Case-based Acquisition of User Preferences for Solution Improvement in Ill-Structured Domains

Katia Sycara
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
katia@cs.cmu.edu

Kazuo Miyashita
Production Engineering Division
Matsushita Electric Industrial Co.
Kadoma, Osaka 571, Japan
miyasita@mcec.ped.mei.co.jp

Abstract

[1] We have developed an approach to acquire complicated user optimization criteria and use them to guide iterative solution improvement. The effectiveness of the approach was tested on job shop scheduling problems. The ill-structuredness of the domain and the desired optimization objectives in real-life problems, such as factory scheduling, makes the problems difficult to formalize and costly to solve. Current optimization technology requires explicit global optimization criteria in order to control its search for the optimal solution. But often, a user's optimization preferences are state-dependent and cannot be expressed in terms of a single global optimization criterion. In our approach, the optimization preferences are represented implicitly and extensionally in a case base. Experimental results in job shop scheduling problems support the hypotheses that our approach (1) is capable of capturing diverse user optimization preferences and re-using them to guide solution quality improvement, (2) is robust in the sense that it improves solution quality independent of the method of initial solution generation, and (3) produces high quality solutions, which are comparable with solutions generated by traditional iterative optimization techniques, such as simulated annealing, at much lower computational cost.

Introduction

We present an approach, implemented in the CABINS system, to demonstrate the capability of acquiring user context-dependent optimization preferences and reusing them to guide iterative solution optimization in ill-structured domains. This capability is very important for two main reasons. First, traditional search methods, both Operations Research-based and AI-based, that are used in combinatorial optimization, need explicit representation of objectives in terms of a cost function to be optimized (Reeves 1993). In many practical problems, such as scheduling and design, optimization criteria often involve context- and user-dependent tradeoffs which are impossible to realistically consolidate in a cost function. Second, expert system approaches, while having the potential to capture context-dependent tradeoffs in rules, require considerable knowledge acquisition effort (Prerau 1990). Our approach uses case-based reasoning (CBR) which has been successful in dealing with exceptional data (Golding & Rosenbloom 1991; Ruby & Kibler 1992), acquiring user knowledge in complex domains (Chaturvedi 1993; Mckay, Buzacott, & Safayeni 1988), and expending less effort in knowledge acquisition compared with knowledge acquisition for rule-based systems (Lewis, Minior, & Brown 1991). CABINS acquires, stores and reuses two categories of concepts that reflect user preferences (1) what heuristic local optimization action to choose in a particular context, and (2) what combinations of effects of application of a particular local optimization action constitutes an acceptable or unacceptable outcome. These are recorded in the case base and are used by CABINS to guide iterative optimization and induce optimization tradeoffs to evaluate the current solution. The optimization criteria are not explicitly represented as case features or in terms of a cost function but are implicitly and extensionally represented in the case base.

Previous case-based systems for incremental solution revision (e.g. (Hammond 1989; Veloso 1992)) have been motivated only by concerns of computational efficiency, preserving plan correctness rather than improving plan quality, and have assumed the existence of a strong domain model that provides feedback as to plan correctness. Case-based knowledge acquisition systems, (e.g. (Bareiss 1989)) require causal explanations from an expert teacher to acquire domain knowledge. In our approach neither the user nor the program are assumed to possess causal domain knowledge. The user's expertise lies in his/her ability to perform consistent evaluation of the results of problem solving and impart to the program cases of problem solving experiences and histories of evaluation tradeoffs.

In this paper, we present initial experimental re-

[1] This research was partially supported by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001. Most of the work was performed when the second author was a visiting scientist at the Robotics Institute at Carnegie Mellon University under the support of Matsushita Electric Industrial Co.

sults to test three hypotheses. First, our CBR-based incremental revision methodology shows good potential for capturing user optimization preferences in ill-structured domains, such as job shop scheduling, and re-using them to guide optimization. Second, the method is robust in the sense that it improves solution quality independent of the method of initial solution generation. Third, CABINS produces high quality solutions. To test this, we compared the solutions produced by CABINS with explicit optimization criteria, with solutions produced by simulated annealing (a well known iterative optimization technique (Johnson *et al.* 1991; Zweben, Deale, & Gargan 1990; Laarhoven, Aarts, & Lenstra 1992)) for the same criteria. Our investigation was conducted in the domain of job shop schedule optimization and the experimental results, shown in section confirmed these hypotheses.

Job Shop Schedule Optimization

The job shop scheduling problem is one of the most difficult NP-hard combinatorial optimization problems (French 1982). Job shop scheduling deals with allocation of a limited set of resources to a number of activities (operations) associated with a set of jobs so as to respect given temporal relations (e.g. precedence relations among activities), temporal constraints (e.g. job release and due dates) and resource capacity restrictions in order to optimize a set of objectives, such as minimize tardiness, minimize work in process inventory (WIP), maximize resource utilization etc. Due to the tight interactions among scheduling constraints and the often conflicting nature of optimization criteria, it is impossible to assess with any precision the extent of schedule revision or the impact of a scheduling decision on the global satisfaction of optimization criteria. For example, in figure 1 moving forward the last activity of ORDER3 creates downstream cascading constraint violations. Therefore, a repair action must be applied and its repair outcome must be evaluated in terms of the resulting effects on scheduling objectives. In addition, the evaluation itself of what is a "high quality" schedule is difficult because of the need to balance conflicting objectives and trade-off among them. Such tradeoffs typically reflect user preferences, which are difficult to express as a cost function. For example, WIP and weighted tardiness are not always compatible with each other. As shown in figure 2, there are situations where a repair action can reduce weighted tardiness, but WIP increases. Which is a better schedule depends on user preferences.

CABINS *incrementally revises a complete but sub-optimal schedule* to improve its quality, based on flexible optimization tradeoffs. Revision-based approaches to scheduling have also been investigated by (Minton *et al.* 1990; Zweben, Deale, & Gargan 1990; Biefeld & Cooper 1991; Laarhoven, Aarts, & Lenstra 1992). In those systems, the initial schedule is repaired by several techniques, such as the min-conflict heuristic or simulated annealing, to minimize the number of constraint violations or optimize a simple cost function (e.g. make-span) of the schedule. The value of incorporating context-dependent user preferences in operational scheduling environments is becoming increasingly recognized (e.g. (Mckay, Buzacott, & Safayeni 1988)) but adequate techniques are lacking.

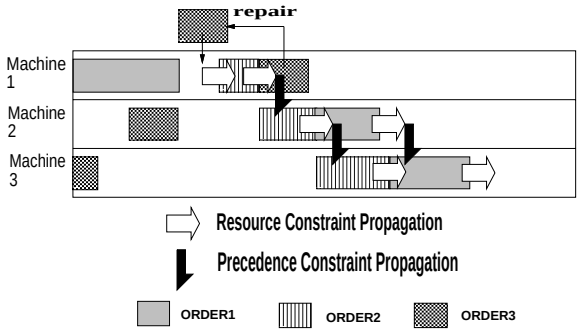

Figure 1: **Example of Tight Constraint Interactions**

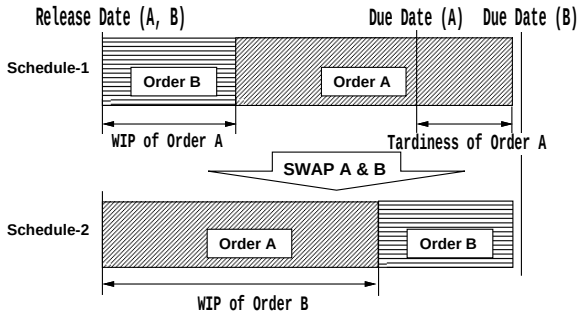

Figure 2: **Example of Conflicting Objectives**

CABINS Overview

CABINS is composed of three modules: (1) an initial schedule builder, (2) an interactive schedule repair (case acquisition) module and (3) an automated schedule repair (case re-use) module. To generate an initial schedule, CABINS can use any of several scheduling methods (e.g. traditional dispatching rules or a constraint-based scheduler).

Case representation

In each repair iteration, CABINS focuses on one activity at a time, the *focal_activity*, and tries to repair it. A case in CABINS describes the application of a particular modification to a focal_activity. Figure 3 shows the information content of a case. Our assumption, borne out by the experimental results, is that despite the ill-structuredness of the domain, the global, local and repair history features express (in an approximate

manner) domain regularities. The global features reflect an abstract characterization of potential repair flexibility for the whole schedule. High 'Resource Utilization Average', for example, often indicates a tight schedule without much repair flexibility. Associated with a focal_activity are local features that we have identified, based on those reported in (Ow, Smith, & Thiriez 1988), and which potentially are predictive of estimating the effects of applying a particular repair tactic to the schedule. For example, 'Predictive Shift Gain' predicts how much overall gain will be achieved by moving the current focal_activity earlier in its time horizon. In particular, it predicts the likely reduction of the focal_activity's waiting time when moved to the left within the repair time horizon.

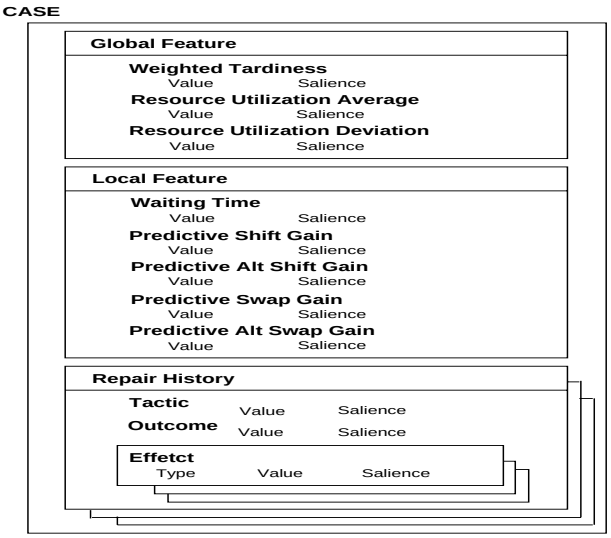

Figure 3: **CABINS Case Representation**

The repair history records the sequence of applications of successive repair tactics, the repair outcome and the effects. Repair effect values describe the impact of the application of a repair action on scheduling objectives (e.g. weighted tardiness, WIP). A repair outcome is the evaluation assigned to the set of effects of a repair action and takes values in the set ['acceptable', 'unacceptable']. Typically the outcome reflects tradeoffs among different objectives. If the application of a repair tactic results in a feasible schedule, the result is judged as either acceptable or unacceptable with respect to the repair objectives. An outcome is 'acceptable' if the user accepts the tradeoffs involved in the set of effects for the current application of a repair action. Otherwise, it is 'unacceptable'. The effect salience is assigned when the outcome is 'unacceptable', and it indicates the significance of the effect to the repair outcome. This value is decided subjectively and interactively. The user's judgment as to balancing favorable and unfavorable effects related to a particular objective constitutes the explanation of the repair outcome.

Case acquisition

To gather cases, sample scheduling problems are solved by a scheduler. CABINS identifies jobs that must be repaired in the initial sub-optimal schedule. Those jobs are sorted according to the significance of defect, and repaired manually by a user according to this sorting. For example, if the user's optimization criterion is to minimize order tardiness, the most tardy order is repaired first. The user selects a repair tactic to be applied. Tactic application consists of two parts: (a) identify the activities, resources and time intervals that will be involved in the repair, and (b) execute the repair by applying constraint-based scheduling to reschedule the activities identified in (a). Currently CABINS has 11 tactics and a flexible interface through which the user can define more.

After tactic selection and application, the repair effects are calculated and shown to the user who is asked to evaluate the outcome of the repair. If the user evaluates the repair outcome as 'acceptable', CABINS proceeds to repair another focal_activity and the process is repeated. If the user evaluates the repair outcome as 'unacceptable', s/he is asked to supply an explanation in terms of rating the salience/importance of each of the effects. The repair is undone and the user is asked to select another repair tactic for the same focal_activity. The process continues until an acceptable outcome for the current focal_activity is reached, or the repair is given up. Repair is given up when there are no more tactics to be applied to the current focal_activity; in this situation, CABINS carries on repair of another activity. The sequence of applications of successive repair actions, the effects, the repair outcome, and the user's explanation for failed application of a repair tactic are recorded in the repair history of the case. In this way, a number of cases are accumulated in the case base.

Case re-use

Once cases have been gathered, CABINS repairs sub-optimal schedules without user interaction. CABINS repairs the schedules by (1) recognizing schedule sub-optimalities, (2) focusing on a focal_activity to be repaired in each repair cycle, (3) invoking CBR with the set of global and local features as indices to decide the most appropriate repair tactic to be used for each focal_activity, (4) invoking CBR using the repair effect features (type, value and salience) as indices to evaluate the repair result, and (5) when the repair result is unacceptable, deciding which repair tactic to use next. Note that in contrast to traditional local iterative optimization approaches, (e.g. tabu search, simulated annealing) where the schedule generated in the current iteration as a result of local revision is directly compared (in terms of its associated cost function) with the current schedule, in CABINS, evaluation of the re-

vision is provided by the case base, thus obviating the need for the presence of an explicit cost function.

The similarity between i-th case and the current problem is calculated as follows :

$$exp(-\sqrt{\sum_{j=1}^{N}(SL_j^i \times \frac{CF_j^i - PF_j}{E\_D_j})^2})$$

where SL_j^i is the salience of j-th feature of i-th case in the case-base, and its value has been heuristically defined by the user. CF_j^i is the value of j-th feature of i-th case, PF_j is the value of j-th feature in the current problem, E_D_j is the standard deviation of j-th feature value of all cases in the case-base. Feature values are normalized by division by a standard deviation of the feature value so that features of equal salience have equal weight in the similarity function.

An Example

We briefly illustrate the repair process with a very simple example schedule to be repaired shown in figure 4. The example has ten jobs $(J_1, \ldots, J_{10})$ and each job has five activities with linear precedence constraints. (e.g. O_1^n BEFORE O_2^n, ... , O_4^n BEFORE O_5^n). Resources R_1 and R_2, R_3 and R_5 are substitutable; resource R_4 is a bottleneck. Suppose that the job under repair is J_8. This job has a weight of 2, a due date of 1250 and the scheduled end-time of its last activity is 1390. Hence it has a weighted tardiness of $2 \times (1390 - 1250) = 280$. Suppose the current focal_activity is O_4^8. CBR is invoked with global features (weighted tardiness= 280, resource utilization average=0.544, resource utilization deviation=0.032) plus the set of local features as indices and selects swap as a repair tactic. One can see from the figure that this is a good choice since the focal_activity is scheduled on machine R_4, which doesn't have any substitutable machine and any idle time in the repair time horizon (time between the end of O_3^8 and the end of O_4^8).

To apply swap, CABINS calculates the activity with which O_4^8 will be swapped. To do this, CABINS selects the activity which, if swapped with O_4^8, will result in least amount of precedence constraint violations. In the example, activity O_4^4 is selected as the activity to be swapped with the current focal_activity O_4^8. Job J_4 has weight 3 and weighted tardiness $3 \times (1370 - 1320) = 150$. The effect of applying the swap tactic is that O_4^8 and O_4^4 are unscheduled on R_4 and O_4^8 is re-scheduled to start at time 1090 (the start time of activity O_4^4 prior to the swap). The repair process resolves occurring constraint violations. The repaired schedule is shown in figure 5.

The effects of repairing O_4^8 are calculated. CABINS calculates the effects on J_8 and J_4, the jobs affected by the application of the swap on O_4^8. Machine utilization did not change but J_8 had an estimated decrease in weighted-tardiness of 180 time units and an estimated decrease in WIP of 200 units, J_4 had an increase in