Tabu search and genetic algorithms: a comparative study between pure and hybrid agents in an A-teams approach

Carlos A. S. Passos (CenPRA) carlos.passos@cenpra.gov.br
Daniel M. Aquino (UNICAMP, PIBIC/CNPq) dmadruga@gmail.com

Summary
This paper presents a comparative study between pure and hybrid agents in an A-Teams approach. The A-Teams architecture is intended to solve scheduling problems. The study is based on Tabu Search (TS) and Genetic Algorithms (GA) meta-heuristics. The pure approach implements classical TS and GA scheduling algorithms. The hybrid approach is based on the technique proposed by Zdanski & Pozivil (2002), where the TS and GA are encapsulated in a single algorithm. These algorithms are implemented as agents in the A-Teams architecture. Two different data sets are used to compare the pure and hybrid algorithms’ implementations. The first one, is based on data collected from a practical industrial case and, the second, is based on a set of benchmark instances, originally proposed by Taillard (1993). The main contribution of this paper is to demonstrate the behavior of A-Teams architecture using hybrid algorithms in comparison to one using pure algorithm for solving scheduling problems.

Keywords: Scheduling, A-Teams, Tabu Search, Genetic Algorithms and Hybrid Agents.

1. Introduction
This paper presents a comparative study between two approaches toward solving flowshop scheduling problems, which was initially motivated by the large use of flowshop production model in chemical process industries. Both approaches make use of heuristic techniques to build agents for an A-Teams architecture. Owing to this, the major concern of the here described research is to analyze the behavior of these agents and their contribution to the quality and speed of problem solutions convergence.

The flowshop model is basically a unidirectional flow of products through the production facilities. In other words, the product flows through the plant in the same direction, eventually a product can “jump” a production stage, but keeping the same direction (PASSOS & NAZARETH, 2002). The main goal for a flowshop problem solver is the deployment of an optimal schedule for N tasks on M machines. This is proved to be a NP-hard problem Garey et al. (1979), and hence the various approaches, tools, and techniques to solve practical flowshop related scheduling problems are often heuristics-based.

A-Teams stands for Asynchronous Teams and was initially proposed by Sarosh Talukdar in Talukdar & Souza (1990 & 1992) as an open, high-performance organization for solving difficult problems, particularly, problems from the areas of planning, design, scheduling and real-time control. It is based on the use of autonomous agents that share memories containing a population of trial-solutions to a part of the problem-to-be-solved.

In this work, the concept of meta-heuristic is used as a general algorithmic framework applied to the flowshop problem. Definitions for classical heuristic methods were implemented according to the solver necessities and formed the base for A-Teams agents’ creation.
The two approaches compared in this study are based in the widely used meta-heuristics Tabu Search (TS) and Genetic Algorithms (GA), which are combined in a synergetic way in an A-Teams architecture. The first one makes use of A-Teams agents based on the two meta-heuristics working separately. Thus, in spite of sharing the A-Teams common solutions memory they do not have direct access to each other’s output during its execution and a set of solutions being improved are rarely processed sequentially by the two agents. The second approach is based on the technique proposed by Zdansk & Pozivil (2002), where the TS and GA are encapsulated in a single algorithm. In this approach this hybrid algorithm is the core of a third agent introduced in the used A-Teams architecture.

2. Flowshop Scheduling Problems

Flowshop scheduling, first considered by Johnson (1954), has been one of the most extensively studied problems in scheduling research. A formal definition of this problem is presented in (BAKER, 1974). The considered flowshop model in this research consists in scheduling N tasks (i = 1...N) on M machines (j = 1...N). A task consists in M operations linked by precedence constraints. The order in which operations are executed is the same for all products because these products belong to a common family and because links between units cannot be changed easily (PASSOS & NAZARETH, 2002). Each task shares the utilization of common resources that are necessary for certain operations to take place. These resources have, depending on its availability, constraints in its offer that should be respected when the operations are scheduled.

For each operation there is a known associated processing time p_{ij} for the task i to be executed on machine j. The objective function to be minimized, respecting the availability of machines and shared resources, and the precedence constraints, is the overall completion time for all the tasks, known as makespan.

Both optimizing and heuristic techniques have been used as a solution methodology for flowshop problems. The optimizing techniques used so far are mostly branch and bound and mixed integer programming see Zdansk & Pozivil (2002) but, as long as this scheduling problem is NP-hard see Garey et al., (1976), the search for an optimal solution is more a theoretical than a practical question. Hence, since the sixties many heuristics approaches have been formulated for scheduling of jobs in industrial environments and some of them find good solutions for this kind of problems.

3. A-Teams overview

The Asynchronous Teams (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.
3.1. Memories

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 (≤ N) and normally they have some idle positions. As it is not possible to calculate the cost function (makespan) of partial and non-complete solutions, the selection process is based on the FIFO (First-In-First-Out) policy, and the memory works as a solution buffer.

3.2. Agents

A-Teams use agents to optimize a population of solutions and cooperate by sharing access to populations of candidate solutions. Each agent 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 (MURTHY et al., 1997).

4. The proposed A-Team

This section presents the A-Team architecture proposed, describes the data flow structure and presents the improvers and destructors agents used in the implementation. The initial population is generated randomly. In the proposed algorithm there are two memories – one for complete solutions and another for partial solutions, called respectively the main and partial memories –, a destroyer agent (D), a constructor agent (R), improver agents that read and write at the main memory – Tabu Search (TS), Genetic Algorithm (GA), Hybrid Agent (HA: TS + 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.

In conformance with this pattern it was proposed two different A-Teams architecture for comparison purposes. The only difference between them is the hybrid agent that was introduced in the second one. Figures 1 and 2 show the data flow structure used in both proposed A-Teams, where agents are represented by arrows and memories by rectangles.

5. Tabu Search and Genetic Algorithms Description

This section presents a brief introduction about the two main meta-heuristics on which the implemented A-Teams agents were based, the Tabu Search and the Genetic Algorithms.
5.1. Tabu Search

Glover, in Glover (1986), describes Tabu search as neighborhood search with a list of recent search positions. The term Tabu comes from the fact that the recent visited positions may not be repeated while in the active list. During the search, several moves are performed in order to have an improvement in the solution. This process is repeated until the stop criteria have been met.

The stop criteria adopted and details about the implementation of this agent can be found in Passos & Nazareth (2002). In the A-Team algorithm here proposed the TS algorithm, like the GA, acts as an improver agent working on the population of complete solutions stored in the main memory. It reads a single solution from the memory to use as its initial solution.

Considering that the TS is a deterministic agent, in all reading protocol it is not permitted to the same solution to be read twice. The following policies apply: i) Select solutions from the best one to the worst with linear probability of choice (the choice probability is inversely proportional to the makespan); ii) Select solutions from the worst one to the best with linear probability of choice (the choice probability is proportional to the makespan) and iii) Select the best solution never read before.

5.2. Genetic Algorithm

The genetic algorithms were first introduced by Holland in 1975, see 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.

Considering that the GA is an agent with non-deterministic behavior, the following reading policies were proposed: i) Select the best solution from the memory based on the makespan value; ii) Select randomly a set of solutions; iii) Select solutions from the best one to the worst with linear probability of choice (the choice probability is inversely proportional to the makespan) and iv) Select solutions from the worst one to the best with linear probability of choice (the choice probability is proportional to the makespan).

The details about the implementation of this agent can be found in (Passos & Fonseca, 2003). In the A-Team algorithm here proposed the GA algorithm acts as an improver agent working on the population of complete solutions stored in the main memory.

6. Hybrid Algorithms

The hybrid algorithm implemented was based on the TS-GA hybrid model proposed by Zdansk & Pozivil in 2002. This model is intended to combine the TS and GA in a single algorithm that firstly creates a set of random valid solutions, and for several iterations it optimizes them using a TS-based method (ZDANSK & POZIVIL, 2002). Afterwards, it takes this set of optimized solutions as the initial population for the GA, and iterates until the adopted stop criteria have been met.

For the implemented hybrid agent, the TS output is not written in the main memory, as the TS pure agent, but it is stored in a transition memory, which has an interface with the module that implements the GA. After the stop criteria for the TS have been met, the GA module uses the solutions in this memory as initial population and performs iterations until the stop criteria have been met. At last, the best solution produced by the GA is written to the main memory. This process is illustrated in Figure 3.

For the implementation of the hybrid A-Teams agent some adaptations were necessary. Firstly, instead of creating another set of random solutions for the TS, the algorithm retrieves one valid solution from the A-Teams main memory. It is reasonable since there is a set of available solutions randomly-generated and already improved concerning makespan. Secondly, the initial population for the GA with N individuals is created not based on N TS executions, once it would take more time to be completed than the maximum reasonable amount of time to be spent by the agent. For this reason, the TS is processed only once, and instead of inserting only the final result of the TS in the transition memory, the solution achieved after each iteration is tested against the last solution found, and whether it is better it is written to the transition memory. In such way it is possible to create a complete set of initial solutions for the GA running the TS only once.
The third adaptation concerns the initial number of individuals created by the TS for the GA. Since the TS is run only once, and the stop criteria is respected, there are no guarantees that the minimum number of individuals necessary to GA to take place will be created by TS. Thus, the GA is configured to introduce random solutions in the initial set to fulfill the minimums population size requirements. After these adaptations, the GA runs normally and its output is written to the main memory.

The idea behind this approach is to combine the advantages of the two algorithms and mitigates the disadvantages. TS provides better capabilities concerning local optimization if compared to GA, but it can easily miss some promising areas of the search space. Submitting a set of solution to the TS and then to the GA firstly result in some locally optimized solutions by the TS, and then the GA promotes information exchange among the solutions, and more globally optimized solutions are found.

7. Experimental Results

The first set of tests performed a straight comparison between the pure and the hybrid A-Team’s performances, in order to measure the impact caused by the introduction of the hybrid agent in the former pure A-Teams.

These tests made use of large instances, with makespan varying from 600 to 1000 units of time. The number of iterations for the first case was choose based on analyses performed with benchmark instances which showed that the rate of improvement to solutions found after 5 A-Teams iterations is quite slow. In order to have a deep test strategy, a second test case with 20 iteration, 300% higher than the first, was also performed.

The results for these tests are shown in Figures 4 and 5. The introduction of the hybrid agent did not bring any performance enhancement to the A-Teams, since the achieved results were almost the same for both configurations.

The second set of tests aims to compare the two A-Teams architectures, with pure and hybrid agents, and the hybrid classical algorithm working stand-alone. It was used a set of benchmark instances proposed by Taillard (1993) and available at ORLIB’s (Operational Research Library) website at http://mscmga.ms.ic.ac.uk/info.html. These benchmarks allowed an overall performance analysis for the three tested approaches, the pure A-Teams, the hybrid A-Teams and the stand-alone hybrid agent, since the achieved results could be compared to the optimal solutions.
The results found for this set of tests are illustrated in Figure 6. A set of 100 benchmark instances were used as input data, and the output was ordered according to makespan, from the lowest to the highest. The lines for the pure A-Teams, the hybrid A-Teams and the optimal solutions are overlapped between the majority of the points in the graph.

Once more, with this second set of tests the results for the pure and for the hybrid A-Teams were almost the same. Furthermore, these results were, for the very most of cases, equal or not more than 2 units of time higher, concerning the found makespan, than the optimal results. It shows that the used A-Teams delivery good results as a problem-solving approach to flowshop scheduling problems.

![Figure 5 – Hybrid x Pure A-Teams for 20 iterations: Performance comparison for large instances](image)

The stand-alone hybrid agent did not present as good results as the ones provided by the A-Teams. Its behavior was not regular. Some solutions found were very close to the optimal line, but many others have an expressive increase in makespan.
8. Conclusions and Future Work

The first meaningful conclusion that can be drawn from the analysis of the above presented data is that the A-Teams architecture provide a better way to combine the advantages of the two used meta-heuristics than the method stated by Zdansk & Pozivil (2002) for solving the flowshop scheduling problem, since the convergence of the solutions toward a lower bound is faster with the use of the A-Teams than with the use of the stand-alone hybrid TS-GA algorithm.

The second important conclusion is that, instead of presenting better results than the TS or GA agents working stand-alone, the hybrid agent TS-GA does not offer any additional performance enhancement in terms of makespan cost-function when introduced as an agent in the A-Teams that already counts with the two agents based on the two meta-heuristics separately. It appears to be a consequence of the fact that the A-Teams structure already combines the two pure agents in such a way that the improvements achieved by the hybrid configuration are already obtained. Even in terms of execution time, the A-Teams with pure algorithms have better performance than the hybrid approach.

Since the A-Teams architecture makes use of autonomous agents that can be added or removed without any supervisory system getting in the way, the introduction of other agents in the developed A-Teams structure is straightforward. Consequently, the research and development of new models to serve as base for the implementation of new agents to be introduced in the A-Teams is certainly a feasible way to enhance the performance of the
current system. It is assumed to be a good continuation for this research.

A-Teams architecture allows the agents to work asynchronously (each at its own speed) and in parallel. Therefore, porting the developed system to work as a distributed system is a reasonable task and offers guarantees of performance enhancement since the processing for each agent can be done by a dedicated processor unit.

References


