Best Practices in Reporting Results with Parallel Genetic Algorithms

Disdain for rules is as harmful as their excessive observation.

Juan Luis Vives (1492 - 1540) - Spanish philosopher

Most optimization tasks found in real world applications impose several constraints that frequently prevent the utilization of exact methods. The complexity of these problems (they are often NP-hard [78]) or the limited computational resources available to solve them (time, memory) have made the development of metaheuristics a major field in present research. In these cases, metaheuristics provide optimal or suboptimal feasible solutions in a reasonable time. Although the use of metaheuristics allows to significantly reduce the time of the search process, the high dimension of many tasks will always pose problems and result in time-consuming scenarios for industrial problems. Therefore, parallelism is an approach not only to reduce the resolution time, but also to improve the quality of the provided solutions. This last holds since parallel algorithms usually run a different search model with respect to sequential ones [35] (see also Chapter 1).

Unlike exact methods, where time-efficiency is a main measure for evaluating their success, there are two chief issues in evaluating parallel metaheuristics: how fast solutions can be obtained, and how far they are from the optimum. We can distinguish between two different approaches for analyzing metaheuristics: a theoretical analysis (worst-case analysis, average-case analysis, ...) and an experimental analysis. Several authors [80, 81] have developed theoretical analyses of some importance for a number of heuristics and problems. But, these theoretical achievements have a difficulty that makes it hard their utilization for most realistic problems and algorithms, severely limiting their range of application. As a consequence, most of metaheuristics are evaluated empirically in an ad-hoc manner.

An experimental analysis usually consists in applying the developed algorithms to a collection of problem instances and comparatively report the observed solution quality and consumed computational resources (usually time). Other researchers [82, 83] have tried to offer a kind of methodological framework to deal with the experimental evaluation of heuristics. Each a
3.1 Parallel Performance Measures

There are different metrics to measure the performance of parallel algorithms. In the first subsection we discuss in detail the most common measure, i.e., the speedup, and address its meaningful utilization in parallel metaheuristics. Later, in a second subsection, we summarize some other metrics also found in the literature.

3.1.1 Speedup

The most important measure of a parallel algorithm is the speedup [82, 85]. This metric computes the ratio between sequential and parallel times. Therefore, the definition of time is the first aspect that we must face. In a uniprocessor system, a common performance measure is the CPU time to solve the problem; this is the time the processor spends executing algorithm instructions, typically excluding the time for input of problem data, output of results, and system overhead activities. In the parallel case, time is not a sum of CPU times on each processor, neither the largest among them. Since the objective of parallelism is the reduction of the real-time, time should definitely include any overhead activity because it is the price of using a parallel algorithm. Hence the most prudent choice for measuring the performance of a