Package ‘CausalFX’

May 20, 2015

Type Package

Title Methods for Estimating Causal Effects from Observational Data

Version 1.0.1

Date 2015-05-20

Imports igraph, rcdd, rje

Maintainer Ricardo Silva <ricardo@stats.ucl.ac.uk>

Description Estimate causal effects of one variable on another, currently for
binary data only. Methods include instrumental variable bounds, adjustment by a
given covariate set, adjustment by an induced covariate set using a variation of
the PC algorithm, and an effect bounding method (the Witness Protection Program)
based on covariate adjustment with observable independence constraints.

License GPL (>= 2)

URL http://github.com/rbas2015/CausalFX

BugReports http://github.com/rbas2015/CausalFX/issues

NeedsCompilation no

Author Ricardo Silva [cre, aut],
Robin Evans [aut]

Repository CRAN

Date/Publication 2015-05-20 17:10:47

R topics documented:

`bindagCausalEffectBackdoor`

Estimates Average Causal Effects by Covariate Adjustment in Binary Models

Description

Estimate the average causal effect (ACE) of a given treatment variable X on a given outcome Y by adjusting for a set S of covariates (equivalenly, assuming S blocks all backdoor paths in the causal graph that generates the observations). . Bounds are based on finding conditional instrumental variables using the faithfulness assumption relaxed to allow for a moderate degree of unfaithfulness. Candidate models are generated from the method described in `covsearch`.

Usage

```
bindagCausalEffectBackdoor(problem, prior_table, S)
```

Arguments

`problem`	a `cfx` problem instance for the ACE of a given treatment X on a given outcome Y.
`prior_table`	effective sample size hyperparameter of a Dirichlet prior for testing independence with contingency tables.
`S`	an array indicating the indices of the covariates used in the adjustment.

Details

The algorithm implemented is a naive one. If the dimensionality of S is larger than around 20, this may crash the system. This function is assumed to be used along other methods such as `covsearch` and `wpp`, which generate relativelty small covariate sets.

Value

the estimated ACE.

References

Pearl, J. (2000) *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Examples

```
## Generate a synthetic problem
problem <- simulateWitnessModel(p = 4, q = 4, par_max = 3, M = 1000)

## Idealized case: suppose we know the true distribution,
## get "exact" ACE estimands for different adjustment sets
sol_pop <- covsearch(problem, pop_solve = TRUE)
effect_pop <- synthetizeCausalEffect(problem)
cat(sprintf(
  "ACE (true) = %1.2f\nACE (adjusting for all) = %1.2f\nACE (adjusting for nothing) = %1.2f\n",
   effect_pop$effect_real, effect_pop$effect_naive, effect_pop$effect_naive2))

## Perform inference and report results
covariate_hat <- covsearch(problem, cred_calc = TRUE, M = 1000)
if (length(covariate_hat$witness) > 1) {
  sol_ACE <- bindagCausalEffectBackdoor(problem, prior_table = 10, S = covariate_hat$Z[[1]])
  cat(sprintf("Estimated ACE = %1.2f\n", sol_ACE))
}
```

---

cfx *Creates a CausalFX Problem Instance*

Description

Set up an object describing a causal inference problem of finding the average causal effect of some treatment on some outcome. Currently, only binary data is supported. The problem specification also allows the specification of a synthetic model, for simulation studies.

Usage

```
cfx(x, y, latent_idx = NULL, dat = NULL, g = NULL, model = NULL,
  num_v_max = 20)
```

Arguments

`x`	the index of the treatment variable.
`y`	the index of the outcome variable.
`latent_idx`	an array with the indices of variables which should be considered latent
`dat`	a matrix of binary data, can be ignored if a model is provided.
`g`	a binary matrix encoding a causal graph, where g[i, j] == 1 if a directed edge from vertex j to i should exist, 0 otherwise. This is only required if a ground truth model exists.
`model`	if g is specified, this needs to be specified too. This argument should be a list of conditional probability tables, each encoding the conditional probability of each vertex in g given its parents. Entry `model[[i]]` is an array of non-negative numbers, describing the probability of random variable/vertex i being equal to 1. In particular, `model[[i]][j]` is the conditional probability of this event given

	that the parents of i are in state j. States are indexed as follows. If S is the binary string corresponding to the binary values of the parents of i in g, sorted by their index, then j is given by $1 + bin2dec(S)$, where $bin2dec$ is the transformation of a binary string into a decimal number.
`num_v_max`	the maximum dimensionality in which the joint distribution implied by a model is pre-computed. Having this pre-computed can speed up some computations for methods that use the provided ground truth model. Because the space required to store a joint distribution grows exponentially with the dimensionality, this quantity cannot be too large.

Value

A `cfx` object, which contains the following fields:

`X_idx`	the index of the treatment variable in the data/graph.
`Y_idx`	the index of the outcome variable in the data/graph.
`latent_idx`	the array of latent variable indices given as input.
`data`	the data given as input.
`graph`	the graph given as input.
`varnames`	an array of strings with the names of the variables, as given by `data`. If `data` has no column names or it is not provided, this is given a default value, where variable i is assigned the name "`X`".
`model`	the model given as input.
`ancestrals`	a list of arrays (if g is provided), where `ancestrals[[i]]` is the array of the indices of the ancestrals of i in g, excluding i itself.
`probs`	a multidimensional array (if g is provided) of the same dimensionality as g, where each entry corresponds to the probability of that particular assignment of variable values. This is `NULL` if the dimensionality of g is greater than `num_v_max`.

`covsearch` *Search for Causal Effect Covariate Adjustment*

Description

Find the witnesses and adjustment sets (if any) for the average causal effect (ACE) between a given treatment variable X on a given outcome Y. This is done by an exhaustive search on a (reduced) set of possible candidates. Currently, only binary data is supported.

Usage

```
covsearch(problem, max_set = 12, min_only = TRUE, prior_ind = 0.5,
  prior_table = 10, cred_calc = FALSE, M = 1000, stop_at_first = FALSE,
  pop_solve = FALSE, verbose = FALSE)
```

Arguments

`problem`	a `cfx` problem instance for the ACE of a given treatment X on a given outcome Y.
`max_set`	maximum size of conditioning set. The cost of the procedure grows exponentially as a function of this, so be careful when increasing the default value.
`min_only`	for each witness, once a set of a particular size is found, don't look for larger ones.
`prior_ind`	prior probability of an independence.
`prior_table`	effective sample size hyperparameter of a Dirichlet prior for testing independence with contingency tables.
`cred_calc`	if `TRUE`, compute conditional credible intervals for the ACE of highest scoring model.
`M`	if necessary to compute (conditional) credible intervals, use Monte Carlo with this number of samples.
`stop_at_first`	if `TRUE`, stop as soon as some witness is found.
`pop_solve`	if `TRUE`, assume we know the population graph in `problem` instead of data.
`verbose`	if `TRUE`, print out more detailed information while running the procedure.

Details

The method assumes that the variables given in `problem` (other than `problem$X_idx` and `problem$Y_idx`) are covariates which causally precede treatment and outcome. It then applies the faithfulness condition of Spirtes, Glymour and Scheines (2000, *Causation, Prediction and Search*, MIT Press) to derive an *admissible set*: a set of covariates which removes all confounding between treatment and outcome when adjusted for. The necessary and sufficient conditions for finding an admissible set using the faithfulness assumption were discussed by Enter, Hoyer and Spirtes (2013, *JMLR W&CP*, vol. 31, 256–264). In order for a set to be proved an admissible set, some auxiliary variable in the covariate set is necessary - we call this variable a "witness." See Entner et al. for details. It is possible that no witness exists, which in this case the function returns an empty solution. Multiple witness/admissible sets might exist. The criterion for finding a witness/admissible set pair requires the testing of conditional independence constraints. The test is done by performing Bayesian model selection with a Dirichlet prior over the contingency table of the variables in `problem` using the effective sample size hyperparameter `prior_table`, and a prior probability of the independence hypothesis using the hyperparameter `prior_ind`.

For each witness/admissible set that passes this criterion, the function reports the posterior expected value of the implied ACE for each pair, by first plugging-in the posterior expected value of the contingency table as an estimate of the joint distribution. For a particular pair of witness/admissible set, chosen according to the best fit to the conditional independencies required by the criterion of Enter et al. (see also Silva and Evans, 2014, NIPS 298-306), we calculate the posterior distribution of the ACE. This posterior does not take into account the uncertainty on the choice of witness/admissible set, but instead is the conditional posterior given this choice.

The search for a witness/admissible set is by brute-force: for each witness, evaluate all subsets of the remaining covariates as candidate admissible sets. If there are too many covariates (more than `max_set`), only a filtered set of size `max_set` is considered for each witness. The set is chosen by first scoring each covariate by its empirical mutual information with the witness given problem$X_idx and picking the top max_set elements, to which a brute-force search is then applied.

Value

A list containing problem plus the following items:

witness	array containing the indices of the witness variables.
Z	a list, where Z[[i]] is the i-th array containing the indices of the variables in the admissible set corresponding to witness witness[i].
witness_score	array containing the scores of each witness/admissible set pair.
hw	witness corresponding to the highest scoring pair.
hZ	array containing admissible set corresponding to the highest scoring pair.
ACEs	array of average causal effects corresponding to each witness/admissible pair.
ACEs_post	array of samples corresponding to the posterior distribution of the ACE associated implied by hW and hZ.

References

http://jmlr.org/proceedings/papers/v31/entner13a.html

http://papers.nips.cc/paper/5602-causal-inference-through-a-witness-protection-program

Examples

```
## Generate a synthetic problem
problem <- simulateWitnessModel(p = 4, q = 4, par_max = 3, M = 1000)

## Idealized case: suppose we know the true distribution,
## get "exact" ACE estimands for different adjustment sets
sol_pop <- covsearch(problem, pop_solve = TRUE)
effect_pop <- synthetizeCausalEffect(problem)
cat(sprintf(
  "ACE (true) = %1.2f\nACE (adjusting for all) = %1.2f\nACE (adjusting for nothing) = %1.2f\n",
    effect_pop$effect_real, effect_pop$effect_naive, effect_pop$effect_naive2))

## Perform inference and report results
covariate_hat <- covsearch(problem, cred_calc = TRUE, M = 1000)
summary(covariate_hat)
```

---

iv *Bayesian Analysis of Binary Instrumental Variables*

Description

Perform Bayesian instrumental variable analysis for binary data. This assumes the number of covariates is small enough. Notice that a set of size greater than 20 will raise a flag and require explicit permission from the user by setting force_use to TRUE.

Usage

```
iv(problem, w, Z, prior_table, M, verbose = FALSE, reject_level = 0.9,
  force_use = FALSE)
```

Arguments

`problem`	a `CausalFX` problem instance.
`w`	index of instrumental variable within the data.
`Z`	array of indices for covariate adjustment.
`prior_table`	Dirichlet hyperparameter for contingency tables.
`M`	number of Monte Carlo samples to compute posterior distributions.
`verbose`	print relevant messages.
`reject_level`	if first iteration of rejection sampling has a proportion of rejected samples larger than this, reject the model and return no bounds.
`force_use`	if TRUE, ignore any warnings about size of `Z`.

Details

Given a joint distribution, this function finds a lower bound and an upper bound on the average causal effect of treatment X on outcome Y, adjusting for covariate set Z. The joint distribution is estimated by assigning a prior to joint contingency table of `w`, `Z` and the treatment/outcome pair indexed in `problem`, and ACE bounds are inferred by computing the posterior on the contingency table implied by the instrumental variable assumption. The prior is proportional to a Dirichlet distribution with an effective sample size `prior_table`. It assigns probability zero to models which do not satisfy the constraints of the (conditional) instrumental variable, as described by Balke and Pearl (1997, *Journal of the American Statistical Association*, vol. 92, 1171–1176). Hence, the prior is not an exact Dirichlet distribution, but only proportional to one. Posterior samples for the lower and upper bound are generated by rejection sampling using the unconstrained model as a proposal. If the model is a bad fit to the data, this might take much computing time. Setting `reject_level` to a level smaller than 1 may stop the rejection sampler earlier and reject the model, returning no bounds.

Value

A list containing two fields,

`bound`	the point estimate of lower and upper bounds.
`bound_post`	samples from the posterior distribution over bounds.

References

http://ftp.cs.ucla.edu/pub/stat_ser/r199-jasa.pdf

Examples

```
## Generate synthetic problems until a (conditional) instrumental variable can be found

while (TRUE) {
  problem <- simulateWitnessModel(p = 4, q = 4, par_max = 3, M = 1000)
  s <- covsearch(problem, pop_solve = TRUE)
  if (length(s$witness) > 0) {
    w <- s$witness[1]
    Z <- s$Z[[1]]
    break
  }
}

## Calculate true effect for evaluation purposes
sol_pop <- covsearch(problem, pop_solve = TRUE)
effect_pop <- synthetizeCausalEffect(problem)
cat(sprintf("ACE (true) = %1.2f\n", effect_pop$effect_real))

## Binary IV bounds
sol_iv <- iv(problem, w, Z, prior_table = 10, M = 1000)
summary(sol_iv)
```

---

print.cfx *Prints a CausalFX Problem Instance*

Description

Prints some of the information regarding a `cfx` object.

Usage

```
## S3 method for class 'cfx'
print(x, ...)
```

Arguments

`x`	a `cfx` object.
`...`	other parameters, ignored.

Details

The information that is displayed includes the identifiers of the treatment and outcome variables, the names of all variables and, if a theoretical causal graph is part of the object specification, its causal structured described in terms of the parent ids for each variable.

`print.summary.covsearch` *Print Summaries of Covariate Search Outputs*

Description

Print output of `summary` method for class `"covsearch"`.

Usage

```
## S3 method for class 'summary.covsearch'
print(x, ...)
```

Arguments

`x`	an object of class `"summary.covsearch"`, usually a result of a call to `summary.covsearch`.
`...`	other parameters, ignored.

Details

Variable names with more than 20 characters are truncated when printing.

`print.summary.iv` *Print Summaries of Binary Instrumental Variable Analyses*

Description

Print output of `summary` method for class `"iv"`.

Usage

```
## S3 method for class 'summary.iv'
print(x, ...)
```

Arguments

`x`	an object of class `"summary.iv"`, usually a result of a call to `summary.iv`.
`...`	other parameters, ignored.

Details

Variable names with more than 20 characters are truncated.

`print.summary.wpp` *Print Summaries of Witness Protection Program Outputs*

Description

Print output of `summary` method for class "`wpp`".

Usage

```
## S3 method for class 'summary.wpp'
print(x, ...)
```

Arguments

`x`	an object of class "`summary.wpp`", usually a result of a call to `summary.wpp`.
`...`	other parameters, ignored.

Details

Variable names with more than 20 characters are truncated when printing.

`simulateWitnessModel` *Generates Synthetic CausalFX Problems*

Description

This function generates simple synthetic problems that can be used to test the methods in the `CausalFX` package. `CausalFX` problems are objects of class `cfx`, and specify a causal inference task of estimating the effect of a given treatment X on a given outcome Y, with a corresponding dataset. This function generates only binary data from a multinomial distribution.

Usage

```
simulateWitnessModel(p, q, par_max, M, no_sol = FALSE)
```

Arguments

`p`	number of background variables (besides X and Y).
`q`	number of sink variables.
`par_max`	maximum number of parents in the background set.
`M`	sample size.
`no_sol`	if `TRUE`, then latent variables are parents of both X and Y, meaning no adjustment set will theoretically be found (barring sampling variability) if a method such as `covsearch` is applied.

Details

The function first generates a directed acyclic graph with a given number of variables which have no latent common parents with treatment X and outcome Y, which we call "background variables". Conditioning on a subset of the background variables will block all measured confounding in this problem. The function then generates a set of "sink" variables K which have one common latent parent with either X or Y, but are otherwise not adjacent to any observed variable. Conditioning on the sink variables will generate confounding paths between treatment and outcome. Latent variables are a pool of independent variables with no parents. If `no_sol` is `FALSE`, they are parents of either X or Y but not both. If `no_sol` is `TRUE`, then all latent variables are parents of both X and Y and as such no adjustment set with observed variables will remove unmeasured confounding between treatment and outcome. Remaining parents for observed variables are sampled uniformly at random from the pool of background variables obeying the constraint on the maximum number of parents given by `par_max`.

Given a graph structure, each variable i is given a binary conditional distribution, defining the probability of i being equal to 1 given its parents in the graph. This conditional distribution is generated randomly by a logistic regression model with pairwise interactions, where coefficients are generated by samples from independent Gaussians with zero mean and standard deviation 10 / number of parents.

Value

An object of class `cfx`.

`summary.covsearch` *Summarize Covariate Search Outputs*

Description

`summary` method for class "`covsearch`".

Usage

```
## S3 method for class 'covsearch'
summary(object, ...)
```

Arguments

`object`	an object of class "`covsearch`", usually a result of a call to `covsearch`.
`...`	other parameters, ignored.

Value

Besides fields inherented from the `covsearch` object, a list summary statistics is included:

`q`	an array of 5 entries corresponding to evenly space quantiles of the ACE of the highest scoring witness/admissible set pair.
`ci`	95% marginal posterior credible interval for the chosen ACE.

See Also

The model fitting function `covsearch`.

`summary.iv` *Summarize Binary Instrumental Variable Analyses*

Description

`summary` method for class `"iv"`.

Usage

```
## S3 method for class 'iv'
summary(object, ...)
```

Arguments

`object`	an object of class `"iv"`, usually a result of a call to `iv`.
`...`	other parameters, ignored.

Value

Besides fields from the `iv` object, this includes a list summary statistics,

`lq`	an array of 5 entries corresponding to evenly space quantiles of the lower bound ("minimum", 25%, median, 75 given by a Monte Carlo approximation.
`lci`	95% marginal posterior credible interval for lower bound.
`uq`	an array of 5 entries corresponding to evenly space quantiles of the upper bound ("minimum", 25%, median, 75 given by a Monte Carlo approximation.
`uci`	95% marginal posterior credible interval for upper bound.

See Also

The model fitting function `iv`.

summary.wpp *Summarize Witness Protection Program Outputs*

Description

`summary` method for class "`wpp`".

Usage

```
## S3 method for class 'wpp'
summary(object, ...)
```

Arguments

`object`	an object of class "`wpp`", usually a result of a call to `wpp`.
`...`	other parameters, ignored.

Value

Besides fields inherented from the `wpp` object, a list summary statistics is included:

`lq`	an array of 5 entries corresponding to evenly space quantiles of the lower bound of the ACE of the highest scoring witness/admissible set pair.
`lci`	95% marginal posterior credible interval for the chosen ACE lower bound.
`uq`	an array of 5 entries corresponding to evenly space quantiles of the lower bound of the ACE of the highest scoring witness/admissible set pair.
`uci`	95% marginal posterior credible interval for the chosen ACE lower bound.
`hscore_bound`	score of the witness/admissible oair of highest score.
`chosen_w`	witness of the witness/admissible pair of highest score.
`chosen_Z`	admissible set of the witness/admissible set of highest score.
`min_l`	estimated minimum lower bound of all lower bounds found by the procedure.
`max_u`	estimated maximum upper bound of all upper bounds found by the procedure.
`narrowest_bound`	lower and upper bounds corresponding to the narrowest ACE interval found by the procedure.

See Also

The model fitting function `wpp`.

---

synthetizeCausalEffect *Computes Average Causal Effects by Covariate Adjustment in Binary Models using a Given Causal Model*

Description

Computes the average causal effect (ACE) of a given treatment variable X on a given outcome Y for the models generated by `simulateWitnessModel`. This assumes the synthetic models are small enough, as adjustment is done by brute force calculation.

Usage

```
synthetizeCausalEffect(problem)
```

Arguments

`problem`	a `cfx` problem instance for the ACE of a given treatment X on a given outcome Y. This problem instance should have a fully specified causal model, including a graph and conditional probability tables. It must also be small enough so that the joint probability must have been pre-computed.

Details

The algorithm implemented is a naive one. When creating the `cfx` object, field `num_v_max` must be large enough so that the joint distribution is computed in advance. Only for relatively small models (approximately 20 variables in total) this will be feasible.

Value

A list containing three different types of estimand:

`effect_real`	the true ACE.
`effect_naive`	the result of a naive adjustment using all of the observed covariates.
`effect_naive2`	the result of a naive adjustment using no covariates.

Examples

```
## Generate a synthetic problem
problem <- simulateWitnessModel(p = 4, q = 4, par_max = 3, M = 1000)

## Idealized case: suppose we know the true distribution,
## get "exact" ACE estimands for different adjustment sets
sol_pop <- covsearch(problem, pop_solve = TRUE)
effect_pop <- synthetizeCausalEffect(problem)
cat(sprintf(
  "ACE (true) = %1.2f\nACE (adjusting for all) = %1.2f\nACE (adjusting for nothing) = %1.2f\n",
   effect_pop$effect_real, effect_pop$effect_naive, effect_pop$effect_naive2))
```

wpp *The Witness Protection Program for Causal Effect Estimation*

Description

Perform a search for bounds on the average causal effect (ACE) of a given treatment variable X on a given outcome Y. Bounds are based on finding conditional instrumental variables using the faithfulness assumption relaxed to allow for a moderate degree of unfaithfulness. Candidate models are generated from the method described in `covsearch`.

Usage

```
wpp(problem, epsilons, max_set = 12, prior_ind = 0.5, prior_table = 10,
  cred_calc = TRUE, M = 1000, analytical_bounds = TRUE,
  pop_solve = FALSE, verbose = FALSE)
```

Arguments

`problem`	a `cfx` problem instance for the ACE of a given treatment X on a given outcome Y.
`epsilons`	an array of six positions corresponding to the relaxation parameters. In order: (1) the maximum difference in the conditional probability of the outcome given everything else, as the witness changes levels; (2) the maximum difference in the conditional probability of the outcome given everything else, and the conditional distribution excluding latent variables for the witness set at 0; (3) the maximum difference in the conditional probability of the outcome given everything else, and the conditional distribution excluding latent variables for the witness set at 1; (4) the maximum difference in the conditional probability of the treatment given its causes, and the conditional distribution excluding latent variables (5) the maximum ratio between the conditional distribution of the latent variable given the witness and the marginal distribution of the latent variable. This has to be greater than or equal to 1; (6) the minimum ratio between the conditional distribution of the latent variable given the witness and the marginal distribution of the latent variable. This has to be in the interval (0, 1].
`max_set`	maximum size of conditioning set. The cost of the procedure grows exponentially as a function of this, so be careful when increasing the default value.
`prior_ind`	prior probability of an independence.
`prior_table`	effective sample size hyperparameter of a Dirichlet prior for testing independence with contingency tables.
`cred_calc`	if `TRUE`, compute conditional credible intervals for the ACE of highest scoring model.
`M`	if necessary to compute (conditional) credible intervals, use Monte Carlo with this number of samples.
`analytical_bounds`	if `cred_calc` is `TRUE`, use the analytical method for computing bounds if this is also `TRUE`.

`pop_solve`	if `TRUE`, assume we know the population graph in `problem` instead of data. Notice that data is still used when computing posteriors over bounds.
`verbose`	if `TRUE`, print out more detailed information while running the procedure.

Details

Each pair of witness/admissible set found by `covsearch` will generate a corresponding lower bound and upper bound. The bounds reported in `bounds` are based on the posterior expected contingency table implied by `prior_table`, which uses a numerical method to optimize the bounds. Besides these point estimates, posterior distributions on the lower and upper bound for the highest scoring witness/admissible set can also be computed if the flag `cred_calc` is set to `TRUE`, and reported on `bounds_post`. If the option `analytical_bounds` is set to `FALSE`, the posterior distribution calculation will use the numerical method. It provides tighter bounds, but the computational cost is much higher. Please notice these posteriors are for the bounds conditional on the given choice of witness and admissible set: uncertainty on this choice is not taken into account.

A complete explanation of the method is given by Silva and Evans (2014, "Causal inference through a witness protection program", *Advances in Neural Information Processing Systems*, 27, 298–306).

Note: messages about numerical problems when calling the bound optimizer are not uncommon and are accounted for within the procedure.

Value

An object of class `wpp` containing the copies of the inputs `problem`, `epsilons`, `prior_ind`, `prior_table`, `analytical_bounds`, plus the following fields:

`w_list`	a list of arrays/lists, where each `w_list$witness[i]` is a witness, each `w_list$Z[[i]]` is the corresponding admissible set, and each `w_list$witness_score[i]` is the corresponding score for the witness/admissible set.
`hw`	witness corresponding to the highest scoring pair.
`hZ`	array containing admissible set corresponding to the highest scoring pair.
`bounds`	a two-column matrix where each row corresponds to a different witness/admissible set combination, and the two columns correspond to an estimate of the lower bound and upper bound as given by the posterior expected value given an inferred causal structure.
`bounds_post`	a two-column matrix, where rows correspond to different Monte carlo samples, and the two columns correspond to lower and upper bounds on the ACE as implied by `epsilons` with witness `hw` and admissible set `hZ`.

References

http://papers.nips.cc/paper/5602-causal-inference-through-a-witness-protection-program

Examples

```
## Generate a synthetic problem
problem <- simulateWitnessModel(p = 4, q = 4, par_max = 3, M = 200)

## Calculate true effect for evaluation purposes
```

```
sol_pop <- covsearch(problem, pop_solve = TRUE)
effect_pop <- synthetizeCausalEffect(problem)
cat(sprintf("ACE (true) = %1.2f\n", effect_pop$effect_real))

## WPP search (with a small number of Monte Carlo samples)
epsilons <- c(0.2, 0.2, 0.2, 0.2, 0.95, 1.05)
sol_wpp <- wpp(problem, epsilons, M = 100)
summary(sol_wpp)
```

Index