
Journal of Intelligent Manufacturing, 15, 455–462, 2004

Teams of autonomous agents for job-shop scheduling problems: An experimental study

M. EMIN AYDIN and TERENCE C. FOGARTY

South Bank University, School of Computing, Information Systems and Mathematics, London, UK, SE1 0AA
E-mail: aydinme@sbu.ac.uk; fogarttc@sbu.ac.uk

Received February 2003 and accepted December 2003

ATeams—teams of autonomous agents co-operating by sharing solutions through a common memory—have been proposed as a means of solving combinatorial optimization problems. In this paper, the ATeam architecture is tested on the job-shop scheduling problem. The results show that the method can work, but that it depends on the portfolio of agents and on the way in which the memory is managed.

Keywords: Multi-agent systems, distributed and parallel computing, simulated annealing, taboo search

1. Introduction

Scheduling has been one of the most studied problems for a long time. This is because of its NP-hardness and time-consuming nature and the successes of proposed algorithms in solving this kind of problems. There is a growing tendency to use distributed computing techniques in solving scheduling and timetabling problems. One of the justifications for this is the difficulty of solving such large-scale problems by single algorithms, which are unlikely to perform efficiently across a range of problems. The search space for systematic constraint-based methods can be prohibitively large, while local search methods can provide acceptable performance, but cannot guarantee to give the optimal results. Local search methods work by iteratively searching neighborhoods of solutions, and algorithms frequently become trapped in local optima (defined by their neighborhood function). Since different algorithms may have different neighborhoods, it is possible that a local optimum for one algorithm will not be a local optimum for another. If the local optimum is passed from one algorithm to another, it is possible that further progress can be made. Second, job-shop scheduling (JSS) in manufacturing industry is more complicated than those problems abstracted for theoretical studies. The manufacturing companies get larger through time by getting new suppliers, partners and customers. As a result, the relation chain enlarges and the scheduling of jobs has to be done in the context of physical distributedness. So, the companies have to consider the situation of their suppliers and sub-suppliers when they generate their plans and schedules.

Distributed scheduling has been discussed since the late of 1980s, but the software and hardware technologies were not as sophisticated as they are today. Tharumarajah and Bemelman (1997) give a review of this topic classifying the studies undertaken into three categories: hierarchical, heterarchical and centralized control. In a hierarchical approach like distributed asynchronous system DAS (Burke and Prosser, 1994), the system is co-ordinated level by level and the corresponding parts are synchronized. In the heterarchical approach, the systems are co-ordinated by allowing the agents to negotiate amongst themselves. Anke *et al*. (1997) and Wellner and Dilger (1999) applied that approach to timetabling and JSS problems. Lastly, the agents can be co-ordinated by a

single module utilizing centralized control. Talukdar (1993) and Talukdar *et al*. (1996) proposed the teams of autonomous agents (ATeams) to solve large combinatorial optimization problems using a multi-agent-based distributed problem solving method where the agents asynchronously build shared solutions. This method allows the system either to be centrally controlled or decentralized. In an ATeam, a collection of agents co-operates by sharing solutions through a common memory. The architecture is asynchronous and the agents are autonomous—each agent decides when and how to act. Some agents may operate solely to keep the population in check, destroying selected inferior solutions.

Talukdar (1993) reports success on a number of different problems. However, many questions remain unanswered. How should the population of agents be selected? Which agents work best in combination? How should agents decide when to act, and on what solutions to act? How effective are the destroyer agents? The research reported in this paper is an attempt to answer some of these questions, and in general, establish whether ATeams work, and if so, under what conditions. Thus, the interest of this work is not concerned with finding the best individual algorithms for solving problems, nor even with demonstrating that ATeams can provide comparable results to algorithms published elsewhere. The main aim is whether or not the ATeam results are better than the results of the individual algorithms that make up the team.

The rest of the paper is organized as follows. In Sections 2 and 3, brief introductions to ATeams and JSS problems are given. Then, the system architecture and the algorithms implemented are presented in Section 4. After that, the experimental results are discussed in Section 5 and the conclusions drawn in Section 6.

2. Asynchronous teams (ATeams)

Talukdar (1993) proposed one architecture of autonomous agents operating asynchronously on a shared population of solution attempts, which they call ''ATeams''. In the basic architecture, each agent is completely independent from the rest, and operates by selecting a solution from the memory, carrying out some operations on that solution, and then placing it back in the memory. Co-operation is thus achieved by sharing solutions. The population of solutions is controlled by a subset of destroyer agents, which evaluate solutions according to certain criteria and remove unwanted solutions. The organization of the agents is that loose-agents may appear and disappear from the team without penalty, may be widely distributed and do not communicate directly with other agents. An instance of ATeam architecture is shown in Fig. 1. Here, the system consists of a team of agents and a single memory, which has particular communications with each agent.

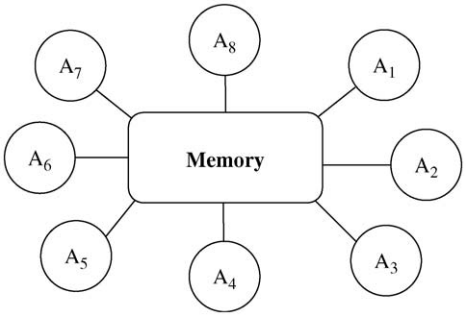

Fig. 1. An instance of ATeams architecture.

ATeams are, in many respects, similar to blackboard system, in that a collection of processes co-operates to solve problems by posting the results of actions to a shared memory (or blackboard). However, there are some differences. In blackboard systems, the problem solving process is typically centrally controlled, with a control process deciding which of the available knowledge sources should be activated at which point. Blackboards are typically structured to suit a particular problem, being hierarchically sub-divided, with problems also being sub-divided and sub-problems combined in pre-determined ways. In a basic ATeam, there is no control and each agent operates without knowledge of the others. The memory is typically on a single level. Agents can be significant for the behavior of the system, and thus merit serious investigation. ATeam systems can also encompass evolutionary methods. A genetic algorithm, for example, could be instantiated as one of the agents. More fundamentally, however, the basic operators of a genetic algorithm could be implemented as independent agents—e.g., one agent implementing the crossover operator and another implementing a mutation operator—and go through repeated distinct generations, but allows reproduction and selection to operate concurrently. Additionally,