SOLVING A SCHEDULING PROBLEM

3G-1

BY RULE-BASED SYSTEM WITH DP AND LP ENGINES

Tsunehiko Tanaka, Masayuki Numao, and Shin'ichi Morishita
IBM Research, Tokyo Research Laboratory, Tokyo

1. Introduction

Scheduling problems are both classic and new. They are classic since they were recognized many years ago as a challenge, especially in the field of manufacturing. They are new in that their environments have changed in two important aspects: (1) the newly emerging concept of flexible manufacturing systems (FMS) that require complex decision-making has introduced more complexity into scheduling problems, and (2) to cope with this complexity, it has become essential to use highly advanced computer systems.

A scheduling problem is characterized by two difficulties. One is combinatorial explosion: an n-machine, m-job problem has $(m!)^n$ possible schedules, so that without elaborate and intelligent methods, a prohibitively large number of cases must be checked. The other is the diversity of conflicting constraints: a problem is usually constrained by due date, cost limits, production levels, machines, order characteristics, resources, and other factors.

Scheduling problems have been extensively studied by a technique called Operations Research (OR). OR is an analytical method for obtaining an optimal solution by modelling. Recently, many complex problems have been investigated by using Artificial Intelligence (AI) techniques, since it is very difficult to address them by using analytical methods and conventional computer technologies. One important success in the field of AI is Expert Systems (ES), also known as Knowledge-Based Systems. ES exploit human experts' knowledge represented in the knowledge base.

The purpose of the present paper is to propose a new approach to scheduling problems that uses a rule-based system with deterministic algorithms in an engine. In section 2, an actual scheduling problem for steel-making processes is described. In section 3, a basic AI approach and analytical solutions are presented. In section 4, a rule-based system with Dynamic Programming and Linear Programming engines is proposed.

2. Scheduling Problem in Steel-Making Processes

2.1 Steel-Making Processes

There are three major manufacturing processes: iron-making, steel-making, and rolling. Our target process is steel-making, which consists of converters, refining devices, and continuous casters.

2.2 Constraints

Typical constraints for the steel-making process are as follows:
1. Fixed sequence of process stages
2. No process overlaps
3. Continuous processing
4. Waiting time limitation

2.3 Objective Functions

The following criteria are related to the quality of the schedule:
1. Waiting time minimization
2. Total lots maximization

3. Approaches

3.1 Basic AI Approach

Our AI approach to making a schedule that satisfies the above constraints is not to get an optimal solution, but to get a feasible solution efficiently. The reason for this is that there could be several objective functions for determining optimality, and that a combinatorial explosion might prevent a schedule from being obtained in a reasonable time. We introduced a cooperative scheduling method, in which the system efficiently generates a candidate schedule by a *subscheduling and merging* method, and the user evaluates and modifies the candidate schedule by *interactive refinement*. Figure 1 shows the flow of cooperative scheduling. More details can be found in [1].

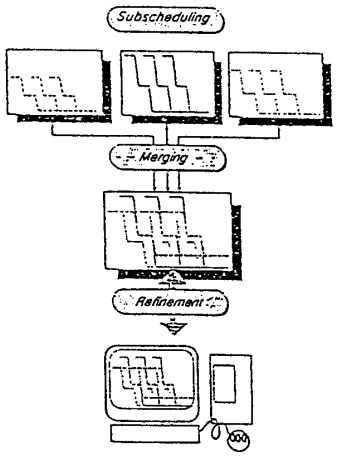

Figure 1 Scheduling Flow

We start off with an almost-good order of subschedules. However, the solution is infeasible, because some of the operations are overlapping. The system helps the user to make a global decision by maintaining the local constraints, that is, the system generates possible schedules, each of which is then checked by the user against his evaluation criteria, such as total waiting time, to test whether it is satisfactory or not. If not, the system revises the schedule, the user checks it again, and so on. There is no guarantee that a solution has short waiting times. Evaluation of the total schedule is left to the user.

3.2 Analytical Solution

The following two programmings provide a method of computing an analytical solution. We would like to minimize the max. of waiting times as well as to find a feasible solution.

3.2.1 Dynamic Programming

A subschedule, which consists of three or four lots, and looks like one in Figure 2. We suppose that a pair of subschedules that are separated by two subschedules are independent.

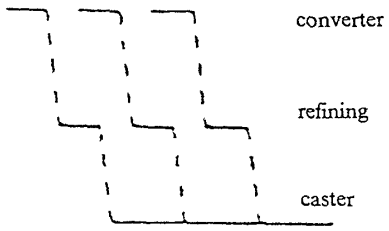

Figure 2 Subschedule

First, the following notation is introduced.

Notation

m : Number of subschedules

S_k : Set of k subschedules

T_k : Time period $[0, T_k]$ in which the first $(m-k)$ subschedules should be scheduled

$t_k = T_{k-1} - T_k$, $T_m = 0$

$f(S_k, T_k, p_{m-k+1}, q_{m-k-2}, q_{m-k-1}, q_{m-k})$:
Minmax of waiting times of the last $k(\in S_k)$subschedules starting with subschedule p_{m-k+1} ,when $(m-k)$ subschedules in a period $[0, T_k]$ are optimally scheduled and when the the $(m-k-2)$th, $(m-k-1)$th, and $(m-k)$th subschedules are q_{m-k-2}, q_{m-k-1}, and q_{m-k}, respectively.

$f^{opt}(S_k, T_k, q_{m-k-2}, q_{m-k-1}, q_{m-k})$:
Minmax of $f(S_k, T_k, p_{m-k+1}, q_{m-k-2}, q_{m-k-1}, q_{m-k})$ over all possible subschedules of p_{m-k+1}

$w^i(T, a, b, c)$: i waiting times when a, b, and c subschedules are scheduled in a time span T

The recursive equation of a dynamic programming formulation is given by

$$f^{opt}(S_k, T_k, q_{m-k-2}, q_{m-k-1}, q_{m-k})$$

$$= \underset{p_{m-k+1} \in S_k}{\text{Minmax}}$$

$$\{f^{opt}(S_k - \{p_{m-k+1}\}, T_k + t_k, q_{m-k-1}, q_{m-k}, p_{m-k+1}),$$
$$w^i(T_{m-k+1} - T_{m-k-1}, q_{m-k-1}, q_{m-k}, p_{m-k+1})\}$$

$$(k = 1, \ldots, m-3)$$

Prior to the application of the recursive equation, we can compute $w^i(T, a, b, c)$ for

$$m \times (m-1) \times (m-2)$$

permutations of subschedules.

At stage k, there are $\binom{m-3}{k}$ different S_k 's. Thus, the number of equations at stage k is

$$m \times (m-1) \times (m-2) \times \binom{m-3}{k}.$$

The total number of equations is

$$\sum_{k=1}^{m-3} m \times (m-1) \times (m-2) \times \binom{m-3}{k}.$$

3.2.2 Linear Programming

When we are given three subschedules in a time span T, we must have a method of finding the best assignment of those subschedules. This can be done by *linear programming*.
The objective function:

$$\underset{i}{\text{Minmax}} \{w^i\}$$

Subject to:

Inequalities of non-overlapping conditions

The objective function does not seem to be linear. However, by letting $Z = \max w^i$, we can convert the problem to a linear programming problem with an objective function Min Z and additional constraints $w^i \leq Z$.

4. Rule-Based System with DP and LP Engines

As we saw in section 3.2, we can obtain an optimal solution by the use of dynamic programming and linear programming, at the cost of a tremendous amount of computation time. In practice, however, we may not be able to reach an optimal solution because of the severe limitation of computation speed.

We here propose "a rule-based system with DP and LP engines for solving a scheduling problem." The architecture of this rule-based system is depicted in Figure 3.

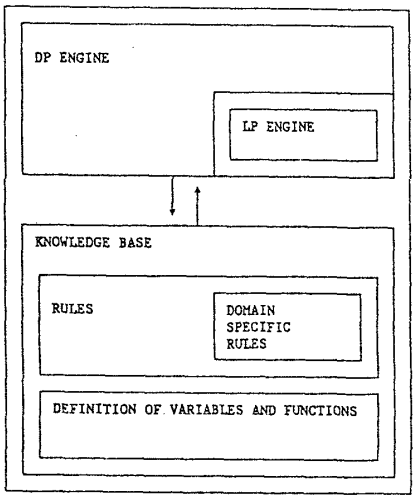

Figure 3 Architecture of the Proposed Rule-Based System

Heuristic rules can be used to exclude a number of unnecessary and insignificant cases from consideration. Especially, rules specific to the application area must be utilized. Dynamic programming shell with powerful library routine of linear programming gives the best solution from cases considered.

Reference

[1] M.Numao and S.Morishita, "Scheplan - A Scheduling Expert for Steel-Making Process," International Workshop on Artificial Intelligence for Industrial Applications, Hitachi, Japan, May, 1988.