Integer Permutation Programming and the Loading Pattern Optimization Code SUPERLPOS Used at SNERDI

Shengyi Si
Shanghai Nuclear Engineering Research & Design Institute
Reactor Physics and Thermal-Hydraulic Department
No.29, Hongcao Road,
Shanghai 200233, P. R. China
ssy@snerdi.com.cn

ABSTRACT

The loading pattern optimization code system SUPERLPOS has been in production usage at Shanghai Nuclear Engineering Research & Design Institute (SNERDI) since 1999. This code system is based on the methodology of integer permutation programming developed at SNERDI (ref[1]). This paper will introduce the integer permutation programming method, the core modeling theory and the design application results of SUPERLPOS.

1. INTRODUCTION

Optimization design of loading pattern is a challenging work of in-core fuel management for Nuclear Power Plant. There are a lot of optimization methods and papers about loading pattern design developed and published during past 2 or 3 decades. Generally, all these methods can be classified into 2 categories. The first one is non-algorithm optimization, e.g. direct search method, expert system method and others methods based on artificial intelligence (AI). Since these methods are dependent on human experience accumulated during manual shuffling of loading pattern and some principles set by human, it is difficult to get global optimization by using these methods, and also the solution is not repeatable. The second one is algorithm optimization, e.g. linear programming, mixed integer programming and recently popularized stochastic programming such as simulated annealing algorithm and genetic algorithm. Linear programming and mixed integer programming are based on a linear model, however, the response of core reactivity and power distribution to loading pattern is nonlinear, and therefore the crucial point for these two methods is to create a effective linear model to simulate the nonlinear response. Stochastic programming can simulate linear system or nonlinear system with continued or discrete (integer) variables explicitly, but because it is a stochastic algorithm, the reliability of the solution is naturally dependent on the history simulated by user.

The Integer Permutation Programming, presented in this paper, is an algorithm optimization method based on the idea of combinatorial optimization using an accurate linear model. The method can give a definite global solution without human interference; therefore, it can serve as an effective tool for loading pattern design optimization.

Some basic ideas of Integer Permutation Programming are introduced in Sec.2. Section 3 presents how we implement these ideas for engineering application. The forth section will introduce application of Integer Permutation Programming at SNERDI. The last section gives some conclusions about our development and application on Integer Permutation Programming.

2. METHODOLOGY BACKGROUND

Intuitively, loading pattern design is a problem related to permutation. If we transform 2D core positions into 1D vector and think the available fuel assemblies as elements in that vector, a loading pattern is just a permutation of available fuel assemblies in a given discrete space. Shuffling of fuel assemblies is just a transformation from one permutation to another permutation, and the optimum loading pattern is just the optimum permutation.

If one can scan all these permutations, of course, we can find the optimum permutation, which is absolutely the global optimization. Unfortunately, the number of the permutations is so huge that human can not scan it even with the fastest computer in the world. Nevertheless, after a second thought, we will find that it is unnecessary to scan all these permutations. Because most of these permutations, in practical applications, can be pruned at the very beginning, and under some preconditions, we can utilize mathematical programming to help the permutation. The inspiration of Integer Permutation Programming comes just from this consideration.

Let's assume a reload core with N batch fuel (for a given reload core, the available fuel assemblies can be grouped into several batches according to the initial enrichment and depletion history), and each batch has M_n fuel assemblies, the number of core locations is $L=\Sigma M_n$. The number of permutations without any constraint in this case can be written as:

$$NT = P_{L_1}^{M_1} \cdot P_{L_2}^{M_2} \cdots P_{L_N}^{M_N} \qquad L_n = L - \sum_{i=1}^{n-1} M_n \qquad n = 1, N \tag{1}$$

$$NT = C_{L_1}^{M_1} \cdot P_{M_1}^{M_1} \cdot C_{L_2}^{M_2} \cdot P_{M_2}^{M_2} \cdots C_{L_N}^{M_N} \cdot P_{M_N}^{M_N} \tag{2}$$

$$NT = [C] \cdot [P] \qquad [C] = C_{L_1}^{M_1} C_{L_2}^{M_2} \cdots C_{L_N}^{M_N} \qquad [P] = P_{M_1}^{M_1} P_{M_2}^{M_2} \cdots P_{M_N}^{M_N} \tag{3}$$

From the expression above, we can find that the permutation of loading pattern can be decomposed into 2 steps. The first step is to get the combination of batch fuel, which is only identified with batch ID and without any information about individual fuel assembly. We can call this kind of batch ID arrangement in 2D core as *batch pattern*. Actually, if there is no constraint, the combinatorial number of batch patterns is still an astronomical figures ranging 10^{12}~10^{21}. Nevertheless, for a given reload core, with the constraints of design objectives (such as Out-In, In-Out-In, In-In-Out), core symmetry and other specific design requirements, the number of acceptable batch patterns can be dramatically reduced to certain acceptable magnitude. We can write a special computer code to filter and pick up the entire acceptable batch patterns for a given reload core. The second step is to get permutation of available fuel assemblies based on a selected batch pattern. At this step, each permutation is just a loading pattern identified with individual fuel assembly's information. Because these permutations are performed within the same batch, the magnitude of perturbations of fuel reactivity or burnup can be controlled within a smaller range. Mathematically, the influence of these smaller perturbations on core reactivity and core power distribution can be linearized and supposed to be added up linearly. That means we can write some linear model to simulate the effect of fuel assemblies' shuffling on core reactivity and power distribution, so long as the shuffling always happens within the same batch, and then we can utilize integer linear programming to find the optimum permutation.

All above are general ideas of Integer Permutation Programming for loading pattern optimization. In brief words, we will enumerate batch patterns at the first step, and then we will utilize integer linear programming to perform permutations of the available fuel assemblies in each batch pattern.

3.METHODOLOGY OF INTEGER PERMUTATION PROGRAMMING

3.1 Batch Pattern Enumeration

Even though the number of batch patterns for any core is astronomical, actually, the number can be reduced to certain acceptable magnitude dramatically after some engineering requirements are considered. As we know, most of technical ideas of reload core design are closely related to the batch pattern, and also, the performance of the reload core is mainly dependent on the batch pattern, especially dependent on the assignment of feed batch. These relations are shown as follows:

(a) Assignment of feed fuel is dependent on what kind of core the customer want. Out-In, In-Out-In or In-In-Out has special requirement on assignment of feed fuel.

(b) Symmetrical requirement on radial power distribution implies that the batch pattern should keep 1/8 core symmetry.

(c) Flatting radial power distribution requires that the fuel assemblies in the same batch should not congregate together too much, some kind of distribution like checkerboard is recommended.

(d) If you want to maximize the averaged discharge burnup, the batch to be discharged should not be assigned on core periphery.

(e) The requirement on signal response of NIS (Nuclear Instrument System) says that twice or more times burned fuel should not be assigned around the NIS locations, even though you want to have a lower leakage core design.

(f) Efficiency of main adjustment rod and integral worth of all control rods also has some requirements on batch pattern.

All above is only a part of available constraints, anyone or any plant may have other special constraints, and with these constraints, the number of acceptable batch patterns can be reduced dramatically from an astronomical number to several thousands or just hundreds. Using recursive and backtrack logic, a constrained enumeration algorithm was developed at SNERDI, and the computer module has been written and incorporated in our SUPERLPOS code system to enumerate the batch patterns that satisfy the location constraints.

While the acceptable batch patterns are enumerated, we use a 2D-core code to evaluate them one by one. For every batch pattern, we assume that all fuel assemblies in a batch have the same reactivity or burnup, which is just the batch averaged value. After we finished the scanning calculation by using 2D-core code, we can rank all the batch patterns according to selected criteria, e.g. power peaking factor, cycle length, averaged discharge burnup etc. At this point, we can choose some better batch patterns that match our design purpose.

As we mentioned above, the performance of a loading pattern is mainly dependent on the base batch pattern. For example, the difference of BOC critical boron for different batch patterns may range from 0ppm to 200ppm. However, the difference of BOC critical boron between a given batch pattern and the optimal pattern, which can be obtained from shuffling distinct individual assemblies inside the batches of that batch pattern, is only a few ppm. Therefore, if the performance of the batch patterns is optimal, it is reasonable to believe that the global optimum to the problem is just among those loading patterns based on these selected batch patterns. Next step we will utilize integer linear programming to explore the local optimum in these specific domains, which is just the global optimization.

3.2 Linear Programming Model

For a given batch pattern, the permutation number of individual fuel assemblies within each batch is also very huge ranging 10^{13}~10^{26}, and we can not enumerate all these loading patterns as we enumerate batch pattern, because we do not have enough constraints to prune bad permutation. But if we consider a batch pattern as a reference loading pattern, where all fuel assemblies in the same batch have the same burnup or reactivity i.e. the batch averaged value, the final loading pattern is just a series of local burnup or reactivity perturbations based on this reference pattern. Since these perturbations are around the averaged batch fuel reactivity or burnup, the magnitude of the perturbations is smaller, according to the first-order Taylor's Theorem, the influence of these perturbations on core reactivity and core power distribution can be linearized and assumed to be added up linearly. This section will introduce the linear model to simulate the effect of perturbations on core reactivity and power distribution.

The crucial point to establish a linear model is to get the relation between perturbation of fuel burnup or local reactivity and variation of radial power distribution. Ref.[5] introduced a method to generate so-called "Power Response Matrix" by using Generalized Perturbation Theory. We will use the similar method to get the power response matrix, but will not present its details in this paper. One can find the detailed description in ref[3] and ref[5].

The power response matrix is generated based on the selected batch pattern, which is considered as a reference case. The depletion calculation for this batch pattern can give the reference power distribution at every burnup step. If there are M locations in the core, we can write this reference power distribution as a series of M-dimensional vectors $\boldsymbol{P}^t_{ref}$, where t identifies different burnup step. And then the perturbed calculation can give the perturbed normalized power at each core location. Assuming that there are L perturbation locations (L ≤ M) in the core, which may be grouped into several batches. Then the power response matrix $[F]^t$ is a L × M matrix, the elements of the matrix can be approximated as:

$$f^t_{lm} = \frac{\partial P^t_m}{\partial \rho^0_l} \approx \frac{\Delta P^t_m}{\Delta \rho^0_l} \qquad (4)$$

where

$\Delta\rho^0_l$ is the reactivity perturbation of location l at BOC

ΔP^t_m is the perturbation in normalized power of location m at burnup step t due to reactivity perturbation at location l at BOC

For a given reload core, the available fuel assemblies are given and can be grouped into several batches according to the initial enrichment and depletion history, the difference between each fuel assembly reactivity and its batch averaged reactivity is the available perturbation. Therefore, the perturbation vector corresponded to a loading pattern can be written as $\boldsymbol{\Delta\rho^0} = (\Delta\rho_1, \Delta\rho_2, \ldots \Delta\rho_L)^T$. Applying the first-order Taylor's theorem, the power distribution corresponded to the loading pattern can be written as a linear combination of reference power distribution and the perturbed value,

$$\boldsymbol{P^t} = \boldsymbol{P}^t_{ref} + \boldsymbol{\Delta P^t} \qquad (5a)$$

$$\boldsymbol{\Delta P^t} = [F]^t \boldsymbol{\Delta\rho^0} \qquad (5b)$$

Start from equation (5), we can establish various linear models for different design purpose. Here we present 2 typical linear models: (maximization of) remaining reactivity at EOC and (maximization of) averaged discharge burnup at EOC.

(Maximization of) remaining reactivity at EOC

According to the Point Reactivity Model (ref[2]), the core reactivity ρ_c can be expressed as a weighted average of individual fuel assembly reactivity over the assembly-wise normalized power distribution as follows:

$$\rho_c^t = \boldsymbol{P}^t \cdot \boldsymbol{\rho}^t / M \quad t=0,1,\ldots\ldots,T \qquad (6)$$

where

t is identified for burnup step
T is total step number
M is total number of core fuel assembly locations
$\boldsymbol{P}^t$ is normalized power distribution vector at burnup step t, (dimension size M)
$\boldsymbol{\rho}^t$ is reactivity distribution vector at burnup step t (dimension size M)

By applying a piecewise continuos linear approximation for fuel assembly reactivity vs. burnup introduced in ref [5], EOC reactivity distribution vector $\boldsymbol{\rho}^T$ has a recursive relation with the initial reactivity distribution vector $\boldsymbol{\rho}^0$ as follows:

$$\boldsymbol{\rho}^T = \boldsymbol{\rho}^0 - \sum_{t=1}^{T} \Delta \boldsymbol{B}_c^t [\boldsymbol{A}]^t \boldsymbol{P}^t \qquad (7)$$

where

ΔB_c^t is the burnup step length at $\boldsymbol{t}$ step
$[\boldsymbol{A}]^t$ is a diagonal matrix $=\boldsymbol{diag}(\alpha_1^t, \alpha_2^t, \alpha_3^t, \ldots\ldots \alpha_M^t,)$; t=1,….T
α_m^t is slope of piecewise linear reactivity for location m at step t; m=1,M; t=1,T

$$\alpha_m^t = \frac{\rho_m^{t-1} - \rho_m^t}{B_m^t - B_m^{t-1}} = \frac{\rho_m^{t-1} - \rho_m^t}{\Delta B_c^t P_m^{t-1}} \qquad (8)$$

B_m^t is assembly-wise burnup for location m at step t

Since all our work are based on a given batch pattern, for each location in the core, the characteristics of the batch fuel is known, therefore we can calculate α_m^t during batch pattern depletion calculation and consider $[\boldsymbol{A}]^t$ as a constant matrix. Now incorporating equation (6) and (7), we can get the EOC core reactivity $\boldsymbol{\rho}^T$ as follows:

$$\boldsymbol{\rho}_c^T = \frac{1}{\boldsymbol{M}} \left(\boldsymbol{\rho}^0, \boldsymbol{P}^T \right) - \frac{1}{\boldsymbol{M}} \sum_{i=1}^{T} \Delta \boldsymbol{B}_c^i \left([\boldsymbol{A}]^i \boldsymbol{P}^i, \boldsymbol{P}^T \right) \qquad (9)$$

The second term in equation (9) is still non-linear because of power distribution vectors. The application of the first-order perturbation can transform it in to a linear equation:

$$\boldsymbol{\rho}_c^T \approx \boldsymbol{\rho}_{ref,c}^T + \Delta \boldsymbol{\rho}_c^T \qquad (10\boldsymbol{a})$$

$$\Delta \boldsymbol{\rho}_c^T = \frac{1}{\boldsymbol{M}} \left[\left(\Delta \boldsymbol{\rho}^0, \boldsymbol{P}_{ref}^T \right) + \left(\boldsymbol{\rho}_{ref}^0, \Delta \boldsymbol{P}^T \right) \right] - \frac{1}{\boldsymbol{M}} \left[\sum_{t=1}^{T} \Delta \boldsymbol{B}_c^t \left([\boldsymbol{A}]^t \Delta \boldsymbol{P}^t, \boldsymbol{P}_{ref}^T \right) + \sum_{t=1}^{T} \Delta \boldsymbol{B}_c^t \left([\boldsymbol{A}]^t \boldsymbol{P}_{ref}^t, \Delta \boldsymbol{P}^T \right) \right] \qquad (10\boldsymbol{b})$$

Now it is obvious that maximization of EOC reactivity $\rho_c{}^T$ is equal to maximize $\Delta\rho_c{}^T$ in equation (10b). By incorporating equation (5) and equation (10) and applying matrix transformation technique, we can get a concise linear programming model for maximization of remaining reactivity at EOC:

$$\begin{aligned} &\text{Objective Function:} \quad \max \ \boldsymbol{C}^{\mathrm{T}} \cdot \boldsymbol{\Delta\rho}^{\boldsymbol{0}} \\ &\text{Subject to:} \quad [S] \cdot \boldsymbol{\Delta\rho}^{\boldsymbol{0}} \le \boldsymbol{b} \qquad (11) \\ &\qquad\qquad lb_l \le \Delta\rho_l^{0} \le ub_l, \quad l=1,L \end{aligned}$$

$\boldsymbol{C}^{\mathrm{T}}$ in equation (11) is the coefficient vector of objective function, which corresponds to the design purpose, i.e. maximization of remaining reactivity at EOC.

The inequality $[S] \cdot \boldsymbol{\Delta\rho}^{\boldsymbol{0}} \le \boldsymbol{b}$ in equation (11) represents constraints or constraint functions, which are mathematical expressions of the engineering requirements on loading pattern design. For example, if we want to control assembly-wised power peaking within a limited value P^{lim}, the corresponding constraint function can be written as $[F]^{\boldsymbol{t}} \cdot \boldsymbol{\Delta\rho}^{\boldsymbol{0}} \le P^{lim} - \boldsymbol{P}^{\boldsymbol{t}}_{\boldsymbol{ref}}$. t=1,2,……T. Similarly, we can transform other requirements on, for example, global reactivity and geometric symmetry to constraints.

lb_l and ub_l in equation (11) are the ***l***ower ***b***ound and ***u***pper ***b***ound of variables $\Delta\rho_l, l=1,L$. For a given batch pattern, lb_l and ub_l correspond to the maximum lower perturbation and upper perturbation, (lb_l , ub_l) is the available perturbation range related to different batch. After we classified all available fuel assemblies in batch, we can get the differences between individual assembly burnup or reactivity and the batch averaged value, then we can get the available perturbation range for this batch.

<u>*(Maximization of) averaged discharge burnup at EOC*</u>

Since our search is based on a selected batch pattern, and we also know which batch should be discharged at end of this cycle, it is easy to work out a linear programming model to maximize the averaged discharge burnup.

In the quasi-static depletion analysis model, the incremental assembly-wise burnup for location m at step t is related to the core burnup step length and the normalized power factor:

$$\Delta B_m{}^t = B_m{}^t - B_m{}^{t-1} = \Delta B_c{}^t P_m{}^{t-1} \qquad (12)$$

Assuming that there are ND locations in the core (ND≤M) that belong to the batch to be discharged, the averaged discharge burnup can be written as:

$$DBU = \frac{1}{ND} \sum_{m=1}^{ND} \sum_{t=1}^{T} \Delta B_c^t P_m^{t-1} \qquad (13)$$

By applying the first-order perturbation approximation, equation (13) can be transformed into:

$$DBU = DBU_{ref} + \Delta DBU \qquad (14a)$$

$$\Delta DBU = \frac{1}{ND} \sum_{m=1}^{ND} \sum_{t=1}^{T} \Delta B_c^t \Delta P_m^{t-1} \qquad (14b)$$

The maximization of averaged discharge burnup is equivalent to maximizing equation (14b). Combining equation (5) and equation (14b), we can get a concise linear programming model similar to equation (11) for maximization of averaged discharge burnup.

The algorithm to solve linear programming was well studied during past several decades. Every linear

programming module based on Revised Simplex Algorithm can accept the format of programming model in equation (11).

3.3 Integer Permutation Programming

As we know, the solution space for the variables in the linear programming model of equation (11) is a continuous real space, while a realistic loading pattern must take an integer solution. So we should add some other integer constraints in the linear programming model of equation (11) to transform the linear programming to an integer programming.

Let's focus on a core with N batch fuel and each batch has M_n fuel assemblies, the number of core locations is $M=\Sigma M_n$. The solution space of loading pattern integer programming is composed of N integer subsets $\boldsymbol{I}_n$, n=1,N, and every subset has M_n integers. By adding these integer solution space constraints into the linear programming model of equation (11), we can get an integer programming model,

$$\begin{aligned} &\text{Objective Function:} \quad \max \ \boldsymbol{C}^{\mathrm{T}} \cdot \Delta\boldsymbol{\rho}^{\boldsymbol{0}} \\ &\text{Subject to:} \quad [S] \cdot \Delta\boldsymbol{\rho}^{\boldsymbol{0}} \le \boldsymbol{b} \\ &\qquad lb_l \le \Delta\rho_l^0 \le ub_l, \quad l=1,L \\ &\qquad \Delta\rho_l \in \boldsymbol{I}_n, \text{ if } l \in n, \ l=1,L \text{ and } n=1,N \end{aligned} \tag{15}$$

The algorithm to solve integer programming of equation (15) is called Branch & Bound (B&B) or Implicit Enumeration, which is a classical algorithm to solve hard integer optimization problem. B&B method is a divide-and-conquer approach trying to solve a hard integer optimization problem by splitting it into smaller problems denoted as subproblems. For every subproblem, the upper or lower bounds of partial variable and subproblem are computed recursively. And specially, the computation of upper bound for these subproblem is quite crucial, it can help you to prune a lot of branches and converge to the optimum solution quickly. Well used algorithm to calculate upper bound of subproblem is linear programming relaxation, which is just a new linear programming by dropping the constraints on integer variables. With today's software and hardware technology, we can realize these ideas successfully.

The methodology outlined so far does not yet solve the problem of loading pattern optimization completely, because we can not guarantee that the integer solution to integer programming of equation (15) is just a permutation of the available fuel assemblies. It is possible that some of variables may take identical solution value, which means one fuel assembly is loaded in several locations in the core. Therefore, we must add some extra constraints in integer programming model of equation (15) to keep the solutions of all variables different from each other. Therefore, the final integer permutation programming model is thus written as follows,

$$\begin{aligned} &\text{Objective Function:} \quad \max \ \boldsymbol{C}^{\mathrm{T}} \cdot \Delta\boldsymbol{\rho}^{\boldsymbol{0}} \\ &\text{Subject to:} \quad [S] \cdot \Delta\boldsymbol{\rho}^{\boldsymbol{0}} \le \boldsymbol{b} \\ &\qquad lb_l \le \Delta\rho_l^0 \le ub_l, \quad l=1,L \\ &\qquad \Delta\rho_l \in \boldsymbol{I}_n, \text{ if } l \in n, \ l=1,L \text{ and } n=1,N \\ &\qquad \Delta\rho_i \ne \Delta\rho_j, \ i,j=1,N \end{aligned} \tag{16}$$

With all these constraints, now, we can guarantee that the solution of integer permutation programming of equation (8) is a valid loading pattern.

4.SUPERLPOS CODE SYSTEM AND APPLICATION RESULTS

The above methodology has been implemented in the code system SUPERLPOS (Super Loading Pattern Optimization System), and successfully applied to loading pattern design at Shanghai Nuclear Engineering Research and Design Institute (SNERDI). This section will introduce the code system and the application results.

4.1 SUPERLPOS Code System

SUPERLPOS is a PC-based optimization platform with a friendly Graphic User Interface (GUI). The code system is composed of several main functional modules, and all the modules are written in Visual FORTRAN90. These modules include:

a). ***GUI Drive Module*** gives a friendly graphic user interface and makes the optimization process visualized. By using the menu system and dialog box system in this module, user can control various calculation functions interactively, some important calculation results and calculation progress are also be displayed on the graphic screen.

b). ***Refuel Module*** helps user to collect available fuel assemblies for loading pattern design. It displays the batch ID and burnup of all the assemblies at EOC of last cycle on graphic screen, user can select the assembly to discharge by using mouse, and then user can input the information for refuel assemblies via dialog boxes. Finally, the module saves the available assemblies information in a file.

c). ***Core Neutronics Module*** is a 2-dimensional depletion routine based on Nodal Green's Function Method. With the same thermal-hydraulic model and depletion model as that used in nuclear design codes, and tuned to the results of 3-dimensional nuclear design code, the results of this module, such as power distribution and critical boron etc., can have good agreement with the 3-dimensional nuclear design code.

d). ***Batch Pattern Enumeration Module*** uses recursive and backtrack algorithm to enumerate batch patterns that satisfy all the geometrical and engineering constraints input by user. The constraints include: fixed region for a batch, available region for a batch, forbidden region for a batch, and various shapes of forbidden clusters of assemblies in the same batch.

e). ***Batch Pattern Evaluation Module*** performs the neutron diffusion and depletion calculation for every enumerated batch pattern. As we mentioned above, in a batch pattern, all the assemblies in a batch take the batch averaged burnup information as BOC conditions. The evaluation has two levels, the first level is only to get critical boron and power distribution at BOC for every enumerated batch pattern, and then rank the critical boron and the power peaking factor. At this point, the user can throw away a part of the enumerated batch patterns, whose BOC critical boron is extremely low or power peaking is extremely high compared with some criteria. The second level is to perform depletion calculation for the survived batch patterns. The cycle length of depletion calculation for every survived batch pattern is the same and fixed, therefore we can rank the EOC critical boron, power peaking in the cycle and batch averaged burnup for all survived batch patterns. And then we can pick up some top batch patterns according to our design purpose as the base case of loading pattern design.

f). ***Power Response Matrix generation Module*** incorporates neutronic module and Generalized Perturbation Theory to generate power response matrix for a selected batch pattern. The module at first does depletion calculation for the selected batch pattern without perturbation to get the reference power distribution, and then performs Generalized Perturbation Theory calculation for every available perturbation location to get the perturbed power distribution. Finally, the module

generates the power response matrix based on reference power distribution, perturbed power distribution, the perturbed reactivity, and saves it as a file.

g). ***Integer Programming Module*** presents a dialog box on the graphic screen, via which user can input the type of objective function (maximization of remaining reactivity at EOC or maximization of averaged discharge burnup at EOC) and the limiting value for the power peaking during the cycle life. With these input and a selected power response matrix file, the module constructs a linear programming model. The linear programming solver used in this module is an intrinsic routine in IMSL bound with Visual FORTRAN90. A routine implementing Branch & Bound algorithm will call that linear programming solver recursively to explore the acceptable integer solutions. The progress of recursive level and the number of explored integer solution are displayed on the graphic screen instantly. Finally the module will give the optimum integer solution to current problem if there are solutions under current constraints, which is just the optimum loading pattern based on the selected batch pattern.

4.2 Application Result and Discussion

SUPERLPOS code system has been in production usage since 1999 at SNERDI. In order to demonstrate the efficiency of the code system, this section presents a sample calculation of an operating Nuclear Power Plant, which is a 3-loop Westinghouse-type core containing 157 Fuel assemblies. The sample calculation is the loading pattern optimization design for the Cycle 2, in which the available fuel assemblies include once-burned assemblies of 2.1w/o (9 boxes), once-burned assemblies of 2.6w/o (52 boxes), once-burned assemblies of 3.1w/o (52 boxes) and fresh assemblies of 3.15w/o (44 boxes). The actually used loading pattern in this cycle is presented in Figure 1 for the purpose of later comparison.

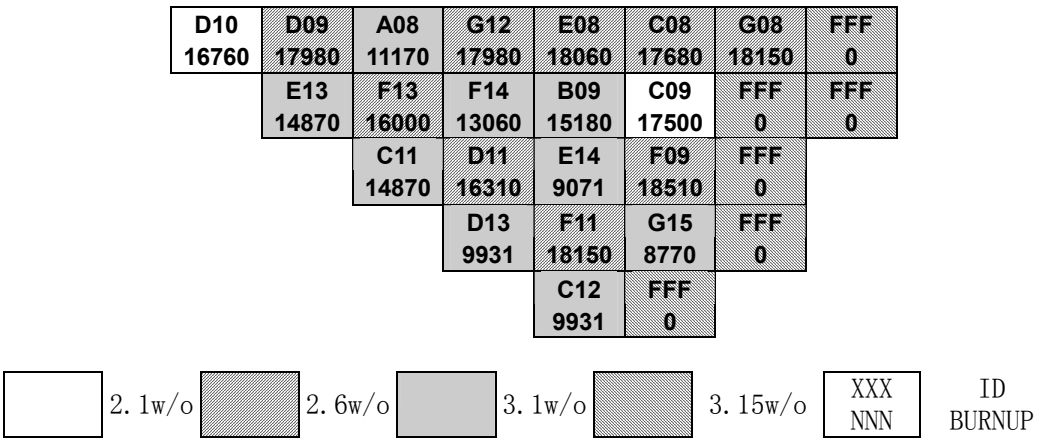

Figure 1. Reference loading pattern for Cycle 2

When we use SUPERLPOS to do optimization for loading pattern design, the first step is to utilize Refuel Module to collect the available assemblies for Cycle 2 based on the burnup distribution at EOC of cycle 1. And then we run the Enumeration Module to enumerate acceptable batch patterns. It is obvious that the actual loading pattern is a typical Out-In arrangement. In order to maintain the consistence and comparability with the reference case, we only enumerate the batch patterns in Out-In model by using the constraints of forbidden region, i.e. the fresh assemblies are only arranged on periphery of the core. As for the burned assemblies, we use the constraints of forbidden clustering to prune batch patterns with clusters of more than 3 assemblies in the same batch except for 4 assemblies in a row.

While the batch patterns are enumerated, the Batch Pattern Evaluation Module and Nutronics Module are called to evaluate them one by one. As we mentioned above, there are two levels at this step. Firstly, we only perform criticality calculation at BOC for each enumerated batch pattern to get the critical boron and power peaking factor at BOC and rank them. Now we can throw away a part of batch patterns, whose BOC critical boron (HFP) is less than 1100 ppm or power peak at BOC (HFP) is over 1.35. In this way, out of the 1129 batch patterns that are enumerated, only 83 of them survive. In the second level, we do depletion calculation with a fixed cycle length of 8000 MWD/MTU for each survived batch pattern to get the maximum power peak factor in this cycle, critical boron and batch averaged burnup at EOC (8000 MWD/MTU) and rank them.

By analyzing the ranked results and considering our design purpose, we can pick several top batch patterns for further exploration. For each selected batch pattern, we run Power Response Matrix generation Module to get the power response matrix, and then run Integer Programming Module to get the optimal loading pattern based on the selected batch pattern. Because it is batch-pattern-dependent, this optimal loading pattern should be a local optimum. And because we believe that the global optimum is among those loading patterns based on the selected top batch patterns, we can find the global optimal loading pattern for our design objectives from those local optimal loading patterns. In our sample calculation, we will present the results of several optima arising from two batch patterns. One of these batch patterns is that used in reference loading pattern, identified here as batch pattern A; another one is a competitive batch pattern picked from the ranked results, identified as batch pattern B.

Figure 2~5 show the loading patterns optimized by using SUPERLPOS code system. Figure 2 and Figure 3 use the same batch pattern as that used in actual loading pattern, but the loading patterns correspond to maximum remaining reactivity at EOC and maximum batch averaged discharge burnup at EOC respectively. Figure 4 and Figure 5 share another batch pattern, but the loading patterns are generated with the similar idea as that in Figure 2 and Figure 3. Table 1 presents the results of these loading patterns together with the results of reference loading pattern and the batch patterns. These results are calculated by using nuclear design code of SNERDI.

Firstly, let's focus on the results of the two batch patterns. The main difference between the two batch

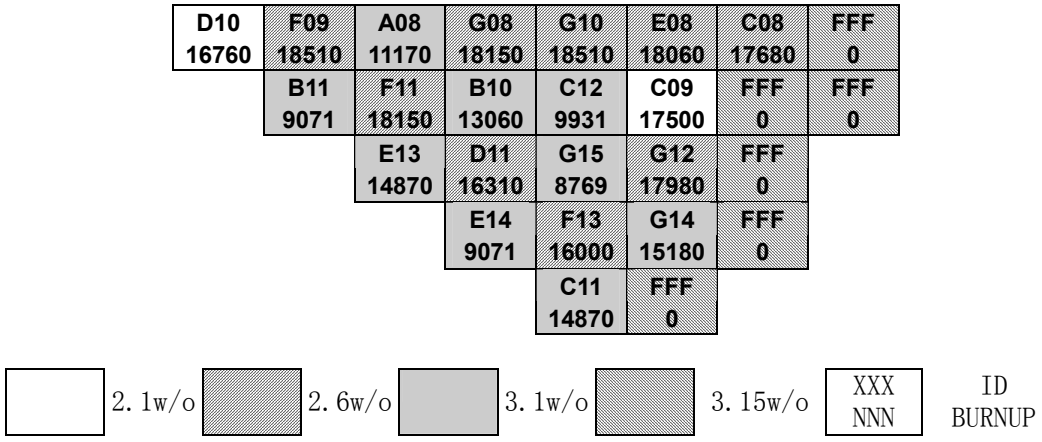

Figure 2. Optimized loading pattern for Cycle 2 based on batch pattern A
(Maximization: remaining reactivity at EOC)

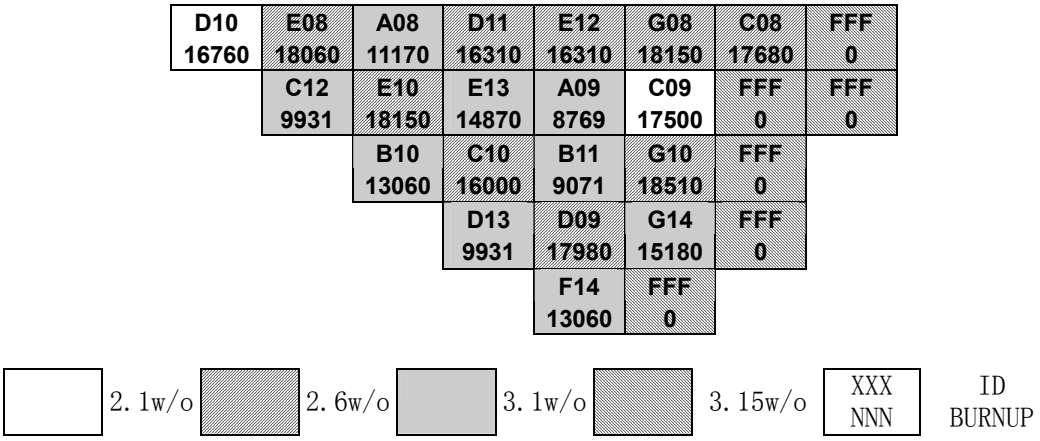

Figure 3. Optimized loading pattern for Cycle 2 based on batch pattern A
(Maximization: averaged discharge burnup at EOC)

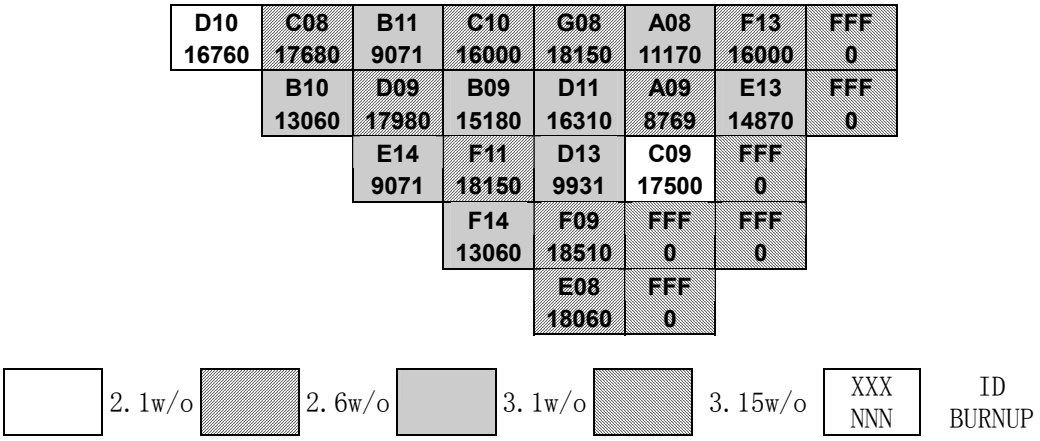

Figure 4. Optimized loading pattern for Cycle 2 based on batch pattern B
(Maximization: remaining reactivity at EOC)

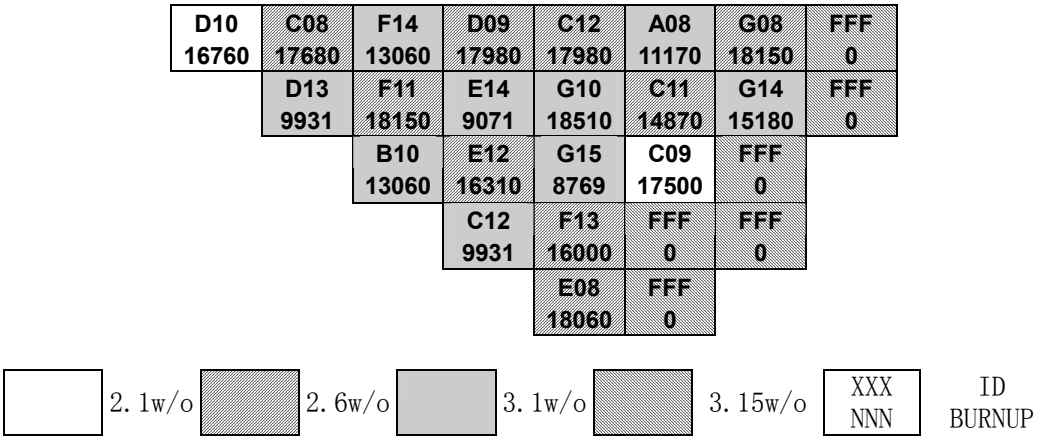

Figure 5. Optimized loading pattern for Cycle 2 based on batch pattern B
(Maximization: averaged discharge burnup at EOC)

Table 1. Comparison of optimized loading pattern with reference loading pattern
(Calculated by nuclear design code)

	Batch Pattern A				Batch Pattern B		
	Batch Pattern	Reference Loading Pattern	Optimized Loading Pattern Max: Reactivity at EOC	Optimized Loading Pattern Max: Averaged Discharge Burnup at EOC	Batch Pattern	Optimized Loading Pattern Max: Reactivity at EOC	Optimized Loading Pattern Max: Averaged Discharge Burnup at EOC
Critical Boron ppm (BOC,HFP)	1199	1195	1213	1212	1205	1205	1210
Power Peak (Cycle, HFP)	1.3166	1.3042	1.3023	1.3053	1.3184	1.2967	1.3132
Cycle Length (MWD/MTU)		8843	8891	8880		8904	8885
EOC Boron (8000MWD/MTU)	89	86	91	90	91	92	90
EOC Batch Burnup (8000MWD/MTU)							
2.1w/o (9)	24286	24257	24462	24521	24292	24686	24550
2.6w/o (52)	25485	25432	25530	25541	25488	25407	25579
3.0w/o (52)	20878	20887	20968	20955	21094	21024	20961
3.15w/o (44)	6923	7018	6764	6754	6665	6797	6695
EOC Batch Burnup (10 ppm)							
2.1w/o (9)		25001	25267	25320		25518	25356
2.6w/o (52)		26269	26420	26421		26301	26464
3.0w/o (52)		21835	21974	21949		22055	21966
3.15w/o (44)		7764	7538	7517		7578	7457

patterns is assignment of feed fuel. The assignment of feed fuel in batch pattern B reduced the power sharing on core periphery and the neutron leakage from the core (since it is a Out-In pattern), therefore batch pattern B can have a little bit longer cycle length and lower accumulated burnup of fresh batch at EOC.

Now let's check the results of the optimized loading pattern. It is clear in table 1 that the performance of the 4 loading patterns generated by SUPERLPOS code system is obviously better than that of reference loading pattern, either in term of cycle length or in term of averaged discharge burnup. Since batch pattern B has a little bit longer cycle length than batch pattern A, loading patterns based on batch pattern B are expected to have a longer cycle length than those based on batch pattern A. Furthermore, even though the cycle length in the case of maximizing averaged discharge burnup is shorter than that of maximizing the remaining reactivity at EOC for both batch patterns, it is interesting that the averaged discharge burnup in the former case is still greater than that in the latter case. That means the design outcome is quite consistent with the design objective as represented by the objective function.

5. CONCLUSIONS

Based on the idea of combinatorial optimization, Integer Permutation Programming Method decomposes the optimization of loading pattern into 2 steps: batch pattern (combination) enumeration by using recursive and backtrack algorithm and loading pattern permutation by using integer programming on a given batch pattern. Since the permutation is controlled within the same batch, the magnitude of perturbations can be controlled within a smaller ranger; therefore the method can get an effective linear model suitable for application of Integer Programming. Moreover, by predetermining the domains where the global optimum is located, the method can definitely find the global optimum without using any technique for escaping from trapping.

Integer Permutation Programming Method has been implemented in the code system SUPERLPOS, which integrates each steps in the method into an effective and powerful design tool by using visualization programming technique. Validation calculation and engineering application demonstrate that the development of the method and the code system is successful.

REFERENCES

1. Si Shengyi, "Development and Application of Integer Permutation Programming to PWR Core Loading Pattern Design", *The 8th Reactor Numerical Computation and Particle Transport Conference*, Dayabay, Guangdong, China, October 30 - November 3, 2000
2. Geoffrey Thomas Parks and Jeffery David Lewins, "a Point Reactivity Model for In-core Fuel Cycles", *Nuclear Technology*,**Vol.82**,P.267-274 (1998)
3. W.S. Yang and T.J.Downer, "Generalized Perturbation Theory for Constant Power Core Depletion", *Nuclear Science and Engineering*,**Vol.99**,P353 (1988)
4. Wei Heping, *Application of Optimization Method*, Beijing University Press,(1987)
5. Taek Kyum Kim and Chang Hyo Kim, "Mixed Integer Programming for Pressurized Water Reactor Fuel-Loading-Pattern Optimization" *Nuclear Science and Engineering*,**Vol.127**,P67 (1997)