NAT Tutorial 6: Particle Swarm Optimization

1. Recall the main algorithms that we were dealing with (i.e. GA, ES, GP, ACO, PSO and possibly variants of these, if this makes a difference) and classify them according to Dorigo's the criteria for the classification of solvers of combinatorial optimisation problems:
   a. Is the solution obtained by direct construction or by the use of local search? What local search is used? (this was not asked for, but it's still interesting)
   b. Are population of solutions used or not?
   c. Is a memory used within the search process or not?
   d. Is the evaluation function fixed or is it modified during search?
   e. Several neighbourhoods or only a single one?
   f. Inspired by nature or artificial?

   You could represent the answer to this question as a table containing check-marks.

   Answer: Not all variants covered here, further combinations exist or are possible.

Criterion/algorithm	GA	ES	GP	ACO	PSO
Solution	Assumed to be constructed from building blocks	Direct construction	Direct construction or by using autonoumously defined subroutines	Probabilistic compostion from partial solutions from several ants	Direct construction
Local Search	In order to ensure admissibility; hill-climbing	Possible (hill-climbing)	Editing for syntactical correctness or based on a grammar; hill-climbing to fix numerical constants	In order to ensure admissibility	often not used
Population	Large population	Usually population of subpopulations	Large population	Small number of ants, but solution from a single pheromone matrix	Small number of swarm members
Memory	Only when elitism is used	Mutabilities	Only when elitism is used	Taboo lists	Personal best and generation best
Evaluation	Fitness function	Fitness function (objective function)	Fitness cases (with cross-validation) +bloat control	Tour length (or other fitness function) used for pheromone update	Fitness function (objective function)
Neighbourhood	Restricted in the island variant; manu neighbourhoods in the multi-objective variant	Neighbourhood structure is learned by correlations of mutabilities	Usually all-to-all competition; islands possible	Ants do not interact directly, but exchange information via the common environment (stigmergy)	Often on graphs, i.e. neighbourhood chosen in dependence on problem
Inspired by nature	Natural evolution	Directed search, not too similar to natural evolution	Some analogy to epigenetic processes	In some respect similar to real ants	Inspired by swarm intelligence in flocks and schools

2. How can optimisation problems be solved if the search space is continuous and the fitness function is analytically known? (This relates to our topic only indirectly.)

   Answer: see http://en.wikipedia.org/wiki/Mathematical_optimization. It is interesting to consider metaheuristic optimisation algorithms in this context (see in particular section 6 of this Wikipedia article). Such comparisons can help to evaluate the efficiency and scaling properties of metaheuristic algorithms. Familiarity with mathematical optimisation techniques is highly recommended but not a learning goal of this course.

3. Suppose you want to evaluate the usefulness of metaheuristic algorithms
   a. in comparison to the canonical version or a version which has been proposed by a competitor,
   b. in comparison to standard or problem-specific direct algorithms for a specific application,
   c. in comparison to other types of metaheuristic algorithms for a specific application,
   d. same as c) but in more general sense.

   How would you design the experiments in each of these tasks? How would you present the results?

   Answer: a. The competitors has probably used several benchmark problems to show that their algorithm works well. You will need to show that your algorithm works (in most cases) better on the same problems. In addition, you may like to test your algorithm also on different set of problems in order to show (it that's possible) that your algorithm has a wider range of applicability. It should be avoided to test the algorithm only on a subset of the relevant problems, because by choosing only "suitable" problems it is often easy to achieve a apparent improvement.
   In general, also other advantages might justify to propose a new algorithm: Lower computational complexity, similarity to a biological example, better convergence properties, improved robustness etc. even if some of these "advantages" are detrimental to the performance, although one should strongly prefer algorithm that are better in the Pareto sense.
   Comparisons to a canonical version are often not fair, because canonical version are usually not proposed for reasons of performance. Therefore a comparison to a canonical version rather help to explain the proposed variant or can be presented as a further development of the canonical algorithm.
   b. This may be hard, but not hopeless because direct algorithms (see previous question) often rely on some assumptions that may not be needed for the metaheuristic algorithm. The success of the metaheuristic approach will depend on the accuracy of the model that was used in the specific case. Here, even more than in a., it will be important that your solution is using exactly the same problem as the existing solution.
   c. Same as b., but here it would be of interest what modification of the algorithm lead to the superior performance: Is it merely a more clever choice of the parameters? A clever modification of the operators? A more suitable local search method? In addition one should face the question whether merely a record was broken or whether a better understanding of either the algorithm or the problem was achieved.

d. In a more general sense, one should (in addition to the points made so far) consider the set of problems that are selected for the comparison. This includes questions such as: Can the set of problems where are particular algorithm is successful be characterised in some way? Is the performance on typical benchmark problems always acceptable or are there some bad fails? Are the problems sufficiently complex to predict a success of the algorithm in real-world tasks? How does the algorithm scale with problem complexity?

4. Compare the representations of GA and PSO as model-based search/optimisation.

   Answer: see Lecture.

5. No answer provided

6. No answer provided