On the application of the Cascade Optimization Algorithm in distributed computer networks and grids

Du Du\textsuperscript{a}, Franjo Cecelja\textsuperscript{a}, Antonis Kokossis\textsuperscript{b}
\textsuperscript{a}PRISE, FEPS, Unverisity of Surrey, Guildford, Surrey, GU2 7UB, U.K., f.cecelja@surrey.ac.uk
\textsuperscript{b}School of Engineering, National Technical University of Athens, Zofrafou Campus, 9, GR-15780, Athens, Greece, akokossis@mail.ntua.gr

Abstract
Cascade Optimisation Algorithm (COPT) has been implemented on computer grid. The series of experiments, especially ones on reactor network synthesis for biocatalytic application, demonstrated better suitability of COPT for parallel execution than conventional stochastic algorithms, in particular TABU Search model, with execution time used as a measure of performance.

Keywords: stochastic optimization, parallel execution, computer grid.

1. Introduction
The stochastic optimization algorithms have found many applications especially in complex engineering problems. All of them are based on random probabilistic-driven search, which guarantees full exploration of search space and hence avoids local optimal solutions, combined with various strategies directing search in the most promising direction, hence increases search efficiency. Based on these search strategies there are numerous algorithms developed with perhaps the best known being Simulated Annealing, Tabu Search, Genetic Algorithms and Ant Colony optimization algorithms. The characteristics of these algorithms have been reviewed by many authors [Fouskakis et al. 2002, Mourcoulaki at al. 1999] and all of them agree that the common characteristic is that randomness introduces inefficiencies in search requiring long computational time to converge. Many attempts were made to address this inefficiency either by improving the search strategy of individual algorithms or by parallelizing or distributing the search effort. One such an attempt to parallelize Tabu Search and Simulated Annealing stochastic algorithms is presented later in this paper. Parallelizing the execution of these algorithms is particularly attractive in the light of increasingly available computational resources in the form of grids or clouds. However, most of these attempts follow the natural suitability of optimization algorithms to be implemented in so called synchronous form where synchronization is ‘fixed’ to completion of natural tasks and hence the execution is dictated by the slowest worker (computing unit). In consequence, parallelizing stochastic algorithms has offered limited improvement in reducing computational time.

We have, however, developed new stochastic algorithm, namely the cascade optimization algorithm (COPT) (Kokossis at al. 2099) which enables the asynchronous operation along with the possibility to implement the knowledge base support. The algorithm benefits from the Simulated Annealing concept of Markov process expanded by introduction of partitions and pools to store intermediate solutions and objective
values. The redistribution of population within partitions and pools is redistributed periodically according to predefined criteria and which depends on the progress of optimization and ‘quality’ of solutions. The use of partitions and pools inherently enables execution of Markov chain simultaneously without the need for synchronization after each section; execution is independent and redistribution of solutions and objective values in partitions and pools assures that all solutions are accounted for in searching for most promising search direction/region. This particular feature of COPT enables full scale parallelism and concomitant asynchronous execution.

In this paper we present a brief synopsis of COPT and its implementation on computer grid. The performance has been verified by using benchmark case of reactor network synthesis for biocatalytic application (Giorno at al. 2000) and compared with performance of Simulated Annealing and Tabu Search algorithms.

2. Theoretical analysis

2.1. Cascade Optimization Algorithm

Cascade optimization algorithm (COPT) has been described in details by Kokossis at al (Kokossis at al. 2011). It is suffice to say here, however, that it uses the concept of Markov process adopted from Simulated Annealing algorithm together with preserving/storing solutions and objective values organized in partitions (Kokossis at al. 2011). It is suffice to say here, however, that it uses the concept of Cascade optimization algorithm (COPT) has been described in details by Kokossis at al.

Theoretical analysis

2.1. Cascade Optimization Algorithm

Cascade optimization algorithm (COPT) has been described in details by Kokossis at al (Kokossis at al. 2011). It is suffice to say here, however, that it uses the concept of Markov process adopted from Simulated Annealing algorithm together with preserving/storing solutions and objective values organized in partitions $G_j$, $j \in \{1, 2, \cdots, w\}$, and pools $P_j$, $j \in \{1, 2, \cdots, w\}$, respectively, as shown in Figure 1. Each pool is assigned a global parameter, the temperature, with the highest temperature $T_1$ of the pool $P_1$ and the lowest temperature $T_w$ of the pool $P_w$. Redistribution of solutions is based on the quality of their objective values and is performed every time when new solution from worker $W_j$ arrives. By specifying the higher objective value $y_j^h$ and lower objective value $y_j^l$ of every pool $j$, the new solution $y^s$ is placed to satisfy $y_j^l \leq y^s \leq y_j^h$. The boundaries of every pool are determined by the range of objective values $y_{\min}$ and $y_{\max}$ of currently available solutions and temperatures between $T_1$ (highest) and $T_w$ (lowest) as

$$y_j^h = \frac{(y_{\max} - y_{\min})(T_j - T_w)}{T_1 - T_w} + y_{\max}, \quad y_j^l = \frac{(y_{\max} - y_{\min})(T_j + 1 - T_1)}{T_1 - T_w} + y_{\max}$$  \hspace{1cm} (1)

Every worker $W_j$ executes one section of Markov chain independently of other workers. The initial solutions $S^0$ for Markov chains executed on worker $W_k$ are randomly selected from randomly selected pool $P_j$ with temperature $T_j$. New generated solutions $S_k^s$ are accepted (or rejected) for redistributions in partitions and pools according to its objective value $y^k$ and Metropolis acceptance criterion

$$\exp \left( \frac{y^k - y^s}{T_j} \right) \geq R$$  \hspace{1cm} (2)

where $y^s$ is the current best solution in the pools and $R$ is the random number between 0 and 1. The whole process terminates when the number of solutions $N_w$ in the lowest partition becomes greater than predefined value $\mu$ (= 20 selected in our case) in a number of
It has been noted here that the highest temperature $T_1$ should be and normally is selected such to ensure random attempts to move uphill from local optimum, while the lowest temperature $T_w$ should have value close to the freezing point ensuring high quality solution. The cooling schedule (temperature $T_j$ of individual pools) is then determined by $T_j(j) = a \times j^\theta + b$, where $a$, $b$ and $\theta$ are real numbers defining the shape of the cooling schedule as explained in (Kokossis et al., 2011). In our work we have used $\theta = 1$ which ensures a linear distribution of the temperature between pools.

2.2. Tabu Search Algorithm on Computer Grid

The operation of Tabu Search (TS) algorithm has been described elsewhere. To implement the TS algorithm on the computer grid, the whole Markov chain is split into sections called slots (Figure 2). From the initial randomly selected solution $S_1$, the best solution $S_{EN}$ in the neighborhood at each iteration is selected as initial solution for the next iteration of the slot, all repeated for the fixed number of iterations, 5 in our case. The neighborhood size $N(S)$ is kept fixed and we have experimented with $N(S) = 6$ and $N(S) = 7$.

After a slot is completed, the current best solution $S^*$ is selected as the initial solution for the next slot. The optimization process continues until no new better solutions are generated to replace $S^*$ in several consecutive slots or all $n$ slots in a Markov chain are completed. Note that a Markov chain is a sequential process, which is due to the data dependencies between the previous slot and the current slot. Each Markov chain is distributed to a worker and new session can start after all workers have completed their tasks so that new initial solutions can be redistributed from a complete set of solutions.

3. Numerical Examples

The performance of COPT on computer grid has been extensively studied on several test problems. In this paper we use the reactor network synthesis for well surveyed byocatalitic reaction as benchmark problem with results of COPT compared to those of TS.

3.1. Experimental set-up

The reactor network in its superstructure form (Achenie et al., 1990) is shown in Figure 3. Each of reactors $r_1$, $r_2$, $r_3$ and $r_4$ is modeled as an ideal continuous stirred tank reactor (CSTR) or a plug flow reactor (PFR). In the
reactive phase these reactors are physically connected in every possible combination. The key parameters of a reactor network to be considered in the process of synthesis include the number of active reactors, reactor types, reactor sequence, single/multiple feeds, reactor volumes, and volumes passed through bypasses and recycles. The aim of optimization with superstructure model is to find out a combination of all these parameters for the best performance of a biocatalytic process for Saccharomyces Cerevisiae problem (Giorno et al. 2000). Saccharomyces Cerevisiae is a yeast organism used throughout food and drink industries. Variety of catabolic and anabolic reactions are used for cell metabolism and biomass production starting from a substrate of glucose. The key interest of these reactions is to optimize and explore the condition under which ethanol is produced.

3.2. Experimental results

For the TS model we used Markov chain with 50 slots, each having 5 iteration with neighborhood size $N(s)$ of 6 and 7. All workers were with the same Central Processing Unit (CPU) type; Intel(R) Pentium(R) 4 3.00GHz. The whole process was implemented using Grid Superscalar (GRIDS), which is a programming paradigm for grid enabling applications (Badia et al. 2007), and with a complete Markov chain being a task for a single worker. The estimated execution time was monitored for each worker using GFlops parameter within GRIDS. Figure 4 shows an example of elapsed time against the number of workers used for biocatalytic application. In this particular case the improvement is limited to 6 workers because the communication overhead becomes dominant after that.

For the COPT model, we deliberately used workers with different CPU power, ranging from Intel(R) Pentium(R) 3 700MHz to Intel(R) Pentium(R) 4 3.00GHz. Again, the whole process was implemented using Grid Superscalar (GRIDS) and with all workers working independently in an asynchronous mode. Other COPT parameters were taken as explained in Section 2.1. The performance of each worker is shown in Figure 5. Here, the elapsed time $t$ is the time from the beginning of the whole execution to the end of the current section of Markov process. It is evident from
comparison diagram in Figure 6 that the optimization search converges faster with more workers added. The reason is in the fact that when using more workers, not only more solutions are generated, but also more search regions are explored so that the optimization search is easier to find the global optimal solutions. Also, compared to the TS performance for the same experiment, the improvement is obvious from total execution time of 2 hrs for TS to below 1 hrs for COPT.

It has to be noted here that we performed similar experiments with some other reactor network applications, such as Van de Vuse and Lacoste reaction scheme (Marcoulaki et al. 1999) and the results were very similar.

4. Conclusion

Computer grid implementation of both Tabu Search and Cascade Optimisation models were presented. The performance of these has been tested by a series of experiments in the form of complex reactor network with biocatalytic one being presented in the paper. From the results, a conclusion can be made that the COPT model is better suited for parallelized execution as it can be executed in asynchronous application on computer grid which was clearly demonstrated using execution time as a measure of performance. This is primarily due to the fact that COPT algorithm enables full parallelization and independent operation of each worker and still benefiting from aggregated analysis of past history of solutions.

References