn=TRUE,n_in=5,n_out=1,QCA=TRUE)
```

---

l.robot1 *Truth table from a Genetic Programming Use Case*

Description

This is an 8 input 3 output truth table from A.J. Keane that is used in a Genatic Programming use case for robot control. The truth table has been modified slightly from the paper (referenced below) to specify Boolean outputs for the three possible output values of "zero", "one", and "minus" one. This is slightly more informative than the example in the paper.

Usage

```
data(l.robot1)
```

Format

Espresso compatible truth table

Source

Keane, A.J. 2015. "Genetic Programming, Logic Design and Case-Based Reasoning for Obstacle Avoidance." Learning and Intelligent Optimization: 9th International Conference, pages 104:118.

Examples

```
## Not run:
# steps to recreate l.robot1
inpath <- system.file("extdata/espresso/robot1_in.esp", package="LogicOpt")
l.robot1 <- logicopt(esp_file=inpath,mode="echo")

## End(Not run)

# load l.robot1
data(l.robot1)

# optimize l.robot1
robot1_opt <- logicopt(l.robot1,8,3)

# optimized results have 13 rows that cover outputs zero, one, and minus
robot1_opt[2]

# print optimized equations (where each output is 1)
print_multi_tt(robot1_opt,TRUE,8,3)
```

`l.small` *Espresso truth table with 4 inputs and 3 outputs*

Description

Espresso compatible truth table generated from espresso format file small.esp.

Usage

```
data(l.small)
```

Format

R data frame table

Examples

```
## Not run:
# steps to recreate
inpath <- system.file("extdata/espresso/small.esp", package="LogicOpt")
l.small <- logicopt(esp_file=inpath,mode="echo")[1]

## End(Not run)
```

logicopt *Truth Table Logic Optimization*

Description

This function provides various options to optimize and analyze an input truth table that represents a sum of Boolean or multi-valued input product terms. This function leverages the powerful logic minimization algorithms from Espresso. These algorithms are a standard for optimizing large functions in digital logic synthesis and have been modified to handle general logic minimization problems to serve the R community. The input truth table is an R data frame or alteratively an Espresso format file. See examples section for more details.

Usage

```
logicopt(in_tt = NULL, n_in = 0, n_out = 0, find_dc = FALSE,
  input_sizes = NULL, exact_cover = TRUE, esp_file = "",
  mode = "espresso")
```

Arguments

in_tt	An R data frame table representing a sum of product terms (PTs) truth table. The PTs have one or more inputs with a positive integer value or a "-" which means the input is not specified. The outputs are Boolean and have possible values 1, 0, "-", or "~" and specify that the corresponding PT is part of the ON set, the OFF set, the DC (don't care) set, or is unspecified (not a part of any set) respectively. PTs should not be in both the ON set and OFF set. When logicopt optimizes, it attempts to find the fewest number of PTs in the ON set and uses PTs in the DC set to further reduce the solution.
n_in	Integer number of input columns in the truth table. Inputs must come first in the truth table.
n_out	Integer number of output columns in the truth table. Outputs must come after the n_in inputs of the truth table. The number of columns in the in_tt must be n_in + n_out.
find_dc	FALSE (default) means any unspecified input conditions are added to the OFF set for that output. TRUE means that any unspecified input conditions are added to DC set for that output. The DC set is used to further optimize the ON set. Don't cares can also be explicitly defined by using "-" for the output in the input truth table.
input_sizes	Integer vector which represents the number of possible values for each input. Default is NULL which means the size for each input will be determined automatically by the software by analyzing the input truth table and counting the number of values used. Specifying input_sizes is important when the input table has unspecified ("-") or unused input values and all possible values are not used.
exact_cover	Do an exact covering of prime implicant table. Option applies to QM based algorithms (mode = "qm", "multi-min", and "multi-full"). If FALSE, the covering algoirthm is heuristic and runs faster but may not find an exact solution.

If TRUE, algorithm is exact and finds an exact minimum solution. Default is TRUE.

`esp_file` File name for espresso format file to read and process. If esp_file is specified, the input truth table options (in_tt, n_in, n_out, input_sizes, and find_dc) are ignored. The mode and exact_cover options still apply.

`mode` A string that specifies the mode to use for optimization:

- "espresso" – Use the classic espresso algorithm to optimize the input truth table in_tt or the espresso format table in esp_file. Returns a single solution of the optimized ON set. This option should be used for very large truth tables.
- "qm" – Use Quine-McCluskey (QM) algorithm to optimize in_tt or esp_file and return a single solution of the optimized ON set. Use with caution for large truth tables.
- "primes" – Return set of prime implicants for in_tt or esp_file. Three solutions are returned in a single truth table. These represent the ESSENTIAL PRIMES, the PARTIALLY REDUNDANT PRIMES, and the TOTALLY REDUNDANT PRIMES. Use the `print_primes_tt` function to print the results.
- "multi-min" – Use QM to find the minimum set of non-redundant solutions that cover all the ESSENTIAL PRIMES and the minimal set of PARTIALLY REDUNDANT PRIMES. Solutions are ordered by size. The number of solutions found is capped at 50.
- "multi-full" – Find additional coverings of prime implicants beyond what is found in multi-min. A exhaustive covering of all PARTIALLY REDUNDANT PRIMES is found. The number of solutions is capped at 50.
- "echo" – Echo the ON, OFF, and DC sets for the in_tt or esp_file truth table without any optimization. Note the resulting truth table can be extremely large because it will cover the complete Boolean (or MV) space for all inputs and outputs. Use with caution.

Value

The logicopt function returns a list of two items: an truth table and a vector. The vector represents the size and number of solutions in the output truth table. For example, a vector [10] means there is a single solution in the truth table which has 10 rows. A vector [5 8 2] means there are three solutions in the truth table with 5, 8, and 2 rows respectively.

Examples

```
######################### EXAMPLE #1 ###############################
# create a truth table with 4 inputs A, B, C, D and 2 outputs X and Y
e.ex1 <- data.frame(
 A = c(0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1),
 B = c(0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0),
 C = c(0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1),
 D = c(0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0),
 X = c(1,1,1,1,0,1,1,1,"-",1,1,0,1,1,0,0),
```

```
 Y = c("-",1,1,1,1,1,1,1,1,1,1,0,0,0,0,"-"))

# show the unoptimized equations
tt2eqn(e.ex1,4,2)

# optimize the truth table
tte <- logicopt(e.ex1,4,2)

# show the optimized equations
tt2eqn(tte[[1]],4,2)

# generate and print the prime implicants from optimized tte
ttp <- logicopt(tte[[1]],4,2,mode="primes")
print_primes_tt(ttp,TRUE,4,2)

######################### EXAMPLE #2 ###############################
# get path to an espresso format file
file <- system.file("extdata/espresso/small2.esp", package="LogicOpt")

# get the espresso truth table without optimization
small2 <- logicopt(esp_file=file,mode="echo")

# print the unoptimized equations
print_multi_tt(small2,TRUE,4,3)

# optimize with espresso algorithm
small2_opt <- logicopt(small2[[1]],4,3,mode="espresso")

# print the optimized equations
print_multi_tt(small2_opt,TRUE,4,3)

######################### EXAMPLE #3 ###############################
# load up truth table created from a QCA dataset
data(l.represent.1)

# read documentation on how truth table was created
?l.represent.1

# find the set of minimum solutions that cover QCA dataset (mode="multi-min")
# treat unspecified input conditions as don't cares (find_dc=TRUE)
# find a exact covering (exact_cover=TRUE)
lomm <- logicopt(l.represent.1,n_in=5,n_out=1,find_dc=TRUE,
  exact_cover=1,mode="multi-min")

# print the solutions in equation format
print_multi_tt(lomm,TRUE,5,1,QCA=TRUE)

######################### EXAMPLE #4 ###############################
# optimize a truth table from Genetic Programming
inpath <- system.file("extdata/espresso/robot1_in.esp", package="LogicOpt")
robot1 <- logicopt(esp_file=inpath,mode="echo")

# unoptimized truth table has 273 rows (256 in ON set and 18 in OFF set)
```

```
robot1[2]

# optimize l.robot1
robot1_opt <- logicopt(robot1[[1]],8,3)

# optimized results have 13 rows that cover outputs zero, one, and minus
robot1_opt[2]

# print optimized equations (where each output is 1)
print_multi_tt(robot1_opt,TRUE,8,3)

######################### EXAMPLE #5 ###############################
# show how to use input_sizes

# get vector of number of unique values for each input
data(l.partybans.1)
pb_in_vals <- num_input_values(l.partybans.1,5)
pb_in_vals

# optimize with mode=espresso
epb <- logicopt(l.partybans.1,5,1,find_dc=TRUE,mode="espresso")
epb
pb_in_opt_vals <- num_input_values(epb[[1]],5)

# note how some input values have been optimized away and are no longer used!
pb_in_opt_vals

# we need original input sizes to process the optimized truth table
qmpb <- logicopt(epb[[1]],5,1,find_dc=FALSE, input_sizes=pb_in_vals,mode="qm")
print_multi_tt(epb,TRUE,5,1)
print_multi_tt(qmpb,TRUE,5,1)
```

`num_input_values` *Find size of input values*

Description

Find number of unique input values for each input in tt.

Usage

```
num_input_values(tt, n_in)
```

Arguments

`tt`	a truth table where first n_in columns are inputs
`n_in`	number of inputs

Value

returns a vector of number of unique input values for each input

---

`print_multi_tt` *Print logicopt() results*

Description

This function prints the the results from logicopt() in truth table or equation format.

Usage

```
print_multi_tt(esp_multi, eqn = FALSE, n_in, n_out, max_sol = 50,
  QCA = FALSE)
```

Arguments

`esp_multi`	An R data frame table representing a truth table with 1 or more solutions.
`eqn`	Print in equation format. Default is FALSE.
`n_in`	Number of inputs in the esp_multi truth table.
`n_out`	Number of outputs in the esp_multi truth table.
`max_sol`	Maximum number of solutions to print. Default is 50.
`QCA`	Attempt to print out in a QCA like format

Value

None

Examples

```
data(l.partybans.0)
tt <- logicopt(l.partybans.0,n_in=5,n_out=1,find_dc=TRUE,mode="multi-full")
print_multi_tt(tt,5,1,eqn=TRUE,max_sol=5)
```

print_primes_tt *Print the Primes from logicopt(mode="primes")*

Description

This function prints the results from logicopt(..,,mode="primes") option.

Usage

```
print_primes_tt(primes, eqn = FALSE, n_in, n_out)
```

Arguments

primes	An R data frame table generated by logicopt(...,mode="primes").
eqn	Print in equation format. Default is FALSE.
n_in	Number of inputs in the primes truth table.
n_out	Number of outputs in the primes truth table.

Value

None

Examples

```
data(l.small)
ptt <- logicopt(l.small,n_in=4,n_out=3,find_dc=TRUE,mode="primes")
print_primes_tt(ptt,eqn=TRUE,4,3)
```

QCAtt2LOtt *Create logicopt tt from QCA tt*

Description

This function takes a truth table produced by the package QCAGUI (or QCApro) function truthTable() and creates an logicopt() format truth table.

Usage

```
QCAtt2LOtt(qcatt)
```

Arguments

qcatt	An R data frame table generated by QCAGUI or QCApro truthTable().

Value

truth table for logicopt()

Examples

```
## Not run:
inpath <- system.file("extdata/raw_qca/partybans.csv", package="LogicOpt")
partybans <- read.csv(inpath,row.names=1,na="")
library(QCAGUI)
q.partybans.1 <- truthTable(partybans, conditions = c("C","F","T","R","V"), outcome = "PB{1}")
l.partybans.1 <- QCAtt2LOtt(q.partybans.1)

## End(Not run)
```

---

tt2eqn *Equations from a Truth Table*

Description

This function generates the ON set equations for a truth table. Inputs are uppercase if they are positive and lowercase for negative.

Usage

```
tt2eqn(tt, n_in, n_out, QCA = FALSE)
```

Arguments

tt	R data frame truth table.
n_in	Number of inputs in the tt.
n_out	Number of outputs in the tt.
QCA	Print in QCA format.

Value

Vector of equations strings. One for each output.

Examples

```
data(l.small)
tt <- logicopt(l.small,n_in=4,n_out=3)
eqn <- tt2eqn(tt[[1]],4,3)
```

Index