MAS IN PROJECT MANAGEMENT

COMP-575 Sec 001 – Multi-Agent Systems

Under guidance of Prof. Jason Isaacs

Project Report by Geetanjali Agarwal

Motivation for the project:

Global markets often require the coordination of firms across the world to gain maximum competence. The organization structure and the management of a firm tend to be project oriented, and members of a project team are distributed throughout a network. In this Project, I used multi-agent systems as a technique to support project management in a distributed environment. The conventional systems for manufacturing are centralized. A central database provides a consistent global view if the state of the enterprise, which is hard to be maintained in current distributed manufacturing environment. All the production activities are carefully planned, which are hard to be changed when uncertainty occurs. Agent approach replaces the central database and the central control computer with a network of agents, each endowed with a local view of its environment and the ability and authority to respond locally to that environment.

I adopted multiagent system as information infrastructure to support project management in a highly distributed environment. Each activity and resource needed in a project are represented by an agent. The resource agents and activity agents reside at the site of the project team members who own the resource or implement the activities. The functions of project management are taken by service agents. Only three types of agents are defined, and they jointly or separately can represent any object. Therefore, this structure is suitable to all applications.

Literature Review:

The paper [1] lays part of the groundwork for a domain theory of negotiation, that is, a way of classifying interactions so that it is clear, given a domain, which negotiation mechanisms and strategies are appropriate. State Oriented Domains are defined as a general category of interaction. Cases where agents have incomplete information on the goals and worth of other agents are analyzed. First, the case where agents' goals are confidential information is considered, and then what goal declaration strategies the agents might adopt to increase their utility is analyzed. Then, the situation where the agents' goals are common knowledge is considered. Another paper [2] mentions about an innovative approach to the problem of scheduling of design activities with precedence and multiple resource constraints is proposed. A procedure is proposed to transform an IDEF3 model into a set of alternative precedence networks. In the networks selected, the activities that are resource independent are grouped with a partitioning procedure. To increase the efficiency of the search for the best schedule, a procedure based on Christofide’s reduction procedure is introduced to determine a lower bound on the completion time of the hierarchically structured design activity network.

Negotiation [3] is a form of decision making in which two or parties talk with one another to resolve their opposing interests. This book is a theoretical synthesis of what is known about negotiation as a general phenomenon. The principles presented are illustrated with examples of negotiation from many specific realms, but no realm is stressed over any other. Negotiation is defined and contrasted with other forms of multiparty decision making, the significance of negotiation is explored, information is provided about the nature of research on this phenomenon, and two fundamental theoretical notions are presented: the strategic choice model and the goal/expectation hypothesis. In another paper [4], a condensed survey of multiagent systems is presented, with special emphasis on cooperation coordination, conflict resolution and closely related issues; issues that are critical for the development of large-scale, distributed complex software systems. Also, three different cooperative MAS architecture types are discussed, their drawbacks and propose the need for a service driven framework for the development of cooperative multi agent systems. A hierarchical multi-agents model is also discussed which aims to overcome the limitations of hierarchical agent architectures, to allow for intelligent information transfer among multiple agents and to answering for other applications which wasn't concerned by the previous models.

Implementation:

I have chosen "forward calculation" method with the strategy of keeping "best agent". A mobile agent is sent out to trace down the MPM network. It collects the duration data of activity A, and then clones itself into three clones to follow the different paths. When paths combine (e.g. before activity G), only the clone storing the longest path duration continues travelling. After visiting activity K, the service agent come back to the project manager and reports the duration.

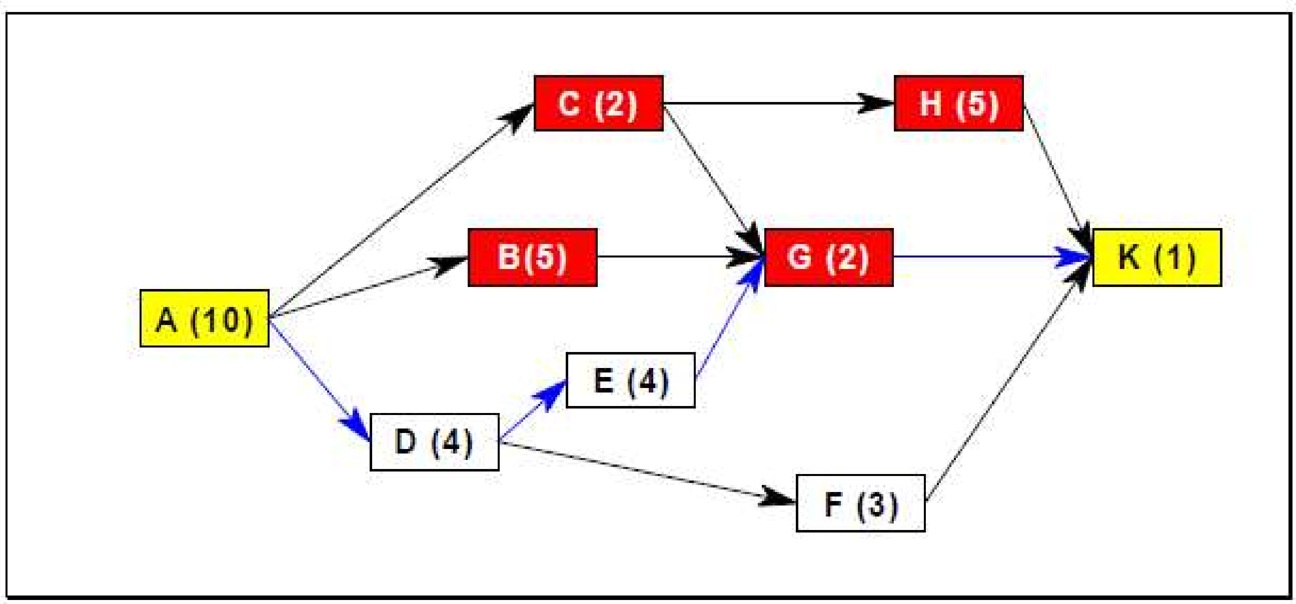

Figure 8: MPM Graph of the example

Algorithm:

The below mentioned algorithm explains the steps I have referred to write the code.

1. Create cost matrix C[ ][ ] from adjacency matrix adj[ ][ ].
C[i][j] is the cost of going from vertex i to vertex j. If there is no edge between vertices i and j then C[i][j] is infinity.

2. Array visited[ ] is initialized to zero.

```
for(i=0;i<n;i++)
        visited[i]=0;
```

3. If the vertex 0 is the source vertex then visited[0] is marked as 1.

4. Create the distance matrix, by storing the cost of vertices from vertex no. 0 to n-1 from the source vertex 0.

```
for(i=1;i<n;i++)
        distance[i]=cost[0][i];
```

Initially, distance of source vertex is taken as 0. i.e. distance[0]=0;

5. for(i=1; i<n; i++)
– Choose a vertex w, such that distance[w] is minimum and visited[w] is 0. Mark visited[w] as 1.
– Recalculate the shortest distance of remaining vertices from the source.
– Only, the vertices not marked as 1 in array visited[ ] should be considered for recalculation of distance. i.e. for each vertex v

```
if(visited[v]==0)
        distance[v]=min(distance[v],
        distance[w]+cost[w][v])
```

Software:

I have used C programming language to implement the above explained algorithm. C is an imperative procedural language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming.

Results:

Result 1: Output:

```
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node 1=10
 Path=1<-0
Distance of node 2=50
 Path=2<-3<-0
Distance of node 3=30
 Path=3<-0
Distance of node 4=60
 Path=4<-2<-3<-0
Process returned 5 (0x5)   execution time : 47.471 s
Press any key to continue.
```

Result 2: Complexity of algorithm:

The program contains two nested loops each of which has a complexity of O(n). n is number of vertices. So, the complexity of algorithm is **O(n2).**

References:

[1] Mechanisms for Automated Negotiation in State Oriented Domains
(Gilad Zlotkin and Jeffrey S. Rosenschein)

[2] Resource Constrained Scheduling of Hierarchically Structured Design Activity Networks
(By Upendra Belhe and Andrew Kusiak)

[3] Negotiation Behavior (D.G Pruitt, Academic Press, New York (1981))

[4] Coordination, Cooperation and Conflict Resolution in Multi-Agent Systems (W. Alshabi*, S. Ramaswamy+, M. Itmi*, H. Abdulrab*)
[5] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.1502&rep=rep1&type=pdf

[6] https://link.springer.com/chapter/10.1007/978-1-4020-6268-1_87

[7] https://www.jair.org/media/72/live-72-1411-jair.pdf

[8]https://books.google.com/books?hl=en&lr=&id=WvFFBQAAQBAJ&oi=fnd&pg=PP1&dq=D.G+Pruitt+Negotiation+Behavior,+Academic+Press,+New+York+(1981)&ots=Eq2cbZgCQC&sig=ALgZpnvCcKTDvv1O_WYg6W70U8w#v=onepage&q=D.G%20Pruitt%20Negotiation%20Behavior%2C%20Academic%20Press%2C%20New%20York%20(1981)&f=false