Memetic Algorithms
and
Memetic Computing

Memetic Algorithm (MA)

generalities:

- The term Memetic Algorithm (MA) is coined by Moscato [Moscato and Norman, 1989]. . .
- . . . but the same idea was also given under the name of
  - Hybrid GAs (already in the 80s);
  - Baldwinian GAs;
  - Lamarckian GAs;
  - others. . .

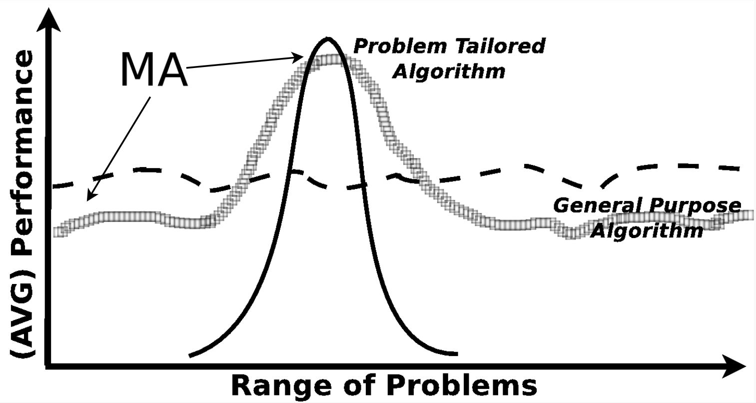
MA: the metaphor

- The word Meme is borrowed from from Dawkins' Universal Darwinism theory, see the "Selfish Gene" [Dawkins, 1976].
- The Meme is a unit of "cultural transmission" in the same way that genes are the units of biological transmission.
- In EAs, genes are encoding of candidate solutions, in MAs the memes are also "strategies" of how to improve the solutions.
- We may think that on the top of the genetic evolution, solutions can "go to school" and learn during their life-time to become fitter.

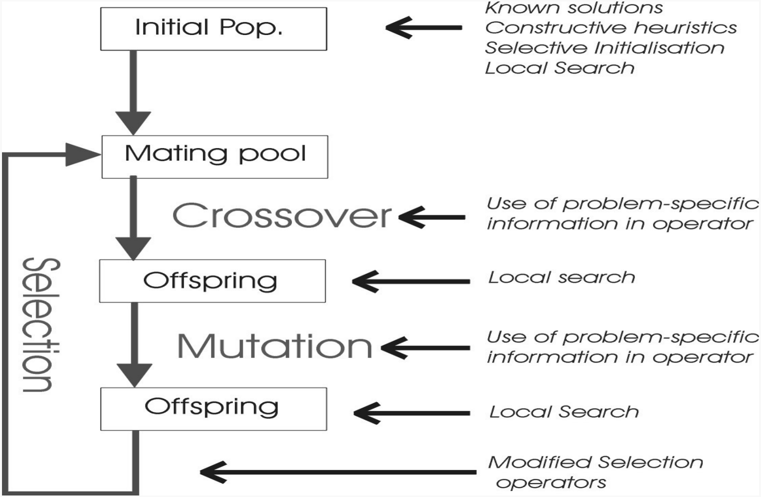
What is an MA?

- The combination of Evolutionary Algorithms with Local Search Operators that work within the generation loop has been termed "Memetic Algorithms".
- MA can refer also to EAs that use instance specific knowledge in operators
  - such ad-hoc XOs or mutations, etc. ...)
- MAs have been shown to be orders of magnitude faster and more accurate than EAs on some problems, and are the "state of the art" on many problems.

NFLT reinterpretation



MA general scheme



Intelligent initialisation

- The initial population is not given at pseudo-random but it is given according to a heuristic rule.
  - e.g. quasi-random generator, orthogonal arrays, super-fit,...
- N.B. It increases the average fitness but it decreases the diversity!

Intelligent variation operators

- Intelligent Crossover: finds the best combination between parents in order to generate the most performing offspring (e.g. heuristic selection of the cut point).
- Intelligent Mutation: tries several possible mutated individuals in order to obtain the most "luck" mutation (e.g. bit to flip).

Local search
neighborhood

- Optimisation algorithms generate trial solutions. At each step, from a starting point (or set of solutions) an algorithm can potentially reach a set of solutions;
- the set of point that can be reached with a single move is named "neighborhood":
  - If the neighborhood is (potentially) the entire decision space the metaheuristic is a global optimiser
  - if it is a proper sub-set of the decision space, the metaheuristic is a local search operator/local searcher/local search algorithm.

Locality of the search

- It follows that while metaheuristics can be divided into two crisp categories, the locality is an intensive property, a local search can be more or less local than another (analogous to exploration/exploitation).

Deterministic VS stochastic

- Local searchers can be deterministic or stochastic:
  - deterministic local search usually select the new base points on the basis of the gradient direction;
  - in the latter case the algorithm, given a proper radius, is supposed to reach the local optimum belonging to the corresponding basin of attraction.

Properties of local search

local Searchers can be classified according to:

- order;
- pivot rule;
- depth;
- neighborhood generating function.

Local Search Order

- Order zero if it uses just the function (direct search, metaheuristics);
- we don't have any of them in this module but for completeness:
  - order one if it uses the first derivative;
  - order two if it uses the second derivative.

Pivot rule

- *Steepest Descent Pivot Rule*: the Local Search explores all the possible moves before selecting the new base point (set of points).
- *Greedy Pivot Rule*: the Local Searcher chooses the first better search direction found.

The pivot rule is also an intensive property:

A local search can have a pivot rule that is neither fully greedy nor fully steepest descent.

Depth and Neighborhood Generating Function

- The depth of the Local Search defines the termination condition for the outer loop (stop criterion).
- The neighborhood generating function $\phi(i)$ defines a set of points that can be reached by the application of some move operator to the point i.

Main properties of some popular LSs
from [Caraffini, 2014]

LS algorithm	Search Logic	Derivatives	Memory Footprint	Processed Points	Convergence
Newton	*Deterministic (gradient descent)*	1^{st} *and* 2^{nd} *order*	$\mathcal{O}\left(n^2\right)$ *(Hessian matrix)*	*Single-solution*	q-quadratically[a]
Hook-Jeeves	*Deterministic*	*Derivative free*	$\mathcal{O}\left(n\right)$	*Single-solution*	No proof
S	*Deterministic (along the axis)*	*Derivative free*	$\mathcal{O}\left(n\right)$	*single-solution*	No proof
Nelder-Mead	*Deterministic*	*Derivative free*	$\mathcal{O}\left(n^2\right)$ *(simplex vertices)*	*Multiple-solution*	Convergence[b]
Rosenbrock	*Deterministic (diagonal move)*	*Derivative free*	$\mathcal{O}\left(n^2\right)$ *(rotation matrix)*	*Single-solution*	Convergence[c]
Powell	*Deterministic (diagonal move)*	*Derivative free*	$\mathcal{O}\left(n^2\right)$ *(directions matrix)*	*Single-solution*	$n\left(n+1\right)$ steps[d]
Solis-Wets	*Stochastic (diagonal move)*	*Derivative free*	$\mathcal{O}\left(n\right)$	*Single-solutions*	Convergence[e]
SPSA	*Stochastic (gradient descent)*	1^{st} *order*	$\mathcal{O}\left(n\right)$	*Single-solutions*	Convergence[f]

[a] Only for uni-modal and locally twice Lipschitz continuously differentiable functions.

[b] For convex functions in 1 and 2 dimensions.

[c] Under hypothesis on the fitness, e.g. differentiability, and on the line search method [Rinaldi, 2012]

[d] Using the classic method in [Powell, 1964] on quadratic forms.

[e] If f quasi-convex and inf-compact, converges in a neighbourhood of $\mathbf{X}^*_{\text{local}}$ [Solis and Wets, 1981].

[f] Under conditions on f and the distribution of probability used as in [Spall, 1992].

Lifetime learning:
LS acting on offspring

- Solutions undergo local search at the generation time;
- the LS is applied to the offspring in order to have more performing individuals;
- a LS can be viewed also like a special mutation operator and it is often (but not only!) used to speed-up the “endgame” of an EA by making the search in the vicinity;
    - in fact the EAs are efficient in finding solutions near the optimum but not in finalising the search!

How to apply a local searcher?

- How many iterations of the local search are done?
- Is local search applied to the whole population?
    - or just to the best solution?
    - or just to the worst?
    - or to a certain part of the population according to some rules?
- Basically, (as usual) the right choice depends on the problem!

Two models of Lifetime Learning

- **Lamarckian:**
  traits acquired by an individual during its lifetime can be transmitted to its offspring (refreshing of the genotype), e.g. replace individual with fitter neighbour.

- **Baldwinian:**
  traits acquired by individual cannot be transmitted to its offspring (suggests new direction search), e.g. individual receives fitness (but not genotype) of fitter neighbour.

Memetic vs Multi-Meme

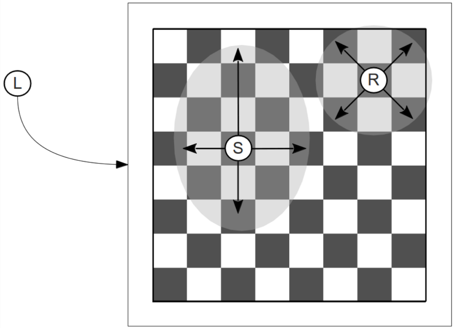
- A Meme Algorithm uses one LS (usually complex);
- while a Multi-Meme Algorithm (M-MA) employs a set (a list) of LSs (usually simple).

If a M-MA is implemented the problem of how and when to run the LSs arises

and as usual, some rules are therefore needed.

Diversity of the operators

- [Krasnogor, 2002] shows that there are theoretical advantages to using a local search with a move operator (LS to the offspring) that is different to the move operators used by mutation and crossover
- Multiple operators (e.g. LS) are like pieces on a chessboard [Caraffini et al., 2013a].



Coordination

- The more parts compose an algorithm, the more difficult is to coordinate them:
  - sometimes is better to employ few and simple operators [Iacca et al., 2012] than too many complex ones,
  - and select them so that are not redundant (high diversity of moves) [Caraffini et al., 2013a],
  - and focus on their efficient coordination logic rater than on the efficiency and complexity of the single operator (algorithmic structure is as important as the choice of the operators! [Caraffini et al., 2012a]).
- In order to enhance the efficiency and the robustness of a MA an adaptive or self-adaptive scheme can be used:
  - meta-lamarckian learning, hyper-heusristic selections strategies, etc.

Adaptation

- *Adaptive*: the operators are controlled during the evolution by means of some rules depending on the state of the population or on a feedback.

Philosophy:

if the "necessities" of the problem are efficiently encoded it is possible to use different LSs in different moments and on different individuals (or set of individuals).

Self-Adaptive MAs

- *Pure Self-Adaptive [Krasnogor and Smith, 2000]*: the adaptive rules are encoded in the genotype of each individual.
- *Co-evolutionary [Smith, 2007]*: Two populations, one of solutions and one of operators. The solutions are somehow linked to the operators and co-evolve.

Similar to strategy parameters in EP

- LSs are evolved as well: useless LSs get discarded.
- LS is sensitive to the application point: if successful then the LS is re-used but could stop working at some point. By including/linking the LS in/to the genotype it can be exchange and applied to a new point.

Meta-Lamarckian Learning
[Ong and Keane, 2004]

- A pool of operators, e.g. LS operators, are in a list;
- each operator is associated to a score;
- on the basis of the success of each operator, a selection probability is adjusted (similar to the parent selection in GAs) o that promising operators are more likely selected.

Hyper-heuristics
see [Burke et al., 2010]

- A list of operators (e.g. any meta-heuristic, Ls etc.) is coordinated by a machine learning supervisor.
- The supervisor selects the components and on the basis of the results trains for selecting the proper operators
  - many rules are available: random, linear, exponential and many other selections...

Diversity Adaptive Control

e.g. Fast Adaptive MA (FAMA) [Caponio et al., 2007]

- Population diversity: in high diversity conditions the algorithm needs to exploit available genotypes, in low diversity conditions the algorithm needs to detect new genotypes and search directions.
- A measurement of the diversity can be employed for enlarging population size in low diversity condition and shrinking in high diversity condition (analogously for mutation rate)
  - or can be employed for deciding the most proper local searcher for assisting the evolutionary framework.

Fitness diversity

- A measure of the diversity of the solutions can be computationally very expensive (especially in high dimensions);
- an indirect measure of the population diversity can be carried out throughout the fitness values:
  - diverse fitness values mean diverse solutions while unique could potentially mean solutions on a plateau.

Idea:

- If the fitness diversity is low, regardless of the reason, a more intensive exploration is needed, e.g. to jump out from a plateau.
- If the fitness diversity is high, the points are likely to be spread and a an action to focus the search is needed.

FAMA
an implementation example

- MA with dynamic parameter setting i.e. population size and mutation rate.
- Adaptive coordination for avoiding premature convergence and stagnation by means of diversity measurement λ.
- Two local searchers, Hooke Jeeves (steepest descent pivot rule) and Nelder Mead (greedy descent pivot rule):
  - if the diversity is decreasing still not critical, the Nelder-Mead is applied since it is greedy and explorative in order to jump out from the nearest basin of attraction;
  - if the convergence is very near the Hooke-Jeeves is run since it is a LS with steepest descent pivot rule and can then finalise the work in the hopefully found global optimum.

FAMA I

working principle and implementation details

1. Create an initial EA population: $S_{pop} = 200$ and evaluate fitness values;
2. select a diversity measure $\lambda \in \{\xi, \Psi, \nu\}$
   - $\xi = min\left\{1, \left|\frac{f_{best} - f_{avg}}{f_{best}}\right|\right\}$, i.e. how close is f_{avg} to the f_{best}?
     - better handles landscapes with a strong global optimum!
   - $\Psi = 1 - \left|\frac{f_{avg} - f_{best}}{f_{worst} - f_{best}}\right|$ i.e. linear sorting, what position if f_{awg}?
     - better handles flat landscapes!
   - $\nu = min\left\{1, \frac{\sigma_f}{|f_{avg}|}\right\}$, i.e. how sparse are the fitness values?
     - better handles landscapes with multiple strong optima!
3. perform linear ranking selection with selection pressure 1.8

FAMA II
working principle and implementation details

4. recombine solutions via blend crossover [Herrera et al., 1997] and get 200 offspring
5. mutate them ($\mathbf{x_m}[i] = \mathbf{x}[i] + \delta$) with probability $p_m = 0.4(1 - \lambda)$
6. merge parents and offspring
7. if $\lambda < 0.1$ AND g[1] > 8 perform Hooke-Jeeves method on the elite individual to improve upon its fitness value
8. if $0.05 < \lambda < 0.8$ AND $g > 4$ select at random $n + 1$ points from the population and apply the Nelder-Mead simplex method on them
9. update $S_{pop} = S^f_{pop} + S^v_{pop}(1 - \lambda)$ ($S^f_{pop} = 40$ and $S^v_{pop} = 120$ in [Caponio et al., 2007])
10. survivor selection: only the best performing S_{pop} individuals survive (the others are simply discarded).
11. updated λ, and go back to step *3*

[1] Number of generations.

Another interesting approach

MA based on LS chains [Molina et al., 2010]

- Every individual of a GA can be selected to undergo LS;
- an instance of the LS strategy is executed for a given amount of iterations and then frozen:
  - if the same individual is still present in the population and selected for LS, the same instance starts over from where it was stopped.
- Individuals that stay for too long in the populations without improving their fitness value are re-sampled.
- A version using CMA-ES as LS showed extremely good results
  - ...but can be very heavy for handling large scale problems due to the number of CMA-ES instances to be executed.

Memetic Algorithms VS Memetic Computing

- Memetic Algorithm have a fairly clear definition: EAs + Local Search.
- Would an algorithm PSO + local search be Memetic?
- Memetic Computing (MC) is an umbrella name that covers hybrid algorithmic structures regardless of their nature, e.g. single-solution operators, etc.
- Memetic Computing definition becomes "blurry" as it includes all the possible algorithms.

Some suggested readings

some examples of simple MC approaches can be downloaded directly from DORA

- [Iacca et al., 2012]: "Ockham's razor in MC", the paper proposes a simple bottom-up approach for finding optima.
- [Caraffini et al., 2013b]: this is a seriously simple memetic approach with a high performance.
- [Caraffini et al., 2014]: an adaptive modification of [Caraffini et al., 2012b] based on a "separability test" of the problem at hand.
- [Caraffini et al., 2013c]: a DE framework with extra moves along the axes.
- [Iacca et al., 2014]: a more complex example of hybrid algorithm.

Laboratory and participation work:

- For the four usual problems under consideration in $10D$, $50D$, and $100D$, test your creativity and design your own Memetic Computing approach containing at least two algorithms (a safe option could be to draw inspiration from the suggested readings, or simply incorporate a single solution metaheuristic within a population-based framework, but if you feel confident try something less standard!).
- Test the algorithm over the problems and attempt to outperform the results obtained in the previous weeks.
- Write a short report (one paragraph) that justifies the algorithmic choices and its pseudo-code.

References I

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., and Woodward, J. (2010).
Classification of hyper-heuristic approaches.
In *Handbook of Meta-Heuristics*, pages 449–468. Springer.

Caponio, A., Cascella, G. L., Neri, F., Salvatore, N., and Sumner, M. (2007).
A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives.
IEEE Transactions on System Man and Cybernetics-part B, 37(1):28–41.

Caraffini, F. (2014).
Novel Memetic Computing Structures for Continuous Optimisation.
PhD thesis, De Montfort University, Leicetser, United Kingdom.
https://www.dora.dmu.ac.uk/xmlui/handle/2086/10629.

References II

Caraffini, F., Iacca, G., Neri, F., and Mininno, E. (2012a).
The importance of being structured: A comparative study on multi stage memetic approaches.
In *Computational Intelligence (UKCI), 2012 12th UK Workshop on*, pages 1–8.

Caraffini, F., Neri, F., Iacca, G., and Mol, A. (2012b).
Parallel memetic structures.
Information Sciences, 227(0):60–82.

Caraffini, F., Neri, F., Iacca, G., and Mol, A. (2013a).
Parallel memetic structures.
Information Sciences, 227(0):60 – 82.

Caraffini, F., Neri, F., Passow, B. N., and Iacca, G. (2013b).
Re-sampled inheritance search: high performance despite the simplicity.
Soft Computing, 17(12):2235–2256.

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

References III

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

Dawkins, R. (1976).
The Selfish Gene.
Press, Oxford University.

Herrera, F., Lozano, M., and Verdegay, J. L. (1997).
Fuzzy connectives based crossover operators to model genetic algorithms population diversity.
Fuzzy Sets and Systems, 92(1):21–30.

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

References IV

Iacca, G., Neri, F., Mininno, E., Ong, Y.-S., and Lim, M.-H. (2012).
Ockham's razor in memetic computing: three stage optimal memetic exploration.
Information Sciences, 188:17–43.

Krasnogor, N. (2002).
Studies in the Theory and Design Space of Memetic Algorithms.
PhD thesis, University of West England.

Krasnogor, N. and Smith, J. (2000).
A memetic algorithm with self-adaptive local search: TSP as a case study.
In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 987–994, Las Vegas, Nevada, USA. Morgan Kaufmann.

Molina, D., Lozano, M., García-Martínez, C., and Herrera, F. (2010).
Memetic algorithms for continuous optimisation based on local search chains.
Evolutionary Computing, 18(1):27–63.

References V

Moscato, P. and Norman, M. (1989).
A competitive and cooperative approach to complex combinatorial search.
Technical Report 790.

Ong, Y. S. and Keane, A. J. (2004).
Meta-lamarkian learning in memetic algorithms.
IEEE Transactions on Evolutionary Computation, 8(2):99–110.

Powell, M. J. D. (1964).
An efficient method for finding the minimum of a function of several variables without calculating derivatives.
The Computer Journal, 7(2):155–162.

Rinaldi, F. (2012).
A class of derivative-free nonmonotone algorithms for unconstrained optimization.
Seminario Dottorato 2012/13, page 107.

Smith, J. E. (2007).
Coevolving memetic algorithms: A review and progress report.
IEEE Transactions on Systems, Man, and Cybernetics, Part B, 37(1):6–17.

References VI

Solis, F. J. and Wets, R. J.-B. (1981).
Minimization by random search techniques.
Mathematics of Operations Research, 6(1):19–30.

Spall, J. C. (1992).
Multivariate stochastic approximation using a simultaneous perturbation gradient approximation.
IEEE Transactions on Automatic Control, 372:332–341.