The variety and the continuous evolution of parallel architecture computers introduce many difficulties in the development of efficient parallel algorithms and, at the same time, give impulse to the research activity in this area.

Common guidelines and goals in the design and implementation of parallel algorithms are concurrency, data locality, minimization of synchronization and communication, load balancing, but the way they are actually achieved depends on the target architecture. Different programming models have been defined (data-parallel, task-parallel, structured-parallel, shared-address space, message-passing, etc.), in order to hide the complexity of the underlying architecture and to simplify the algorithm development. These models also correspond to different possibilities for parallelism exploitation and are not all mutually exclusive.

On the other hand, various abstract models of parallel machines, more “realistic” than PRAM, have been introduced, such as BSP, LogP, CGM, etc., to analyse and compare algorithms and to predict their performance behaviour on a wide range of computers. However, despite the many efforts in this direction, the most widely used and still more effective way to determine the performance of a parallel algorithm is implementing and testing it on a specific architecture.

The five papers included in this session provide a picture of the scenario previously outlined. They present parallel algorithms that share the above common guidelines and goals, but use different programming models to solve various problems on different target architectures.

Guivarch et al. analyse a class of parallel asynchronous iterative algorithms for the solution of convection-diffusion problems. Parallelism is achieved through domain decomposition, and asynchronous iterations allow flexible communication on MIMD distributed-memory machines. Garcia et al. present a parallel algorithm for the Longest Common Subsequence problem. This algorithm is described and analysed in the framework of the Coarse-Grained Multicomputer (CGM) model and experimental results on a Linux cluster are reported to support the theoretical analysis.

Ring-based algorithms for the computation of long- and short-range interactions are discussed by Corana. A performance model is proposed to analyse the behaviour of these algorithms on heterogeneous computing systems, and numerical experiments on a network of workstations confirm the results predicted by the model. Banos et al. describe a coarse-grained parallel evolutionary algorithm for circuit partitioning, using a message-passing paradigm. Parallelism is exploited here not only to speed up the algorithm, but to improve the quality of the solution.

Finally, a fine-grain parallel version of quicksort for cache-coherent shared-address space multiprocessors is presented by Tsigas et al.. Efficiency is achieved in this algorithm by using suitable techniques to reduce synchronization.