

JOINT WORK TO BUILD A FRAMEWORK TO SOLVE COMBINATORIAL OPTIMIZATION PROBLEMS

Abstract

This paper presents the main results of the employed effort into works on the Project of Center Optimization Distributed (PeCOD), Memetic Algorithms to Combinatorial Optimization Problems (MemePool) and the Framework to Combinatorial Optimization Problem, each one of them executed in collaboration between the following universities: PUC/RS, UNICAMP and UFSM respectively. The preliminary definitions, specifications and developmental patterns which help the Operational Research professional, among others, the reusability and reification (i.e., how to incorporate behaviors of a kind of problem into another one) data and methods of solution of Optimization Problem which are already available, as well as the differences observed in the use of new technologies, such as: use of Class Library with Java Native Interface (JNI), Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), to build a structure named framework were presented.

Key Words

Object Oriented Modeling, Distributed Systems, and Combinatorial Optimization Problems.

Introduction

The class of problems of our interest is the one with combinatorial structure, belonging to the group of NP - Full problems, for which efficient algorithms are not known, algorithms which are able to find excellent solutions in polynomial times. This difficulty can be explained by the theory of complexity (GAREY and JOHNSON, 1979).

Since accurate polynomial algorithms are not likely to exist, we opted to use heuristic methods, which were satisfactory for all the problems of the class in question. However, which heuristic is more adjusted for a determined problem? This can lead us to an exhausting analysis of all possible ones, taking as a matching parameter the implementation complexity, quality of the solutions, computational efficiency, robustness and many other features (ZANAKIS and EVANS, 1981 and BALL and MAGAZINE, 1981).

The Design of Distributed Centers of Optimization developed a support resolution environment of heuristic methods applied to the problems of Symmetrical Traveling Teller (STAGGEMEIER, 1998 and 1999) and Anti-symmetrical (BURIOL, 2000), Problem of Bin Packing (VIEIRA, 2000) and Problem of the Knapsack (MÜLLER et al., 1998), to which archetype versions with diverse instances and solution methods, shaped in accordance with the considered one for STAGGEMEIER (1999) are already available. After the first phase of the design, that deals with constructive heuristic and improvement methods, as well as allows the inclusion of existing accurate methods (which solve small instances), we realized the necessity to increase the power of resolution of the framework, not only in relation to its efficiency in computational time but also to the quality of the gotten solutions. To obtain this, a new layer must be included in this

framework to handle metaheuristics. This article presents in its next sections the solutions, which were found in a detailed way.

Guided model to objects for framework of optimization problem

To develop the modeling framework, the Language of Unified Modeling (Unified Modeling Language, UML) was used, as its name says, being the unifying force of all the modeling techniques described up to that time (FURLAN, 1998).

An object, according to UML, reflects an element of the real world, that in this case are the problems of combinatorial optimization (COP), for instance: a COP is an object, a data set of a COP is an object, a resolution method (heuristic or accurate algorithm) is an object of a COP, at last everything that can be defined by features and proper behaviors.

Thus, the first object considered for the framework of the COP is defined as *Problem of Combinatorial Optimization*. A Problem has features, which are also called attributes, such as: Problem Name, Instance Name, Method Name, Initial Solution, F Objective, Solution, which represent the name of the problem of combinatorial optimization for which the framework is being applied, the name of the instance (or data set) under which some method of resolution will be applied and that indicates the place where this instance lies on the Internet (URL), the name of the method that will solve the indicated combinatory problem, an initial solution for a resolution method that will improve it, the value found for the objective function of the combinatory problem, as well as the Solution which indicates a joined result for a COP through a chosen method of resolution, respectively.

The behavior of an object is defined by the methods, which this implements, i.e., the actions that it manipulates. For the example of the object *Problem*, an action is defined as Request to Server. This action is in charge of sending all the data of a determined problem to a computer (Serving) that it will solve it and send it back to the computer that requested the solicitation (Customer) an objective function value and the gotten data sequence for *the Problem*. This way, similar objects and objects that present the same attributes and operations are grouped together, we are no longer talking about a sole object *Problem*, but about the Class of Problems of Combinatorial Optimization, indicated by *Problem*.

One framework, therefore, involves much more than a static structure of object classes that has relationships, such as: specialization, generalization, aggregation, and association.

According to (FIRESMITH, 1993) cited in (BUZATO and RUBIRA, 1998), *frameworks* guided to objects are building blocks much bigger than objects and classes, that define collections of collaborative classes that allow the reusability in a wide scale of both design and code. That is, one *framework* is a set of classes (concrete and abstract) that provide a generic infrastructure of solutions for a set of specific problems. Therefore, this is considered a half-finished *software* architecture for a problem domain, which can be adapted in order to solve specific problems of this domain.

One of the strong reasons to develop such an architecture, is the difficulty of reusability (i.e., total or partial reusability of data and/or codes) and reification (i.e., the use of behaviors destined for a type of object by another one) generated through traditional methodologies, structured approach, what does not occur when we are talking about methodology guided to objects.

Static vision - diagram of classes

The first stage consists of the identification and definition of the structure of framework classes. The general and specific aspects that the entire framework should understand were determined, what resulted in the diagram of Classes presented in figure 1.

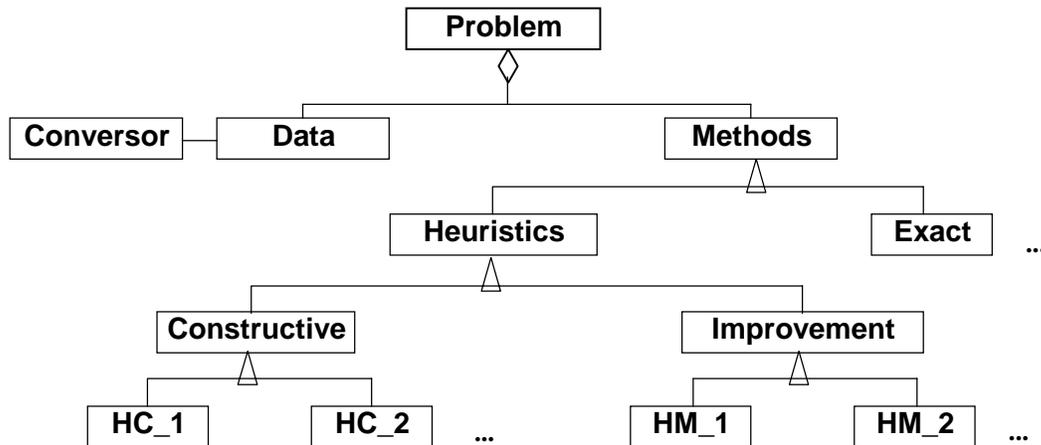


Figure 1: Diagram of Classes of the Framework for COP, in (STAGGEMEIER, 1999).

We opted to present the classes of the diagrams in the form of rectangular boxes, what simplifies the drawing and guarantees the same representation, also called diagram of Goal-classes.

A Combinatorial Optimization Problem is formed by two basic pieces, the instances (called data set here so that it is not confused with the object concept of the UML) and the resolution methods that will act on the Data to find a solution for a COP. A clarification on the methods is also persistent, therefore in Object Oriented, the word method refers to operations of an object, however, here it is related to the class of methods for resolution of the COP treated by the framework.

The class COP is represented by the private attributes (not visible to the users of the class) and that is only available through the get and set methods. These methods are the ones, which manipulate the internal information of each object COP. The most important method of this class, called *to request To Server*, is called this way for being responsible for making the solicitation of a job to a serving computer. The requested job involves the sending of the name of the problem, the name of the instance of the problem, the name of the resolution method and sends back a solution for the COP, the value of solution found in the objective function of the COP and the solution itself.

The second stage of the development of *frameworks* consists of the identification of adaptable points. In this phase it was identified that in a COP the form of the data of each problem can vary.

This way the associative relationship proposed to the Data class with the Converting class, allows to keep the same format for any COP that is requested in the interface of the application.

The class Methods identifies what type of strategy of resolution for the COP will be applied, the existing Heuristic strategy and the strategy of resolution through accurate methods.

In the same way each one of these types has subtypes, here subclasses, in the case of the Heuristic, Construction or Improvement methods.

A Heuristic method of Construction is called this way because it finds a solution for the COP only from the data set, that is, it builds a solution for the problem. A Heuristic method of Improvement, needs to find a solution for the COP, of an initial solution, that can be generated by a constructive method or not, and the respective data set.

Since there are more than one kind of heuristic for each COP, we opted to determine the specialization of this group of resolution methods as being the belonging objects to the Constructive and Improvements classes, remembering that the main difference between them is exactly the necessity of supplying an initial solution for the improvement methods.

The existing specializations in each subclass of Constructive or Improvement Heuristic, will define the COP methods that will really be executed, being called Objects Executors (legacies) from now on in this work, that is, those objects responsible for the determination of the solution in accordance with the solution strategy.

These objects encapsulate codes which were already implemented in other languages, or not, and they define a policy of reutilization, being responsible for the generation of the solution of the selected COP.

These executor object classes, are redefined for each type of COP that the framework will be used, for example: the Constructive Heuristic method of Nearest Neighbour for the TSP is implemented in different ways of the Nearest Neighbour for the Problem to Scheduling of tasks to processors. Thus, each COP has as adaptable points in the framework the Data and the Executor Objects, keeping all the remaining modeling as a fixed point.

Functional vision of the model - diagram of use cases

A use case describes the functionality of the system perceived through actors. An actor interacts with the system being able to be a user, a device or another system.

In this representation we defined only one actor - the user itself because only the interactions of this with the customer layer of the framework are represented. The other interactions as well as the definition of the other layers of the architecture of the framework are suggested in the extensions of this work.

An interaction diagram is represented by the use of scenes of the application, that is, the description of the system functionality. An example of this diagram in the sequence of steps is described: the user must select an instance (or a data set) of the COP; the user must choose the type of resolution that he desires to apply. About the provided strategies for the architecture, the results of constructive methods with improvement methods can be matched, or to execute them separately; the user can opt to visualize a joined solution graphically; the user can desire to visualize some instructions of help during the processing of the system.

The Use Case of described in fig. 2, reports the functionality of the application to solve the problem of one determined data set through a heuristic method of construction, for clarifications on the use case see (FURLAN, 1998).

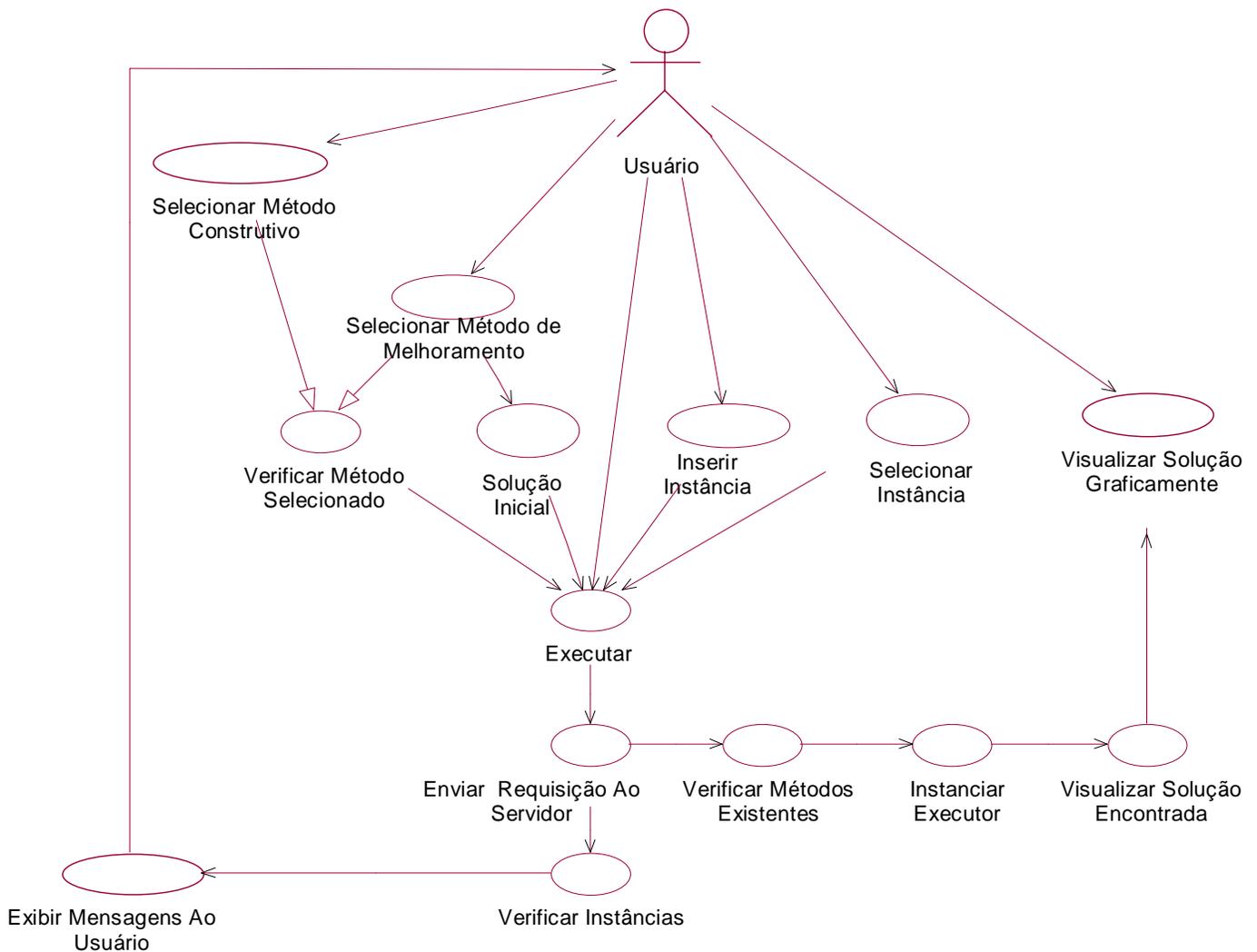


Figure 2: Use Case Diagram.

It is important to emphasize that the diagrams presented for the framework were generated through the support tool to the modeling of guided systems to Rational Rose 98 objects, free version on the Internet, that runs in Windows environment. For further details about the tool see its tutorial on <http://www.rational.com>, as well as doubts about the UML can also be clarified on the same address on the network.

The interface customer of the application

The interface of application of the layer Customer of the *framework* was developed in programming language JAVA, for the version of compiler JDK 1.1.5 (*Java Development ToolKit*), and with the support of the developmental environment IMB Visual Age For Java, free version 1.0, which allowed a greater agility in the form of presentation of the available interface components for the language and the chosen compiler JDK. The Java language was chosen for the implementation, for possessing multiplatform features, that

is, it has a virtual machine, which can be adapted to any kind of operational system, regardless of the *hardware* manufacturer.

The interfaces of the layer customer of the framework described in this article correspond respectively to the problems of Symmetrical Traveling Salesman (given a set of cities and the respective distances between each one of them, to find the total minimum distance, leaving an origin city, covering all the other ones and returning to the departure city, passing by each one of them, only once) and Bin Packing (given a set of boxes of a determined capacity, this has to be smaller than the sum of all the weights of each item - and a set of items with its respective weights, the problem has as an objective to place all the items in the boxes so that the quantity of boxes can be minimized).

The Interfaces Customers of each problem obey a standardization that facilitates the user to carry their tasks through in a way they do not need to invest time in learning about the application, however some differences are inevitable, for instance, in the window of the Problem of Symmetrical Traveling Salesman there is a graphical viewer of the solution available, because this type of problem can be represented by a graph of n nodes where each node corresponds to a city of the route and the links between two of them represent the arcs.

Yet for the interface of the Problem of Bin Packing the graphical visualization of the solution in the form of a graph is not possible, but the detail of this interface is in the possibility of generation of an instance from the definition of the data of the problem.

Once the parameters are definite and the type of solicitation, which the user intends to effect, the distribution layer is responsible for identifying where, how and when the communication between the Centers of Optimization is carried through. For better details about how these transactions work nowadays see the next section.

Structure and functioning of the distributed centers of optimization

There are some features, which benefit the architecture and the functioning of the centers and keep its independence; here they are autonomy, distribution and heterogeneity.

Autonomy - the centers work with autonomy, being able to interact in several ways for the resolution of its problems. This intention, when carried through, can be called center federation, where each center can participate in several problem resolution federations.

Distribution - the centers are distributed geographically and have distributed *software*, that is, the equipment used for the optimization has software which makes the execution of algorithms in a distributed way possible, locally in the center or through Internet with other centers;

Heterogeneity - For the execution of the algorithms in the centers, a great amount of existing equipment in each institution is used. Since these pieces of equipment are produced by different manufacturers and have different operational systems, we will adopt a strategy of integration of heterogeneous systems through the specification CORBA (*Common Object Request Broker Architecture*) or any other way that allows this integration.

The structure of the design was divided in layers such as in a distributed architecture of systems, among all these can be identified as the main ones:

Layer Customer: composed of *applet* and browser HTML, that is, the interface of the optimization problem with the user. Place where the solicitations are required, such as: to execute one determined heuristic method on one determined instance, to show the

graphical solution. This layer also receives the solution for the problem, in the reserved area the Joined Solution and control messages about the process which is being executed, such as the processing time of a method.

Layer Distribution: composed of manager of distributed objects and the interfaces of communication between objects, controls which optimization centers are available to execute the required jobs making a load balance between the centers, to prevent a job overload, and finally;

Serving Layer: composed of optimization programs, parallel processing environments and data base (statistical), that is, not only controls the activation of methods, the reading and the encapsulation of the data in the standard format for its solution, but also sends the responses, solution of the problem and processing messages to the layer Customer again so that the user of the system can analyze them and request, when available, the graphical visualization of the solution, see Fig. 3.

It was necessary to evaluate the framework before even having the architecture working fully, where for this a local archetype was used, however hardware dependent and that allowed comments, such as: for instances of very large problems, that is, with a great number of cities or items, some methods took a reasonable computational time in order to find its solution.

The construction of this archetype made use of the technology of language JAVA native methods. This choice was made by the rapidity of implementation and by considering the Test phase even before the conclusion of the other layers of the utmost importance. Thus, the proposed alteration for the synchronization problem of the results was the insertion of a choice field, in which the user can opt if he desires a synchronous transaction (he waits for the reply with the window of browser open) or an asynchronous one (the reply is sent via the user's e-mail).

Another important aspect that must be emphasized is that each layer has programming language, operational systems and *hardware* independence. This independence is minimized by the object manager that can invoke methods in heterogeneous environments remotely, in accordance with CORBA specifications.

Main conclusions

As complementary results to the related archetype development, the necessity to create input and output standards for the interface was noted, because in a combinatorial optimization problem, it is practically impossible to foresee all the possibilities of data input, as well as the form to represent the processing outputs.

A decrease in the software development time was observed once the user through this framework can compare his own methods with the available ones in the interface, thus speeding the creative process of new heuristic or even using part of the existing codes, so that the implementation work of an optimization problem is improved.

Besides the insertion of his own methods in the *framework*, provided that these methods follow the data format standards, the user will be able to insert and to evaluate his data sets, COP instances, about all the methods available in the interface, making the attainment of optimal, or quasi-optimal solutions (for being accurate and heuristic methods) be an important factor of support of the decision taking process by the most experienced users in Operational Research.

Yet, the layer customer implementation of this goal-structure we observed that an interface for the Internet is a way of releasing the Problems dealt by it, and that it guarantees reutilization and reification, once these problems can serve as "cloths of deep" for the resolution of other types of COP, through the existing changing between them, and described by the theory of computational complexity.

The main result of this implementation and modeling for the COP we obtained the evidence that an ample and generic design was necessary, a design that made these be reused in an efficient way (MÜLLER and STAGGEMEIER, 1999).

TSP Modeling were compared with other COPs, for instance, the Problem of the Knapsack and the Problem of Bin Packing, where the hierarchy and the behavior of the framework were completely preserved, needing only pertinent adequacies for the particular features of each study object problem, such as, the definition of the class *Data* which is distinctive in each COP and the inclusion of heuristic methods that are specific for the new COP, (MÜLLER et al., 1998). We must emphasize as a negative aspect in the applications based on CORBA an increase in the system reply time, due to a bigger flow of messages, if compared to the JNI use (ORFALI and HARKEY, 1998). However in the case of the COP framework this factor is not relevant due to the low incidence of solicitations if compared to the total time used to solve a problem. The main advantage of the use of the Architecture CORBA in the framework was the creation of an Object Server Executors (SILVA, 2000), to manage this layer better, allowing autonomy, distribution and heterogeneity, all vital to the COD design success.

Future extensions and works

The Evolution of the current architecture became essential when there is a necessity to include metaheuristic methods.

From this idea, we opted for the memetic algorithms, defined by functioning similar to the genetic algorithms, however, before the generated children can be evaluated, a growth environment is available to them (for example: algorithms of local search) where the solution quality is improved (CORNE, DORIGO and GLOVER, 1999 and MOSCATO, 1989). They are the ones, which adapt to the developed structure in a more natural way, because they use the legacies integrally (improvement algorithms, among others) supplying improved solutions to be included in the population (KRASGNOR and SMITH, 1999). The implementation of the memetic algorithm is done for the anti-symmetrical traveling salesman problem as the first metaheuristic representative of the new layer of the structure distributed to Optimization Combinatorial Problems (BURIOL, 2000), where there is a framework for memetic algorithms perfectly adaptable to the evolutionary structure of the design.

The new layer of distribution includes the manager of distributed objects, that allows the object handling in Client/Server environments in a flexible way (application and language), it is better than via RPC, using the distributed object concepts and having its interfaces exported. This interface exportation allows the customers to establish a communication with the methods of objects implemented in the server, thus making the customers make calls to objects, sending/receiving the parameters specified in the interface.

The interconnection between the centers will be done through the IIOP (*Internet Inter-ORB Protocol*) protocol which allows the manager of distributed objects to change

information about methods through Internet. Thus, this interconnection will be transparent to the user who will be able to request implemented algorithms in several optimization centers, without worrying about their location and their execution environment. Fig. 3 presents the proposed architecture for the PeCODE.

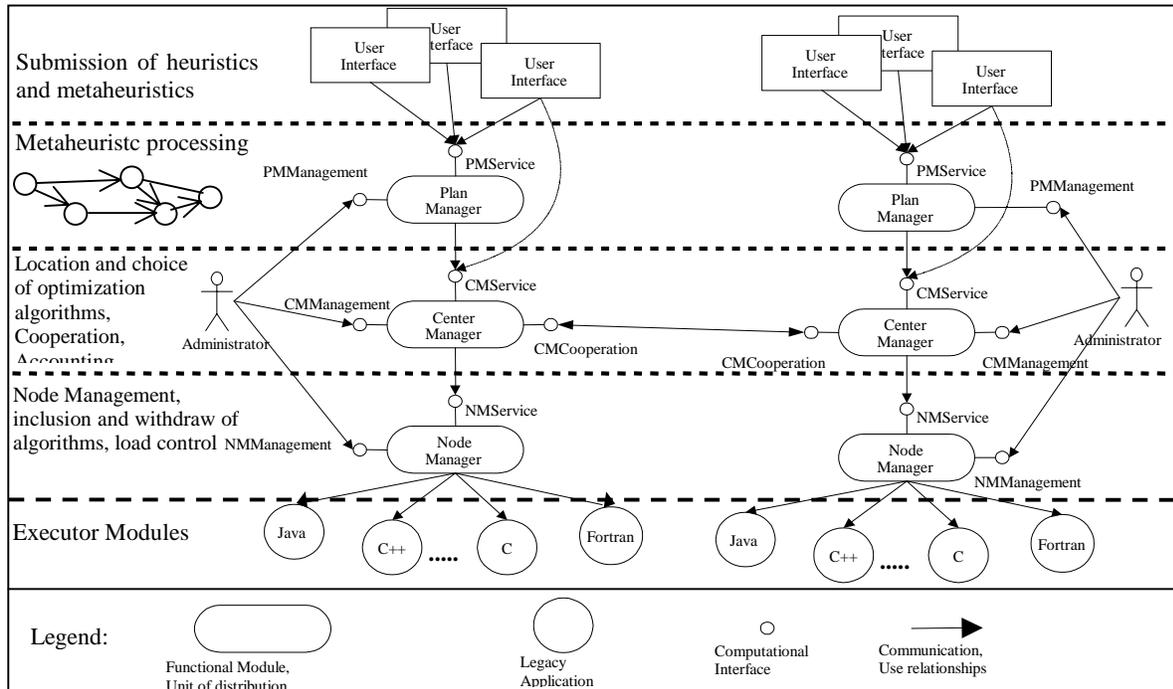


Figure 3: PeCODE Architecture

Bibliographical references

- BALL, M. & MAGAZINE, M., The design and analysis of heuristics, **Networks**, vol. 1, 11, 1981.
- BURIOL, L. S., **Algoritmo Memético para o Problema do Caixeiro Viajante Assimétrico como parte de um Framework para Algoritmos Evolutivos**, Dissertação de Mestrado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, 21 de fevereiro de 2000.
- BUZATO, L.E., RUBIRA, C.M.F., Construção de Sistemas Orientados a Objetos Confiáveis, DCC/IM, COPPE Sistemas, NCE/UFRJ, **Anais da 11ª Escola de Computação**, Rio de Janeiro, 1998.
- CORNE, D., DORIGO, M. e GLOVER, F. (Eds.), **New Ideas in Optimization**, McGraw Hill, 1999.
- FURLAN, J.D., **Modelagem de Objetos Através da UML – The Unified Modeling Language**, SP: Makron Books, 1998.
- GAREY, M. R. e JOHNSON, D. S., **Computers and Intractability: a Guide to the Theory of NP-Completeness**, San Francisco : Freeman, 1979.
- KRASGNOR, N. e SMITH, J., A java framework for memetic algorithms, acessível via internet em <http://www.ics.uwe.ac.uk/~natk/MAFRA/mafra.html>.

MOSCATO, P., **On evolution, search, optimization, genetic algorithms, and martial arts: towards memetic algorithms**, Technical Report C3P Report 826, Caltech Concurrent Computation Program, 1989.

MÜLLER, F. M., VIEIRA, V. G., STAGGMEIER, A.T., GARCIA, V. J., CAMOZZATO, M. M., e CAMPOS, T. J., Framework para problemas de otimização combinatorial: uma implementação orientada a objetos para o problema da mochila, **Anais da XIII Jornada Acadêmica**, Santa Maria - RS, dezembro de 1998.

ORFALI, R., HARKEY, D., **Client/Server Programming with JAVA™ and CORBA**, Second Edition, New York: Wiley Computer Publishing, 1998.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, W., **Modelagem e Projetos baseados em Objetos**, RJ: Ed. Campos, 1994.

SILVA, M.W., **Implementação do mecanismo de ativação de executores na camada de distribuição para framework de otimização combinatoria**, Trabalho de Conclusão de Curso de Informática, Universidade Federal de Santa Maria, Santa Maria – RS, 17 de fevereiro de 2000.

STAGGMEIER, A.T., **Meta-estrutura para Problemas de Otimização Combinatória**, Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia de Produção, Centro de Tecnologia, Universidade Federal de Santa Maria, Santa Maria - RS, 24 de novembro de 1999.

STAGGMEIER, A. T., **Framework para Problemas de Otimização Combinatorial: uma Implementação Orientada a Objetos para o Problema do Caixeiro Viajante Simétrico Euclidiano**, Trabalho de Conclusão de Curso de Sistemas de Informação, Centro Universitário Franciscano, Santa Maria – RS, 10 de dezembro de 1998.

VIEIRA, V. G., **Modelagem e Implementação do Problema de Empacotamento**, Trabalho de Conclusão de Curso de Informática, Universidade Federal de Santa Maria, Santa Maria – RS, 20 de janeiro de 2000.

ZANAKIS, S.H. e EVANS, J.R., Heuristic “optimization”, why, when, and how to use it, **Interfaces**, vol. 1, 11, 1981.