Multi-Agent Scheduling using Constraint Satisfaction Problem

Ioana COBEANU, Vasile COMNAC
Transilvania University of Braşov, Romania

***Abstract* — In multi-agent systems, allocating resources to each activity of the agents is very important. The allocation of resources must be done so that all agents can accomplish their given tasks and at the same time a resource must not be allocated to more than one agent. The decision making process whose goal is to allocate optimally resources to certain tasks in a given timeframe is named scheduling. This paper presents the comparison of two scheduling problem implementations, which extend the traveling salesman scenario. The first implementation was realized using ASP (A*nswer* S*et* P*rogramming*) system DLV, the second one using the CP (C*onstraint* P*rogramming*) solver Choco.**

***Index Terms* — Answer Set Programming, Constraint Satisfaction Problem, mobile agents, scheduling, Travelling Salesman Problem**

I. INTRODUCTION

Multi-agent systems technology is successfully used to model distributed systems. Distributed systems are those systems whose components can decide for themselves, where each component performs a task so that the entire system can achieve its main goal. The distributed system's environment is dynamic and continuously changing. Agents have the ability to decide for themselves based on the conditions of the environment to which they belong. The decisions of each agent will be in accordance with the needs of the entire system.

In multi-agent systems, allocating resources to each agents' activity is very important. Agents must accomplish certain tasks in order to achieve the main goal, tasks which run sequentially or in parallel. At the same time, these tasks require one or more resources. The allocation of resources must be done so that all agents can accomplish their given tasks and at the same time a resource must not be allocated to more than one agent.

The decision making process whose goal is to allocate optimal resources to certain tasks in a given timeframe is named scheduling [1]. The purpose of scheduling is to optimize one or more objectives defined by resources and tasks. In [2], it is shown that solving a scheduling problem generally consists of allocating scarce resources to a given set of activities over time. This allocation tends to become more complicated as soon as the number and type of resources and tasks required to achieve a business goal increases or when the availability of these resources is dependent on external factors.

This paper presents the comparison of two scheduling problem implementations, which extend the traveling salesman scenario. The case scenario presented is one in which construction materials and equipment must be delivered to a number of construction sites distributed around the city. The distribution company can have one or more depots used for material storage. The agents must pick up the materials and deliver them in the time intervals established at each client (construction sites around the city).

The first implementation was realized using ASP (A*nswer* S*et* P*rogramming*) system DLV, the second one using the CP (C*onstraint* P*rogramming*) solver Choco. The comparison was made based on the following criteria: ease of modeling the scheduling problem, support for search control, constraints for modeling and solution generation time.

The paper is structured as follows: in section II we present the scheduling case scenario with the related research work; the topic of constraints satisfaction problems and Choco solver are discussed in section III; in section IV ASP and DLV systems are presented. In section V the DLV and Choco implementations, the obtained results in section VI, and the implementations comparison in VII are described. Final conclusions are given in section VIII.

II. THE SCHEDULING PROBLEM

The scheduling problem refers to the delivery of construction materials and equipment, referred from now on as items, to a number of construction sites distributed around the city. The items are to be delivered from large storage houses or depots located at the outskirts of the city using minivans. The goal is to deliver all items to all required destinations while complying with the time window defined for each destination.

The scheduling problem extends the travelling salesman scenario. Added to this scenario where the time windows at each construction site, representing the moment in time at which the materials should be delivered and more than one depot can exist. Through these additions, the problem becomes MDVRPTW (M*ulti*-D*epot* V*ehicle* R*outing* P*roblem* with T*ime* W*indows*). The scheduling algorithm takes into account the situations when the items can be found in more than one depot and they need to be delivered by more than one agent. The agents' scheduling problem must find the optimal routes so that the items are delivered from the depots to all construction sites. The time spent by each agent in traffic must be less than the time of a working day.

The scheduling algorithm's goals are: to minimize the route costs (time spent in traffic), each construction site must be visited only once by one agent, the agent leaves one depot, visits the customers assigned to it, and arrives to the destination depots, the construction sites time windows are respected (in calculating the time it is taken into account also the time spent by each agent to unload the items).

The proposed case scenario is similar with other delivery scenarios such as: delivery of components to production plants or workstations, delivery of various packages by the logistics companies.

A. Related work

The TSP (T*ravelling* S*alesman* P*roblem*) scenario was and still represents a problem difficult to optimize. The TSP problem is NP-hard. The existing solutions have placed great emphasis on finding algorithms which generate solutions in more and more complex situations.

The majority of implemented algorithms are oriented towards search methods. The solutions generated by the scheduling algorithm do not need to be optimal because the time required for finding the solution tends to increase consistently (the majority of problems being NP-hard). It is possible that these optimal solutions will not correspond anymore to the actual state of the system, thus becoming irrelevant.

In the multi-agent systems, each agent must adapt its behavior to the actual state of the system. Because the system is dynamic (its state modifies in a short time), the scheduling algorithms must generate solutions in the shortest time possible, so that they correspond to the actual state of the system.

In order to minimize the time required for solution generation, in solving a scheduling problem it is necessary to define heuristic commitments and propagation techniques [3].

Examples of algorithms used for solving TSP are simple algorithms such as: *genetic algorithms*, *branch-and-bound*, or hybrid algorithms like a combination between more algorithms such as: *constraint programming propagation algorithms* to find the routes and *operations research techniques* to find the best route [4], or *ant colony optimization* with *beam search* [5].

The search methods used to solve the case scenario presented in this paper are the ones used by ASP systems (these have a search method based on *backjumping* and *leaning*) and by the Choco solver (*backtracking* / *branch-and-bound* algorithms).

In the case of this algorithm, unlike the algorithm presented in [6], the traveled distance was not taken into consideration in the route selection but the time spent in traffic (main goal being to respect the time to get to the clients). As in [6] there is a maximum time every agent can spend in traffic.

Unlike the algorithm presented in [7], for solving MDVRPTW, the one presented in this paper does not take into consideration the spatial location of the agents to associate them with their items. The main goal is to meet the time intervals. A constraint was also added to minimize the distance traveled between two clients. The distribution of the packages is done over one day.

The MDVRP (Multi) problem, solved with a *genetic algorithm* is presented in [8]. The algorithm used for route selection contains three steps: grouping/clustering, routing, scheduling/optimization. The first step in the case of both implementations is not taken into consideration because the initial data contains the location (M*ulti*-D*epot* V*ehicle* R*outing* P*roblem*) where each package is stored.

III. CONSTRAINT SATISFACTION PROBLEM

Scheduling represents an important applications field for CSP. Using variables and constraints for modeling scheduling problems good flexibility of problem representation and good power of reasoning is achieved [9].

Constraints are those conditions which restrict decisions taken in certain situations. A constraint is defined in [10] as being a logical relation between certain variables. Each variable has values in a certain domain, and constraints restrict these values.

The goal of CSP (C*onstraint* S*atisfaction* P*roblem*) is to establish the decision variables values for which all defined constraints are met, thus providing a possible solution to the problem.

A. Choco solver

The constraint programming solver in which the application was implemented is Choco [11]. More constraint programming solvers have been developed such as: JaCoP, IBM ILOG CP, JOpt, Gecode, ECLiPSe.

The Choco solver was chosen because is a Java open-source constraint programming library. It is built based on an event-based propagation mechanism with backtrackable structures. It can manipulate a great number of variables. In this library over 70 constraints are defined.

It delivers a good separation between modeling the problem (establishing the decision variables, defining constraints) and solving the problem (implementing the domain types and constraint propagation algorithms).

The solver allows the change of search strategies based on each application's requirements, so that the performances obtained are the best.

IV. ANSWER SET PROGRAMMING

ASP is a form of logic programming, which proposes a better manipulation of negation, for programs that can be interpreted on small constraint domains. Even if = is the only built-in constraint supported by ASP, the method represents an elegant formalism to model CSP.

The approach of ASP is to first model the problem so as to find an answer set for a logic program and then, by using an answer set solver, to obtain some answer sets. ASP written programs can have multiple answer sets which are equivalent to multiple solutions, or they can have no answer set which is equivalent to not having any solutions [12].

The disadvantage of ASP solvers is the fact that there is little support for search control [2].

A. DLV System

The ASP system in which the application was implemented is DLV [13]. Examples of others ASP systems are: ASSAT, CMODELS, Smodels, clasp.

This system was chosen because the programs written in DLV can easily be integrated in Java applications with the help of the DLV Wrapper. This is useful because the calculation of the shortest paths used in the simplified graph variant (representing the road network) was done using a Java graph library.

V. IMPLEMENTATION

Both implementations must receive as input data the road network. The road network is represented as a graph in which the arcs represent the streets, and the nodes represent the intersections. Every arc is associated with a cost. The arc costs represent the approximation of the time needed to pass through them.

In [14] the necessity to obtain a simplified graph of the road network is presented. The graph will be given as input data for the two implementations. The nodes of the simplified graph consist only of the starting depot, the

destination depot and the construction sites. The graphs arcs represent the most cost effective connections between depots and every construction site and the connections between all construction sites.

A. Implementation using DLV system

The implementation that uses the DLV system is presented in detail in [14]. The input data for the scheduling algorithm implemented with the DLV system is defined as: the agents, the departure depots, the destination depots, the items with their location (departure depots), the construction sites with their time windows, and the simplified graph (the connections generated in the previous step).

The language of the DLV system allows the definition of strong constraints as well as weak constraints. The strong constraints were used to calculate the agents' routes which leave the departure depot and reach the destination depot passing through all the construction sites where the items assigned should be delivered, taking into account the time windows defined for each construction site. The weak constraints were used to minimize the time spent in traffic by each agent.

The answer sets generated by the DLV system contains the arcs associated to each agent. The agents' routes are generated based on these arcs.

B. Implementation using Choco solver

The input data for the scheduling algorithm implemented with Choco solver are the number of agents, the number of depots, the construction sites with their time windows, and the road network simplified graph.

In order to solve the proposed problem, a global constraint was used, constraint which was defined in the solver as $tree(x, restrictions)$ [15]. This constraint states that G_x is a forest that satisfies some conditions specified by restrictions. The restrictions are: the number of trees, the proper number of trees, the sum of the edges' weights, incomparability, precedence, conditional precedence, the in-degree of the nodes, the time windows defined in nodes. The number of nodes for G_x is the x parameter.

In the case of the proposed problem, the number of nodes of G_x is given based on the number of depots, number of agents and the number of construction sites.

In order that agents leave from the starting depot, the number of arcs that enter in these nodes is 0. In the case of the final depot, the number of arcs that enter is given based on the number of agents in the system.

The cost matrix is a square matrix in which the approximate times needed to cross the arcs are given. The number of lines and columns is given by the number of nodes for G_x. In order to obtain results in the shortest time possible, the ordering of the nodes is done as follows. The first columns represent the arc costs from the starting depots to all x nodes. The following columns represent the costs from the construction sites to all x nodes; the last column represents the costs from the final depot to all x nodes. Construction sites have been ordered based on the time window defined for each of them. It was considered the ascending order of the middle of the defined interval. In case there is more than one starting depots, the clients are ordered by the depot where the items which are to be delivered to them are stored. The first clients are the ones for the first depot, the following clients for the second depot and so on. In this way an ordering of the n nodes is made so that the number of arcs that enter each node, the successors and the predecessors of each node can be easily defined.

The sum of the edges' weights represents the objective variable. This should not exceed the legal working time of all agents (product between the number of working hours per each agent and the number of agents). The aim is to minimize this value, so that the time spent by all agents in traffic is the shortest.

In order for the agents not to pass multiple times by the same construction sites, the constraint $allDifferent(< x_1,...,x_n >)$ has also been used.

Heuristic strategy chooses the successors variables as the only decision variables. The solver output represents the successors for each node.

VI. RESULTS

Both implementations have started from the case of one agent and based on it the final implementations were developed.

Since the first implementation, it was observed that in the case of both implementations, the optimization part of the application is the one conducting to the increased time in solution finding.

In the case of the DLV system, the weak constraints could not be eliminated from the implementation. In order to improve the solution generation times, it was necessary to add to the system invocation, the limiting of the total cost of a route (- *costbound)* and limiting the maximum integer (- *N*). The closer this number is to the desired value, the faster the solution is generated. The system does not generate solutions when the maximum integer is bigger than the value that needs to be obtained.

In the case of the Choco solver, optimizing the time spent by the agents in traffic is made by imposing the value interval for the objective variable, representing the sum of the edges' weights as close as possible to the desired value. Obtaining the minimum value can be achieved by minimizing this value, but at the same time the solution generation time increases.

The solution generation time for the two solvers is very important because the agents must adapt their behavior based on the actual state of the system. Table I, Table II and Table III presents the results obtained in the case of a single agent, a single depot and multi-depot implementations on a computer with Inter Core I5 CPU at 2,27GHz and 4GB of RAM. From these results it can be concluded that the solution generation time is much shorter in the case of the implementation with the Choco solver.

VII. IMPLEMENTATIONS COMPARISON ANALYSIS

The criteria used to perform the comparison of the two implementations are as follows: easy modeling of the scheduling problem (how intuitive the modeling is and if the user needs to be acquainted with CSP), constraint modeling (the possibility of defining new constraints) support for search control (if the user can define new search strategies), and solution generation time.

From the point of view of the modelling of scheduling problems, the DLV system proved to be a form that permits an easier modeling of the problems regarding scheduling, especially for those unfamiliar with CSP. On the other side, modeling of the scheduling problems using Choco is not as

TABLE I
SINGLE AGENT IMPLEMENTATION

Depots	Agents	Construc-tion sites	DLV Time [second]	Choco Time [second]
1	1	10	0.79	0.05
1	1	15	3.38	0.11
1	1	20	8.81	0.12
1	1	25	23.3	0.14

TABLE II
SINGLE DEPOT WITH MULTI-AGENT IMPLEMENTATION

Depots	Agents	Construc-tion sites	DLV Time [second]	Choco Time [second]
1	3	10	3.59	0.32
1	3	15	11.33	0.45
1	3	20	13.97	0.54
1	4	10	12.45	0.36
1	4	15	18.87	0.43
1	4	20	23.75	0.56

TABLE III
MULTI-DEPOT IMPLEMENTATION

Depots	Agents	Construc-tion sites	DLV Time [second]	Choco Time [second]
2	6	20	25.75	1.12
2	8	20	12.31	1.88
3	9	25	13.35	1.98
3	12	25	33.55	2.12
3	12	50	-	8.25

intuitive as DLV, requiring knowledge about constraint programming. Choco can manipulate a great number of variables in comparison to DLV in which only constant variables can be defined.

In the Choco solver, modeling methods are permitted for the input data, so that, for example, they would reduce the search for the successors of a node only in those nodes that have the bigger time frames. In the DLV system, these aspects cannot be controlled by the system's user.

The disadvantage of the DLV system is the fact that there is little support for search control. In the Choco solver different search strategies can be defined by the user in order to improve the performances of the application. Simple constraints but also global constraints defined in the solver confer the user the possibility to define new constraints. Global constraints offer dedicated filtering algorithms. In order to obtain the best performances for the applications it is attempted to model them through global constraints.

Out of the previous section emerges the fact that in the case of the proposed scheduling problem, the Choco solver generates solutions in a much shorter time than the DLV system.

VIII. CONCLUSION

In this article a comparison of two scheduling problem implementations is made, implementations which extend the travelling salesman scenario. The case scenario proposed refers to the delivery of construction materials and equipment to a number of construction sites distributed around the city. The scheduling algorithm generates the agents' routes so that they deliver all the materials in the time frame established by the construction site. Implementing this algorithm was done using two solvers: the first solver used is an ASP system named DLV, the second one is a CP solver named Choco. In this article two implementations were presented, and based on the results obtained, a comparative analysis of them was made. The criteria used to perform the comparison of the two implementations are as follows: easy modeling of the scheduling problem, constraint modeling support for search control, and solution generation time.

ACKNOWLEDGMENTS

This paper is supported by the Sectorial Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number POSDRU: ID59321.

REFERENCES

[1] M. L. Pinedo, "Scheduling. Theory, Algorithms, and Systems", 3rd edition, Springer Science+Business Media, LLC, 2008.
[2] F. Rossi, and P. Van Beek and T. Walsh, "Handbook of Constraint Programming", Elsevier, 2006.
[3] R. Dechter, "Constraint Processing", Elsevier Science USA, 2003.
[4] F. Focacci, A. Lodi, and M. Milano, "A hybrid Exact Algorithm for the TSPTW", INFORMS Journal on Computing, vol. 14, issue 4, pp. 403-417, October 2002.
[5] M.L. Ibanez, and C. Blum, "Beam-ACO for the travelling salesman problem with time windows", Computers & Operations Research, vol. 37, pp. 1570-1583, 2010.
[6] E. Alba, and B. Dorronsoro, "Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithm", Evolutionary Computation in Combinatorial, Coimbra, Portugal, vol. 152, issue 2, pp. 11-20, 2004.
[7] J.F. Cordeau, G. Laporte, and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows", Journal of the Operational Research Society, vol. 52, pp. 928-936, 2001.
[8] P. Surekha, and Dr.S. Sumathi, "Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms", World Applied Programming, vol. 1, no. 3, pp. 118-131, August 2011.
[9] N. Policella, A. Cesta, A. Oddi, and S.F. Smith, "From precedence constraint posting to partial order schedules: A CSP approach to Robust Scheduling", Journal AI Communications – Constraint Programming for Planning and Scheduling, vol. 20, issue 3, pp. 163-180 , August 2007.
[10] R. Bartak, "Constraint Programming: In Pursuit of the Holy Grail", Proceedings of the Week of Doctoral Students (WDS99), Part IV, MatFyzPress, Prague, pp. 555-564, June 1999.
[11] Choco, http://www.emn.fr/z-info/choco-solver/, accessed in January 2012.
[12] V. Lifschitz, "Introduction to Answer Set Programming", unpublished draft, 2004.
[13] DLV, http://www.dlvsystem.com/dlvsystem/index.php/Home, accessed in November 2011.
[14] I. Cobeanu, et.al., "Real-time Scheduling of Mobile Agents Using Answer Set Programming", to be published.
[15] N. Beldiceanu, P. Flener, X. Lorca, "The tree Constraint", International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, LNCS vol. 3524, 2005, pp. 64-78.