

Tractable Resource Allocation Using Approximate Dynamic Programming

Gordon Rios
gordon.rios@certive.com

January 17, 2003

Abstract

This paper explores a general method for solving resource allocation problems using a dynamic programming approach that limits the set of paths examined.

1 Resource Allocation: Dynamic Programming

1.1 Resource Allocation

Consider the resource allocation problem for a set of time ordered tasks scheduled over a shared pool of resources. Assume the tasks are composed of blocks of subtasks that each require a specific class of resource for a given set of discrete time periods. Assume that each resource in the pool may belong to multiple *resource classes*. Tasks are ordered by time with discrete probabilities of occurrence; thus, the resource allocation is performed *under uncertainty*. In addition, alternative variants of the tasks may exist that affect the specific structure of resource requirements for the task.

1.2 Formulation of the Approximate Dynamic Program

This dynamic programming formulation solves for the *expected minimum cost* of a given set of tasks. There are two specifications required: the recurrence relation and the boundary conditions.

1.2.1 Recurrence Relation

The recurrence relation is as follows:

$$f_i(X) = p_i \cdot \min_{\substack{x_i \subset \{h_{i,v}(X), \emptyset\} \\ v \in V_i}} \{g_{i,v}(x_i) + f_{i+1}(X - x_i)\} + (1 - p_i) \cdot f_{i+1}(X)$$

Where:

$i \in 1 \dots N$ is an index of tasks ordered by time

V_i is the set of variants for task i .

p_i is the probability of getting project i .

$g_{i,v}(x_i)$ is the cost of allocating x_i to variant v of task i .

$h_{i,v}(X)$ is a function that selects a set of possible allocations for variant v of task i from a set of resources X (e.g. $h_{i,v}(X) \subseteq X$).

1.2.2 Formulation Discussion

This formulation is not exact because in general the set of allocations generated by the h_i 's is far smaller than the complete set of possibilities generated by X . The exact formulation where each allocation x_i from X is strongly exponential since at each step it generates all task compatible subsets the size of which is exponential in the number of resources. Complete references for dynamic programming formulations of the resource allocation problem can be found in [1] and [2].

1.2.3 Probability Branching and Computational Complexity

Aside from the combinatorics addressed by the h_i 's the probability branching nominally creates 2^N paths. However, caching the values of the various function calls in each path's computation will reduce the runtime complexity from $O(2^N)$ computations. Clever specification of the h_i 's could also yield more hits on the set of cached functions.¹

1.2.4 Boundary Conditions

Here are the boundary conditions:

$$f_N(X) = p_N \cdot \min_{\substack{x_i \subset \{h_{N,v}(X), \emptyset\} \\ v \in V_i}} \{g_{N,v}(x)\}$$

¹A given implementation may offer opportunities to select resource allocations that yield profiles of resource availability in common across different paths of the problem – narrowing the possibilities for X in the calls the $f_i(X)$ will yield more cache hits.

$$f_i(\emptyset) = \sum_{i \leq j \leq N} p_j \cdot g_j(\emptyset)$$

where $g_j(\emptyset) = \min_{v \in V_j} \{g_{j,v}(\emptyset)\}$.

1.2.5 Solution

The solution is simply $f_1(R)$ where R is the initial matrix of resources available during each time period. The standard textbook problem usually involves activities that draw from a single type of resource during a single time period. Since the formulation is not exact the solution is not guaranteed to be optimal.

1.3 Computational Strategies

The key element of the formulation is constraining the search through careful specification of the h_i 's. This framework allows specific named resources to be assigned to project requirements. In general, the h_i 's can be used to embed project-resource constraints and preferences.

One approach to generating a good candidate allocation is to use a heuristic scheduling rule such as *Least Flexible Task*.² For example, put the tasks in increasing order by flexibility and assign each subtask to the resource (from the required *resource class*) with the least number of possible subtask assignments.

The size of the set generated by a given h_i is a natural parameter for controlling the size of the overall computation. The space of solutions reachable with the formulation above is enlarged by increasing the number of solution candidates generated by each h_i . Thus, a natural way to parameterize the computational effort is to specify the number of candidates generated by each h_i .

1.4 Key Features

The model specified above has several key features.

- Enables the modeling of recourse in the decision making process with direct control over the computational burden.

²The use of various heuristic scheduling rules such as Least Flexible Task (LFT) or Least Flexible Machine (LFM) are discussed in standard scheduling texts like [3].

- Allows rich modeling of the constraint structure for resources and requirements.
- Allows for flexibility in the choice of objective functions since the $g_{i,v}$'s can be specified arbitrarily.
- Allows the user to modify any task or resource feature in application domain and update the solution.

References

- [1] Bellman, R., and Dreyfus, S., 1962. Applied Dynamic Programming, Princeton Univ. Press, Princeton, N.J.
- [2] Dreyfus, S., and Law, A., 1977. The Art and Theory of Dynamic Programming, Academic Press, New York, NY.
- [3] Pinedo, M., 2002. Scheduling: Theory, Algorithms, and Systems, Prentice Hall, Upper Saddle River, NJ.