Integrating CBR and Heuristic Search to Solve Complex Real-Time Scheduling Problems

Juan Manuel Adán Coello* **and Ronaldo Camilo dos Santos****

* Instituto de Informática, PUC-Campinas, Cx.P. 317, CEP 13.020-904, Campinas, SP, BRAZIL
juan@zeus.puccamp.br
** Faculdade de Engenharia Elétrica e de Computação, UNICAMP
ronaldoc@dca.fee.unicamp.br

Abstract

This paper presents the Case-Based Reasoning Real-Time Scheduler system (CBR-RTS), that integrates into a case-based reasoning framework a heuristic search component to schedule complex real-time tasks. The problem addressed involves scheduling sets of tasks with precedence, ready time and deadline constraints. CBR-RTS uses the solution of known cases to simplify and solve new problems. When the systems does not have applicable cases, the new problem, complete or already simplified, is passed to a learning algorithm that searches for a solution using a dedicated algorithm, currently an implementation of a searching algorithm proposed by Xu and Parnas. A particularly interesting feature of CBR-RTS is its learning ability. New problems solved by the learning module can be added to the case base for future reuse. Performed tests have shown that small bases of cases carefully chosen allow to substantially reduce the time needed to solve new complex problems.

Introduction

Most scheduling problems of practical interest are NP-complete (Blazewicz, Lenstra and Kan 1983) and are usually solved using heuristic search methods. An inherent characteristic of traditional schedulers based on heuristic search is its lack of ability to learn from experience. New problems identical, or very similar, to other problems already solved in the past have to be solved again form first-principles every time they are found. This procedure wastes time and resources usually scarce to deal with problems previously faced, for which it was already found a solution or discovered that the problem can not be solved applying available methods.

This paper present the architecture of the Case-Based Real-Time Scheduling System (CBR-RTS), that integrates into a case-based reasoning framework a heuristic search component to solve complex real-time scheduling problems.

The problem addressed here involves scheduling a set of tasks with precedence relations and timing constraints (ready time, execution time and deadline) on a monoprocessed computer.

CBR-RTS Architecture

In CBR-RTS, scheduling problems are described by acyclic directed labeled graphs, where nodes denote the tasks to be scheduled and arcs the precedence relations between the connected tasks. Nodes have attributes that specify their timing constrains, as shown in figure 1. Although not used in this version, arcs may also have attributes, for example do express the time needed to transmit the results of a computation made by a predecessor node to a successor node in a distributed system (Adán, Magalhães and Ramamaritham, 1995).

Figure 2 presents the structure of CBR-RTS. It consists of a case base (CB), a case retrieval module, a case reusing module and a learning module. The case base stores the description of scheduling problems solved in the past and

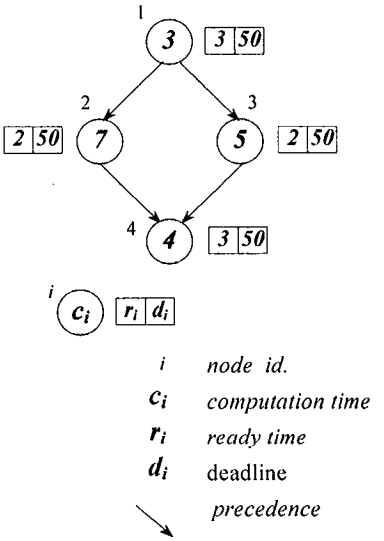

Figure 1: Representation of a Scheduling problem

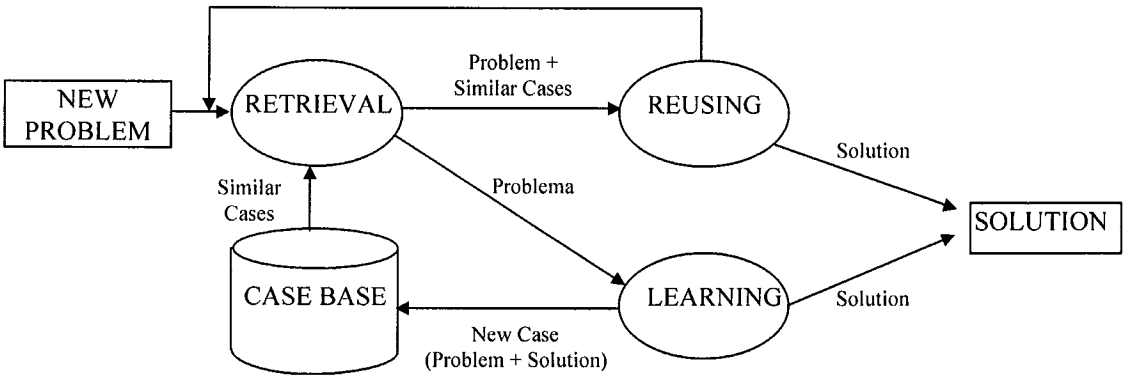

Figure 2: CBR-RTS System Structure

their respective solutions.

The retrieval module is responsible for finding in the case base old problems similar to the new problem or to parts of it. The reusing module employs retrieved cases to solve or simplify the problem. The learning module is responsible for searching for solutions to problems that could not be solved using stored cases.

The retrieval module can retrieve past cases similar to the new problem as a whole or to parts of it (subproblems). In the first situation, the retrieved solutions are adapted to solve the new problem. In the second situation, the retrieved cases are used to simplify the new problem, and new cycles of retrieval and simplification are performed until the complete problem is solved or until it is necessary to use the learning module.

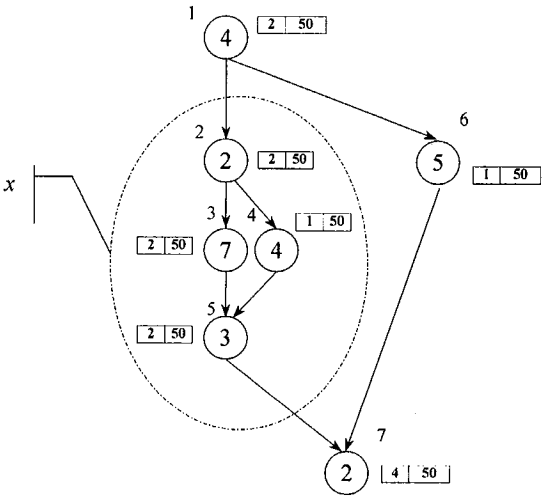

Figure 3: Graph describing a new problem

The Case Retrieval Module

The case retrieval module searches the case base looking for stored cases (old problems) similar to the current problem or to parts of it. A case and a new problem (or subproblem) are similar if they are described by structurally identical graphs and if all tasks in the new problem have timing constraints at least as strict as the corresponding tasks in the old problem. That is, if r_i , c_i ,

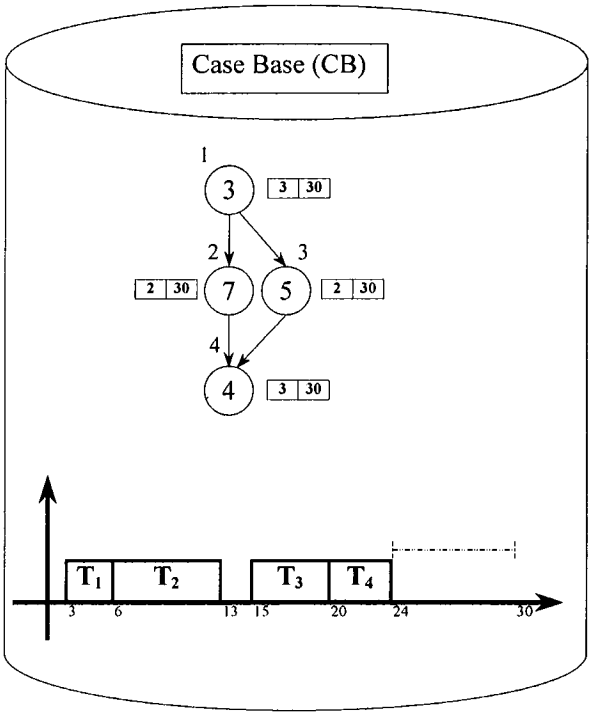

Figure 4: Old problem in the case base similar to subproblem x in new problem

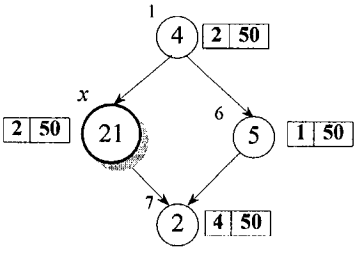

Figure 5: New problem simplified

and d_i are the ready time, computation time and deadline, respectively, of task i in the new problem, and r'_i, c'_i, and d'_i the equivalent times for the corresponding task in the old problem, the following relations have to be observed for all corresponding tasks in both problems:

$r_i \leq r'_i$; $c_i \leq c'_i$ and $d_i \geq d'_i$

The verification of the structural identity of new and old problems is the most complex activity of the retrieval process and involves the detection of subgraph isomorphims. In the current version of CBR-RTS this is done by an implementation of Messmer (1996) of an algorithm proposed by Ullman (1976).

Figure 3 shows an example of a new problem to be solved and figure 4 shows and old case retrieved from the case base that is similar to subproblem x of the new problem.

Case Reusing Module

When the retrieval module finds an old case identical to the new problem, the solution of the old case is integrally reused without any adaptation to solve the new problem. When it retrieves a past case structurally identical to the current problem, but with different ready time, execution time or deadline, the old problem schedule is adapted. Because the retrieved case has timing constraints at least as strict as the new problem, the adaptation consists simply in replacing the execution times of the old problem tasks by the corresponding times of the new problem tasks in the retrieved schedule.

When it is retrieved a case similar to a subproblem of the new problem, the solution to the old problem is adapted to

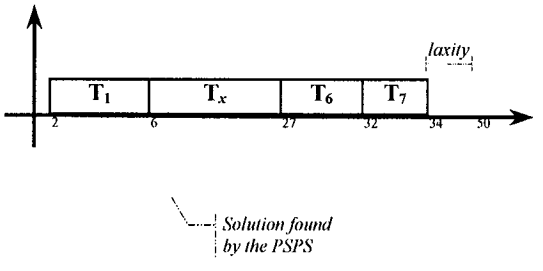

Figure 6: Schedule produced by PSPS for the new problem simplified

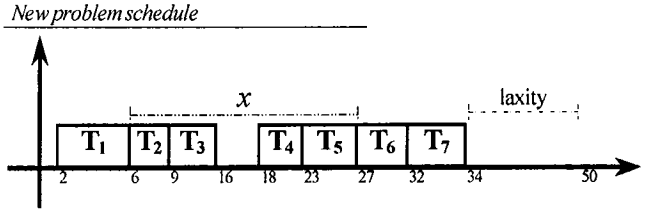

Figure 7: Final schedule for new problem

solve the subproblem, as described above for the complete problem, and the new problem is simplified substituting the subgraph that represents the subproblem by a single node. Then, the system performs new retrieval and reusing cycles to try to solve the simplified problem.

Besides the observation of the structural identity and the timing restrictions described before, to substitute a subproblem by an equivalent task (a single node in the graph) the case reusing module verifies if the subproblem forms a group.

A group is characterized by a graph composed of an entry node, an exit node and internal nodes. In a group, internal nodes and the exit node can have as predecessors only the entry node or other internal nodes, in the same way, the entry node and the internal nodes can have as successors only other internal nodes or the exit node.

Figure 5 shows the new problem or our example (figure 3) simplified using the old case found in the case base (figure 4). We can see that subproblem x was replaced by the equivalent node x..

The Learning Module

When it is not possible to solve a new problem using only past experiences stored in the case base, the system passes the problem to the learning module. The learning module may find a solution to the problem or discover that the problem is not schedulable. The problem description and the solution found (a schedule) or an indication that the problem is not schedulable are stored in the case base. That is, the system can learn solutions for a new type of problem or can learn that this type of problem has no solution. Both lessons are worth remembering.

In the current version of CBR-RTS, the learning module uses a scheduler implemented by Mello Jr. (1993), named PSPS (Periodic and Sporadic Processes Scheduler), based on a branch and bound search algorithm proposed by Xu and Parnas (1990).

In our example, because the simplified version of the new problem, shown in figure 5, has no similar cases in the case based, it is passed to the learning module that produces the schedule shown in figure 6.

Outputting a Problem Solution

When a problem is solved, the schedule produced to the simplified problem is expanded to restore the subproblems

that were transformed into single nodes during the simplification phase. This is done placing the solution to the original subproblems (resulting from the adaptation of the old problems schedules found in the case base) in the time intervals reserved for the corresponding nodes.

In our example, figure 7 shows the final solution produced by CBR-RTS for the new problem presented in figure 3.

CBR-RTS Evaluation

This section discusses the performance of CBR-RTS when applied to solve nine problems of increasing size and complexity. The performance of CBR-RTS in solving these problems is compared with the performance of the PSPS system alone (the heuristic scheduler used by the learning module).

The hypothesis that is being tested on this set of experiments is that the reuse of solved old cases can contribute to reduce the time needed to find feasible schedules for new problems. As our retrieval module involves isomorphism graph detection, a NP-hard problems as is our original scheduling problem, several parameters, for example, the size of each case and of the whole case base, will have a major impact on the performance of the system. In the described experiments we decided to evaluate the behavior of CBR-RTS when working with a case base composed of small cases. This case base will rarely permit to solve new problems in one single step, but could be used to simplify a high number of large problems that will be solved in multiple retrieve-simplify steps.

The Case Base (CB)

As Miyashita and Sycara (1995), we assume that although scheduling is an ill-structured domain, it exhibits some regularities that could be captured in a case. In our context, we assume that scheduling problems tend to present typical structural regularities and attribute values that characterize the main classes of problems handled. Several problems that have identical precedence relations can be represented by graphs that share a common structure. These structures can be automatically learned by the system as it faces new problems.

In order to simulate a situation in which CBR-RTS had already passed for a learning period, fourteen small problems, described by graphs from 2 to 6 nodes, were presented to the system. Since the CB is initially empty, and because the presented problems are not similar to each other, they are completely solved by the learning module and stored in the CB. The structure, ready times and execution times of these problems were chosen in a way that they could be highly reusable in the solution of the testing problems described below.

Testing Problems

The nine testing problems (P_1, P_2, ... P_9) have increasing size and complexity, varying from 5 to 60 nodes with several combinations of precedence relations and timing constraints. The system was able to find a complete solution for all but the 9th problem using only the cases stored in the CB. For P_9, after two retrieve-simplify cycles the system had to use the learning module, because there were no similar cases on the CB. The solution of these problems gives some insight in the performance of the CBR-RTS system with a stable CB, that is, a case base that makes possible to solve most new problems without having to employ the learning module.

CBR-RTS Performance

Table 1 shows the total processing times required by CBR-RTS and PSPS for solving problems P_1 to P_9 in a 167 MHz Sun Ultrasparc 1 workstation with 64 MB of RAM. We can see that as problem size and complexity increases there is also a sensible increase in the relative performance of CBR-RTS compared with PSPS, as the consequence of reusing past solutions.

Table 1 also shows CBR-RTS processing times by phase. As expected, we can see that the retrieval phase accounts for most of the processing time of CBR-RTS, indicating that this is an important point to be focused in future work.

The solution of problem P_9 is an example of a situation where the system is learning the solution for a new type of problem. In this example, CBR-RTS can not find a final solution to the problem using only stored cases, but it is able to simplify the original problem. The simplified problem is submitted to PSPS (the learning module of the system) that finds a solution in approximately 30% of the time it will require to solve the original problem.

	PSPS	CBR-RTS			
Problem	Total Time	Total Time	Retrieval	Reusing	Learning
1	0.10	0.25	0.24	0.01	0
2	0.07	0.51	0.42	0.09	0
3	0.15	0.28	0.26	0.02	0
4	0.16	0.40	0.36	0.03	0
5	0.29	0.93	0.76	0.17	0
6	0.62	0.58	0.51	0.06	0
7	1.77	1.03	0.91	0.11	0
8	6.99	4.60	3.90	0.69	0
9	4.66	2.49	0.94	0.11	1.43

Table 1 *CBR-RTS and PSPS processing times (seconds)*

Related Work

In this section we will discuss systems that adopt integrations strategies similar to ours, particularly in the domain of scheduling, and some systems, as Casey (Koton

1988), that in some degree have inspired the design of CBR-RTS.

Cunningham and Smyth (1996) explore solution reuse in job scheduling. Cases represent highly optimized structures in the problem space, produced using simulated annealing. They address single machine problems where job setup time is sequence dependent (an example of a non Euclidean Traveling Salesman Problem). Their objective is to produce good quality schedules in very quick time. Although we share the same conceptual framework, our work differs in a number of ways. We address distinct types of scheduling problems and we employ different case representations and retrieval and reusing strategies that seem to make our approach amenable for a wider category of scheduling scenarios.

Other systems also combine CBR with some other strategy to solve scheduling problems. CABINS (Miyashita and Sycara 1995), for example, integrates CBR and fine granularity constraint-directed scheduling. CABINS constructs cheap but suboptimal schedules that are incrementally repaired to meet optimization objectives based on the user preferences captured in cases. As in CABINS, we also assume that although scheduling is an ill-structured domain, it exhibits some regularities that can be captured in a case.

Some of the basic ideas of the CBR-RTS system can be found in Casey (Koton, P. 1988), a well know example of system that integrates CBR and search. Casey is built on top of a model based program implemented using rules that diagnoses heart defects. The case library is constructed using this rule based program. Casey searches the case library to see if it has old cases that can be used to diagnose a new patient, if no similar cases are found the problem is passed to the rule based program. When know solutions are reused, Casey can be 2 to 3 orders of magnitude more efficient that the rule based program.

PRODIGY/ANALOGY (Veloso 1994) is also a well know system that combines CBR with search in the solution of planning problems. The case library is seeded by cases solved by a general problem solver, based in a combination of means-ends analysis, backward chaining and state space search. Cases permit to acquire operational knowledge that can be used to guide the generation of solutions for new problems, avoiding a completely new search effort.

Although most CBR systems use flat representations in the form of attribute-value pairs, the issues raised by structured representations, as the graphs used in CBR-RTS, have been addressed by several authors. The interested reader can find more details in (Bunke and Messmer 1994), (Messmer 1996) and (Gebhardt 1995).

Conclusions

The experiments described in this paper suggest that the CBR-RTS system, based on the integration of CBR with heuristic search, can contribute to an expressive reduction in processing times required to schedule complex problems. However, in order to better evaluate the potential and behavior of the system, and the subjacent architecture, it must be submitted to a testing procedure with a wider coverage than that provided by the experiments described in this paper.

CBR-RTS has a modular architecture that easily supports evolution. Each component of the systems constitutes itself an interesting research subject.

The current structure of the case base and the corresponding retrieval algorithm seem adequate to case bases storing a moderate number of cases of small size, as the ones considered in the experiments described in this paper. New organizations and retrieval strategies might have to be considered to deal with case bases with a high number of complex cases.

A particularly important problem in real-time systems is to develop deterministic schedulers that can compute schedules in bounded time. Although in the general case (for any new problem) this can not be achieved by the NP-hard nature of scheduling problems, the possibility of doing this in a reasonable amount of situations permits the implementation of some interesting strategies. The use of polynomial time subgraph isomorphism algorithms, as the one proposed by Messmer (1996), to address this issue is an interesting topic that should be considered in our future work.

An interesting extension to the reusing module is to try to employ old cases to solve subproblems structurally identical even when they do not form a group.

The learning module can also evolve in a number of ways, for example with the creation of a library of methods for solving different types of scheduling problems, besides the one currently considered.

The management of the case base is also an interesting theme. The definition of criteria to be used in deciding which new cases to incorporate to the case base is one of the relevant questions to be considered. There are several possibilities here. For example, the case base can be formed only of carefully chosen small cases that permit to simplify a extensive number of large problems, as done in the experiments described here. It could be also interesting to prioritize the memorization of unschedulable problems that require the searching algorithms to spend a lot of time and resources to reach that conclusion.

Currently, the learning module is only used after a problem can not be further simplified. Other integrations between the retrieval and the learning modules are possible and could be interesting to study. For example, instead of trying to find occurrences of stored cases in the problem graph, as done in the current version, the graph could be initially divided into groups and them the system could try to see if these groups are present in the case base. After reusing the best retrieved cases to simplify the problem, the groups for which there were no applicable cases could be scheduled by the learning algorithm before proceeding in the retrieval-simplification process.

Acknowledgments.

This research has been supported in part by grant #1996/11200-3 from Fundação de Amparo à Pesquisa no Estado de São Paulo (FAPESP).

References

Adán, J. M., M. F. Magalhães and K. Ramamritham. (1995). Meeting Hard Real-Time Constraints Using a Client-Server Model of Interaction. *Proc. of the 7Th Euromicro Workshop on Real-Time Systems*, Odense, Denmark.

Bunke, H. and B. T. Messmer. 1994. Similarity Measures for Structured Representations. In Topics in Case-Based Reasoning, S. Wess, K. Althoff and M. Richter (Eds.) *Lecture Notes in Artificial Intelligence*, Springer-Verlag.

Blazewicz, J., J.K. Lenstra and A.H.G.R. Kan. 1983. Scheduling Subject to Resource Constraints: Classification and Complexity. *Discrete Applied Mathematics* 5: 11-24.

Cunningham, P. and B. Smyth. 1996. Case-Based Reasoning in Scheduling: Reusing Solution Components. Technical Report TCD-CS-96-12, Department of Computer Science, Trinity College Dublin, Ireland.

Gebhardt, F. (1995). Methods and systems for case retrieval exploiting the case structure. FABEL report no. 39. GMD. Germany.

Koton, P. 1988. Reasoning about evidence in causal explanation. In *Proceedings of AAAI-88.* Cambridge, MA. AAAI Press/MIT Press.

Melo, Jr., A. 1993. Uma estratégia de escalonamento de processos periódicos e esporádicos em sistemas de tempo real crítico monoprocessados. MSc. Thesis, FEEC, Unicamp.

Miyashita, K., K. Sycara. (1995). CABINS: A framework of Knowledge Acquisition and Iterative Revision for Schedule Improvement and Reactive Repair. CMU Technical Report CMU-RI-TR-94-34. The Robotics Institute, Carnegie Mellon University, USA. Also Artificial Intelligence Journal. Forthcoming.

Messmer, B. T. 1996. Efficient Graph Matching algorithms for Preprocessed Model Graphs. PhD Thesis. Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland.

Ullman, J.R. 1976, An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31-42.

Veloso, M. 1994. PRODIGY/ANALOGY: Analogical Reasoning in General Problem Solving. In Topics in Case-Based Reasoning, S. Wess, K. Althoff and M. Richter (Eds.) *Lecture Notes in Artificial Intelligence*, Springer-Verlag.

Xu, J. and Parnas, D.L. 1990. Scheduling processes with release times, deadlines, precedence, and exclusion relations. *IEEE Transactions on Software Engineering*, 16(3):360-369.

Appendix A

1. **Integration name/category:** CBR-RTS
2. **Performance Task:** Scheduling sets of tasks with precedence, ready time and deadline constraints
3. **Integration Objective:** Reduce the CPU time needed to find a schedule or find out that the problems is unschedulable
4. **Reasoning Components:** Heuristic search
5. **Control Architecture:** CBR as master
6. **CBR Cycle Step(s) Supported:** Learning (heuristic search is used to seed the case base and to learn how to solve new types of problems)
7. **Representations:** Acyclic labeled graphs
8. **Additional Components:** Subgraph isomorphism detection algorithm for case retrieval
9. **Integration Status:** Empirically Evaluated (initial results)
10. **Priority future work:** Empirical evaluation