CS537: Scheduling Algorithms

Constraint Programming for Scheduling

By: Hesham Darwish

Supervised by: Professor G.S.Young

June 2015

Abstract

“Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.” [E. Freuder]. Constraint Programming is the study of computational systems based on constraints. The idea of constraint programming is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding solution satisfying all the constraints. Constraint programming has been successfully applied to solve scheduling problems in 1992 and since then it has became one of the dominants form of modeling and solving scheduling problems. In this paper, constraint programming (CP) and constraint satisfaction problem (CSP) will be defined, then a general algorithm to solve a CSP will be introduced. Then constraint based scheduling will be discussed in terms of modeling of scheduling problems and Scheduling-specific propagation technique. Moreover, Selected CP Applications in Scheduling will be reviewed from the scheduling literature as well as from real life.
I. INTRODUCTION

Scheduling is the process of allocating resources to activities over time [1]. In a typical scheduling problem, resources are scarce and constrained in various ways (e.g., in the capacity of resources and the order of activities), and one is looking for a schedule of the activities that both satisfies the constraints and is optimal according to some criterion (e.g., the length of the schedule). Scheduling problems are ubiquitous and appear in many forms, from classic job-shop scheduling to manpower and service scheduling, from product assembly sequencing to logistics resource allocation and scheduling. Scheduling has become an important component of business processes such as supply chain management, and it has its own range of successful software tool vendors.

In the last 20 years, computer scientists developed another approach to solve these scheduling problems, instead of the traditional special purpose algorithms and the integer programming (IP). This new approach is extended from logic programming and it utilizes the usage of a declarative style of problem formulation and associated constraint resolution algorithms for solving optimization problems. This approach is Constraint Programming which have nowadays a significant impact for the operational research in general and the scheduling research in particular. [3]

A. CONSTRAINT PROGRAMMING

Constraint Programming (CP) is basically an approach which formulates and solves discrete variable constraint satisfaction or constrained optimization problems. It systematically employs deductive reasoning which reduces the search space and allows for a wide variety of constraints which helps in solving the problem in a more efficient and speedy way. Constraint Programming has attracted high attention among experts from many areas of computer
science due to its potential for solving hard real life problems, which is actually why the ACM (Association for Computing Machinery) identified it as one of the strategic directions in computing research. Despite that, CP is still one of the least known and understood technologies. The history behind constraint programming can be traced back to the sixties and seventies in the research in Artificial Intelligence(AI). [3]

B. CONSTRAINT SATISFACTION PROBLEM (CSP)

The problems of Constraint Programming deals mostly with Constraint Satisfaction Problems (CSP). Roughly, a CSP is a mathematical model with constraints and the goal is to find a feasible solution i.e. to assign values to the variables of the model such that every constraint is satisfied. The CP godfathers; Brailsford, potts and Smith in [2] define CSP as a given set of discrete variables which can be of different types (int, boolean, symbolic, set of elements) along with finite domains and a set of constraints involving these variables, and the findings should be a solution that satisfies all the constraints.

A variety of constraints can be used, including:

- Mathematical constraints: e.g. completion time = start time + processing time.
- Disjunctive constraints: e.g. specifying two jobs that can’t start at the same time.
- Relational constraints: e.g. specifying maximum number of jobs on a machine.
- Explicit: e.g. specifying explicitly jobs to be processed on a certain machine.

A feasible solution for a CSP is an assignment of a value from its domain to every variable, in which all the constraints has to be satisfied. [3]
II. SOLVING A CONSTRAINT SATISFACTION PROBLEM (CSP)

In this section, a general algorithm for solving CSP will be explained in 7 steps.

1. **CSP Formulation:** This is the first step in solving any CSP, which includes defining variables, the domain and all the constraints.
2. Constraints Propagation: Constraint propagation uses the concept of logical consistency checking to communicate information about variable domain reduction within and across constraints involving the specific variables. There are two types in constraint propagation:
   a. Within-Constraint Domain Reduction:
   In this type, there is one constraint and there is a type of communication between two domains. In this case, the constraint reduces the first domain, and then some sort of communication happens between the reduced domain and the second domain, leading to a reduction in the second domain.
   b. Between-Constraint Domain Reduction
   In this type, there can be multiple constraints and there is a type of communication between the constraints and the domains. In this case, the goal is to schedule multiple jobs, with multiple constraints and this happens by the effective full communication between the constraints and their domains which leads to reduction in the domains.
3. If a solution is found, the algorithm terminates, and in case all solutions are required then the basic process is repeated.
4. If a solution is not found, check for inconsistency, where inconsistency is the state where the domain of at least one variable has become empty.
5. If inconsistency is proven, and if all branches have been explored, then problem inconsistency is proven, so the algorithm fails and ends.
6. If inconsistency in not proven, then a search is undertaken using some search strategy for branching. Branching divides the main problem into a set of mutually exclusive subproblems by temporarily adding a constraint. Branching select one of the branches and propagates all constraints again using the filtering algorithms.
Branching is typically, a depth-first strategy is deployed but what to branch on is open, so basically we could choose what to branch depending on a lot of factors, including:

- Selecting a variable whose domain is not yet reduced to a single value.
- Selecting a variable based on heuristic such as “smallest current domain first”, once a variable is selected; the branch is established by instantiating the variable to one of the values of its current domain (heuristic again such as “smallest value in the current domain first”).
- Decisions can be made based on temporary constraints.
- Choice points: The points at which the search strategy makes an advance along the search tree.

7. If inconsistency is proven but not all branches are visited, then backtrack to step number 6 and go to branching.
III. CONSTRAINT-BASED SCHEDULING

A. MODELING SCHEDULING PROBLEMS

Building on the CP representations and techniques introduced above, various variable and constraint types have been developed specifically for scheduling problems. Variable domains include:

- Interval domains where each value is an interval (e.g., start and duration);
- Resource variables for various classes of resources.

In scheduling applications, integer variables might be used to represent timings, interval variables to represent tasks, logical variables to represent mutual dependencies or exclusions, resource domains to denote classes of resources, etc. Higher-level domains may also be defined in terms of lower-level domains; for example, interval variables are often represented as tuples of integer variables that denote start and duration of the interval.

Scheduling-specific constraints include:

- interval constraints for interval variables (e.g., \( t_1 \leq t_2 \) to express that task 1 has to occur before task 2);
- resource constraints for timing (integer or interval) variables (e.g., allocate\((r,t)\) for resource \( r \) and interval \( t \) to express that the task occupies resource \( r \) during interval \( t \)).

B. SCHEDULING SPECIFIC PROPAGATION TECHNIQUES

Several scheduling-specific propagation techniques have been developed, mostly concerning the reasoning about resources and intervals. In one technique, resource timetables, a timetable with required and available capacity at any time is maintained for each resource. Propagation between resources and tasks works both ways: as a task time becomes fixed, the task resource usage is entered in the timetable; conversely, as available capacity in the timetable is reduced, the interval domains of associated tasks are updated.
Another technique, *edge finding*, reasons about the order in which tasks can execute on a given resource. Each task is evaluated with respect to a set of other tasks. If it is determined that the task must or cannot execute before (or after) these tasks, it may be possible to infer new precedence constraints and new bounds on the task’s interval domain. There may also be domain-specific redundant constraints and search heuristics that lead to increased constraint propagation and smaller search trees (smaller domains).
IV. SELECTED CP APPLICATIONS IN SCHEDULING

A. CP APPLICATIONS IN SCHEDULING LITERATURE

In this section, selected applications of CP to scheduling-related problems are reviewed. This section will review three important scheduling subject areas: Job shop scheduling, Timetabling, and Single Machine Sequencing.

Job Shop Scheduling Problem

There are n jobs, each with a set of operations which must be conducted in a specific order. The jobs are scheduled on m machines which have a capacity of one job at a time and the objective is to minimize the makespan. Brailsford in [2] noted that this problem couldn't be solved with the basic branch and bound algorithm when given 15 jobs and 10 machines, while it could be solved easily by the use of CP. Also Nuijten and Aarts in [5] discovered that their CP-based approach that they used to solve this classic job shop problem, had much better results compared to the typical branch and bound algorithms used in terms of solution speed.[3]

Timetabling Problem

Scheduling a double round-robin college basketball schedule in a minimal number of dates subject to a number of constraints imposed by the league (including constraints like: No more than two away games in a row, no two final away games) when the CP was used to solve this problem (Henz in 2001), they discovered a dramatic improvement in performance over the OR methods (Nemhauser and trick in 1997) which used to solve the problem in 24 hours while the same problem was solved in less than 1 min using CP-approach. That less than 1 min schedule was done using a CP software called Friar Tuck. [6]
Single Machine Sequencing

The single machine sequencing problem assumes that every job must be completed before its deadline, and all jobs are available at time zero. When solving this problem using CP, Jordan and Devel in [7] discovered that the CP solution is the optimal when the machine was at high capacity, where there are tight constraints and smaller feasible solution spaces.[3]

B. CP APPLICATIONS IN REAL-LIFE SCHEDULING

CP Scheduling for Paper-path Control

This section will counter a real life application example of a constraint programming scheduling problem (Paper-path Control).

Modern digital reprographic systems come in many forms, from low-end printers to high-end multi-function devices. They typically consist of a source of paper and images, a paper path that brings these together at the right time, place and orientation, and finishing components that collate, sort, staple and bind the resulting, marked sheets. As a concrete example for how Constraint Programming in Scheduling can be used in a real-time control environment. [4]
V. APPROPRIATENESS OF CP IN SCHEDULING PROBLEMS

There are certain attributes of problems that researchers can be aware of when deciding if CP is an appropriate methodology to employ for solving scheduling problems, Including the following:[3]

- CP is most appropriate for Integer and Boolean decision variable problems, on the other hand it’s not effective for floating point decision variables. CP in general is more appropriate for variables with finite domains at the outset because it means better domain reduction.

- CP is most appropriate for optimization problems with large number of logical, global and disjunctive constraints that CP handles well with its rich set of operators and special purpose constraints.

- CP is most appropriate for problems with large numbers of interrelated constraint with few variables in each constraint, which results into better constraint propagation and more domain reduction as well as a general better performance for CP algorithm when dealing with this type of problems.

- CP is most appropriate to problems that can be viewed as an optimal mapping of one ordered set to another ordered set, where exists a relation between variables that can be expressed in mathematical terms. This is basically the case of most of the scheduling problems.
VI. CONCLUSION

We have defined CP as a method for formulating and solving discrete variable constraint satisfaction, CP involves choosing variables and data structures, representing the relations between these entities in a constraint store, and designing the search strategy.

Employing CP for scheduling problems is a craft involving:

- Capturing relations about the nature of the problem to enhance constraint propagation and domain reduction, where the goal is to use knowledge about the nature of the solution to decrease the search trees (search domain).
- Ability to embed scheduling knowledge into the constraint set or into the design of the search strategy.

We can look to CP as a tool that complements scheduling algorithmic knowledge by serving as a vehicle for its easy implementation.
REFERENCES


