TASK 3
supporting material

Fabio Caraffini

De Montfort University

School of Computer Science and Informatics
– Institute of Artificial Intelligence (IAI) –

Report's format

The format of the following papers can be used to structure your (TASK 3) report. Paper [3] in particular is relevant to TASK 3 as it describes a simple, yet efficient, single-solution Memetic Computing algorithm and therefore can be read to draw some inspiration[1].

- Paper [3]: aip.scitation.org/doi/abs/10.1063/1.5089971
  (for extended results CLICK HERE)
- Paper [11]: aip.scitation.org/doi/abs/10.1063/1.5089969
  (for extended results CLICK HERE)
- Paper [4]: aip.scitation.org/doi/abs/10.1063/1.5089972
  (for extended results CLICK HERE)

[1] source code is also made available in the remainder of these slides

Some ideas to design simple MC algorithms[2]:

- Paper [10]: "Ockham's razor in MC"(a simple bottom-up approach).
- Paper [6, 3]: seriously simple iterated local serches.
- Paper [5, 7, 8, 1]: multi-mems MC/hyperheuristic examples.
- Paper [2]: DE framework with extra moves along the axes.
- Paper [9]: a more complex hybrid algorithm.

[2]N.B. The MC algorithm does not necessarily need to be population based!

Source code of [3]

- Source code is available in this shared folder (CLICK HERE);
  - Download "CMAES_11.java" and "RI_CAMES11.java";
  - move them in the "algorithms" folder of your SOS installations;
- create an experiment file to compare the two algorithms;
- launch the experiment from "RunExperiment.java"!

N.B. The previous steps should be very simple at this stage but should you need further help please watch click HERE and HERE for watching SOS HOW TO videos.

Task 3 report

some examples from the past

To you give you further support some good submissions are made available in this folder (CLICK HERE)

N.B. Parts displaying source code or solutions have been intentionally removed.

SOS
some advice, tricks, dos & dont's

- If you are not planning on plotting fitness trends, then minimise the use of "FT.add(...)" as that command is only used to save fitness values into .txt files and will **significantly slow down** the optimisation process!
- Sometimes also using "<" instead of "<=" in the survivor selection process will speed up your simulations!
- A smart way for randomly selecting a number "n" of **different** individuals from the population is via the method "randomPermutation()":
  - simply import the "RandUtils" class: *import utils.random.RandUtils;*
  - initialise an array, e.g. *int[] index= new int[NP-1];* where NP is the population size, with numbers from 0 to NP-1;
  - then shuffle indices with *RandUtils.randomPermutation(index);* and select the first n indices of *index*
- always use *toro(...);* after perturbing an individual and before calling *Problem.f(...);*
- always increase the the budget counter after calling *Problem.f(...);*
- Do not use *Problem.f(...);* if not necessary (e.g. if you already have the fitness value stored in memory) as it will unnecessarily eat your available computational budget.

Advice for your MSc project

Make to most out of TASK 3!

- If you enjoyed this module and have a specific idea/interest that you want to investigate (it has to do with optimisation) use TASK 3 as a pilot study!
- Select an application that can be formulated as an optimisation problem or a topic that you like (i.e. design a novel optimisation algorithm, optimisation in robotics or image processing, etc.) and use it for TASK 3...
- ...then extend it in your MSc project!
- Some students have expressed the interest in publishing their project (N.B. it has be very good) and I always give support and help to transform it in a paper and submit it!

References I

Fabio Caraffini and Ferrante Neri and Michael Epitropakis.
Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain.
Information Sciences, 477:186 – 202, 2019.

F. Caraffini, F. Neri, and I. Poikolainen.
Micro-differential evolution with extra moves along the axes.
In *IEEE Symposium Series on Computational Intelligence, Symposium on Differential Evolution*, pages 46–53, April 2013.

Fabio Caraffini, Giovanni Iacca, and Anil Yaman.
Improving (1+1) covariance matrix adaptation evolution strategy: A simple yet efficient approach.
AIP Conference Proceedings, 2070(1):020004, 2019.

Fabio Caraffini and Anna V. Kononova.
Structural bias in differential evolution: A preliminary study.
AIP Conference Proceedings, 2070(1):020005, 2019.

References II

Fabio Caraffini, Ferrante Neri, Giovanni Iacca, and Aran Mol.
Parallel memetic structures.
Information Sciences, 227(0):60–82, 2012.

Fabio Caraffini, Ferrante Neri, Benjamin N Passow, and Giovanni Iacca.
Re-sampled inheritance search: high performance despite the simplicity.
Soft Computing, 17(12):2235–2256, 2013.

Fabio Caraffini, Ferrante Neri, and Lorenzo Picinali.
An analysis on separability for memetic computing automatic design.
Information Sciences, 265(0):1 – 22, 2014.

Michael G Epitropakis, Fabio Caraffini, Ferrante Neri, and Edmund K Burke.
A separability prototype for automatic memes with adaptive operator selection.
In *2014 IEEE symposium on foundations of computational intelligence (FOCI)*, pages 70–77. IEEE, 2014.

References III

Giovanni Iacca, Fabio Caraffini, and Ferrante Neri.
Multi-strategy coevolving aging particle optimization.
International journal of neural systems, 24(01), 2014.

Giovanni Iacca, Ferrante Neri, Ernesto Mininno, Yew-Soon Ong, and Meng-Hiot Lim.
Ockham's razor in memetic computing: three stage optimal memetic exploration.
Information Sciences, 188:17–43, 2012.

Anil Yaman, Giovanni Iacca, and Fabio Caraffini.
A comparison of three differential evolution strategies in terms of early convergence with different population sizes.
AIP Conference Proceedings, 2070(1):020002, 2019.