Using Metaheuristic Algorithms Remotely via ROS

José García-Nieto
Dpto. Lenguajes y
Ciencias de la Computación
E.T.S. Ingeniería Informática
University of Málaga, Spain
jnieto@lcc.uma.es

Enrique Alba
Dpto. Lenguajes y
Ciencias de la Computación
E.T.S. Ingeniería Informática
University of Málaga, Spain
eat@lcc.uma.es

Francisco Chicano
Dpto. Lenguajes y
Ciencias de la Computación
E.T.S. Ingeniería Informática
University of Málaga, Spain
chicano@lcc.uma.es

Categories and Subject Descriptors: I.2.8 Artificial Intelligence: Problem Solving, Control Methods, and Search, D.2.11 Software Engineering: Software Architectures

General Terms: Algorithms, Design

In the literature we can find a plethora of libraries of optimization algorithms implemented in different programming languages (for example, evolutionary algorithm libraries); all with their own advantages and drawbacks. However, these libraries can only be used by people knowing the programming language in which the library is implemented and, even in this way, some time is required to be familiar with the algorithms and work with them. Our proposal consists of providing a service for the remote execution of optimization algorithms through Internet, ROS, *Remote Optimization Service* [1]. This way, different algorithms developed in different programming languages are grouped in a repository reachable from anywhere in the world through Internet. Two types of actors that interact with the system are established: the *developer*, who, by means of a the *Wrapper* design pattern, adds an algorithm in a server machine (*Worker*), and the *user*, who, using the *client* application, can interact with the available algorithms in a remote way for solving optimization problems.

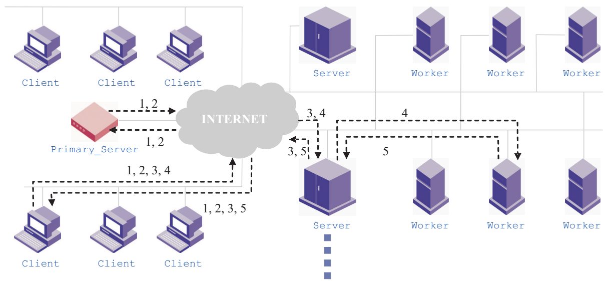

Figure 1: ROS Architecture.

In order to provide a distributed and multiplatform nature to ROS, it is implemented in Java language and the exchange of information is performed by means of XML documents. The global system consists of four kinds of connected entities: the client, the primary server, the distribution server, and the process servers (named workers). The *Client* (see Figure 1) is an application which offers a graphic interface to the users for working with the system. With this interface the users can carry out administrative operations (such as register and authentication), algorithm configuration (selection of algorithms, parameters, machines, and definition of the objective function), and execution control (remote execution of the selected algorithms and results collection). The *Primary Server* is in charge of the user authentication. It also offers information of the algorithms and servers available to the user. The *Distribution Server* (*Server* in Figure 1) is a bridge between connected clients and process servers. The distribution server is therefore a fundamental piece to obtain scalability, flexibility, and the system distribution in network, since it hides the true location of process servers and controls the load balancing among them. Finally, the *Worker* houses the algorithms, and executes them according to the orders specified by the client and routed through the distribution server. Once an execution is finished, it sends the results to the distribution server, which forwards them to the client. Let us illustrate a normal use case of ROS to get an idea of the data flow sequence through the system in a normal operation. We follow the numeration and direction of dotted lines in Figure 1.

1. *User authentication.* The user fills the initial form of the ROS client with his identification and password and connects to the primary server.
2. *Selection of algorithm, server, and programming language.* The primary server sends a list of algorithms available, distribution server addresses, and programming languages. The client receives this information and shows it to the user, who makes a selection.
3. *Execution configuration.* The client, based on the algorithm, server, and programming language selected in the last step, asks to the distribution server the specific parameters and configuration options (via XML) that the user must fill in order to run the algorithm.
4. *Algorithm execution.* Once the user sets the algorithm parameters, the XML file generated filled by the client is sent to the distribution server. This server sends the XML file to the suitable process server, which starts the execution of the algorithm.
5. *Results collection.* After the end of the algorithm execution, the results are sent in XML format to the distribution server, which forwards them to the client.

The installation package of ROS is free available at the URL `http://tracer.lcc.uma.es/ros/index.html`. It is possible to evaluate the system connecting it to active servers already working in our institution and whose IP addresses are configured in the client application.

1. REFERENCES

[1] E. Alba, J. García-Nieto, and F. Chicano. ROS: Servicio de Optimizacin Remota. In J. Riquelme and P. Botella, editors, *Actas de las JISBD, Barcelona 2006*, pages 509–514, 2006.