Simulation and Planning for Food Production Scheduling

Jeff Schneider and Andrew Moore
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
schneide@cs.cmu.edu,awm@cs.cmu.edu

Introduction

Schenley Park Research's PlanIt software is currently being used by a major U.S. food manufacturer to schedule production in one of their factories. An overview of the system is shown in fig. 1 and a sample schedule is shown in fig. 2.

The planner is presented with a problem specification and formulates a schedule to meet the constraints while maximizing profit. A specific problem instance is given as a set of inventory profiles for approximately 20 products. The inventory profiles reflect the current stock of each product and the estimated demand for that product over the period covered by the plan. The goal is to generate a cost optimal production schedule which prevents the inventory of any product from dropping below a danger threshold. This must be done subject to constraints on which products can be produced in combination with other products.

In order to evaluate potential schedules, the planner relies on a simulation of the factory and a knowledge base representing costs and constraints. The user has the opportunity manually edit the plan. The resulting plan is sent off for execution, which is not directly controlled by the software. The results of the execution, which often vary significantly from the expectations of the original schedule, are automatically fed back to the system for re-planning. A typical schedule covers a period of 1 to 4 months and takes the planner 1-12 hours to produce. Replanning is done anywhere from once a day to once a week.

A sample schedule is shown in fig. 2. There are several packagers available to handle about 20 different food products. The schedule shows which products they should be set to produce over time. Additional complexity comes from the constraints and interactions not made explicit by this representation of the schedule. There are several other stages of processing which occur before and after the packaging stage shown. It is not necessary to explicitly schedule all the other operations, but they must be represented in the knowledge base because they constrain what products can be produced simultaneously. Additionally, the rate at which a particular packager produces a particular product is dependent upon what the other packagers are doing at the time. The time required to reconfigure them to produce a new product is also a complex function of the current setup of the factory.

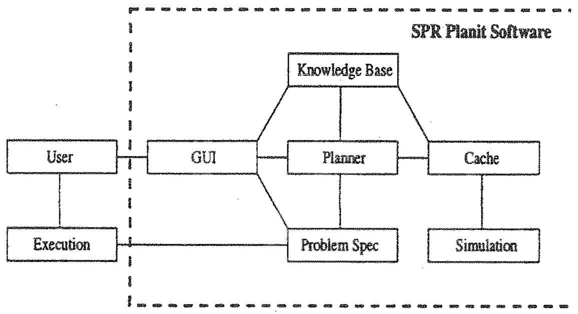

Figure 1: Overview of planning and execution process for production scheduling

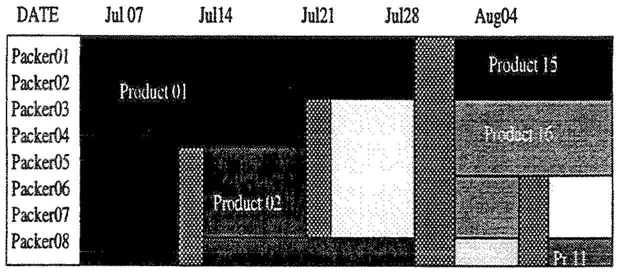

Figure 2: A simplified example of a schedule produced by SPR PlanIt for a single production line. The crosshatching between products reflect the amount of time required to changeover between the products.

Simulation and Caching

The simulation and caching modules are crucial to the solution of this scheduling problem. First, simulation is required to accurately determine how much of each product will be produced with a particular factory configuration. Standard linear models of production are insufficient. The performance of the factory components is highly stochastic, so simulations must be run for a significant amount of time to be accurate. This could make planning infeasible because of the computation required. In order to solve that problem, all simulation results are cached. It is impractical to cache the results for every possible factory configuration, but many of them are never useful. After several planning runs, a cache of 10,000 to 50,000 configurations is generated, which covers almost all the ones the planner is likely to consider.

Planner

The planner uses a heuristic combination of several common search and optimization algorithms including greedy search, hill climbing, simulated annealing, and linear programming. Many of these algorithms are used in several different ways to optimize different aspects of plans. For example, a linear program that optimally sets the length of time each product is run on each resource, is run as a subroutine to a hill climber that changes which products are scheduled for each resource. The talk will discuss the how these algorithms are combined and their pros and cons.

Discussion

There are several properties that make this scheduling problem more difficult than typical formulations such as job shop scheduling. Both constraint satisfaction and cost optimization play a strong part in the planning. algorithms and representations that focus too much on either one are insufficient. Expensive, nonlinear simulations are the only way to accurately model the production process. This restricts both the time available for planning and excludes algorithms that rely on simple production models. Finally, there are no single deadlines for production requirements. The requirement is an inventory curve that must be considered at all points in time. Because of the complexity of this problem, we believe the techniques developed for it can be applied to a variety of space related applications. The scheduling of scientific sensors with interacting resource requirements on a spacecraft is one obvious possibility and we hope to discuss many more at the workshop.

The biggest obstacle during the development of this system was that the factory specifications were a moving target. The capabilities of the factory were changing on nearly a monthly basis. Frequently, new products were being added, or poor sellers were being removed. Occasionally, entire production lines were added, removed, or redesigned. The smallest changes called for updates to the knowledge base and the problem specifications. Larger changes required modification of the simulation and caching algorithms. Some changes also called for re-engineering the algorithms used in the planner. The issue of a constantly changing factory remains the biggest problem in the fielded application. The same problems arise in the goal of rapidly deploying the software to a series of new factories, or in using the software to do what-if analysis on potential factory designs.

Small changes are handled well in the current system. A typical example is the addition of a new product. This can be accomplished by using the GUI to edit the knowledge base. The end users perform that kind of change themselves.

Larger changes generally require modifications to the simulator. A typical example is a change in the distribution system that divides the raw product between the resources. Deploying the system in a new factory will often require the implementation of a new simulator as well. Since this is done in C code in the present system, it can be time consuming and require more expertise than factories have available on the factory floor. One method for dealing with this problem is to use simulation suites that are easy for non-programmers to use. Better simulation tools help, but it is still likely that coping with significant factory changes will be expensive.

An alternative is to expand the sources of information that can be used to fill the cache. In the current system, all information in the cache comes from simulation, but there are other potential sources. One is the knowledge base. In simple cases, approximations to the production statistics can be written down in closed form and stored in the knowledge base. Another source of information is the results of executions. A separate module could be included to learn the behavior of the factory by monitoring it. In addition to storing information, the cache would be responsible for combining the information from various sources depending on how reliable each is. In the best case, simple rules could be added to the knowledge base which would be good enough to allow planning while the learning system came up to speed and/or a simulation is constructed for a new factory.

Finally, some changes affect the performance of the planning algorithms. It requires a significant amount of time from an expert to tune the planner. A big improvement would be to automate the process of testing and tuning the planning algorithms. This is a research problem that requires all the algorithms to be put into a common framework such that they can be slotted in and out of the planner, as well as parameterizing them in a way that can be searched efficiently.