MPI and OpenMP implementations of Branch-and-Bound Skeletons *

I. Dorta, C. León, C. Rodríguez and A. Rojas

Departamento de Estadística, I.O. y Computación

Universidad de La Laguna. Edificio de Física y Matemáticas

E-38271 La Laguna, Tenerife, Spain

(isadorta, cleon, casiano, arodroj)@ull.es

Abstract

This work presents two skeletons to solve optimization problems using the branch-and-bound (`BnB`) technique. Sequential and parallel code of the invariant part of the solvers are provided. The implementation of the skeletons have been made in C++, and is divided into two different parts: One that implements the *resolution pattern* provided by the library and a part which the user has to complete with the particular characteristics of the *problem to solve*, that will be used by the resolution pattern. Required classes are used to store the basic data of the algorithm. `BnB` uses the class `Problem` to define the minimal standard interface to represent a problem, and the class `Solution` to typify a solution. The class `Subproblem` represents the area of not explored solutions. Its method `branch()` generates from the current subproblem the subset of subproblems to be explored. The `lower_bound()` and `upper_bound()` subproblem methods calculate a lower and upper bound respectively of the objective function for a given problem. Furthermore the user must specify in the definition of the `Problem` class whether the problem to solve is a maximization or minimization problem.

The solvers are implemented by the provided class `Solver`. In the class hierarchy there is one skeleton implemented using MPI and another using OpenMP. This is one of the main contributions of this work. Once the user has represented the problem, he/she obtains for free two parallel solvers without any additional effort: one using the message passing paradigm and other with the shared memory one.

In the implementation of the `BnB` message passing resolution pattern, we uses a Master/Slave scheme. The Master is in charge of the generation of new subproblems, and of the coordination between subtasks. The slaves work bounding and branching the received problem. The implementation of the `BnB` shared memory resolution pattern works with a global shared queue of tasks from which, the subproblems are removed and assigned to each processor.

An algorithm for the resolution of the classic Knapsack 0/1 problem has been implemented using the `BnB` skeletons. The obtained computational results for its execution on an Origin 3000 and a cluster of PC are studied.

*This work has been partially supported by the EC (FEDER) and the Spanish MCyT with the I+D+I contract numbers: TIC2002-04498-C05-05 and TI2002-04400-C03-03.