1998年度日本オペレーションズ・リサーチ学会
秋季研究発表会

Solving resource-constrained scheduling problem by constraint-based scheduling

03000500 アイログ(株) *程 宇 CHENG Yu
ILOG フランス MIN-TUNG Daniel

1. Introduction

Given the limited availability of resources, what is the best schedule of the activities in order to complete the project in the shortest possible time? This is called the RCPSP (resource-constrained project scheduling problem). The major objective of the RCPSP is to position the activities on the time line in optimal way so as not to exceed the resource capacity. Project managers who are interested in the scheduling aspects of their projects are typically seeking the way to get the best answer as easy as possible, with the minimum budget and short time.
The approach suggested by this paper is to use a constraint-based scheduling component, which makes it possible to solve complex scheduling problem very quickly. Through this approach, scheduling efforts which used to take months now take days.

2. Constraint programming

One of the main advantages of constraint programming is that it enables you to represent your problem explicitly in a natural and intuitive model. A constraint is a mathematical relation between possible values of variables. Constraint programming is a way of solving constraint satisfaction problems, which consists of a number of variables and a number of relations on and among those variables. Finding solution to a constraint-programming problem is to assign values to the constraint variables of the problem in such a way that all the constraints imposed on the variables are satisfied simultaneously.
The set of possible assignments of values to variables is known as the search space. Constraint programming is efficient because, rather than searching that search space blindly, it exploits the constraints themselves by constraint propagation to reduce its effort in the search. The constraint propagation is known as the most powerful features of the constraint programming.

3. Constraint-based scheduling

Constraint-based scheduling exploits all the facilities of the constraint programming. It propagates more by using domain-dependant knowledge through bi-directional constraint propagation, from activities to resources and from resources to activities.

- Propagation from activities to resources

Information about the start and end times of activities updates the minimal resource capacities required or provided over time. For example, in case when the start time of a consuming activity is known.

- Propagation from resources to activities

Minimal (already required or provided) and maximal capacities over time update earliest and latest start and end times. For example, in case when no capacity remains at a time for a requiring activity.

By this way, the constraint propagation performed in constraint-based scheduling is complete with respect to temporal constraints. This means posting the constraints is sufficient to determine whether a set of temporal constraints is consistent. In addition the earliest and latest start and end times of activities are updated to guarantee: that the earliest start and end times represent a solution, and that the latest start and end times represent a solution, i.e. a set of values for the start and end times which satisfy all of the temporal constraints.

4. Case study

The problem used for case study here was proposed by Prof. Salah E. Elmaghraby (North Carolina State Univ.) in his speech titled "Recent Advances in Project Scheduling" for COM(Computer Optimized Manufacturing) research group of ORSJ (Jan. 13th, 1997, Tokyo).

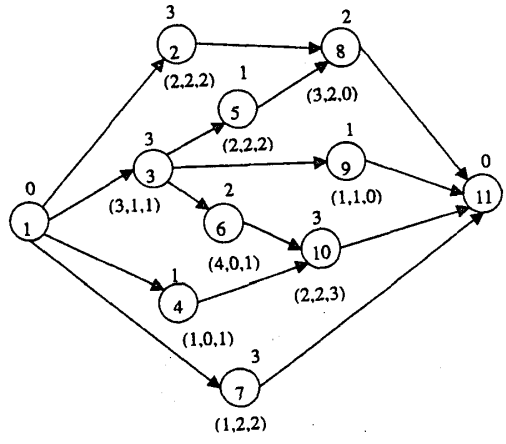
The Problem: Consider the following project in which activity 1 is a dummy activity denoting the "start" of the project, and activity 11 is also a dummy activity denoting the "end" of the project.

There are three resources: R1, R2, R3, each available in 4 units per period.

Each activity is represented by a node, and has a duration

d[i] written on top of the node. Its resource requirements are given below the node. For instance, activity 8 has duration d[8]=2, and requires for each time period, 3 units of resource R1, 2 units of resource R2, and none of resource R3.

What is the minimal duration of the project, and what is the optimal sequence to perform the activity?



The way of modernization is very natural.

Step1: Declare activities and resources;

Step2: Post temporal and resource requirement constraints;

Step3: Search for a solution using constraint propagation and Branch & Bound.

The whole source code on ILOG Solver Studio is as follows.

```
---------------------------------------------------------
Enum Tasks {A1,A2,A3,A4,A5,A6,A7,A8,A9, A10, A11 };
int duration[Tasks] = [0, 3, 3, 1, 1, 2, 3, 2, 1, 3, 0];
int requiredResourceR1[Tasks] = [0, 2, 3, 1, 2, 4, 1, 3, 1, 2, 0];
int requiredResourceR2[Tasks] = [0, 2, 1, 0, 2, 0, 2, 2, 1, 2, 0];
int requiredResourceR3[Tasks] = [0, 2, 1, 1, 2, 1, 2, 0, 0, 3, 0];
scheduleHorizon = sum(t in Tasks) duration[t];
DiscreteResource R1(4);
DiscreteResource R2(4);
DiscreteResource R3(4);
Activity a[t in Tasks](duration[t]);
minimize
 a[A11].end
subject to {
    a[A1] precedes a[A2];
    a[A2] precedes a[A8];
    a[A8] precedes a[A11];
    a[A1] precedes a[A3];
    a[A1] precedes a[A4];
    a[A1] precedes a[A7];
    a[A3] precedes a[A5];
    a[A3] precedes a[A6];
    a[A3] precedes a[A9];
    a[A5] precedes a[A8];
    a[A6] precedes a[A10];
    a[A4] precedes a[A10];
    a[A9] precedes a[A11];
    a[A10] precedes a[A11];
    a[A7] precedes a[A11];
    forall(t in Tasks){
        a[t] requires(requiredResourceR1[t]) R1;
        a[t] requires(requiredResourceR2[t]) R2;
        a[t] requires(requiredResourceR3[t]) R3;}
};
 search {
 setTimes(a);};
---------------------------------------------------------
```

The results on PC Pentium 150 MHz under Windows95 is shown as follows. The total memory used is 48476 (bytes), and CPU time is 0.11 (second). The Optimum value is 13.

```
> A11[13 -- 0 --> 13]
> A10[8 -- 3 --> 11]
> A9[5 -- 1 --> 6]
> A8[11 -- 2 --> 13]
> A7[1 -- 3 --> 4]
> A6[6 -- 2 --> 8]
> A5[4 -- 1 --> 5]
> A4[0 -- 1 --> 1]
> A3[0 -- 3 --> 3]
> A2[3 -- 3 --> 6]
> A1[0 -- 0 --> 0]
```

5.Remarks

In the paper, constraint-based scheduling is suggested to solve resource-constrained scheduling problem. A typical RCPSP problem is given as a case study, and solved by an efficient constraint-based scheduling components ILOG SCHEDULER, which offers a natural object model for the representation of finite capacity scheduling and resource allocation problems and is a library of time-vs-capacity constraints defined in terms of "resources" and "activities". With ILOG SCHEDULER, it is easier and faster to model and solve your resource-constrained scheduling problem, to reduce complexity of your combinatorial problems by constraint propagation.

The constraint-based scheduling can also be used to efficiently solve DTCTP (discrete time/cost trade-off problem), MRCPSP (multi-mode resource-constrained project scheduling problem) and ARCPSP (acquired resource-constrained project scheduling problem) problems.

5. Reference

[1] Salah E. Elmaghraby, Scheduling in activity Networks: The resource-constrainted project scheduling problem, Lecture notes, North Carolina State Univ., 1997.

[2] ILOG SCHEDULER, User Manual, version 4.0, May, 1997.

[3]ILOG SCHEDULER, Reference Manual, version 4.0, May, 1997.