Course: Natural Computing

*4. The No-Free-Lunch Theorem

J. Michael Herrmann

School of Informatics, University of Edinburgh

michael.herrmann@ed.ac.uk, +44 131 6 517177

Theory of MHO algorithms: Next steps

1. No free lunch theorem
2. More algorithms (later)
3. Dynamics and convergence (next week)
4. GA & GP: Schema theory (next week)

Optimisation as search

- Search space X, assumed to be discrete
- Space of values of the objective function, e.g. $\mathbb{R}^+$
- Objective function $f : X \to \mathbb{R}^+$, i.e.
  for each $x_i \in X$ we obtain $f_i = f(x_i)$
- Sample $d_m = \{(x_1, f_1), \ldots, (x_m, f_m)\}$ of size m,
  where $x_i \in X$ and $f_i \in \mathbb{R}^+$
- We usually know the search space, but we don't know the objective (fitness) function, except for the few values in the current sample: The fitness function represents the problem!
- How do we find the optimal x?

Optimisation as search

- Sample $d_m = \{(x_1, f_1), \ldots, (x_m, f_m)\}$, $x_i \in X$ and $f_i \in \mathbb{R}^+$
- Optimisation algorithm is a function $A(d_m) = x_{m+1}$, i.e. given the sample, the algorithm decides where to look next.
- Now we get $f_{m+1} = f(x_{m+1})$ and d_m is extended to d_{m+1} etc.
- In many algorithms not all previous sample points are used to determine a new point in the search space, e.g.
  - only the previous generation (N pairs: string + fitness) are used in a GA with population size N.
  - in PSO $N + N + 1$ pairs (x_i, f_i) are used, i.e. the current points, the personal bests, and the global best (which are not necessarily all different) and the respective fitnesses.
  - the pheromone trail in ACO contains implicitly all previous information in this algorithm.

Performance of an optimisation algorithm

- Performance can be measured for a given fitness function without directly referring to the algorithm (for minimisation problems) as
$$\Psi_m = \min_{i \leq m} f_i$$
- Nevertheless the sequence of the fitnesses depends on the algorithm, i.e. we should talk about $\Psi_m(f, A)$ where $A \in \mathcal{A}$, and $\mathcal{A}$is the set of all algorithms
- Likewise, we denote by $\mathcal{F}$ the set of all fitness functions, which is identical to the set of problems
- Try to ignore this item: In principle we should also consider different search spaces $X \in \mathcal{X}$ or sets of constraints $C \in \mathcal{C}$, where $\mathcal{X}$ is the set of all search spaces and $\mathcal{C}$ is the set of all sets of constraints

The no-free-lunch theorem (Wolpert & Macready, 1995)

What is a "free lunch"? An algorithm that is better than any other algorithm for "all the problems". If we try to pinpoint this we get a fundamental result in optimisation, the No Free Lunch Theorem, which shows that, all non-resampling optimisation algorithms perform equally, averaged over all problems, or more formally

No-Free-Lunch Theorem [Wolpert & Macready, 1995] Let $\mathcal{A}$ the set of optimisation algorithms and $\mathcal{F}$ the set of objective functions.

$$\forall A, B \in \mathcal{A} : \sum_{f \in \mathcal{F}} \Psi_m(f, A) = \sum_{f \in \mathcal{F}} \Psi_m(f, B)$$

see: Joyce & Herrmann (2018) A review of No Free Lunch theorems. Springer.

Alternative formulations of the NFL theorem

1. For all possible metrics, no search algorithm is better than another when its performance is averaged over all possible discrete functions.
2. On average, no algorithm is better than random enumeration in locating the global optimum
3. The histogram of values seen, and thus any measure of performance based on it, is independent of the algorithm if all functions are considered equally likely
4. With no prior knowledge about the function $f : X \to Y$, in a situation where any functional form is uniformly admissible, the information provided by the value of the function in some points in the domain will not say anything about the value of the function in other regions of its domain.

Meaning of NFL for algorithm comparison and evaluation

1. No free lunch results preclude meaningful comparison of optimisation algorithms without reference to specific problems.
2. However, almost all restrictions on the set of problem functions result in possible free lunches.

A general way of describing such restrictions is by means of a prior over $\mathcal{F}$, i.e. some problems are more likely to occur than others:

3. Similarly, but more generally, almost all probability distributions over problem functions result in possible free lunches.
4. When free lunches are possible, the algorithms that achieve them are *aligned* with the probability distribution over problem functions.

Block-uniform distributions

Only certain priors over problems lead to NFL (English, 2004)

- If two fitness functions return the same fitness values although possibly in a different order, we say they are connected by a *function permutation.*
- A prior over functions is *block-uniform* if any two functions that are connected by a function permutation have the same prior probability.

5 More specifically, block-uniform distributions capture exactly the scenarios where no free lunch results hold for any metric.

6 However, when we are interested in no free lunch results with respect to particular metrics, and for limited numbers of samples, then free lunches are possible even under block-uniform distributions.

Possible Free lunches

7 When free lunches are possible, their prominence tends to depend crucially on the optimisation metric used.

8 When considering more than just the exploration behaviour of an algorithm, algorithms can be ranked. For example, some optimisers are simpler, some are faster and some tend to resample less than others.

Practical implications

9 Benchmarking alone cannot be used to evaluate an algorithm, but must be used in combination with clear underlying assumptions. The benchmark functions must be representative of the problems and there must be some smoothness, in the sense that being good at a problem means that you are likely to be good at similar problems.

10a We must try to characterise the dynamics of optimisation algorithms, to understand their search behaviour, so that we can better understand which algorithms should be used for which problems.

10b We must try to characterise optimisation problems, to understand their properties, so that we can better understand which algorithms should be used for which problems.

Conclusion on NFL theorems

- "A basic insight of machine learning is that prior knowledge is a necessary requirement for successful learning"

Shai Ben-David et al. Universal learning vs. no free lunch results. In: Philosophy and Machine Learning Workshop NIPS. 2011.

- "you can't do inference ... without making assumptions"

David J.C. MacKay. Information theory, inference and learning algorithms. Cambridge University Press, 2003.

N.B.: It is quite interesting that for continous problems *free lunches* are possible, but this is beyond the scope of this course.

Theoretical Aspects of Metaheuristic Optimisation

Course: Natural Computing (week 4*)

Part 1: Exploration and Exploitation

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

The General Scheme

1. Embrace randomness
2. Use populations of solutions
3. Maintain diversity
4. Select or favour the best individuals
5. Transfer information in the population from the best individuals to others
6. Design fitness function such that it can use building blocks
7. Avoid local minima
8. Store good solutions in memory
9. Tweak the parameters
10. Develop your own variants
11. Use domain knowledge and intuition for the representation of the problem, encoding, initialisation, termination, local heuristics, choice of the algorithm
12. Check what the algorithm is doing

1. Call the user-provided state generator.

2. Print the resulting state.

3. Stop.

Given any two distinct metaheuristics M and N, and almost any goal function f, it is usually possible to write a set of auxiliary procedures that will make M find the optimum much more efficient than N, by many orders of magnitude; or vice versa. In fact, since the auxiliary procedures are usually unrestricted, one can submit the basic step of metaheuristic M as the generator or mutator for N.

Source: en.wikipedia.org/wiki/Metaheuristic (until 15/5/2010)

Why do we need theory in MHO?

- What are the reasons why the algorithms perform better than random search and are effective in practice?
- How do we measure that this is indeed the case? And for what problems?
- Although hybridisations of MHO algorithms are often of practical interest, the simple algorithms are more likely to be accessible to theory.

Why do we need theory in MHO?

1. Identification of essential factors, formulation of relevant concepts (what)
2. Quantitative description of an observable phenomenon (how)
3. Explanation, understanding (why)

Other questions (whither, whence, who, to what end, ...)
are also important, but are usually less accessible to theory

1. What happens in MHO?

Because of the diversity of the "inspirations" of MHO algorithm, a common mechanism is hardly visible. The common theme seems to be to achieve either a balance between to opposed effects or a gradual shift between them:

- Exploration vs. exploitation
- Cooperation vs. competition
- Diversification vs. intensification (Blum and Roli, 2003)
- Global search vs. local descent/ascent
- Randomness vs. greediness (goal-directedness)

These concepts set the scene, but what happens?

Exploration and exploitation in our algorithms

	exploration	exploitation
SA	temperature-based fitness decrease	fitness increase (hill-climbing)
GA, ES	mutation crossover (islands)	selection (elitism)
ACO	probability rule (sampling) $\tau_{\min}$, $\tau_{\max}$	pheromone evaporation, local heuristics (local search)
PSO	inertia (ω) overshooting forces ($\alpha_1 + \alpha_2 > \approx 4$)	forces to bests ($\alpha_1 + \alpha_2 < \approx 4$), "constriction"

How can exploration and exploitation be balanced or controlled?

Exploration and exploitation

"A metaheuristic will be successful on a given optimization problem if it can provide a balance between the exploitation of the accumulated search experience and the exploration of the search space to identify regions with high quality solutions in a problem specific, near optimal way."

T. Stuetzle: *Local Search Algorithms for Combinatorial Problems—Analysis, Algorithms and New Applications.* DISKI. infix, St. Augustin, 1999.

Theoretical Aspects of Metaheuristic Optimisation

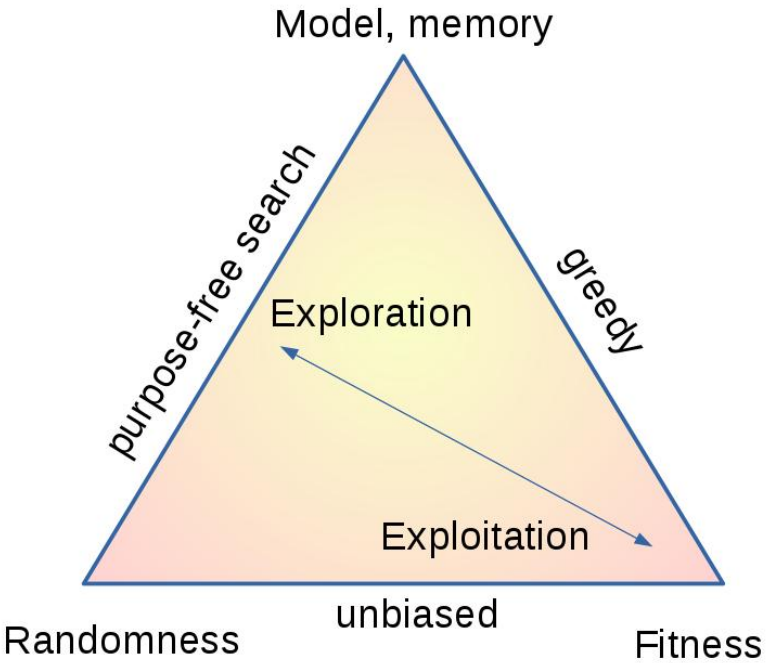
Course: Natural Computing (week 4*)

Part 2: Model-based search

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Intensification and diversification

How does an algorithm guide the solutions in search space?



Beyond the balance between randomness and goal-directedness MHO algorithms are characterised by a specific bias due to, e.g.,

- memory of solutions
- interaction among the solutions
- dependencies in the noise

Fitness is usually a clear criterion (see, however, stochastic problems or multi-objective optimisation), the other two, i.e. randomness and bias, can be complex and will interact in a specific way in actual algorithms.

see C. Blum & A. Roli: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys* **35**:3, 2003, 268–308.

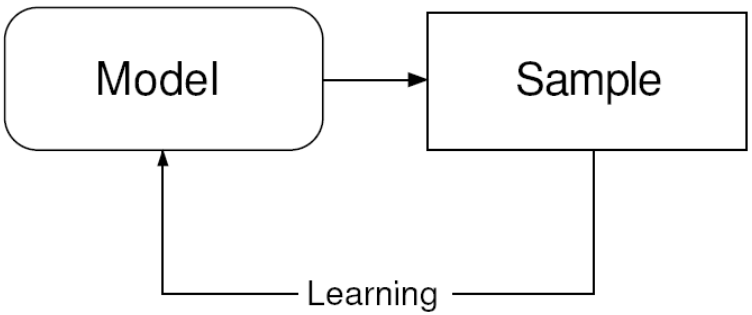
A general framework

Because of the diversity of the "inspirations" of MHO algorithms, a general view is difficult to obtain.
Proposals for a general framework include

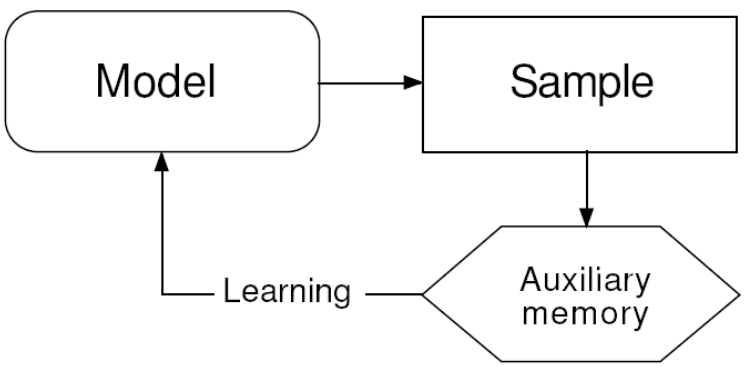
- Model-based search
- Biased random walk
- Bayesian inference
- Dynamical systems

Model-Based Search

Framework for expressing relation between algorithms



Scheme of the MBS approach



MBS approach with memory

E.g. in ACO:

- Model: pheromone matrix
- Sample: ants following pheromone traces
- Learning: pheromone update
- Auxiliary memory: best-so-far solution

Zlochrin, Birattari, Meuleau, Dorigo: Model-based Search for Combinatorial Optimization: A Critical Survey. Annals of Operations Research 2004.

Model Based Search

- Model-based: Candidate solutions are constructed using some parameterised probabilistic model, that is, a parameterised probability distribution over the solution space.
- The model may be merely a theoretical vehicle, namely if the algorithm is instance-based, i.e. improvement based on previous instances.
- The candidate solutions are used to modify the model in a way that is deemed to bias future sampling toward fit solutions.

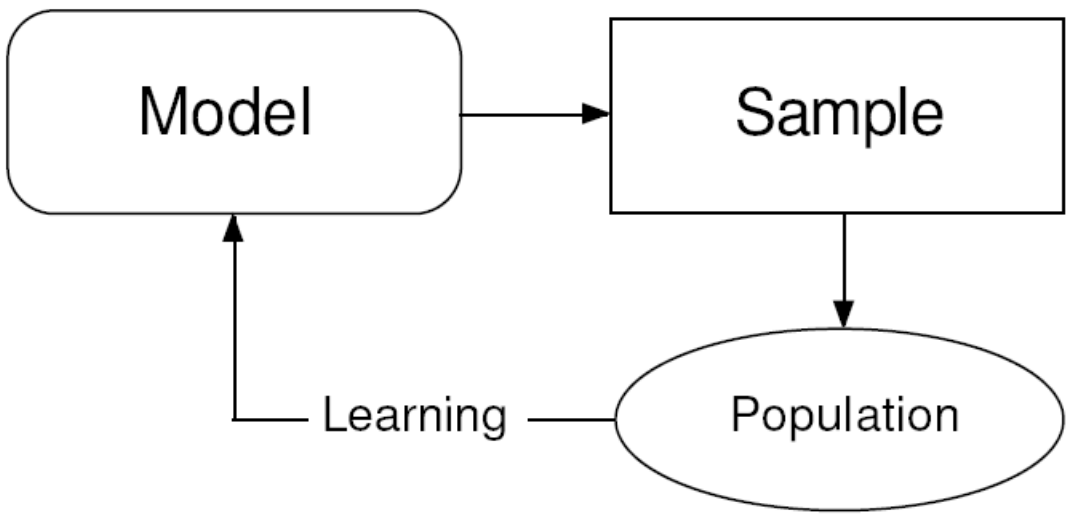
Models enable theoretical predictions.

ACO as MBS

- A finite set $C = c_1, c_2, \ldots, c_n$ of components (n is the number of components; in the TSP c_i is the edge of a graph)
- A finite set X of states of the problem, where a state is a sequence $x = c_i, c_j, \ldots, c_k, \ldots$ over the elements of C. The length of sequence x, i.e., the number of components in the sequence, is expressed by $|x|$. The set of (candidate) solutions S is a subset of X (i.e. $S \subseteq X$).
- A set of feasible states X_f, with $X_f \subseteq X$, defined via a set of constraints Ω
- A non-empty set S^* of optimal solutions, $S^* \subseteq X_f$, $S^* \subseteq S$
- Formulation of the pheromone update
- Result is a fully-connected weighted graph. As many weights will be near zero or near τ_{min} a bias towards good solutions is provided.

GA as MBS

- GA seems to be instance-based, but samples are not drawn independently. Dependencies can be captured by a model:
- Generate new solutions using the current probabilistic model
- Replace (some of) the old solutions by the new ones.
- Modify the model using the new population.



GA as MBS

compact Genetic Algorithm (cGA) (Harik et al., 1999)

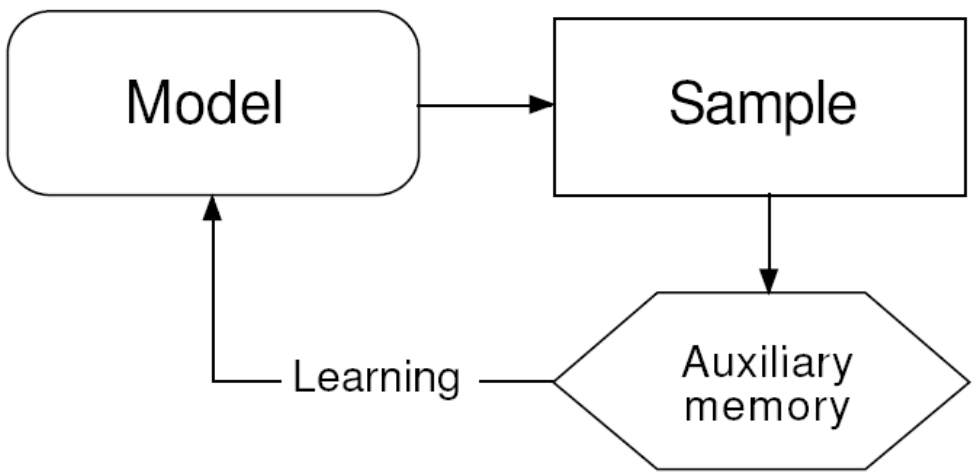
- Probabilistic simulation of a GA with tournament selection
- Probabilistic model of the population: individuals are generated by biased draws based on a probability vector. E.g. if the vector entry p_i is 0.9 it is likely to have a 1 at position i in this individual's string.
- Assume an individual a wins a selection tournament over b

$$\text{if } a_i \neq b_i \qquad \text{then} \quad p_i \leftarrow p_i + \frac{1}{n}(a_i - b_i)$$

  i.e. the model is similar to ACO
- This probabilistic model is different from the original instance-based GA, because if does not capture dependencies between bits in the genome, i.e. the co-occurrences of certain bit combinations (schemas) in the population are ignored

PSO as MBS

- As in GA the "model" is actually a population (which can be represented by a probabilistic model if higher correlations are considered)
- Generate new samples from the individual particles of the previous iteration by random modifications
- Use memory of global (so-far) or personal best (or respective neighbourhood bests) for learning



Conclusions on Model Based Search

- Models may have to be complex in order to replicate the behaviour of the algorithm
- Usually, models capture only certain aspects of the reality.
- What aspects should we consider in order to make statements about
  - convergence,
  - complexity,
  - parameter values?

On exploration in nature

It is thanks to these eccentrics, whose behaviour is not conform to the one of the other bees, that all fruits sources around the colony are so quickly found.

Karl von Frisch, 1927

Maurice Clerc: PSO Mini Tutorial on Particle Swarm Optimisation (2004)