Nuclear Reactor Refueling Optimization

Bell, D.E. and Shapiro, J.F.

IIASA Working Paper

WP-74-032

1974

NUCLEAR REACTOR
REFUELING OPTIMIZATION

D.E. Bell and J.F. Shapiro

August 1974 WP-74-32

In a 1971 paper, Suzuki and Kiyose give a model for light water moderated atomic reactor refueling optimization. Specifically, they present a linear programming formulation for minimizing the number of fresh fuel assemblies required by a reactor over a finite planning horizon subject to power generation and safety requirements and reactor design specifications. The optimal refueling policies found by Suzuki and Kiyose were useful in reducing the fresh fuel required, but two difficulties were encountered. First, the optimal linear programming solutions included small fractional numbers of fresh fuel assemblies which were difficult to round off. The second difficulty was that their formulation had approximately 165H constraints where H is the length of the planning horizon. The problems solved had H=10, but it was desired to analyze the problem for longer planning horizons of 20 to 30 stages without solving prohibitively large mathematical programming problems.

In this paper, we give a reformulation of the reactor refueling optimization problem that consists of approximately 15H constraints and a large number of columns. This reformulation is required because the state-of-the-art of integer programming does not usually permit the solution of integer programs with thousands or even many hundreds of constraints. Moreover, the reformulation should permit the linear programming approximation to be more easily solved, at least approximately. Finally, the reformulation identifies and analyzes explicitly the fundamental activity in refueling optimization; namely, the introduction, degradation and removal of fuel assemblies. This should make it easier to modify the model to take into account additional features of the problem such as a cost for moving an assembly from one location to another.

1. Statement and Reformulation of the Problem

A fuel assembly is introduced into the reactor at a burnup level 1 and degrades with time to burnup level j $j = 1,\ldots,J$. Time is measured in discrete stages and we let $h = 1,\ldots,H$, denote the periods in the planning horizon. The exact degradation of an assembly during a given period depends on the zone in which it operates. Let $i = 1,\ldots,I$ denote these zones and let $T_i(j) > j$ denote the burnup level of a fuel assembly at the end of a period spent in zone i when it was at a burnup level j at the start of the period.

The formulation of Suzuki and Kiyose is as follows. Let x_{ij}^h denote the number of fuel assemblies of burnup level j assigned to zone i in period h. The integer programming problem which minimizes fresh fuel is

$$Z = \min \sum_{h=1}^{H} \sum_{i=1}^{I} x_{i1}^{h} \qquad (1.1)$$

$$\text{s.t.} \quad \sum_{j=1}^{J} a_{ij}\, x_{ij}^{h} \leq b_{i}^{h} \qquad \text{for all } i,h \qquad (1.2)$$

$$\sum_{i=1}^{I} x_{ij}^{h+1} \leq \sum_{i=1}^{I} x_{iT_i^{-1}(j)}^{h} \qquad \text{for all } j,h \text{ except } j=1,\ h=H \qquad (1.3)$$

$$x_{ij}^{h} \geq 0 \text{ and integer for all } i,j,h, \qquad (1.4)$$

where the integer a_{ij} is a technological coefficient for an assembly in zone i at burnup level j and

$T_j^{-1}(j)$ is the burnup level at the start of a period of an assembly located in zone i which degrades to level j by the end of the period. Note that the slacks on the constraints (1.3) are the burnup assemblies of levels j which are discarded at the start of period h+1. In the actual application, there are 3 IH constraints of the type (1.2), including IH equality constraints. Moreover, there are upper bound constraints on the slack variables s_i^h on the constraints. We have stated (1.2) in the simpler form, and omitted the bounds on the s_i^h, in order to be able to present an uncluttered discussion of our approach. These details can be reinstated without difficulty when computation is done. The idea behind our reformulation is that the constraints (1.3) have an implied network structure which is not being exploited and moreover, which is inefficiency described by a large system of inequalities.

We define a fuel assembly schedule to be an H-vector with entries 0,1,2,...,I where the entry in the hth component indicates the zone in which it is located in period h and zero indicates it is not used. The non-zeros must run consecutively. An example of a schedule when H=10 is the vector (0,0,0,3,3,2,2,0,0,0) indicating the assembly is introduced into the reactor in zone 3 at the start of period 4, is relocated in zone 2 at the start of period 6, and is removed at the end of period 7.

Each assembly schedule implies unique burnup levels of the assembly. Specifically, we have

assembly used in periods	$h_0, h_0 + 1, \ldots, h_0 + T$	
located in zones	$i_0, i_1, \ldots, i_T$	
burnup levels	$j_0, j_1, \ldots, j_T$	(2)

where

$$j_s = T_{i_{s-1}}(j_{s-1}) \quad , \qquad s = 1,\ldots,T$$

and

$$j_0 = 1.$$

The information in (2) is used to define the performance coefficients

$$v_i^h = \begin{cases} a_{i_h j_h} & \text{if } h \in \{h_0,\ldots,h_0 + T\} \\ 0 & \text{if } h \notin \{h_0,\ldots,h_0 + T\} \end{cases}$$

Let V denote the IH vector with components v_i^h. In order to state our reformulation of problem (1), we need a complete enumeration of such columns, say V^k, $k = 1,\ldots,K$, with components $v_i^{h,k}$. Let x_k denote the number of times schedule k is to be used. Then problem (1) is equivalent to

$$z = \min \sum_{k=1}^{K} x_k$$

$$\text{s.t. } \sum_{k=1}^{K} v_i^{h,k} x_k \leq b_i^h \text{ for all } i, h \qquad (3)$$

$$x_k \geq 0 \text{ and integer for all } k$$

The number of schedules will in general be quite large and a method is required to generate good schedules iteratively but not exhaustively. The linear programming problem which results if the integrality restriction in (3) is omitted is denoted by L.P. (3) and its minimal objective function value by L.

2. Generation of Fuel Assembly Schedules

It is clear that I.P. (3) has an enormous number of columns for an application of any realistic size; for $i = 5$, $J = 150$, $H = 30$, we estimate I.P.(3) would have between 10,000 and 20,000 columns. Thus, some pricing procedure for generating good columns for I.P.(3) without exhaustively generating all columns is required. Since there is nothing inherently special about I.P.(3), a column generation procedure for it is applicable to a number of similar I.P. column generation problems such as the cutting stock problem, multi-commodity flow problems and others

(Lasdon (1970)). For this reason, the general theory of I.P. column generation will be presented in another paper. We give here only a brief discussion of how columns can be generated.

The idea behind column generation for L.P.(3) is linear programming dual pricing (Lasdon (1970)). Specifically, let π denote a non-negative IH vector of prices on the constraints in L.P.(3). The column generation procedure is to solve

$$\begin{aligned} &\text{minimize } \pi\, V \\ &\text{s.t.} \quad V \text{ feasible column} \end{aligned}$$

in order to find a specific column $\bar{V}$ with the property $\pi\, \bar{V} < -1$. If this last inequality holds, then the column $\bar{V}$ looks attractive for use in L.P.(3) since its reduced cost $1 + \pi\, \bar{V}$ is negative relative to the prices π. In this case, $\bar{V}$ is added with an appropriate variable to L.P.(3).

The column generation problem has a shortest route network interpretation. The nodes and arcs are generated recursively from the following initial set of nodes and arcs. The initial set of nodes are an origin node, a removal node, and nodes i, 1, h, for all i, h. There are arcs drawn from the origin to nodes i, 1, h, with arc lengths $\pi_i^h a_{i,1}$. Starting from node i, 1, h, there are a number of arcs drawn to the removal node. Each arc corresponds to maintaining the fuel assembly in zone i for r additional periods, r = 0,1,2,...,R, where R is a practical upper limit on assembly life; probably R=4 will suffice for the given problem. If r=0, the arc length is 0, whereas if $r \geq 1$, the arc length is

$$\pi_i^{h+1} a_{i,T_i(1)} + ,\ldots,+ \pi_i^{h+r} a_{i,T_i^r(1)}$$

where

$$T_i^r(1) = T_i(T_i^{r-1}(1)) \quad , \quad r = 2,3,\ldots,R \quad ,$$

and

$$T_i^1(1) = T_i(1) \quad .$$

The additional nodes and arcs are generatively recursively from the nodes i, 1, h. Specifically, a node i, j, h previously generated will generate nodes i′, $T_i^{r+1}(j)$, h + r + 1 for all i ≠ i′ and for r = 0,1,...,R, and arcs drawn from i, j, h to these nodes. These correspond to maintaining the assembly in zone i for r additional periods and then shifting the assembly to zone i′. The associated arc length is

$$\pi_i^{h+1} a_{i,T_i(j)} +,\ldots,+ \pi_i^{h+r} a_{i,T_i^r(j)} + \pi_{i'}^{h+r+1} a_{i',T_i^{r+1}(j)}$$

where only the last term is present if r=0.

The column generation problem is solved when we have found the shortest route path from the origin node to the removal node. If the length of this path is less than - 1, then the corresponding path can be used to generate a column to add to L.P.(3). The example illustrated in figure 1 will

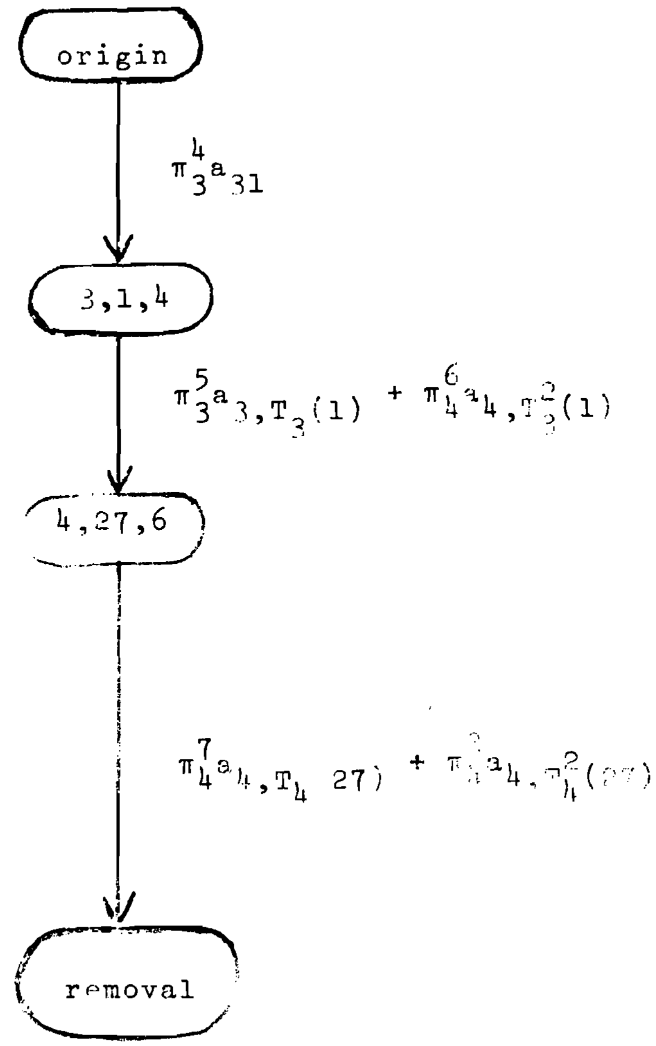
origin
$\pi_3^4 a_{31}$
3,1,4
$\pi_3^5 a_{3,T_3(1)} + \pi_4^6 a_{4,T_3^2(1)}$
4,27,6
$\pi_4^7 a_{4,T_4(27)} + \pi_4^8 a_{4,T_4^2(27)}$
removal

Figure 1.

suffice to show how this is done. Notice that $T_3^2(1) = 27$; that is, a fresh assembly in zone 3 for two periods degrades to a burnup level of 27. The shortest route path corresponds to a schedule (0,0,0,3,3,4,4,4,0,...,0). From this schedule, a column V is uniquely defined.

The network we are describing is clearly very large for the given values I = 5, J = 150, H = 30. However, our proposed method for solving and using the network should eliminate most of the difficulties. The idea is to adapt Dijkstra's algorithm (1959) for solving shortest route problems. The algorithm begins with arcs drawn from the origin to the nodes i, 1, h, with their associated lengths for all i, h. These arcs are ordered according to length, creating a path list, and the minimal one drawn to a specific node i, 1, h, is selected. The algorithm then considers the R + 1 paths drawn out of the specific i, 1, h, to the removal node and selects the minimal length one from among these. This path represents a completed schedule and it becomes the incumbent shortest route path until a better is discovered.

The path to i, 1, h, is also extended to the nodes i', $T_i^{r+1}(1)$, $h + r$, for all $i' \neq i$ and for $r = 0,1,\ldots,R$. These paths are ordered according to length and the ordered list is merged with the previous ordered path list with the minimal element deleted (it is replaced by the newly generated paths). The minimal element of the path list is again selected and the path is extended in the same manner.

L.P. Column Generation Algorithm

Step 1 (Initialization):

For $i = 1,\ldots,I$, $h = 1,\ldots,H$, add $i\ 1\ h$ to path list with associated length $\pi_i^h a_{i1}$. Order path list by increasing length. Set the incumbent length of shortest route path to the best known (or estimated) value $\hat{c}$.

Step 2.

Stop if path list is empty. Otherwise, select first path from path list (i.e., path with minimal length). Suppose it is drawn to node $i\ j\ h$ and has length c. (Optional: search through the list and eliminate all other paths drawn to $i\ j\ h$). Extend path to removal node by shortest path by calculating $r \in \{0,1,\ldots,R\}$ satisfying

$$\sum_{t=1}^{r} \pi_i^{h+t} a_{iT_i^t(j)} = \underset{s=0,1,\ldots,R}{\text{minimum}} \sum_{t=1}^{s} \pi_i^{h+t} a_{iT_i^t(j)} \quad .$$

if

$$c + \sum_{t=1}^{r} \pi_i^{h+t} a_{iT_i^t(j)} < \hat{c} \quad ,$$

replace incumbent by this path and set $\hat{c}$ equal to the left hand sum. Delete all paths from path list with length greater than $\hat{c} - \Delta$ where

$$\Delta = R \cdot \min_{i,j,h} \pi_i^h a_{ij} \quad .$$

Step 3.

For $i_1 \neq i$ and $r = 0,1,\ldots,R$, extend path to nodes i_1, $T_i^{r+1}(j)$, $h + r + 1$ with associated length

$$c + \sum_{t=1}^{r} \pi_i^{h+t} a_{iT_i^t(j)} + \pi_{i_1}^{h+t+1} a_{i_1 T_i^{r+1}(j)}$$

except if this length is greater than $\hat{c} - \Delta$. Merge these paths with the paths on path list so that the augmented path list is still ordered by increasing length. Return to Step 2.

Remarks

Step 1. The shortest route path from the previous calculation with different π_i^h can be used to give a value of $\hat{c}$ using the new arc lengths $\pi_i^h a_{ij}$. Alternatively, we can take $\hat{c} = -1$ since any basis activities in L.P.(3) correspond to paths with length -1.

Step 2(a). Since any column with reduced cost less than -1 can be used to improve the solution to L.P.(3), the stopping criterion can be $\hat{c} \leq -1 -\varepsilon$ for some $\varepsilon > 0$.

(b). There may be relatively few paths drawn to the same node in the network. Therefore, it may not be worth the work at each step to make the optional substep.

(c). The value Δ is selected so that any incompleted path with length greater than $\hat{c} - \Delta$ will not have a completed length less than $\hat{c}$. The value Δ is a gross overestimate and it will probably be preferable to use a smaller value in spite of the small risk that the shortest route path may be deleted before it is completed.

Step 3(a). There may be a cost associated with moving an assembly from one zone to another. If the objective function of the problem (3) were changed to one of minimizing cost rather than the number of fresh fuel assemblies used, then the moving cost could be included as well.

This completes our discussion of column generation for L.P.(3). The problem we really want to solve is I.P.(3). Thus, the question remains: How do we adapt or continue the linear programming column generation process to solve the integer programming problem? In a separate paper we will give a theoretical procedure which allows this to be done. Roughly speaking, the idea is to add additional structure to the shortest route problem so that paths other than those corresponding to the optimal linear programming basic activities are generated.

From a practical viewpoint, however, the procedure for generating additional columns for I.P.(3) needs to be combined with branch and bound and heuristics. We will be in a better position to judge these practical matters when computational experiments currently underway are completed. We plan to write another version of this paper including computational experience.

REFERENCES

Bell, D.E., "Bounds for Generalized Integer Programs" IIASA Working Paper No. 73-9 (1973).

Dijkstra, E.W., "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, 1, (1959).

Dreyfus, S.E., "An Appraisal of Some Shortest-Path Algorithms", Operations Research, 17, pp. 395-412 (1969).

Lasdon, L., Optimization Theory for Large Systems, MacMillan, (1970).

Suzuki, A. and R. Kiyose, "Application of Linear Programming to Refueling Optimization for Light Water Moderated Power Reactors", Nuclear Science and Engineering, 46, pp. 112-130, (1971).