Practical Assignments Natural Computing

Due December 18th, 2009

General

There are three assignments from which you have to pick two. Each assignment involves designing and implementing an algorithm from Natural Computing. In the first assignment you will be asked to implement a Simulated Annealing algorithm for the binary low-autocorrelation problem. In the second assignment you will be asked to implement a Particle Swarm Optimization algorithm to maximize the Second Harmonic Generation function. In the third assignment you are asked to implement an Ant Colony Optimization algorithm for three TSP problems.

For each assignment you are required to write a brief report ($\pm$4 pages) that describes your algorithm, your results, and your findings. Guidelines for the report are provided with the assignments.

Creativity will be rewarded. If you want to apply a different Natural Computing algorithm for one or for both problems, or if you can think of a revolutionary new computing paradigm inspired from nature, then we challenge you to do so. However, consult with us first, and make sure that you compare the performance of your algorithm to the standard algorithms.

Mail your solutions before 6:00 pm on December 18th, 2009 (strict deadline!) to jkruisse@liacs.nl.

Your submission should contain:

- Your name(s) and student number(s)
- For each assignment a brief technical report (PDF-format)
- Working code of your all algorithms (mind that the code should be easy to read)

Assignment 1

The low-autocorrelation problem

The so-called *low-autocorrelation problem of binary sequences*, is subject to actual research and is of big interest for industrial applications, e.g. communications and electrical engineering. Its description follows.

Feasible Solutions: Binary Sequences $\vec{y} \in \{-1, +1\}^n$

Objective Function:

$$f(\vec{y}) = \frac{n^2}{2 \cdot E(\vec{y})} \longrightarrow maximization \tag{1}$$

with:

$$E(\vec{y}) = \sum_{k=1}^{n-1} \left(\sum_{i=1}^{n-k} y_i \cdot y_{i+k} \right)^2 \tag{2}$$

The table below provides an overview of the fitness values of the best known solutions that have been found so far for various dimensions.

n	Best Known f
20	7.6923
50	8.1699
100	8.6505
200	7.4738
222	7.0426

The assignment

Design and implement a Simulated Annealing algorithm for tackling the low-autocorrelation problem. Also implement either a Monte-Carlo search algorithm or a Genetic Algorithm for tackling the low-autocorrelation problem. Write a report about your findings containing the following:

- Describe your implementations; give pseudocodes and short descriptions from which one should be able to reconstruct the algorithms. Do your implementations differ from the standard algorithm? If so, why is that?
- Run your algorithms on $n = 10, 20, 50, 100$, and 200. Report the best solution you found (plus its fitness) for each bitstring length. Did you find the global optimum?
- Compare the algorithms. What is the performance of your implementations? If your implementations differs from the standard algorithm, do your implementations work better than the standard algorithms? (hint: give plots to compare fitness versus number of iterations)

For your convenience, MATLAB code for the fitness function is provided at:

`http://natcomp.liacs.nl/NC/pa/autocorrelation.m`

Assignment 2

Laser pulse shaping

Today in the atomic and molecular physics community, it is possible to investigate and control the motion of atoms or molecules by irradiating them with shaped pulses. One of such control applications is the alignment of an ensemble of molecules which finds its use in molecular physics because it can simplify the interpretation of experimental data to a great extend. However, in order to get the desired control on an ensemble of molecules one has to find appropriate laser pulse shapes that can accomplish this. This is an optimization problem.

A simple test case for the alignment problem is the *The Second Harmonic Generation* which turns out to be a good test case in the Physics lab, and its investigation contributes to the understanding of the alignment problem. In this assignment we will consider this test case.

The Second Harmonic Generation

A laser pulse going through a crystal produces light at the octave of its frequency spectrum. The total energy of the radiated light is proportional to the integrated squared intensity of the primary pulse. Explicitly, the time-dependent profile of the laser field in our simulations is given by:

$$E(t) = \int_{-\infty}^{\infty} A(\omega) \exp(i\phi(\omega)) \exp(i\omega t)\ d\omega, \tag{3}$$

where $A(\omega)$ is a Gaussian window function describing the contribution of different frequencies to the pulse and $\phi(\omega)$, the *phase function*, equips these frequencies with different complex phases.

The objective (fitness) function is given by

$$SHG = \int_0^T E^4(t)dt \longrightarrow maximization \tag{4}$$

By changing $\phi(\omega)$, the temporal structure of $E(t)$ can be altered. In a real life pulse shaping experiment, $A(\omega)$ is fixed and $\phi(\omega)$ is used to control the shape of the pulses. To this end, we interpolate $\phi(\omega)$ at n frequencies ω_n; the n values $\phi(\omega_n)$ are our decision parameters to be optimized. In order to achieve a good trade-off between high resolution and optimization efficiency, the value of $n = 80$ turned to be a good compromise.

The assignment

Your task in this assignment is to design and implement a Particle Swarm Optimization algorithm to optimize (maximization) the SHG using the phase function. I.e. find a vector $\vec{x} = \{\phi(\omega_1), \ldots \phi(\omega_n)\}$ with $0 \leq \phi(\omega_i) \leq 2\pi$ that maximizes the SHG function of equation (4). Write a report about your findings containing the following:

- Describe your implementation; give pseudocode and short description from which one should be able to reconstruct the algorithm. Does it differ from the standard algorithm? If so, why is that?
- Run your algorithm a few times for $n = 80$. How well does your algorithm perform? If your implementation differs from the standard algorithm, does your implementation work better? (hint: give plots to compare fitness versus number of iterations)
- What was the best solution that you have found and what was its fitness (visualize it, it might have a strikingly simple shape)?
- How did you tune the parameters? What happens if you change the parameter settings?

For your convenience, MATLAB code for the fitness function is provided at:

`http://natcomp.liacs.nl/NC/pa/SHG.m`

Assignment 3

The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is probably the most intensively studied combinatorial optimization problem. In this assignment we consider two dimensional maps of locations and we are looking for the the least-cost round-trip route that visits each city exactly once and then returns to the starting city. In these two dimensional maps, the distance between two locations is defined by the Euclidean distance.

The assignment

The TSP datasets that we will use for this assignment are city maps of Qatar (194 Cities), Luxembourg (980 Cities) and Italy (16,862 Cities). These datasets are taken from:

`http://www.tsp.gatech.edu/world/countries.html`

Design and implement an Ant Colony Optimization algorithm to solve these TSP problems:

- Describe your implementation; give pseudocode and short description from which one should be able to reconstruct the algorithm. Does it differ from the standard algorithm? If so, why is that?
- Run your algorithm on the dataset of Qatar. Visualize the solutions that you have found. Do they look reasonable?
- What is the performance of your implementation? If your implementation differs from the standard algorithm, does your implementation work better? (hint: give plots to compare fitness versus number of iterations)
- Also run your algorithm on the Luxembourg and Italy datasets. How well does your algorithm perform when the complexity increases? How do your solutions look on these datasets?

Additional information

You can represent candidate solutions as permutations π of the node indices $\{1, 2, \ldots, n\}$, and you can compute the fitness of a candidate solution π as:

$$f(\pi) = \left(\sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} \right) + d_{\pi(n)\pi(1)} \tag{5}$$

Here, $d_{\pi(i)\pi(j)}$ denotes the distance between the nodes i and j.

Generally, in all ACO algorithms, the heuristic desirability η_{ij} is chosen as $\eta_{ij} = 1/d_{ij}$, that is, the heuristic desirability of going from city i directly to city j is inversely proportional to the distance between the two cities.

In Ant Systems algorithms, the initialization of the pheromones is done by

setting $\tau_{ij} = \tau_0 = m/C^{nn}$, where m is the number of ants, and C^{nn} is the length of a tour generated by the nearest-neighbor heuristic (hence, it is the length of a reasonable tour).

For your convenience, MATLAB functions that load the three TSP datasets into a $2 \times N$ matrix can be found at:

`http://natcomp.liacs.nl/NC/pa/qa194.m`
`http://natcomp.liacs.nl/NC/pa/lu980.m`
`http://natcomp.liacs.nl/NC/pa/it16862.m`