Nature-Inspired Computing (NIC) Methods in Wind Generator Design Intermediate Report

Janne Kemppainen & Eero Järviluoma

Aalto University School of Electrical Engineering

Abstract

This document is the intermediate report for the project Nature-Inspired Computing (NIC) Methods in Wind Generator Design for the course AS-0.3200 Automaatio- ja systeemitekniikan projektityöt. NIC methods include various different algorithms for efficiently finding near-optimal solutions for many optimizing problems. Here we summarize what we have achieved thus far and what are the main problems.

I. Overall Thoughts

After the first meeting with the instructor we had a pretty clear vision of the project work. The work was divided between two different algorithms, namely Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), and the difficulty was tunable to fit the work requirements. Originally we planned to implement the Differential Evolution algorithm, but we switched to PSO.

During the first few weeks we were preoccupied with the project work for the course System Dynamics and had no time to concentrate on this project. After finishing the other project there was more time available and we started looking deeper into this project.

Then there appeared some more obstacles as we both had decided to apply for exchange and we had to find information, plan studies, fill application forms etc. In addition to this, Eero had to allocate his time to apply for a summerjob as the application deadlines were closing in. Therefore we had to delay our initial deadlines.

II. The Optimization Problem

Tahan selitysta meidan optimointiongelmasta.========== ======== ======== ======== ======= = ==== ===== ======= ======= ======== ======== =

III. Particle Swarm Optimization

Implementing the Particle Swarm Optimization algorithm was assigned to Janne. The algorithm was first proposed by James Kennedy and Russel Eberhart in 1995 and the aim is to implement their version of the algorithm with only some small modifications.

The algorithm itself is quite simple. The solution space is first initialized with particles set to random locations with zero velocity. Then the fitness function is evaluated for each particle and the best coordinates are stored. Then the velocities are calculated by the following formula: [1]

```
vx[][]=vx[][]+
 2*rand()*(pbestx[][]-presentx[][])+
 2*rand()*(pbestx[][gbest]-presentx[][])
```

The positions and velocities are updated until a set criterion is fulfilled. The velocity function has a constant 2 for the stochastic part so that the particles would have the possibility to move over the previous best value. This value yields the best results as there is no good way to individually guess which one of the

velocity increments should be weighted more. [1]

Our optimization problem is a hybrid of discrete and continuous parameters whereas the original algorithm was developed for continuous case only. Therefore we have to make some special arrangements for the two discrete parameters. Pole pair number is an integer between 20 and 80. For this parameter we can use an approximation of the continuous case and always round the value to the nearest integer. However, the number of slots per pole per phase can only be 1, 2 or 3. Therefore we have to run the optimization three times with each value to find the optimal parameters.

The implementation began with defining the required variables for the swarm and the limitations. The easy part was to fill the solution space with random particles. After implementing the basic algorithm some problems emerged.

The first disturbing observation was that the computer resources weren't used to their full potential as the calculations were performed mostly on one core at a time. After some research the Matlab Parallel Computing Toolbox, and especially the parfor loop, was found. This allowed for the parallelization of the generator simulation and somewhat improved the performance. Not everything is parallelizable though as the particles have to be synchronized.

The next problem was how to enforce the modeling limitations. The initial attempt was to enforce the parameters to stay inside the defined problem space but this is against the nature of the algorithm. As a result the particles would often hit the hard limits and gather near the edges. The decision was made to check the boundaries but skip the calculations and assign a bad fitness value if any of the parameters were out of bounds.

After the change in handling the boundaries it was noticed that the swarm diverges for some reason. In spite of penalizing out of bounds values some particles seemed to wander far from the desired area. This behaviour needs to be looked into.

IV. Structure

The work is divided into 4 sections: Project planning, pre-studying of the algorithms, implementation of the algorithms and finally documentation. All phases need to be completed for both algorithms we chose. The work packages will be completed in the said order, starting from project planning and concluding with documentation. The goals for each work packet are as follows:

1. Project planning: Discussion of the algorithms, workload and other factors with the instructor. When the general structure of the project is clear, written project plan is submitted.

2. Pre-study of the algorithms: Getting familiar with the chosen algorithms. While the exact sources for information are not specified in this report, special care should be taken to only use reliable and trustworthy sources.

3. Implementation of the algorithms: Algorithms are implemented with Matlab software. To make the workload of 4cr realistic, we refrain from using the existing matlab functions for either of the algorithms (such as GA function from global optimization toolbox). This work package also includes the use of the given simulation model for the wind turbine. The feasibility of the results is also assessed to detect any possible errors in the algorithms.

4. Documentation: Finally, the work is documented according to the outline in the AS-0.3200 wiki page. Among other required things, the document will consist of basic theory behind the used algorithms, discussion of the implementation and the possible performance and result differences between the algorithms.

V. Time and Teamwork Management

While both participants will take part in all 4 work packages, Janne will mostly focus on planning, implementing and documentation of the DE algorithm, while Eero will focus on the GA. We will still cross-check each others'

work to avoid and detect errors. We decided to set quite ambitious deadlines for each work packet in order to avoid interference from other courses that start in the 4. period. Progress is monitored with a table, which lists both realized and planned use of time. The planned deadlines and time consumption estimates for each work packet are as follows:

1. Project planning: Deadline for the written report is 28.1.2014. Planned workload is 15 h for each participant.

2. & 3. Pre-study and implementation: Deadline for the initial algorithms is 28.2.2014. By this deadline, we should have optimization algorithms ready for single-objective case. If results are not feasible, or we want to go further into the multiobjective case, these corrections should be made in the following month, and be ready by 23.3.2014. Planned workload for these two packages combined is 70h for each participant.

4. Documentation: Deadline for project documentation is set at 30.3.2014. Planned workload is 15h for each participant.

VI. Risk management

Other courses may prove to be too time-consuming leaving less time for project work. For example it is difficult to estimate the time required for the project work of the course System Dynamics.

Sudden illness may affect the capability to do work. This might mean that a part of the workload falls to the other person.

Multiobjective optimization might be too difficult to implement with our knowledge and experience or it could increase the workload too much. We still have the option to opt for one objective optimization.

The realised workload might be different for different algorithms (difficulty of implementation). To even out the workload the other one can give some support if possible and maybe focus more on the documenting side of the project.

Other responsibilities can affect the time available for working. It is important to get a good head start with the project and start doing it right away piece by piece.

There is probably going to be a need for some code refactoring as we gain more insight and see what are the better ways of implementing different things. This might cause surprising amounts of re-work.

The implementation of either one of the algorithms can be wrong. This should be apparent if the results were to be somehow odd. However, it is possible that the wrong implementation yields results that we fail to see as faulty. It is also possible that due to the nature of these algorithms the results are not optimal.

File corruption or accidental deleting of files might cause irreversible loss of information. Thus we are going to use the Git platform to manage our code and documentation files.

If any of the risks realize, and we have problems reaching the deadlines specified in the section 3, we have the option of delaying our deadlines closer to the deadlines specified by the course staff. However, for the convenience of every party involved, we try to stick to our planned deadlines as much as possible, and delaying these deadlines should only be used as a last resort.

References

[1] Kennedy, J. and Eberhart, R. *Particle Swarm Optimization*, Proceedings of IEEE international conference on neural networks. Vol. 4. No. 2. 1995.