A multi-agent approach to solve job shop scheduling problems using metaheuristics

Carlos A. S. Passos*, Vitor M. Iha, Rafael B. Dominiquini****

**Centro de Tecnologia da Informação Renato Archer, Campinas, São Paulo, Brazil (Tel: 55.19.3746.6118; e-mail: carlos.passos@cti.gov.br).*

***Universidade Estadual de Campinas – UNICAMP, Campinas, São Paulo, Brazil (e-mail: v. massaru@gmail.com; rafaeldominiquini@gmail.com).*

Abstract: This paper presents a multi-agent approach to solve job shop scheduling problem using metaheuristics. The job shop scheduling problem (JSP) is a traditional problem largely exploited in the operational research area through the academic research but belonging also to the domain of practical situations, especially in industrial companies. Several approaches have been used to investigate this problem that is very complex when addressing real industrial cases. The JSP is a NP-hard problem and no exact solution can be obtained in polynomial time. Metaheuristics approaches when solving scheduling problems have proven to be very effective and useful in practical situations. Among them, Tabu Search (TS) and Genetic Algorithms (GA) have been used to solve optimization problems with success. The main reason is that these algorithms have good performance in terms of solution quality and execution time, when compared with optimization or simple heuristics techniques respectively. In this sense, the multi-agent approach proposed in this paper combining these algorithms brings new perspective to solve this kind of problem. In this paper a multi-agent approach based on A-Team that combines TS and GA and other specific agents is presented to solve the JSP. Results for benchmark problems from the literature are presented aiming to demonstrate the applicability of the proposal.

Keywords: Scheduling, Metaheuristics, Multi-agents, A-Teams, Tabu Search, Genetic Algorithms.

1. INTRODUCTION

Job shop, here referred as jobshop, is a production system that process n number of tasks on m number of machines. In this type of system, products are made to order and the volume can vary from small to large quantities. Usually, these orders are different in terms of processing requirements, materials needed, processing time, processing sequence and setup times. Jobshop problems are widely known as a NP-hard problem, Baker (1974) and French (1982). Nowadays, several search algorithms based on optimization techniques have been developed. However, results from those methods sometimes are really unpredictable, require a lot of time and are dependent of the problem size. Thus, the users are usually satisfied with an acceptance result which it is not the optimum.

One of the widely used techniques in industries is the local search, like for example tabu (or taboo) search (TS) and genetic algorithm (GA). Tabu Search was first proposed by Fred Glover in 1986, Glover (1986). The basic principle of TS is to pursue local search whenever it encounters a local optimum by allowing non-improving moves. Cycling back to previously visited solutions is prevented by the use of memories, called tabu lists that record the recent history of the search, a key idea that came from the Artificial Intelligence concepts.

Genetic algorithms are applied to solve a problem using the principle of evolution. In the search process it will generate a new solution using genetic operator such as selection, crossover and mutation. Genetic algorithms start its search space in a population and will maintain the number of population in iteration. It will generate a new schedule by selecting two individuals in population to apply crossover and mutation. There are many procedures that could be applied in the selection, crossover and mutation process. In scheduling, genetic algorithms represent schedules as individuals or a population's members. Each individual has its own fitness value which is measured by the objective function. The procedure works iteratively, and this iteration is a generation. Each generation consists of individuals who survive from the previous generations. Usually, the population size remains constant from one generation to the next generation, see Yamada (1997).

Our intention in this research is to find out if the idea of combining the TS and GA in a multi-agent approach is suitable when dealing with jobshop scheduling problems so that the makespan value can be minimized.

2. JOB SHOP SCHEDULING PROBLEM

Recently, researchers have been focusing on investigating machine scheduling problems in manufacturing and service environments where jobs represent activities and machines represent resources, and each machine can process one job at a time. The jobshop scheduling problem can be described as a set of n jobs denoted by J_j where j =1, 2,..., n which have to be processed on a set of m machines denoted by M_k where k =1, 2,..., m. Operation of j_{th} job on the k_{th} machine will be denoted by O_{jk} with the processing time p_{jk} Each job should be processed through the machines in a particular order or also known as technological constraint. Once a machine starts to process a job, no interruption is allowed. The time required for all operations to complete their processes is called makespan. Our intention, in this paper, is to minimize this makespan value.

3. MULTI-AGENT SYSTEM

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve, Wooldridge (2002). The agents in a multi-agent system have several important characteristics, between then are:

Autonomy: the agents are at least partially autonomous,

Local views: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge and

Decentralization: there is no designated controlling agent.

Typically multi-agent systems research refers to software agents. However, the agents in a multi-agent system could equally well be robots, humans or human teams. A multi-agent system may contain combined human-agent teams. Multi-agent systems can manifest self-organization and complex behaviors even when the individual strategies of all their agents are simple. Agents can share knowledge using any agreed language, within the constraints of the system's communication protocol.

The study of multi-agent systems focuses on systems in which many intelligent agents interact with each other. Their interactions can be either cooperative or selfish. That is, the agents can share a common goal (e.g. an ant colony), or they can pursue their own interests (as in the free market economy).

Several approaches can be used to implement multi-agent systems. One of them which is well adapted to solve scheduling problem is the Asynchronous Teams (A-Teams), see Talukdar et al. (1990). An A-Team is a problem-solving architecture in which the agents are autonomous and cooperate by modifying one another's trial solutions. These solutions circulate continually. Convergence is said to occur if and when a persistent solution appears.

4. A-TEAMS OVERVIEW

The Asynchronous Teams (A-Teams) was first introduced by Sarosh Talukdar from the Carnegie Melon University - CMU/USA, Talukdar et al. (1990, 1992). A-Teams, as defined by Talukdar, are open, high-performance organizations for solving difficult problems, particularly, problems from the areas of planning, design, scheduling and real-time-control.

A-Teams use agents to optimize a set of solutions, called population of solutions. Each population is actually a memory, where a set of solutions is stored. The main characteristics of A-Teams are:

Autonomy of agents – the agents are completely independent of each other,

Cyclical Data Flow – the cyclical data flow gives the possibility to share the results between the agents and therefore allows the cooperation between them in order to get better solutions and

Asynchronous communication – agents read and write in memories without any synchronization between them.

Memories in A-Teams represent a population of solutions of a single type. In the proposed architecture there are two types of memories: complete solution memory and partial solution memory. The first one stores only admissible solutions to the problem under investigation. The solutions in this memory are stored following the makespan value in order to facilitate the agents processing, once all agents use the same performance criterion. The partial solution memory may contain solutions with different lengths and usually they have some idle positions. As it is not possible to calculate the cost function (makespan) of partial and not complete solutions, the selection process is based on the FIFO (First-In-First-Out) policy, and the memory works as a solution buffer.

Each agent of an A-Teams encapsulates a particular problem-solving method along with the methods to modify existent solutions. Agents can be of three types: constructors, improvers and destructors. Constructors create initial solutions and add them to the population (main memory with complete solutions). Improvers select one or more existent solutions from the population and produce new solutions that are added to the population. Finally, destroyers keep the size of the population of solutions in check and focus the efforts of the improvers by removing bad solutions, see Murthy et al. (1997). More details can be obtained also in Passos et al. (2003, 2004).

5. THE PROPOSED A-TEAM

In the proposed architecture there are two memories – one for complete solutions and another for partial solutions, respectively the main and partial memories –, a destroyer agent (D), a constructor agent (R), improver agents that read and write at the main memory – TS and GA and an additional mechanism combing two agents acting between the main memory and the partial memory (C and Cheap-NEH). The C and Cheap-NEH agents act between the two memories creating a cycle of de-constructing and constructing of feasible solutions.

Figure 1 shows the data flow structure used in the proposed A-Team where agents are represented by large arrows and memories by rectangles.

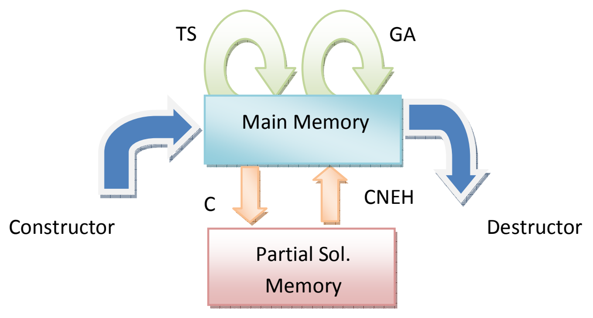

Figure 1 – Multiagent architecture

Tabu search is a higher level heuristic algorithm for solving combinatorial optimization problems. It is an iterative improvement procedure that starts from any initial solution and attempts to determine a better solution. TS was proposed in its present form by Glover (2002). It has now become an established optimization approach that is rapidly spreading to many new fields and applications. Together with other heuristic search algorithms such as GA, TS has been singled out as extremely promising for the future treatment of practical applications, as in the case of scheduling problems. Generally, TS is characterized by its ability to avoid entrapment in local optimal solution and prevent cycling by using flexible memory of search history.

The Tabu Search algorithm can be described in steps as follows:

```
Select an initial solution S, set this solution as the current solution (S*) as well as the best solution (F*): S*=S and F*=F(S);
Repeat until the stop criteria is true
  Find a solution S' belonging to the neighborhood of S (NS), where F(S*) is a minimal in NS;
IF the move from S to S' is not Tabu
Else If F(S') < F*,
S*=S and F*=F(S);
End Else
Do S=S' and update the Tabu List
End If
Else the best solution in NS will be fixed.
End-Repeat
```

Some basic definitions on Tabu Search are the following:

Stop Criteria: these are the conditions under which the search process will terminate. In this study, the search will terminate if one of the following criteria is satisfied: i) the number of iterations since the last change of the best solution is greater than a pre-specified number; or, ii) the number of iterations reaches the maximum allowable and predefined number.

Tabu restrictions: there are certain conditions imposed on moves that make some of them forbidden. These forbidden moves are related to a certain size and known as tabu, called the tabu list. The reason behind classifying a certain move as forbidden is basically to prevent cycling and avoid returning to the local optimum already visited. The tabu list size plays a great role in the search of high quality solutions. Generally, the tabu list size should grow with the size of the given problem. In our case we established a relation with the number of the problem´s operations.

The Genetic Algorithms, other improving agent, were first introduced by Holland (1975), in order to explain the development of artificial systems that retain the natural mechanisms of adaptation. Interacting with the A-Team, the GA incorporates in its initial population some complete solutions from the main memory and writes its best result found in that memory when its evolution process is finished. The solutions that will be incorporated by the GA in the main memory are defined by its reading policy.

The genetic algorithm used in this work can be described in the following way:

```
Initialize the chromosome population with solutions present in the main memory.
For each member of the population:
Repeat until the stop criteria is true
Evaluate each chromosome (makespan calculation);
Select and apply the operators: crossing and mutation.
End-Repeat.
Return the best chromosome and insert it in the main memory.
```

The Consensus Agent, represented by the C arrow in figure 1 and experienced in the former version of the algorithm to solve flow shop problems, see Passos et al (2003), are those that generate a new solution from two or more existent solutions, using some criteria that identify potential fragments present in the optimal solution being searched. The Cheap-NEH Agent is a non-deterministic construction heuristic. It acts at the partial solutions constructing complete solutions by adding new tasks to void positions. It is a derivation from the NEH heuristic proposed by Nawaz, Enscore and Ham, Nawaz et al. (1983) to the flow shop permutation scheduling problem. This additional mechanism, composed of the C and CNEH agents and the partial solution memory, will also be tested in the jobshop version of the algorithm, in order to verify if it can collaborate with the other agents to finding better solutions.

6. EXPERIMENTAL RESULTS

The main objective of this research is developing a solution to be applied to solve large size jobshop scheduling problems, existent in practical cases. The planners when solving these kinds of problems usually do not have the support of tools that build solutions automatically. In such cases, find a feasible solution is already a gain and find optimal solution is only a mirage. Thus one of the important aspects is keep the solution time under control. In several situations it is necessary get new plans, or even review an existent, within time constraints, generally few minutes. In this research the search parameters and the stop criteria were adjusted in order to keep the time under control, but without over penalizing the performance criterion.

As there is no large size jobshop scheduling benchmark problems available in the literature (with 1,000 or more operations) it is not possible to perform a realistic comparison between this proposal and other approaches. A comparison with problems instances obtained from the ORLIB (short for OR-Library), originally introduced by Beasley (1990), will be performed to demonstrate the behavior of the proposal.

6.1 Parameterization for the GA

Some experiments were performed in order to adjust the search parameterization for the algorithms. Figure 2 shows some test results from GA parameterization from a 15X15 size JSP from the first Taillard instance, Taillard (1993). These results were obtained in the very beginning of our experimentation and give a good idea how to choose the best values of the parameters. The graphs obtained show that the results are similar with those from the GA literature. The graph 1 (Figure 2) was obtained with fixed values to mutation, population size and iterations number, and varying crossover probability.

The graph 2 (Figure 3) has fixed values to the crossover probability, population size and iteration number with the variation of mutation probability. The graph 3 (Figure 4) has fixed values to the crossover probability, population size and mutation probability with the variation of iteration number. The graph 4 (Figure 5) has fixed values to the crossover probability, iteration number and mutation probability with the variation of population size.

Similar studies were also performed to the definition of the best values to be used in the tabu search algorithm. It is important to note that in practical situations specific parameterization studies with the real data must be performed in order to have a correct adjustment of the algorithm.

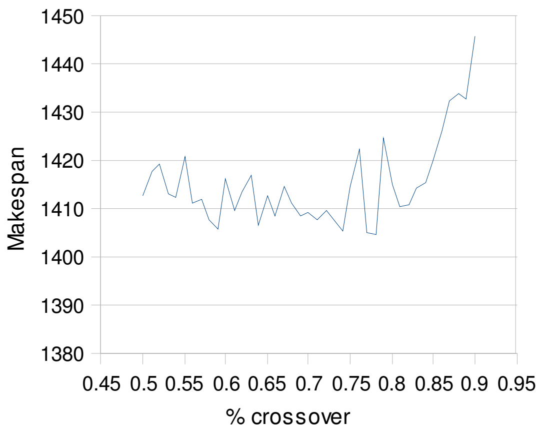

Figure 2 – Graph 1 – Makespan versus Crossover.

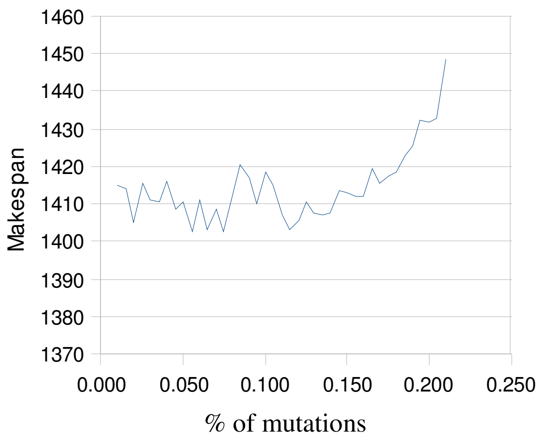

Figure 3 – Graph 2 - Makespan versus Mutation.

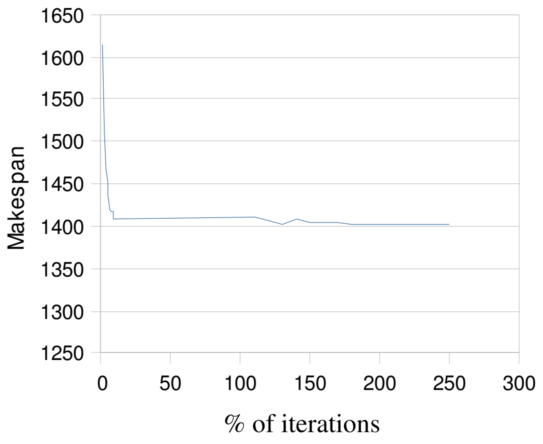

Figure 4 – Graph 3 - Makespan versus Iterations.

Table 1. Instances of the jobshop scheduling problem

Problem	Number of Jobs	Number of Machines	Number of Operations	Optimum value	Makespan obtained	Initial Makespan	Gap
la01	10	5	50	666	666	740	0
la02	10	5	50	655	655	816	0
la03	10	5	50	597	597	776	0
la04	10	5	50	590	590	704	0
la05	10	5	50	593	593	593	0
la06	15	5	75	926	926	966	0
la07	15	5	75	890	890	1068	0
la08	15	5	75	863	863	993	0
la09	15	5	75	951	951	981	0
la10	15	5	75	958	958	958	0
la11	20	5	100	1222	1222	1260	0
la12	20	5	100	1039	1039	1118	0
la13	20	5	100	1150	1150	1184	0
la14	20	5	100	1292	1292	1292	0
la15	20	5	100	1207	1207	1494	0
la16	10	10	100	945	946	1144	0,11
la17	10	10	100	784	784	870	0
la18	10	10	100	848	848	970	0
la19	10	10	100	842	842	1057	0
la20	10	10	100	902	907	1155	0,55
la21	15	10	150	1046	1048	1408	0,19
la22	15	10	150	927	934	1269	0,76
la23	15	10	150	1032	1032	1252	0
la24	15	10	150	935	943	1236	0,86
la25	15	10	150	977	986	1363	0,92
la26	15	10	150	1218	1218	1602	0
la27	20	10	200	1235	1242	1694	0,57
la28	20	10	200	1216	1216	1633	0
la29	20	10	200	(1142 1153)*	1181	1608	3,42
la30	20	10	200	1355	1355	1648	0
la31	30	10	300	1784	1784	2114	0
la32	30	10	300	1850	1850	2238	0
la33	30	10	300	1719	1719	2049	0
la34	30	10	300	1721	1721	2071	0
la35	30	10	300	1888	1888	2498	0
la36	15	15	225	1268	1292	1603	1,89
la37	15	15	225	1397	1418	1935	1,5
la38	15	15	225	1196	1221	1597	2,09
la39	15	15	225	1233	1258	1692	2,03
la40	15	15	225	1222	1233	1679	0,9

Legend: * instance la 29 - (lower bound, upper bound)

6.2 General performance tests

The algorithm has been tested on 40 problems instances provided by the ORLIB. They have different sizes and hardiness level. Table I shows the problem instances and results obtained by the proposed algorithm. The columns represent the problem size (number of jobs, machines and operations), the optimum known value for the instances obtained from Pezzella (2000), the final and initial makespan values and the gap between the optimum and final makespan in % values. The initial value was obtained by the initialization algorithm that generates solutions randomically.

Forty Lawrence instances, problems - la01 to la40 - were tested, varying from 50 to 300 operations. The majority of them, 27/40 or 67.5 %, have the gap equal to 0 meaning that the algorithm got the optimum value. 10/40 or 25 % have the gap less than 1 % and the remaining, 3/40 or 7.5 %, has a gap between 1.0 and 3.5. All of them are excellent values because a gap less than 4 % is very acceptable for heuristics algorithms. With more refinements in the parameterization and stop criteria values it will be possible to improve even more these results.

The good results obtained by the solution here presented show that proposed solution is very interesting to solve the jobshop scheduling problem. Further studies are currently being conducted to improve the algorithms performance as a whole, considering the makespan value and the solution time. The C and CNEH agents already experimented for flow shop problems, when implemented for the jobshop version will open new possibilities to improve also the overall performance.

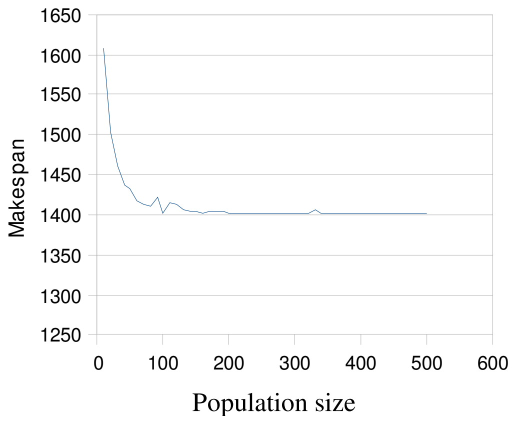

Figure 5 – Graph 4 – Makespan versus Population Size.

7. CONCLUSIONS

Considering that in the practical situations there are a lot of large size jobshop scheduling problems and that the planners usually do not have the aid of tools that generate solutions automatically to them, it is very important that new approaches, such as that presented here, must be investigated.

Multi-agent system using TS and GA to solve job shop scheduling problem provides a rich platform for the constrained combinatorial optimization problems. The main advantage of this kind of approach is the possibility of combining different types of algorithms, which cooperate to solve the problem, so that they can do what they could not do separately.

As results from optimization or single heuristics methods are really unpredictable and time consuming, especially in large size problems, it is natural that the planners are pleased with non-optimal results. Thus, the utilization of metaheuristics with some degree of sophistication, like the multi-agent approach proposed in this research, is an excellent option.

Analyzing the results it is possible to affirm that the overall algorithm performance is very good and also that it can be used to solve practical problems. The data input can be performed in a very simple way and the adjustment for practical cases can be performed by the planners without great difficulty. It is important to note that in practical situation the number of jobs and machines are known a priori, because industrial companies have a finite number of products to be produced and the number of machines is fixed for a determined period. That greatly facilitates the adjustment of the program.

For further research, the technique used in this study will be applied to solve larger size problems and refinements will be also conducted, especially in the search parameterization and in the stop criteria of the algorithms aiming to increase the overall performance of the solution here presented.

Acknowledgement: We acknowledge the PIBIC/CNPq which provided undergraduate scholarships for the students that participate in the development of this work.

REFERENCES

Baker, K.R. (1974). Introduction to Sequencing and Scheduling. John Wiley and Sons, Inc., New York.

Beasley, J.E. (1990). OR-Library: distributing test problems by electronic mail, Journal of the Operational Research Society 41(11), pp1069-1072.

French, S. (1982). Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop. John Willey & Sons Inc. New York, USA.

Glover, F.W., Kochenberger, G.A. (2002). Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, MA, USA

Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

Murthy, S., Rachlin, J., Akkiraju, R., Wu, F. (1997). Agent-Based Cooperative Scheduling; In Constraints & Agents, Technical Report WS 1997-97-05, Menlo Park, AAAI Press.

Nawaz, M., Enscore, E. E., Ham, I., "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem". OMEGA, Int. J. Of Management Science, 11(1): 91-95.

ORLIB - http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/ jobshop1.txt

Passos, C.A.S., Fonseca, S.L.A. (2003). Scheduling of jobs in chemical process industries using metaheuristics approaches. International Conference on Industrial Logistics - ICIL'2003 16, Vaasa, Finland.

Passos, C.A.S., Fonseca, S.L.A (2004). Scheduling of Industrial Jobs Using An A-Team Approach. The Symposium on Professional Practice in AI, 18th IFIP World Computer Congress, Toulouse, France, ISBN 2-907801-05-8, p 1 –10.

Pezzella, F., Merelli, E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem – European Journal of Operational Research. 120, p 297-310.

Taillard, E. (1993). Benchmarks for basic scheduling problems. European Journal of Operational Research, p. 278-285.

Talukdar, S.N., Souza, P.S. (1992). Scale efficient organizations, IEEE Int. Conference on Systems, Man and Cybernetics.

Talukdar, S.N.; Souza, P.S. (1990). Asynchronous Teams. Second SIAM Conference on Linear Algebra: Signals, Systems and Control.

Wooldridge, M. (2002). An Introduction to MultiAgent Systems. John Wiley & Sons Ltd, paperback, 366 pages, ISBN 0-471-49691-X.

Yamada, T., Nakano, R. (1997). Genetic Algorithms for Job-shop Scheduling Problems. Proceedings of Modern Heuristic for Decision Support. Pp. 67-81, UNICOM Seminar, 18-19 March 1997, London.