Proposed Architecture for the Combinatorial Problems Resolution based on BDI Agents in Ubiquitous Environments

Guillermo Cabrera G.
Pontificia Universidad Católica de Valparaíso
Av. Brasil 2241
Valparaíso
CHILE

guillermo.cabrera@ucv.cl

José Miguel Rubio L.
Pontificia Universidad Católica de Valparaíso
Av. Brasil 2241
Valparaíso
CHILE

jose.rubio.l@ucv.cl

Abstract – The objective of global optimality versus the cost that this implies is an unresolved issue by researchers. Classical problems such as Travel Salesman Problem (TSP) or Vehicle Routing Problem (VRP) and its derivatives, relate to combinatorial optimization problems that arise on an everyday basis for a group of people that would like to have the solution to such problems as soon as possible. Thus, the idea that such problems purely operational, would be solved by heuristics in devices makes sense, due to the interest of normal use for sellers or car’s drivers. Within the framework of what we may call ubiquitous distributed computing, this paper proposes an BDI architecture that supports the resolution of problems in complex combinatorial mobile devices across heuristics or metaheuristics embedded in the devices themselves and working from a set of data from a central server using agents for coordination and collaboration.

Keywords: Multi-agent Systems, Combinatorial Optimization Problems, ubiquitous computing, mobile devices, metaheuristic.

I. INTRODUCTION

People currently need access to information anytime and anywhere, this has led to an explosive development of a series of technologies that give support to achieve that goal. Personal Digital Assistants (PDA’s) and mobile phones with Internet access are just a sample of it. This possibility of access to “everywhere” is called ubicomp or ubiquitous computing [12]. However, since today it is not only necessary to be connected, but such connection must afford anything more than surfing the net: It should allow solving problems that men confront every day in his daily life. Some of these operational problems, called “day to day”, are related to complex combinatorial problems such as TSP or VRP.

The ability to integrate solutions for combinatorial optimization problems, through heuristics and metaheuristics, with the development of applications in a ubiquitous environment, is shown as a real alternative resolution of operational problems in different businesses. The idea that sellers, through an application on their cell phones know which customers should visit and, primarily, the path to be followed to incur in the minimum cost of mobilization, appears to be possible through to development of pervasive computing.

Significantly, the architecture that is presented in this article, propose solves the problem directly into the mobile device, making the dependence on a central server is limited only to sending data entry required for proper performance of the algorithm, without neglecting concepts relevant to the moment of development of mobile applications such as those presented in [14][15]. In this sense, it aims to go a step further toward what we call ubiquitous distributed computing, namely the development of distributed solutions in mobile devices.

A. Solving DCSP using BDI

On the other hand, the evolution of multi-agent systems has generated interest by its capacity to confront the problems in a distributed way, and since the first publishing in this scope [1], several algorithms of resolution have been created until today.

Exists a particular type of agent, it is denominated BDI agent, whose abbreviations are the acronyms of Beliefs, Desires and Intentions. This type of agent approaches the problems on the basis of these three mental states, through the observation of its surroundings and the generation of plans.

The aim of this paper is the integration of all these concepts. Exists some studies around it [2], in this occasion, we will focus in the design of the architecture, using the proposed tools by the methodology of development oriented to agents of the Australian Institute of Artificial Intelligence AAII [3].

II. PROBLEM DEFINITION

As previously introduced, the kind of problem that is approached in this work is the CSP or Constraints Satisfaction Problem, which consists of a finite set of variables that are subject to constraints that must be satisfied for solving the problem.

Efforts so far in solving complex combinatorial problems have focused on a traditional not-ubiquitous environment. In this sense, there are many articles in which the resolution of such problems is focused on changing from a centralized architecture, where heuristics resolution were executed in just a processor, to a distributed architecture where algorithms resolution are executed in parallel way. In that sense, [22] presents some improvements associated with solving transportation problems and allocation in a distributed environment. The parallelization of algorithms in multiple processors has also been used in not-ubiquitous environments, and independent of the heuristics resolution to be used, as shown in [23][24].
There are multiple solutions and investigations about this matter in the scope of the programming with restrictions. Nevertheless, we will concentrate to confront the problem in a distributed form, which gives rise to the DCSP.

In this section we have formalized a definition of these concepts.

A. CSP Definition.

On a literature point of view referring to CSP this has formalized it in a very similar way just in the majority of publications. In this case we will refer to the definition published in [4], that says that “A CSP $P$ is a triple $P=\langle X,D,C \rangle$ where $X$ is an n-tuple of variables $X=\langle x_1,x_2,...,x_n \rangle$, $D$ is a corresponding n-tuple of domains $D=\langle D_1,D_2,...,D_n \rangle$ such that $x_i \in D_i$, $C$ is a t-tuple of constraints $C=\langle C_1,C_2,...,C_t \rangle$. A constraint $C_j$ is a pair $\langle R_j,S_j \rangle$ where $R_j$ is a relation on the variables in $S_j=\text{scope}(C_j)$. In other words, $R_j$ is a subset of the Cartesian product of the domains of the variables in $S_j$.”

B. CSP Definition.

The formalization of a distributed CSP is the same one for a CSP, the difference is in the way which the problem is solved. A DCSP distributes to the variables and restrictions in several agents who negotiate and collaborate to each other to solve the problem altogether.

Formally, according to [4], there exist $m$ agents $1, 2, ..., m$. Each variable $x_j$ belongs to one agent $i$ (this relation is presented as $\text{belongs}(x_j,i)$). Constraints are also distributed among agents. The fact that an agent $l$ knows a constraint predicate $p_k$ is represented as $\text{known}(p_k,l)$.

C. BDI Definition.

The BDI architecture (Beliefs, Desires and Intentions) is based on the cognitive model proposed by Bratman [5]. It describes the agents in three particular mental states:

i. Beliefs: it is the information that the agent has about its surroundings.

ii. Desires: they are motivations of the agent, things or events that the agent wants to obtain. A subgroup of related desires conform an objective that the agent persecutes. Also denominated “goal”.

iii. Intentions: Intentions: they are the objectives chosen by the agent, which is committed to obtain.

Typically, an interpreter [6] inside BDI agent will carry out some functions:

i. To generate options: it reads the queue of events and returns a list of options.

ii. To deliberate: it selects an action subgroup to be adopted.

iii. To update intentions: it adds the subgroup obtained to the structure of intentions.

iv. To execute: if exists the intention to realize an atomic strike at this time, the agent executes it.

v. To obtain new external events: any external event that has taken place during the cycle of interpreter, is added to the queue of events. The internal events are added in the same sequence as these happen.

vi. To eliminate successes: it modifies the structures of intentions and desires, leaving successful desires and the satisfied intentions.

vii. To eliminate impossible: it modifies the structures of intentions and desires, leaving impossible desires and the unrealizable intentions.

D. The Traveling Salesman Problem (TSP).

TSP is a paradigmatic NP-hard combinatorial optimization problem which has attracted an enormous amount of research effort [8, 9]. These efforts have focused on developing heuristics and combinatorial optimization metaheuristics that through intelligent searches trying to find solutions of good quality at low cost (or computer time).

TSP has been so studied by researchers, that its different instances are used as a benchmark to validate new techniques developed by researchers. An example of this is the ants Systems (AS) precisely because his first application was on the TSP [10, 11]. Likewise, many other techniques have been applied to this problem: Genetic Algorithm [12], Tabu Search, Simulated Annealing and hybrids heuristics [13] among others. A Classical TSP can be defined as a problem where starting from a node it is required to visit every other node only once in a way that the total distance covered is minimized. This can be mathematically stated as follows:

$$\text{min } \sum_{i \in C} \sum_{p \in C} c_{ij} x_{ij} \quad \text{s.t.} \quad (2.1)$$

$$\sum_{i \in C} x_{ij} = 1 \quad \forall i \neq j \in C \quad (2.2)$$

$$\sum_{j \in C} x_{ij} = 1 \quad \forall j \neq i \in C \quad (2.3)$$

$$u_i = 1 \quad (2.4)$$

$$2 \leq u_i \leq n \quad \forall i \neq 1 \quad (2.5)$$

$$u_i - u_j + 1 \leq (n-1)(1-x_{ij}) \quad \forall i \neq 1, \forall j \neq 1 \quad (2.6)$$

$$u_i \geq 0 \quad \forall i \in C \quad (2.7)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in C \quad (2.8)$$

Where $c_{ij}$ is the cost of transportation from client $i$ to client $j$. Constraint (2.2) and (2.3) ensures that each customer is visited only once by the salesman. Constraints set (2.4), (2.5), (2.6) and (2.7), together are called MTZ constraints and is used to eliminate any sub tour in the solution. (2.8) are the integrality constraints.

D. Vehicle Routing Problem (VRP).

VRP is a very complicated combinatorial optimization problem that has been worked on since the late fifties, because of its central meaning in management distribution. Specific methods as Tabu Search [14], simulated annealing
[15], genetic algorithms [16] and neural networks [17] have been proposed to solve it. The VRP and the TSP are closely related. As soon as the customers of the VRP are assigned to vehicles, the VRP is reduced to several TSPs [18]. The most elementary version of the VRP is the capacitated vehicle routing problem (CVRP). An important extension of the CVRP is the vehicle routing problem with time windows (VRPTW). This problem includes, for the depot and for each customer, a time window during which the customer has to be served. The tours are performed by a fleet of identical vehicles. Mathematical formulation is showed in [24] as follows:

The VRPTW is given by a fleet of homogeneous vehicles (denoted \( V \)), a set of customers \( C \) and a directed graph \( G \). The graph consists of \( |C| + 2 \) vertices, where the customers are denoted 1; 2; …; \( N \) and the depot is represented by the vertex 0 (“the driving-out depot”) and \( N + 1 \) (“the returning depot”). The set of vertices, that is, 0, 1, …, \( N + 1 \) is denoted \( V \). The set of arcs (denoted \( A \)) represents connections between the depot and customers and among the customers. No arc terminates in vertex 0, and no arc originates from vertex \( N + 1 \). With each arc \( (i, j) \), where \( i \neq j \), we associate a cost \( c_{ij} \) and a time \( t_{ij} \), which may include service time at customer \( i \).

Each vehicle has a capacity \( q \) and each customer \( i \) a demand \( d_i \). Each customer \( i \) has a time window \( [a_i, b_i] \). A vehicle must arrive at the customer before \( b_i \). It can arrive before \( a_i \) but the customer will not be serviced before. The depot also has a time window \( [a_0, b_0] \) (the time windows for both depots are assumed to be identical). \( [a_0, b_0] \) is called the horizon scheduling. Vehicles may not leave the depot before \( a_0 \) and must be back or at time \( b_n + 1 \).

It is assumed that \( q, a_i, b_i, d_i, c_{ij} \) are non-negative integers, while the \( t_{ij} \)’s are assumed to be positive integers. It is also assumed that the triangular inequality is satisfied for both \( c_{ij} \)’s and \( t_{ij} \)’s.

The model contains two sets of decision variables \( x \) and \( s \). For each arc \( (i, j) \), where \( i \neq j, i \neq n + 1; j \neq 0 \), and each vehicle \( k \) we define \( x_{ijk} \) as

\[
x_{ijk} = \begin{cases} 
0, & \text{If vehicle } k \text{ does not drive from vertex } i \text{ to vertex } j \\
0, & \text{If vehicle } k \text{ drives from vertex } i \text{ to vertex } j 
\end{cases}
\]

The decision variable \( s_{ik} \) is defined for each vertex \( i \) and each vehicle \( k \) and denotes the time vehicle \( k \) starts to service customer \( i \). In case the given vehicle \( k \) does not service customer \( i \) \( s_{ik} \) does not mean anything. We assume \( a_0 = 0 \) and therefore \( s_{0k} = 0 \), for all \( k \).

We want to design a set of minimal cost routes, one for each vehicle, such that

- every route originates at vertex 0 and ends at vertex \( n + 1 \), and
- the time windows and capacity constraints are observed.

\[
\begin{align*}
\min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} & \quad \text{s.t.} \\
\sum_{k \in V} \sum_{i \in N} x_{ijk} &= 1 & \forall i \in C \\
\sum_{j \in N} x_{ijk} &\le q & \forall k \in V \\
x_{ijk} &= 1 & \forall k \in V \\
x_{ijk} - \sum_{h \in N} x_{hjk} &= 0 & \forall h \in C, \forall k \in V \\
\sum_{i \in N} x_{i,n+1,k} &= 1 & \forall k \in V \\
\sum_{i \in N} x_{ik} + t_{ij} - K(1 - x_{ijk}) &\le s_{ik} & \forall i, j \in N, \forall k \in V \\
a_i &\le s_{ik} & \forall i \in N, \forall k \in V \\
x_{ijk} &\in \{0,1\} & \forall i, j \in N, \forall k \in V 
\end{align*}
\]
total travel time it reduces to a TSP. This paper aims to present an architecture that supports the implementation of an application in a ubiquitous environment, which resolves both problems just mentioned (TSP and VRP). Here is a brief presentation of a formulation of the TSP. There is no need to enter into detail because it is considered a problem well known and studied in the literature.

III. AGENT ARCHITECTURE AND MODELING

In order to form an idea of the problem to model, we will take as example a CSP distributed in four agents. In the same way that in [4], we will suppose each agent has exactly one variable, all constraints are binary and each agent knows all constraint predicates relevant to its variable. Figure 1 illustrates the previous idea, normally a DCSP will be modeled as a non directed graph where each node represents an agent who handles a unique variable within a domain, and each edge represents the restrictions between each pair of variables.

In the present work, the architecture of DCSP agent will be the used one by PRS system created by Georgeff and Lansky [19], which is informed in Figure 2, where an agent receives stimuli from the outside through sensors and on the basis of their sets of beliefs, objectives, plans and intentions, conducts actions by means of effectors.

Using AAII [20] methodology for the development of agents oriented systems, it is observed that exists a variable number of instances of the agent who will depend in particular on the DCSP (in the case of Figure 1 we have four instances of the agent).

A. External Point of View.
From the external point of view, the system is decomposed in agents, who must be modeled, as well as its instances and the interactions among them. Thus, the general model of DCSP agent will be as it is in Figure 3, the circle in the origin of arrow indicates that multiple instances of the agent that can exist.

Figure 4 shows in KQML language, an approach to the interaction model that agents will have to follow, these need to communicate for:

i. To suggest changes in the neighbors variable.

ii. To respond to originating requests of change of the neighbors.

iii. To inform the change into value of the variable.

iv. To inform the state in which is the agent (if it fulfills or not all constraints).

B. Internal Point of View.
In order to make the model of beliefs, some parameters that would be useful for the agents are: (i) the present value of its variable, (ii) the state of the agent, that is to say, if it fulfills all the constraints and if it is stable or not (iii) the domain of the assigned variable, (iv) its neighbors and the domain in which these move, (v) the constraints with its neighbors, (vi) the present value of the variables in their neighbors and (vii) the state in which their neighbors are, that is to say, if these fulfill or not all the constraints. The beliefs are specified in a predicates form.

In [5] three beliefs are identified which can be created by the agent at any moment: (i) it can change its variable assignment to improve its current situation, (ii) it cannot change its variable assignment and some constraints violations cannot be resolved and (iii) it does not need to change its variable assignment as all the constraints are satisfied.

The objectives are a set of related desires one to another which the agent could dedicate itself. The model of objectives of Figure 5 shows the objectives of the DCSP agent. Two main targets are identified, the objective to change the value when the present one does not fulfill the constraints, in the event that this would not be possible, there is the alternative to suggest the change of value of its neighbors (preferably of whom are in a similar state, that is to say, those that do not fulfill all constraints as well).

In order to finalize, a plan is defined in order to determine the objectives like which options the agent has to reach. The model of plans of Figure 6 represents the plan that will try to change the value of the variable X that handles the agent, a plan that will be activate when it notices that the value assigned to its variable does not fulfill the restrictions which is shared with all neighbors, subsequent to it, if it is possible to find a value, then will realize the change and will inform it to his neighbors.

### IV. APPLICATION ARCHITECTURE

This section presents the architecture of resolution proposed for both problems (TSP and VRP). Strictly speaking, this presents a generalization to troubleshoot combinatorial optimization, which will be subsequently applied to the two issues specified in the previous section.

At first, we have the main components of the proposed architecture:

1. **Central Server.** The central server contains data from places that each vendor must visit and the distances between them. It also contains the Web Service (WS) that meets the requirements of a mobile device.

2. **Mobile Device:** It is considered a mobile device that has a Java Virtual Machine (JVM) to run an application that implements J2ME heuristics resolution and a General Packet Radio Service (GPRS) data service for data transmission.

3. **Data Transmission:** The communication between the server and mobile device is accomplished via GPRS, which is a mobile service-oriented package and an HTTP server which is responsible for storing and sending data entry to services requested by Mobile devices.

Once the main components of the proposed architecture, figure 7 shows the organization of these components and how they interact with each other.
Optimization Problems in Ubiquitous Computing Environment.

More details about this architecture (Infrastructure Components, Deployment Paradigm, etc.) found in [25].

IV. CONCLUSIONS AND FUTURE WORK

Efforts to develop the concepts associated with pervasive computing suggest the resolution of problems in environments highly interconnected. This represents an architecture that will support applications able to solve problems identified, as an example is the in case of this paper in which combinatorial complex problems, may grant high benefits for all those individuals whose work involves such problems as operational activities.

This paper presents an architecture that supports the implementation of well known algorithms for complex combinatorial problem solving in an environment of ubiquitous computing. As future work, it is expected to implement a prototype based on this architecture, which is capable of solving specific problems (benchmark), so we can compare their performance based on predefined indicators and others that may arise in the course of their development and experimentation. The implementations of other well known heuristics solvers are also part of future studies, determining which are feasible to implement in an ubiquitous environment and which one due to its use of resources are not. The performance of each of the techniques in a ubiquitous environment will also be something interesting to compare.

V. REFERENCES


[7]. Francesca Rossi, Peter Van Beek, and Toby Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006.


