RANDOM SEARCH WITH ADAPTIVE BOUNDARIES (RSAB) & REPULSIVE FORCES OPTIMIZATION ALGORITHM (REF)

This document introduces an optimization model for unconstrained/bounded & constrained continuous optimization problems. This model includes two algorithms can either be applied single or hybrid.

The first one is structured as a generic method that can be applied in the initialization stage of any algorithm for optimization problems and it depends on updating given upper and/or lower boundaries dynamically according to parameters. The outstanding feature of the first algorithm is the ability to reach better initial solutions by reducing the diversity of pure randomness, to some extent, for continuous unconstrained/bounded and constrained nonlinear optimization problems that may have many local optimums.

The REF algorithm which can be defined as the main algorithm is based on physics to provide the best-known solutions for constraint handling. This algorithm assumes that the particles that are likely charged repel each other considering their neighbors to reach new locations where better solutions may exist. The calculation of repulsive forces mimics Coulomb’s Law and the movement is calculated according to the Momentum Law. Furthermore, Tabu Search and Elitism selection are also inspired in terms of the memory structure of the algorithm. This algorithm handles constraints with the help of a multiplicative penalty approach that considers satisfaction rate and the deviations of constraints besides objective function value.

These algorithms were prepared under the supervision of Prof. Dr. Sabri Erdem within the scope of Gülin Zeynep Öztaş's PhD dissertation. More detailed information can be obtained from the thesis itself or from the authors. In addition, updated information will be added here as current publications.

RSAB and REF algorithms have been coded in Python language, and PyCharm developed by the Czech company JetBrains was utilized as IDE.

Python Libraries

Package	Version	Latest Version
beautifulsoup4	4.9.0	4.9.3
cmp	0.0.1	0.0.1
compareMe	1.0	1.0
cycler	0.10.0	0.10.0
docutils	0.16	-
et-xmlfile	1.0.1	1.1.0
jdcal	1.4.1	1.4.1
kiwisolver	1.2.0	1.3.1
matplotlib	3.2.1	3.4.2
numpy	1.18.3	1.20.3
openpyxl	3.0.4	3.0.7
opfunu	0.8.0	0.8.0
pandas	1.0.3	1.2.4
pip	19.0.3	21.1.1
pstats2	0.1.0	0.1.0
pyparsing	2.4.7	2.4.7
pyprof2calltree	1.4.5	1.4.5
pyroots	0.3.2	0.5.0
python-dateutil	2.8.1	2.8.1
pytz	2019.3	2021.1
recordtype	1.3	1.3
recursive-itertools	0.2.2	0.2.2
scipy	1.4.1	1.6.3
setuptools	40.8.0	56.2.0
six	1.14.0	1.16.0
soupsieve	2.0	2.2.1
statistics	1.0.3.5	1.0.3.5
xlwt	1.3.0	1.3.0

Content of Python codes

- atom.py (functions)
- Best_Three.py (saves best three solutions)
- check_duplication.py
- create.py (create result tables)
- determine.py
- dictionary.py (database of unconstrained/bounded benchmarks and engineering design problems)
- displacement.py (alternative)
- duplication.py (alternative)
- evaluate2.py
- export.py (export results)
- findneighbors.py
- ForceIt.py
- init.py
- main.py
- main_2.py
- ReLocate.py
- rsab_v3.py (functions)
- update.py
- UpdateBests.py
- variables.py (definition of variables)

Pseudo-code of RSAB [main_2.py]

```
 1:  Determine intervals (Initial Limits) [determine.py]
 2:  Create initial 1000 sized random solution vectors
 3:  For each solution vector
 4:         Evaluate constraints [evaluate2.py]
 5:  For each iteration
 6:         If Improvement = FALSE
 7:              Initial Limits
 8:              If unconstrained problem
 9:                   Update intervals by using the midpoint
10:                 Else
11:                   Update intervals by using the holdbest
12:         Else
13:              If unconstrained problem
14:                   Update intervals by using the holdbest
15:                 Else
16:                   Update intervals by using the midpoint
17:         For each variable
18:                Create random values based on new intervals
19:         For each solution vector
20:                Evaluate constraints [evaluate2.py]
21:         Store Updated Interval
22:  Loop Until maximum iteration given
23:  For each variable
24:      Calculate means, modes, medians of lower-upper limits
25:      Updated Lower Limit = min (Mean_L, Mode_L, Median_L)
26:      Updated Upper Limit = max (Mean_U, Mode_U, Median_U)
```

Pseudo-code REF [main.py]

```
 1: Determine Intervals [determine.py]
 2: Create Initial Particles [init.py]
 3: Evaluate Constraints [evaluate2.py]
 4: For Each Iteration
 5:      Until stopping condition is met
 6:      For Each Particle
 7:           Find Neighbors [findneighbors.py]
 8:           For Each Neighbor
 9:                Find Incremental Replacements [update.py & ForceIt.py]
10:                If f(x) reduces Go to New Location [ReLocate.py]
11:                Update New State [UpdateBests.py]
12:      Check Duplication [check_duplication.py]
```

Note!

Do not forget to update the location to save the results as excel documents in export.py file and main_2.py.

The user interface where the algorithm can be operated easily will be developed and released soon.