C514: Parallel Dynamic Scheduling in a Cluster of Computers

We would like to thank the referees for their useful and constructive suggestions. We have included some paragraphs and modified parts of the paper as indicated in the following:

Referee 1 ---
I can suggest the following changes

(1) *The authors discuss the water supply problem in one particular Spanish city: How does this problem scale to other cities -- are there big cities which are hundreds times bigger and so more computationally demanding. This could motivate problem*

We have implemented simulations with different numbers of tanks and we provide a figure (Figure 14) with the behaviour in the computing time versus the number of tanks. We have also included the following paragraph in Section 5:

As the number of tanks increases in the case of cities bigger than the one considered in these experiments, the number of bits that codify each individual in the population grows linearly with the number of tanks. Thus, the search space grows, and the time required by the genetic algorithm to find the solutions corresponding to the feasible points increases. This increase in the computing time comes from the increment in the time required to process each individual in the population, the possible increase in the population size, and in the number of generations required to find a solution. With respect to the population size, there exists some theoretical analysis of the optimal size [32,33], and algorithms that adjust this size according to the probability of selection error [34]. Moreover, some studies have attempted to find the adequate population size empirically and have recommended the use of populations with size ranges between some proposed values [35,36]. Thus, in order to get an insight into the effect of increasing the complexity of the water supply network, we fixed a population size across the different number of tanks (we selected a population that enabled us to find feasible solutions for different numbers of tanks with an error rate below 0.5%), and we performed some experiments that indicate the effect of the increase in the complexity of each individual of the population and the possible increase in the number of generations to obtain a feasible solution. Figure 14 represents the increase in the time required to execute a fixed number of iterations when the number of tanks grows (line *iter=175*). As can be seen in the Figure the time grows less than with a power of two, and slightly more than as a power of 1.9. Figure 14 shows the number of iterations executed to obtain a feasible solution with a 0.5% of error (line *err<0.5%*). In this case, the increase in the time required by the algorithm grows slowly. Thus, a reasonable increase in the number of tanks to be scheduled does not have an important effect on the practical applicability of our scheduling procedure.

(2) *The English generally reads well but there are occasional glitches. For instance obtention page 3, llow and nexte page 4, most part page 5. Please run a Spell check!*

Spelling mistakes have been corrected.

***(3)** I think section 5 could be strengthened. Limits of scaling should be discussed. The figure 13 should show speed up and not time*

We have included in Section 5 the paragraph corresponding to suggestion (Referee 1,1) and also the following paragraphs:

As can be seen, the prediction procedure has a high degree of accuracy. Moreover, as the prediction module determines the parameters of the prediction model concurrently and asynchronously with the processing of the trajectories by the neuro-dynamic module, an on-line improvement of the prediction accuracy is possible.

The parallelization of the procedure implemented by the Neuro-Dynamic Programming module is based on two of the alternatives to take advantage of the parallel processing in genetic algorithms that are shown in Figure 8 (alternatives 1 and 2). More specifically, a parallel genetic algorithm is run on the P processors several times, in order to obtain a set of K feasible points where the trajectories start (alternative 2 in Figure 8). Once the beginning of K trajectories (the maximum number of trajectories to be processed) are obtained, the rest of each trajectory is determined by only one of the processors (alternative 1 in Figure 8). The distribution of the trajectories among the processors is done dynamically by allocating a trajectory to each processor when it finishes a previously allocated one.

To give an idea of the computing times and the speedups obtained by the parallel implementation of the procedure, Table 1 and Figure 13 shows the results for different number of trajectories and processors. We have obtained efficiencies between 0.7 and 0.9. These results correspond to the mean values obtained from five executions of each case. As Figure 13 shows, as the number of trajectories grows, the speedup obtained for the higher number of used processors is improved. This is a consequence of the way the parallel algorithm works: once the beginning of the simulated trajectories are determined by the parallel genetic algorithm UPGA (alternative 2 in Figure 8), the rest of each trajectory is determined by several sequential genetic algorithms that are executed practically without communication (USGA). Thus, as the number of trajectories grows, the rate of communication time (required basically in the UPGA) decreases with respect to the computing time required to build the whole trajectory.

The times presented in Table 1 can also be improved by implementing a new genetic algorithm able to converge to several optima in only one run, by including niching methods to maintain the diversity of population across the processors and in each processor (alternative 3 in Figure 8).

Referee 2 ---

This is an interesting paper that should be published if the authors can make a few changes.

(1) *Most importantly the title must be changed. It currently suggests that the paper is about scheduling jobs on a cluster -- an area where there is much work. Rather title should indicate that the authors are parallelizing' a scheduling application (from the Water Supply network field)*

We have changed the title of the paper in order to indicate that we have considered a parallel application dealing with water supply network scheduling.

(2) *It is well known that genetic algorithms parallelize well and a lot of other optimization approaches do not. What is not well known is what problems are suitable for genetic methods. Some comments on this issue would be good.*

We have included the following paragraph in Section 3:

Genetic algorithms can be considered as meta-heuristics or general purpose optimization algorithms that require very little knowledge concerning the problem at hand. In [31] the relation was explored between the effectiveness of a general purpose algorithm and the optimization problem to which it is applied, and some NFL (No Free Lunch) theorems are proved indicating that for any algorithm, an improved performance over one class of problems is offset by performance deterioration with respect to another class. Nevertheless, it is possible to improve the behaviour of a general purpose procedure in a problem where limitations are presented by modifications of the algorithm that include some problem-specific operators to accelerate the search with respect to a pure genetic algorithm. In the case of the water supply application presented here, the genetic algorithms perform relatively well. Nevertheless, as said above, we have used a real genetic algorithm based on that described in [10], which includes iterations of a gradient-descendent optimization algorithm to accelerate the convergence.

We have also included some more references and provided more details concerning Figure 13.