GODLIKE

GODLIKE is an abbreviation of **G**lobal **O**ptimum **D**etermination by **L**inking and **I**nterchanging **K**indred **E**valuators. This algorithm is an attempt to generalize and improve the robustness of the four meta-heuristic optimization algorithms GA, PSO, DE and ASA, *and* generalize the optimization process by being capable to solve both single-objective and multi-objective optimization problems.

Using GODLIKE for Single-Objective Optimization Problems

Typical Usage

Simple usage for single-objective problems is

```
[solution, function_value] = ...
GODLIKE(obj_function, lb, ub)
```

which returns the solution vector `solution` with associated [function_value], which is the global minimum of the objective function `obj_function`. The upper and lower bounds in each dimension is set by the arguments `lb` and `ub`.

Output Arguments

`solution` the solution vector returned. It is always a row-vector of size [1 × dimensions].

`function_value` the function value associated with the solution vector; the globally minimum function value. It is always a scalar [1 × 1] value.

Input Arguments:

`obj_function` The user-provided objective function `obj_function` must be a function handle to an anonymous (inline) function or a function in MATLAB's current path. This function must accept arguments in the same format as `lb` or `ub` are given in, and return a scalar value.

`lb & ub` In case all bounds are the same for every dimension, either `lb` or `ub` can be a scalar, in which case it is simply resized to the problem's dimensions. However, it is important to note that the dimension of the problem (the length of each trial solution, or equivalently, the number of decision variables) is determined by the maximum size or either `lb` or `ub`. So at least *one* of the arguments `lb` or `ub` must have the same size as the problem's dimensions.

All four of these input arguments are mandatory.

Extended Usage

An example of the most extended usage is

```
[solution, function_value, exitflag, output] = ...
GODLIKE(obj_function, lb, ub, confcn, 'option', 'value', ...)
```

or equivalently,

```
options = set_options('option', 'value', ...)
[solution, function_value, exitflag, output] = ...
GODLIKE(obj_function, lb, ub, confcn, options)
```

All of the arguments are the same as before. However, the additional arguments (which may also be empty) are

Additional Input Arguments:

`confcn` Constraint function(s); placeholder for future implementation of constraint functions. Currently unused.

`options` **(or** `('option', value)` **pairs)** These set the options to be used by GODLIKE. The argument `options` may be provided as a structure created by the function `set_options`, or more directly as ('option', value) pairs. As the possible options are quite numerous, these have been placed in their own section (see below).

Additional Output Arguments:

`exitflag` A simple value which informs the user of the conditions under which GODLIKE exited. As usual, positive values are "good", negative values are "bad". A value of 1 indicates normal convergence: the global optimum found by GODLIKE decreased less than `options.MinDescent(1) * options.MinDescentMultiplier` for more than `options.MinDescent(2)` GODLIKE iterations. A value of -1 indicates the loop was exited because the maximum amount of function evaluations had been exceeded, set by `options.MaxFunEvals`. A value of -2 means that GODLIKE terminated because the maximum amount of GODLIKE iterations has been exceeded, set by `options.MaxIters`.

`output` A structure that contains the following fields:

- `algorithms` lists the algorithms used in the optimization.
- `message` A message that contains a description of the reason GODLIKE terminated.
- `algorithm_info` A structure that contains more information on each of the optimizers. It contains the total number of function evaluations made by that optimizer, the total number of iterations made, and the population and fitness values of the last population before termination.
- `funcCount` total number of function evaluations made.
- `iterations` total number of GODLIKE iterations made.

Using GODLIKE for Multi-Objective Optimization Problems

Usage

Using multi-objective optimization is very much the same as single-objective optimization. Just include additional objective functions:

```
[solution, func_value, ...
Pareto_individuals, Pareto_fitnesses,...
exitflag, output] = ...
GODLIKE({obj_function1, obj_function2,...}, lb, ub, options)
```

where everything is the same as in single-objective optimization, except for the arguments `Pareto_individuals` and `Pareto_fitnesses`; these are the `[popsize` $\times$ `dimensions]` non-dominated individuals and `[popsize` $\times$ `number_of_objectives]` associated function values.

The globally most efficient point is computed by first shifting the whole front to a new origin, defined by the outermost individuals in function-value space, followed by finding the individual that has the minimum distance to that origin (again in function-value space). Note that the output argument `func_value` is now a `[popsize` $\times$ `number_of_objectives]` matrix.

The above may also be accomplished by issuing the command

```
[solution, func_value, ...
Pareto_individuals, Pareto_fitnesses,...
exitflag, output] = ...
GODLIKE(obj_function, lb, ub, options)
```

when `obj_function` is a single function that returns multiple function values in one `[popsize` $\times$ `num_of_objectives]` matrix (or `1` $\times$ `num_of_objectives]` if it only accepts `[1` $\times$ `dimensions]` input).

Available Options

individual options may be set directly by providing ('parameter', value) pairs to GODLIKE, or by first creating a structure in the same way using `set_options`. Here is a direct copy from the help on the function `set_options`, which lists all the available options:

```
% SET_OPTIONS                 Set options for the various optimizers
%
% Usage:
%
%   options = set_options('option1', value1, 'option2', value2, ...)
%
%
%   SET_OPTIONS is an easy way to set all options for the global optimization
%   algorithms PSO, DE, GA, ASA in GODLIKE. All options, and their possible
%   values are:
%
%   ========================================================================
%   General Settings:
%   ========================================================================
%       Display : string, either 'off' (default), 'on' or 'CommandWindow',
%                 'Plot'. This option determines the type of display that
%                 is used to show the algorithm's progress. 'CommandWindow'
%                 (or simply 'on') will show relevant information in the
%                 command window, whereas 'Plot' will make a plot in every
%                 iteration of the current population. Note that 'Plot'
%                 will only work if the number of decision variables is 1
%                 or 2 in case of single-pbjective optimization, or between
%                 1 and 3 objectives for multi-objective optimization.
%                 Please note that using any other display setting than
%                 'off' can significantly slow down the optimization.
%   MaxFunEvals : positive scalar, defining the maximum number of
%                 allowable function evaluations. The default is 100,000.
%                 Note that every objective and constraint function
%                 evaluation will be counted as 1 function evaluation. For
%                 multi-objective optimization, each objective function
%                 will be counted.
%      MaxIters : positive scalar, defining the maximum number of
%                 iterations that can be performed. The default is 20.
%      MinIters : positive scalar. This option defines the minimum amount
%                 of iterations GODLIKE will perform. This is particularly
%                 useful in multi-objective problems with small population
%                 sizes, because this combination increases the probability
%                 that GODLIKE reports convergence (all fronts are Pareto
%                 fronts), while a Pareto front of much better quality is
```

```
%                 obtained if some additional shuffles are performed. The
%                 default value is 2.
%   UseParallel : logical, either false (default), or true. If enabled, it
%                 will use run function evaluations within each
%                 generation in parallel. It uses MATLAB's native parfor
%                 keyword for this, utilizing the current parallel
%                 execution pool (see parfor for more info).
%
%   ========================================================================
%   Options specific to the GODLIKE Algorithm:
%   ========================================================================
%       ItersLb : positive scalar. This sets the minimum number of
%                 iterations that will be spent in one of the selected
%                 heuristic optimizers, per GODLIKE iteration. The default
%                 value is 10.
%       ItersUb : positive scalar. This sets the maximum TOTAL amount of
%                 iterations that will be spent in all of the selected
%                 heuristic optimizers combined. The default value is 100.
%       popsize : Total population size for all global optimization
%                 algorithms used, combined. When omitted, defaults to
%                 25 times the number of dimensions.
%    algorithms : The algorithms to be used in the optimizations. May
%                 be a single string, e.g., 'DE', in which case the
%                 optimization is equal to just running a single
%                 Differential Evolution optimization. May also be a
%                 cell array of strings, e.g., {'DE'; 'GA'; 'ASA'},
%                 which uses all the indicated algorithms. When
%                 omitted or left empty, defaults to {'DE';'GA';'PSO';
%                 'ASA'} (all algorithms once).
%
%   ========================================================================
%   General Settings for Single-Objective Optimization:
%   ========================================================================
%        TolIters: positive scalar. This option defines how many consecutive
%                  iterations the convergence criteria must hold for each
%                  individual algorithm, before that algorithm is said to
%                  have converged. The default setting is 15 iterations.
%           TolX : positive scalar. Convergence is assumed to be attained,
%                  if the coordinate differences in all dimensions for a
%                  given amount of consecutive iterations is less than
%                  [TolX]. This amount of iterations is [TolIters] for each
%                  individual algorithm, and simply 2 for GODLIKE-iterations.
%                  The default value is 1e-4.
%          TolFun : positive scalar. Convergence is said to have been
%                  attained if the value of the objective function decreases
%                  less than [TolFun] for a given amount of consecutive
```

```
%                 iterations. This amount of iterations is [TolIters] for
%                 each individual algorithm, and simply 2 for the
%                 GODLIKE-iterations. The default value is 1e-4.
%  AchieveFunVal : scalar. This value is used in conjunction with the
%                 [TolX] and [TolFun] settings. If set, the algorithm will
%                 FIRST try to achieve this function value, BEFORE enabling
%                 the [TolX] and [TolFun] convergence criteria. By default,
%                 it is switched off (equal to AchieveFunVal = inf).
%
%   ========================================================================
%   General Settings for Multi-Objective Optimization:
%   ========================================================================
%   NumObjectives : Positive scalar. Sets the number of objectives manually.
%                   When the objective function is a single function that
%                   returns multiple objectives, the algorithm has to first
%                   determine how many objectives there are. This takes some
%                   function evaluations, which may be skipped by setting this
%                   value manually.
%
%   ========================================================================
%   Options specific to the Differential Evolution algorithm:
%   ========================================================================
%          Flb : scalar. This value defines the lower bound for the range
%                from which the scaling parameter will be taken. The
%                default value is -1.5.
%          Fub : scalar. This value defines the upper bound for the range
%                from which the scaling parameter will be taken. The
%                default value is +1.5. These two values may be set equal
%                to each other, in which case the scaling parameter F is
%                simply a constant.
%   CrossConst : positive scalar. It defines the probability with which a
%                new trial individual will be inserted in the new
%                population. The default value is 0.95.
%
%   ========================================================================
%   Options specific to the Genetic Algorithm:
%   ========================================================================
%      Crossprob : positive scalar, defining the probability for crossover
%                  for each individual. The default value is 0.25.
%   MutationProb : positive scalar, defining the mutation probability for
%                  each individual. The default value is 0.1.
%         Coding : string, can either be 'binary' or 'real'. This decides
%                  the coding, or representation, of the variables used by
%                  the genetic algorithm. The default is 'Binary'.
%        NumBits : positive scalar. This options sets the number of bits
%                  to use per decision variable, if the 'Coding' option is
```

```
%                 set to 'Binary'. Note that this option is ignored when
%                 the 'Coding' setting is set to 'real'. The default
%                 number of bits is 52 (maximum precision).
%
%   ====================================================================
%   Options specific to the Adaptive Simulated Annealing Algorithm:
%   ====================================================================
%              T0 : positive scalar. This is the initial temperature for
%                   all particles. If left empty, an optimal one will be
%                   estimated; this is the default.
%  CoolingSchedule : function handle, with [iteration], [T0], and[T] as
%                   parameters. This function defines the cooling schedule
%                   to be applied each iteration. The default is
%
%                     @(T,T0,iteration) T0 * 0.87^iteration
%
%                   It is only included for completeness, and testing
%                   purposes. Only in rare cases is it beneficial to change
%                   this setting.
%        ReHeating : positive scalar. After an interchange operation in
%                   GODLIKE, the temperature of an ASA population should
%                   be increased to allow the new individuals to move
%                   over larger portions of the search space. The default
%                   value is
%
%   ====================================================================
%   Options specific to the Particle Swarm Algorithm:
%   ====================================================================
%           eta1 : scalar < 4. This is the 'social factor', the
%                  acceleration factor in front of the difference with the
%                  particle's position and its neighorhood-best. The
%                  default value is 2. Note that negative values result in
%                  a Repulsive Particle Swarm algorithm.
%           eta2 : scalar < 4. This is the 'cooperative factor', the
%                  acceleration factor in front of the difference with the
%                  particle's position and the location of the global
%                  minimum found so far. The default value is 2.
%           eta3 : scalar < 4. This is the 'nostalgia factor', the
%                  acceleration factor in front of the difference with the
%                  particle's position and its personal-best. The default
%                  value is 0.5.
%          omega : scalar. This is the 'inertial constant', the tendency of
%                  a particle to continue its motion undisturbed. The
%                  default value is 0.5.
%   NumNeighbors : positive scalar. This defines the maximum number of
%                  'neighbors' or 'friends' assigned to each particle. The
```

```
%                   default value is 5.
% NetworkTopology: string, equal to either 'fully_connected', 'star', or
%                   'ring'. This defines the topology of the social network
%                   for each particle. In case 'star' is selected (the
%                   default), the setting for NumNeighbors will define the
%                   total number of partiles per star; the same holds in
%                   case 'ring' is selected. When 'fully_connected' is
%                   selected however, the value for NumNeighbors will be
%                   ignored (all particles are connected to all other
%                   particles).
```