bbowda-python

Kevin Kofler <kofler@dagopt.com>

Jan 26, 2026

CONTENTS

CHAPTER

ONE

MODULE BBOWDA

`class bbowda.DoubleArray`

Array of double, wrapping a C double[].

`class bbowda.OptimizationProblem`(*dimx*, *dimy*, *dimy_eq*, *numinitpts*, *c*, *xlow*, *xup*, *Flow*, *Fup*, *initpts*)

Definition of a black box optimization problem.

This class represents both the constants (inherited from 'bbowda_problem)' and the black box function (in the form of the pure virtual method 'evaluateF)' that together represent the black box optimization problem. Subclass this class to implement your optimization problem.

The problem is assumed to be of the form:

```
min cT (x ; y)
s.t. y = F1(x) [explicit equality constraints]
     F2(x) = 0 [implicit equality constraints]
     xlow <= x <= xup
     Flow <= y <= Fup
```

Flow

Lower bounds for explicit equality constraints.

Lower bounds for *y* = F1(*x)*. Must have dimension 'dimy'.

Notes: No bounds need to be specified for the implicit equality constraints F2(*x)* because those bounds are by definition always 0.

Fup

Upper bounds for explicit equality constraints.

Upper bounds for *y* = F1(*x)*. Must have dimension 'dimy'.

Notes: No bounds need to be specified for the implicit equality constraints F2(*x)* because those bounds are by definition always 0.

c

Coefficients in the (linear) objective function.

The coefficient vector *c*. Must have dimension 'dimx' + 'dimy'. First list all coefficients of *x* in order, then all coefficients of *y* = F1(*x)* in order.

dimx

Number of input variables.

The vector dimension of *x*.

dimy

Number of explicit equality constraints.

The vector dimension of y = F1(x). Also known as the number of interval inequality constraints if the coefficients of y in 'c' are 0.

`dimy_eq`

Number of implicit equality constraints.

Vector dimension of F2(x).

`evaluateF()`

Pure virtual callback evaluating the black box function.

Callback evaluating the black box function (or obtaining the evaluation result from an external source) at the point x (of dimension 'dimx)' and writing the result to F (of dimension 'dimy' + 'dimy_eq)'. Must be implemented by the user through subclassing.

Notes: There is no *user_data* pointer because any user data can and should be included in the user-provided subclass.

`property initpts`

User-provided starting points, as a list of lists.

A jagged matrix of dimension 'numinitpts' * 'dimx' (in row-major order, i.e., first all components of the first starting point, then the second one, etc.). If 'numinitpts' is 0, this is just an empty list.

`initptsvec`

User-provided starting points, as a contiguous array.

Must be a contiguous matrix of dimension 'numinitpts' * 'dimx' (in row-major order, i.e., first all components of the first starting point, then the second one, etc.). If 'numinitpts' is 0, this is just an empty vector (and can be NULL).

`numinitpts`

Number of user-specified starting points.

Can be 0. If no or not enough starting points are provided by the user, BBOWDA will automatically sample some starting points within the bounds.

`solve()`

Main entry point of the BBOWDA object-oriented API.

Runs the BBOWDA algorithm on this problem with the given parameters.

Parameters **`params`** (*`SolverParameters`*) – The solver parameters for the BBOWDA algorithm.

`xlow`

Lower bounds for input variables.

Lower bounds for x. Must have dimension 'dimx'.

`xup`

Upper bounds for input variables.

Upper bounds for x. Must have dimension 'dimx'.

`class bbowda.SolverParameters`

Solver parameters for the BBOWDA algorithm.

Parameters allowing to tune how the BBOWDA algorithm operates.

`estimate_constraint_tol`

Tolerance for the constraints estimating the implicit equality constraints during global search.

If 'global_search_ignores_eq_constraints' is false, the global search attempts to estimate global enclosures for the implicit equality constraints. But those enclosures are estimated approximations and not guaranteed to hold exactly. This tolerance specifies by how much the bounds for the estimated enclosures should be relaxed to account for that.

This parameter has no effect if the problem does not include any implicit equality constraints or if 'global_search_ignores_eq_constraints' is true.

`global_search_ignores_eq_constraints`

Whether to ignore implicit equality constraints for global search.

True means that implicit equality constraints will be ignored by the global search. Hence, it will suggest evaluation points to fill any gaps in the search space even if they are nowhere near feasible for the implicit equality constraints.

False means that implicit equality constraints will be honored by the global search. The global search will compute estimates that attempt to globally enclose the implicit equality constraints with the help of an external LP solver, helping to suggest only points that are expected to be approximately feasible.

This parameter has no effect if the problem does not include any implicit equality constraints.

`maxpts`

Maximum number of points to evaluate.

Currently, the algorithm will always run until exactly this many points are evaluated, because no other stopping criteria are implemented yet. In the presence of implicit equality constraints, it may then evaluate one more point for the final extrapolation attempt.

`optimum_tol`

Tolerance for optimum feasibility (infinity norm).

A point will be considered feasible, and thus a valid candidate for the optimum, if the bound constraints for *x* are satisfied exactly, and if none of the other constraints is violated by more than *optimum_tol*, i.e., if the componentwise inequalities *Flow* - *optimum_tol* <= F1(*x)* <= *Fup* + *optimum_tol* and - *optimum_tol* < F2(*x)* < *optimum_tol* hold. In vector terms, this means that the infinity norm of the constraint violation must be less than *optimum_tol*.

`bbowda.srand()`

Set the pseudorandom number generator (PRNG) seed of the C library.

BBOWDA uses the C library to generate pseudorandom numbers. Therefore, if reproducible results are needed, it is necessary to set this seed.

Parameters **`seed`** (*`int`*) – The random seed to set.

PYTHON MODULE INDEX

INDEX