GECCO 20220 - Competition on Single Objective Bound Constrained Numerical Optimization) - Submission

Nelishia Pillay
Department of Computer Science, University of Pretoria
Gauteng, South Africa
npillay@cs.up.ac.za

Mia Gerber
Department of Computer Science, University of Pretoria
Gauteng, South Africa
u15016502@tuks.co.za

May 12, 2022

1 Introduction

An ensemble of single point selection perturbative hyper-heuristics (SPHHs) is being used to perform single-objective bound constrained numerical optimization. Section 2 presents the algorithm for the SPHH and Section 3 explains what the heuristic combination for the SPHH consists of and gives an example. Section 4 lists and explains the low level perturbative heuristics and Section 5 presents how the fitness score is calculated. Section 6 discusses all the heuristic selection techniques that are being used and Section 7 presents all the move acceptance techniques that are being used. Section 8 lists all the hyper-parameters and explains how their values were chosen. Section 9 discusses how ensembles of SPHHs were created and also gives the algorithm for how the ensembles of SPHHs are used to find optimal values for the 12 different functions.

The files attached to this submission are as follows:

- Algorithm_Complexity.txt
- Results_for_10D.txt
- Results_for_20D.txt
- SPHH_Ensemble_1_10.txt

- SPHH_Ensemble_1_20.txt
- SPHH_Ensemble_2_10.txt
- SPHH_Ensemble_2_20.txt
- SPHH_Ensemble_3_10.txt
- SPHH_Ensemble_3_20.txt
- SPHH_Ensemble_4_10.txt
- SPHH_Ensemble_4_20.txt
- SPHH_Ensemble_5_10.txt
- SPHH_Ensemble_5_20.txt
- SPHH_Ensemble_6_10.txt
- SPHH_Ensemble_6_20.txt
- SPHH_Ensemble_7_10.txt
- SPHH_Ensemble_7_20.txt
- SPHH_Ensemble_8_10.txt
- SPHH_Ensemble_8_20.txt
- SPHH_Ensemble_9_10.txt
- SPHH_Ensemble_9_20.txt
- SPHH_Ensemble_10_10.txt
- SPHH_Ensemble_10_20.txt
- SPHH_Ensemble_11_10.txt
- SPHH_Ensemble_11_20.txt
- SPHH_Ensemble_12_10.txt
- SPHH_Ensemble_12_20.txt

Note that all the files are in TSV (tab separated values) format, but have a .txt file extension as per the specification for the competition.

2 Algorithm

The pseudo code for the SPHH algorithm is given below:

```
f = the function to be minimised
problem_dimensions = 10 or 20
current_hc = random matrix of values between -100 and 100
current_fitness = mean( f(current_hc) - f(optimal) )
best_hc = current_hc
best_fitness = current_fitness
exit_threshold = 25
max_iterations = exit_threshold * 2
i = 0

while i < max_iterations:

        if iterations_since_increase > exit_threshold:
                return best_hc as solution

        random_x_indices = select 10 random numbers between 0 and 19
        random_y_indices = select 10 random numbers between 0 and 19

        prev_current_fitness = current_fitness
        prev_best_fitness = best_fitness

        for x in random_x_indices:
                for y in random_y_indices:

                        llph = select_llph()
                        new_hc = apply llph to value at current_hc[x][y]
                        new_fitness = mean( abs( f(new_hc) - f(optimal) ) )
                        move_accepted = check_move_acceptance()

                        if move_accepted == True:
                                current_fitness = new_fitness
                                current_hc = new_hc

                        if new_fitness ¡ best_fitness:
                                best_fitness = new_fitness
                                best_hc = new_hc

        if prev_best_fitness == best_fitness:
                if prev_current_fitness > current_fitness:
                        iterations_since_increase += 0.5
                else if prev_current_fitness < current_fitness:
                        iterations_since_increase += 1.0
        else:
                iterations_since_increase = 0

        i++

return best_hc as solution
```

3 Heuristic combination representation

The heuristic combination will represent the parameters to be optimized, where the size of the array corresponds to the problem dimensions. An example of the representation for a problem with 10x10 dimensions is as follows:

```
[
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
    [1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 1.5, 3.5, 2.2, 6.4],
]
```

4 Low level perturbative heuristics

Each LLPH will operate on a single element of the heuristic combination ($hc[x][y]$) as well as an additional float value. The additional float value is randomly generated in the range specified. The output for all LLPHs are scaled to lie between [-100,100] meaning if the application of an LLPH results in a value that is either larger than 100 or smaller than -100, that value will be capped to either 100 or -100.

- Add two digit value.
  hc[x][y] + float value between 10.0 and 99.9999
- Add one digit value.
  hc[x][y] + float value between 1.0 and 9.9999
- Add fractional value.
  hc[x][y] + float value between 0.0009 and 0.9999
- Subtract two digit value.
  hc[x][y] - float value between 10.0 and 99.9999
- Subtract one digit value.
  hc[x][y] - float value between 1.0 and 9.9999
- Subtract fractional value.
  hc[x][y] - float value between 0.0009 and 0.9999

- Multiply positive value.
  hc[x][y] * float value between 0.0 and 100.0
- Multiply negative value.
  hc[x][y] * float value between -100.0 and -0.0009
- Divide positive value.
  hc[x][y] / float value between 0.0 and 100.0
- Divide negative value.
  hc[x][y] / float value between -100.0 and -0.0009

5 Fitness evaluation

The fitness is evaluated according to the equation below:

$$fitness = mean(abs(f(hc) - f(optimal)) \tag{1}$$

In the equation above hc is the current heuristic combination that is being evaluated and *optimal* represents the known optimal values for f.

6 Heuristic selection techniques

The heuristic selection techniques that are implemented are listed below:

- Ranking
  This heuristic selection technique uses a choice function to rank each LLPH and the LLPH with the highest rank is selected. For more details see [1].
- Random
  This heuristic selection technique selects a random LLPH
- Reinforcement learning
  This heuristic selection technique approaches the choice of which LLPH to select as a reinforcement learning problem. For more details see [2].
- Tabu list
  This heuristic selection technique adds poor performing LLPHs to a tabu list and heuristic selection is done by picking a random LLPH that is not on the tabu list. For more details see [3].

7 Move acceptance techniques

The move acceptance techniques that are implemented are listed below:

- Adaptive Iteration Limited Threshold Acceptance (AILTA)
  AILTA works by accepting moves that result in an equal or higher fitness value than prior to the move. AILTA will also accept moves that result in lower fitness values if certain conditions are met. For more details see [4].

- Improving
  This move acceptance technique will simply only accept improving moves and reject all other moves.

- Accept all
  This move acceptance technique will accept all moves.

- Great deluge
  This move acceptance technique will accept all moves that improve the fitness but moves that degrade the fitness will only be accepted if the amount by which the fitness is degraded is less than the threshold. For more details see [5].

- Late acceptance
  This move acceptance technique works by maintaining a history of previous fitness values. A move is then accepted if it improves upon a past (not the previous) fitness value, otherwise the move is rejected.

- Simulated annealing
  This move acceptance technique works by accepting all improving moves. Any moves that are equal or degrade the fitness value are accepted if the simulated annealing equation returns a value larger than that of a random value drawn from a uniform distribution between 0.0 and 1.0. [5].

8 Hyperparameters

The hyperparameters for the SPHH including the aforementioned heuristic selection techniques and move acceptance techniques are all listed below:

- Exit threshold: 25 iterations
- Maximum iterations: 50 iterations
- AILTA alpha: 0.5
- AILTA beta: 0.5
- AILTA delta: 0.25
- Reinforcement learning epsilon: 0.4
- Reinforcement learning alpha: 0.85
- Reinforcement learning gamma: 0.45

- Tabu list size: 6
- Late acceptance distance: 3 iterations

The hyper-parameters were decided upon after empirical experimentation and manual tuning, taking into account how all 12 functions were affected by changes in hyper-parameters values. After hyper-parameter tuning the hyper-parameter values were kept the same for all 12 functions and problem dimensions.

9 Ensembling

Ensembling is done in two ways. Firstly different selection methods and move acceptance methods are ensembled within a single SPHH, this is referred to as "internal ensembling". Secondly, different SPHH instances are ensembled together, this is referred to as "external ensembling".

9.1 Internal ensembling

Three different modes of internal ensembling exist:

9.1.1 Static selection, ensembled acceptance (SSEA)

A single heuristic selection technique is used and an ensemble of move acceptance techniques are used. Each technique in the move acceptance ensemble will either vote to accept or reject the move, the move is then accepted or rejected based on the majority of votes.

9.1.2 Ensembled selection, static acceptance (ESSA)

An ensemble of heuristic selection techniques are used and a single move acceptance technique is used. Each technique in the heuristic selection ensemble will choose a heuristic and then majority voting is used to select a heuristic to apply.

9.1.3 Ensembled selection, ensembled acceptance (ESEA)

An ensemble of heuristic selection techniques and an ensemble of move acceptance techniques are used. Each technique in the heuristic selection ensemble will choose a heuristic and then majority voting is used to select a heuristic to apply. Each technique in the move acceptance ensemble will either vote to accept or reject the move, the move is then accepted or rejected based on the majority of votes.

9.2 External ensembling

Three different modes of external ensembling exist and are discussed below. In all cases, each SPHH in the ensemble will be given the same heuristic combination to optimise and after a certain amount of time the best performing SPHH in the ensemble will have its heuristic combination chosen as the final solution.

9.2.1 Ensemble of SSSA SPPHs

The default way to use the SPPH is by having a static heuristic selection and static acceptance method (SSSA). There are 4 heuristic selection and 6 move acceptance techniques available, meaning the total number of unique SPPH configurations is 24. When creating an ensemble of SSSA SPPHs, all 24 SPPH configurations will be instantiated and used.

9.2.2 Ensemble of SSEA SPPHs

When creating an ensemble of SSEA SPPHs, 4 SPPH configurations will be instantiated and used as there are 4 heuristic selection techniques available.

9.2.3 Ensemble of ESSA SPHHs

When creating an ensemble of ESSA SPPHs, 6 SPPH configurations will be instantiated and used as there are 6 move acceptance techniques available.

9.3 Ensembling algorithm

```
move_acceptance_methods = ['ailta','improving','accept_all','great_deluge',
'late_accepting','simulated_annealing']
selection_methods = ['rank','random','reinforcement_learning','tabu_list']
current_hc = randomly create heuristic combination
current_hc = Run ESEA on current_hc

counter = 0

while counter < 4:

    worst_selection_method = None
    worst_move_acceptance = None

    SSEA_ensemble = Create ensemble of SSEA SPHHs
    Run SSEA_ensemble on current_hc
    current_hc = get hc from SPHH with the best fitness in SSEA_ensemble

    ESSA_ensemble = Create ensemble of ESSA SPHHs
    Run ESSA_ensemble on current_hc
    current_hc = get hc from SPHH with the best fitness in ESSA_ensemble

    SSSA_ensemble = Create ensemble of SSSA SPHHs
    Run SSSA_ensemble on current_hc
    current_hc = get hc from SPHH with the best fitness in SSSA_ensemble
    worst_selection_method = used by worst SPHH in SSSA_ensemble
    worst_move_acceptance = used by worst SPHH in SSSA_ensemble

    Remove worst_selection_method from selection_methods
    Remove worst_move_acceptance from move_acceptance_methods

    counter++
```

10 Technical specifications

All techniques discussed above were developed in the Python language, specifically Python 3.8. The experiments were conducted in a multicore environment consisting of 24 Intel Xeon E5-2680 CPUs.

References

[1] G. Kendall, P. Cowling, E. Soubeiga *et al.*, "Choice function and random hyperheuristics," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, 2002, pp. 667–671.

[2] Y. Jia, M. B. Cohen, M. Harman, and J. Petke, "Learning combinatorial interaction test generation strategies using hyperheuristic search," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 540–550.

[3] F. Glover and M. Laguna, "Tabu search principles," in *Tabu Search*. Springer, 1997, pp. 125–151.

[4] M. Misir, K. Verbeeck, P. De Causmaecker, and G. V. Berghe, "Hyperheuristics with a dynamic heuristic set for the home care scheduling problem," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[5] L. N. Ahmed, E. Özcan, and A. Kheiri, "Solving high school timetabling problems worldwide using selection hyper-heuristics," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5463–5471, 2015.