Project Management with Computational Trust Models in a Multi-Agent Environment

Ricardo Ferreira da Silva
up201305163@fc.up.pt

Gustavo de Castro Nogueira Pinto
up201302828@fe.up.pt

Tiago Filipe Abreu Figueiredo
ei12069@fe.up.pt

December 8, 2016

Contents

1 OBJECTIVES

A project, in the context of operations research consists of a set of interdependent tasks, a start state and an end state. Each of these tasks has a certain duration and it is required that the tasks preceding it are completed before we can execute it. The start state gives us access to the initial tasks (tasks without precedences), and the final state is reached when all tasks are completed, which means the project itself was completed.
Expanding upon this context, lets say we are in an simple entrepreneurial setting. A company will certainly have a project which it intends to complete and it possesses a set of workers, each one with a different set of skills and different degrees of proficiency at those skills. There will also exist a very special type of worker, that is, the manager. The manager has the crucial function of evaluating all its subordinate workers and assigning them to the tasks they will be most useful at, with the ultimate objective of finishing the overall project in the shortest amount of time.
Our objective, is to model this problem in an multi-agent environment setting and use computational trust model in order to obtain an evaluation of these workers via a manager, thus obtaining an approximation of the information needed to make the correct choices in terms of allocation.

2 SPECIFICATION

Just as it was described in the previous chapter, the objective of this application is to simulate the development of a project. This project consists of a set of interdependent tasks that must be concluded, a a manager responsible of making decisions and a set of workers, whose ratings are not fully known by the manager.

2.1 PROJECT STRUCTURE

2.2 AGENTS

In order to model this problem in a multi-agent environment we decided to design three of the main entities in the problem as agents: Manager, Worker and Task. The worker and the task are mostly reactive agents since all their actions are triggered via a manager order, with an exception we will later discuss. The manager is a BDI agent and he is the true "mastermind" so to say behind the project. He has the final objective of finishing the project in the shortest amount of time and he must get to know the worker agents properly so he can effectively allocate them to the available tasks.

2.2.1 WORKER

2.2.2 TASK

2.2.3 MANAGER

The manager is a "Beliefs-Desires-Intentions" type agent that is responsible for allocating the available workers to available tasks with the ultimate goal of finishing the project in the

shortest amount of time. The manager has knowledge of all tasks and of all workers. Regarding workers, the manager can order them to work on a task and he does this by deciding if a worker is useful at working in a particular task by comparing his skills and ratings with the skills required by the task. However, when the project starts the manager has no idea what skills (or ratings at those) each worker has, so he assumes each worker to have a neutral value at each skill and that value will be updated using FIRE components each time a task is completed.
The manager has two behaviours. The first one, and also the first behaviour to be executed is the critical path method. The manager takes into account the estimated duration values of each task (provided by the user in the configuration) and their dependencies, calculates the critical time of each task and orders the tasks in a list with this heuristic. This is known has the Critical-Time-Algorithm and while there is no efficient scheduling algorithm currently known that always gives an optimal schedule, the Critical-Time-Algorithm is the best general-purpose scheduling algorithm currently known. Later on, we will describe how we implemented parallel allocation of workers (when there is more than an available task) in order to increase the performance of this algorithm.

3 Development

3.1 Platform

3.2 Modules

3.3 Implementation

4 Experiences

5 Conclusions

6 Improvement