Optimization of nuclear reactor reloading patterns

E. de Klerk[a], C. Roos[a], T. Terlaky[a,★], T. Illés[b,☆], A.J. de Jong[c], J. Valkó[c] and J.E. Hoogenboom[c]

[a] *Faculty of Technical Mathematics and Informatics, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands*

E-mail: e.deklerk@twi.tudelft.nl; c.roos@twi.tudelft.nl; t.terlaky@twi.tudelft.nl

[b] *Department of Operations Research, Eötvös Loránd University, Museum rd 6–8, 1088 Budapest, Hungary*

E-mail: illes@cs.elte.hu

[c] *Interfacultary Reactor Institute (IRI), Mekelweg 15, 2629 JB Delft, The Netherlands*

E-mail: ajdejong@iri.tudelft.nl; valko@iri.tudelft.nl; hoogenboom@iri.tudelft.nl

The loading pattern of fuel bundles in a reactor core determines the characteristics of the reloading cycle. The problem of maximizing fuel utilization during a cycle may be formulated as a nonlinear mixed integer programming (NLMIP) problem, but has until now mostly been treated by heuristic optimization methods like simulated annealing. This paper evaluates the performance of state-of-the-art nonlinear programming routines on a simplified model of this problem, and compares results with available heuristic solutions.

1. Introduction

A nuclear reactor is operated in cycles, the duration T being 1 year to $1\frac{1}{2}$ years in practice. An important consideration in operating the nuclear reactor is the design of a reloading pattern for each cycle. Light water reactors are usually *batch refueled*, i.e. equal quantities of fuel from different fuel batches are placed in the core. The different batches consist of fuel with varying degrees of *burn-up*. One typically has three batches of fuel: fresh, once burnt (used once in an earlier cycle), and twice burnt.

★ On leave from the Eötvös Loránd University, Budapest, and partially supported by OTKA No. 2116.
☆ Visited the Optimization Centre of the Delft University of Technology from 1.2.1994 to 1.5.1994, sponsored by Peregrinatio II Foundation, Budapest, and partially supported by OTKA No. 14302.

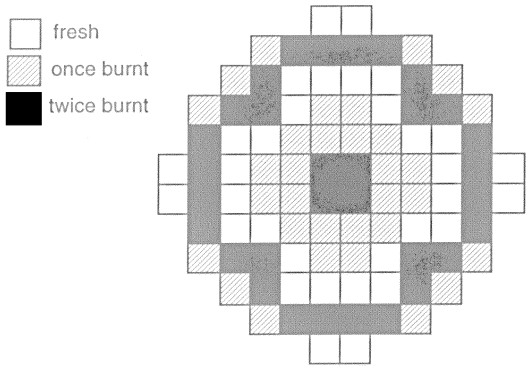

Figure 1. Example of a nuclear core with a three fuel batch loading pattern.

At the end of the cycle, the oldest batch is discharged from the reactor as spent fuel (figure 1).

After a cycle, the discarded fuel elements are replaced by fresh ones. All the fuel elements are now reshuffled to obtain the original loading pattern. The fuel elements which had been fresh at the start of the cycle are moved to the positions reserved for once burnt elements in the loading pattern, etc. If the same procedure of replacing and reshuffling is repeated for a number of years, an equilibrium cycle is reached [9]. Once a reloading pattern is chosen, it can therefore be repeated. The objective becomes to find a pattern that is optimal in some sense.

The most common objective is to maximize the *discharge burn-up*, with the underlying idea of maximal fuel utilization while satisfying nominal power demand. The resulting optimization problem is nonlinear mixed integer (NLMIP). Several optimization approaches to this problem have been tried, with the recent trend towards local search heuristics.

1. Successive linear programming [3, 19].
2. Local search by pairwise interchange of fuel bundles (see [10, 12, 14, 20]).
3. Simulated annealing [4, 15, 22].

Surveys of optimization approaches in reactor physics may be found in [11, 16].

A natural question is whether the advances in software for nonlinear programming may be utilized for this problem. In this paper, a simplified model of the reactor reloading problem is formulated as a nonlinear mixed integer optimization problem. The optimization formulation allows evaluation of the nonlinear mixed integer solver Dicopt [23], as well as a convex branch and bound algorithm in conjunction with state-of-the-art nonlinear (NLP) solvers, like Minos5 [13] and Conopt [6]. These optimization results are compared to known heuristic solutions, obtained by pairwise interchange of fuel bundles. The results show that the optimization approach

consistently gives the best known solutions, even though the heuristic solutions seem to be surprisingly good.

This paper is organised as follows. Section 2 introduces the physical parameters and governing equations of equilibrium cycle calculations, using the simplified model of De Jong [9]. This model is also described to a large extent by Driscoll et al. in [5]. In section 3, the optimization problem is formulated, while section 4 gives some upper bounds for the discharge burn-up in the optimization problem. In sections 5 and 6, numerical results are given for two classes of instances.

2. The reloading parameters

We now introduce two parameters which play a role in the equilibrium cycle of a reloading pattern. The reloading pattern places equal numbers of fuel elements from n batches in N positions in the core, where N is a multiple of n. The first parameter is the average *power densities* vector $p = [p_1,\ldots,p_n]^T$ corresponding to the fuel batches, and the second is the vector of *fuel infinite multiplication factors* of the fuel batches, k_j, ($j = 1,\ldots,n$). The parameter gives the ratio of production of neutrons to the loss of neutrons for the different fuel batches. It refers to an infinite array of identical fuel assemblies, thus excluding neutron losses through leakage out of the reactor. The fuel infinite multiplication factors depend only on the material composition of the fuel assembly during operation and not on the geometry of the reactor or the position of the material therein.

The parameters k_j are closely related to the fuel densities of the fuel batches and decrease with time t during the cycle. The parameter $k_j(0)$, ($j = 1,\ldots,n$) represents **b**egin **o**f **c**ycle (BOC) fuel at time $t = 0$ and $k_j(T)$, ($j = 1,\ldots,n$) represents **e**nd **o**f **c**ycle (EOC) fuel at $t = T$.

The EOC values $k_j(T)$ and parameters p_j appear in three sets of equations, to be discussed in turn.

2.1. Burn-up equations

The first set of equations gives the EOC values of $k_j(T)$ as a function of batch power p_j. The linear approximation is used for this in [9]:

$$k_j(T) = k_j(0) - \alpha \frac{p_j P_{tot}}{M} T \tag{1}$$

for all batches $j = 1,\ldots,n$. In this equation, $\alpha > 0$ is a fuel characteristic constant, M is the (constant) initial mass of a fresh fuel element, and P_{tot} is the (constant) total capacity of the reactor. The burn-up equations give an indication of the reduction in "fuel density" of the different batches during a cycle.

2.2. Equilibrium equations

The second set of equations gives *equilibrium cycle conditions*. The EOC fuel of batch 1, $k_1(T)$, becomes BOC fuel for batch 2, $k_2(0)$, etc. For an equilibrium cycle it therefore holds that

$$k_j(0) = k_{j-1}(T) \tag{2}$$

for $j = 2,\ldots,n$. The value of $k_1(0)$ for the fresh fuel batches is always the same, and $k_n(T)$ is discharged from the reactor as spent fuel. It is desirable to have as little fuel as possible in the discarded batch, i.e. to have maximal fuel utilization. The objective function of reloading pattern design is therefore usually to minimize $k_n(T)$ over all possible reloading patterns.

From combining (1) and (2), we find a system of n equations:

$$\begin{aligned}
k_1(T) &= k_1(0) - \frac{\alpha P_{tot} T}{M} p_1, \\
k_2(T) &= k_1(0) - \frac{\alpha P_{tot} T}{M} (p_1 + p_2), \\
&\vdots \\
k_n(T) &= k_1(0) - \frac{\alpha P_{tot} T}{M} (p_1 + \ldots + p_n) = k_1(0) - \frac{\alpha P_{tot} T}{M}.
\end{aligned} \tag{3}$$

In the last equation of (3), the normalization of the power fractions $\sum_j p_j = 1$ is used. The quantity

$$\alpha B_d \equiv k_n(T) - k_1(0)$$

is called the discharge burn-up, and gives the reduction in "fuel density" of the oldest batch during a cycle. Furthermore, it follows from the last equation of (3) that αB_d depends linearly on the cycle length T:

$$\alpha B_d = \frac{\alpha P_{tot} T}{M}.$$

This shows that the cycle length is maximized by maximizing the discharge burn-up: if the reloading pattern can be improved to increase discharge burn-up by a given percentage, then the cycle length increases with the same percentage.

A full-sized nuclear power plant produces a yearly amount of electricity worth about $500 million. An improved reloading pattern yielding as little as 1% more power from the same amount of fuel can therefore result in savings of several million dollars per year.

2.3. Coupling eigenvalue equation

The third set of equations concerns the $(N \times N)$ *fast group coupling coefficient matrix* G [7], where G_{rs} $(r, s = 1,\ldots,N)$ is the probability that a neutron produced at

fuel location (or *bundle*) s will be absorbed in bundle r. This leads to the following eigenvalue equation for the vector ϕ of *fast neutron fluxes* [7], ϕ_i ($i = 1,\ldots,N$) in the bundles:

$$\phi = \frac{1}{k_{eff}} GK\phi, \tag{4}$$

where K is a diagonal matrix with the infinite fuel multiplication factors on the diagonal which depends on the loading pattern as follows:

$$K_{ii} = k_j \text{ if fuel from batch } j \text{ is placed in bundle } i. \tag{5}$$

Note that K is uniquely determined by a given pattern – since each bundle (or fuel location) can contain one fuel bundle, each k_j, ($j = 1,\ldots,n$) appears exactly N/n times on the diagonal of K.

The eigenvalue k_{eff} is introduced in (4), since for an arbitrary arrangement of fuel elements in the core, the production of neutrons will not balance the loss of neutrons due to absorption and leakage. The parameter k_{eff} is simply the ratio between these two quantities. To have a stationary operating reactor, k_{eff} must be equal to unity throughout the operation cycle. The reactor is then said to be critical. In practice, this is accomplished by introducing sufficient neutron absorbers in the core, e.g. control rods. The cycle ends when all control absorbers are removed and the reactor can no longer be kept critical.

Only the EOC values for the infinite multiplication factors are of interest here, and these appear in (3) and in (4), where the latter is evaluated at EOC, i.e. at time $t = T$.

Each fuel bundle has an associated *nodal power* value defined as

$$P_i = K_{ii}\phi_i, \qquad i = 1,\ldots,N. \tag{6}$$

The batch power values p_j are simply the summed nodal powers taken over bundles containing fuel from batch j. More formally,

$$p_j = k_j \sum_{i \in I_j} \phi_i, \quad j = 1,\ldots,n, \tag{7}$$

where I_j is the index set of bundles containing fuel from batch j.

We have now given the governing equations which determine the discharge burn-up for a given loading pattern. The next step is to formulate an optimization problem to find an optimal loading pattern which maximizes the burn-up.

3. Formulating an optimization problem

To model the set of allowable loading patterns, we introduce the binary variables

$$\xi_{ij} = \begin{cases} 1 & \text{if bundle } i \text{ contains fuel type } j, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

A feasible pattern must therefore satisfy the following *transport equations*:

$$\sum_{i=1}^{N} \xi_{ij} = N/n, \quad j = 1, \ldots, n,$$
$$\sum_{j=1}^{n} \xi_{ij} = 1, \qquad i = 1, \ldots, N. \tag{9}$$

The first equation states that equal amounts of fuel from each of the n batches should be placed in the N bundles. The second equation states that only one type of fuel is placed in a given bundle.

We associate an $(N \times n)$ matrix $X = [\xi_{ij}]$ with a given pattern, and define

$$[W(X, \phi)]_{ij} = [\phi_i \xi_{ij}], \quad i = 1, \ldots, N, \quad j = 1, \ldots, n. \tag{10}$$

In other words, W is the matrix obtained by multiplying each row i of X by ϕ. Using the vector $k(T)$ associated with the different fuel batches, we can construct the diagonal matrix K in (5) as before: $[K]_{ii} = k_j(T)$ if fuel from batch j is placed in bundle i. It is easy to show that $K\phi = W(X, \phi)k(T)$, where ϕ denotes the vector of fluxes. We can therefore rewrite the coupling eigenvalue equation (4) as

$$\phi = GW(X, \phi)k(T). \tag{11}$$

Equation (11) is a trilinear equation in the binary variables ξ_{ij} and the nonnegative variables ϕ_i and $k_j(T)$. Using the new notation, equations (7) become the system of trilinear equations

$$p_j = k_j(T) \sum_{i=1}^{N} \phi_i \xi_{ij}, \quad j = 1, \ldots, n, \tag{12}$$

where we require that the values p_j be normalized:

$$\sum_{j=1}^{n} p_j = 1. \tag{13}$$

Additionally, we have *nodal power peaking* constraints which limit the nodal power density to avoid fuel melt. The constraints take the form

$$P_i \leq \frac{f_{lim}}{N} \sum_{l=1}^{N} P_l, \qquad i = 1, \ldots, N, \tag{14}$$

where the nodal power can be expressed as

$$P_i = \phi_i \sum_{j=1}^{n} \xi_{ij} k_j(T), \tag{15}$$

using the new notation, and f_{lim} is a constant. The value $f_{lim} = 1.8$ is characteristic for the *Borssele nuclear power plant*, a medium sized reactor in the Netherlands for which the calculations in the following sections are done. The nodal power peaking constraints are also trilinear. Since

$$\sum_{l=1}^{N} P_l = \sum_{j=1}^{n} p_j = 1$$

by (13), it follows that (14) simplifies to

$$P_i \leq \frac{f_{lim}}{N}. \tag{16}$$

In other words, the nodal powers may not exceed a critical value.

The variables in our optimization problem are the components of the vectors ϕ, p, P, and $k(T)$, as well as B_d, and the binary variables ξ_{ij}.

The objective function to be minimized is the discharge fuel:

$$k_n(T) = k_1(0) - \alpha B_d,$$

or alternatively the discharge burn-up αB_d can be maximized.

To summarize, we state the optimization model concisely:

Optimization problem

$$\begin{aligned}
&\underset{\phi, X, k(T), p, P, B_d}{\text{maximize}} \quad \alpha B_d \\
&\text{subject to} \quad \sum_{i=1}^{N} \xi_{ij} = N/n, && j = 1, \ldots n, \\
&\qquad \sum_{j=1}^{n} \xi_{ij} = 1, && i = 1, \ldots, N, \\
&\qquad \phi = GW(X, \phi)k(T), \\
&\qquad p_j = k_j(T) \sum_{i=1}^{N} \phi_i \xi_{ij}, && j = 1, \ldots, n, \\
&\qquad \sum_{j=1}^{N} p_j = 1, \\
&\qquad P_i = \phi_i \sum_{j=1}^{n} \xi_{ij} k_j(T), && i = 1, \ldots, N, \\
&\qquad P_i \leq f_{lim}/N, && i = 1, \ldots, N, \\
&\qquad k_j(T) = k_1(0) - \alpha B_d \sum_{l=1}^{j} p_l, && j = 1, \ldots, n,
\end{aligned}$$

$$
\begin{aligned}
p_j &\geq 0, \qquad && j = 1, \ldots, n, \\
k_j(T) &\geq 0, && j = 1, \ldots, n, \\
\phi_i &\geq 0, && i = 1, \ldots, N, \\
\xi_{ij} &\in \{0, 1\}, && i = 1, \ldots, N; \quad j = 1, \ldots, n.
\end{aligned}
$$

A few remarks about this model:

1. The optimization problem is a nonlinear mixed integer programming problem, where the nonlinear constraints are bilinear and trilinear. The number of variables equals $nN + 2(n + N) + 1$ and the number of constraints $5(N + n) + 1$.
2. The problem is non-convex, i.e. a global optimization problem which may have a large number of local optima.
3. For the purpose of computation, the integrality constraints $\xi_{ij} \in \{0, 1\}$ are relaxed to $0 \leq \xi_{ij} \leq 1$, as is customary. The solution of the resulting problem will be referred to as the *relaxed solution*. This relaxed solution has the following physical interpretation: The value ξ_{ij} gives the fraction of fuel type j which is placed in bundle i. That is, the relaxed solution allows a mixture of fuel types in a given bundle.
4. The optimal integer solution need not be an integer "neighbour" of the relaxed solution, but may be arbitrarily far removed from it. One way of searching for a globally optimal integer solution is to "branch and bound" through the whole $0-1$ tree. Even this is not guaranteed to work – a global optimization problem must be solved at each node in the $0-1$ tree, and it is in general not possible to guarantee the global optimality of these solutions. It may happen that a node is "killed" incorrectly if a "bad" local minimum is found for the corresponding subproblem.

4. Upper bounds for the discharge burn-up

The nonconvexity of the relaxed problem makes it difficult to quantify the "goodness" of obtained solutions. Known bounds on the optimal burn-up value are therefore of importance. In this section, some known worst-case bounds are given, and suggestions for alternative (convex) relaxations with guaranteed global optimality are made.

De Jong shows in [9] that the following expression for the discharge burn-up holds:

$$
\alpha B_d = \frac{k_1(0) - (1 + L)}{\frac{1}{2}(1 + \sum_{i=1}^{n} (p_i)^2)}, \tag{17}
$$

where $L = 1 - \sum_{i=1}^{n} \phi_i$ is a first-order approximation to the *out of core leakage probability* [9]. Physical considerations require that $L > 0$ as $L = 0$ corresponds to an infinite reactor. To prove that $L > 0$ holds for this model, we take the l_1-norm on both sides of equation (4):

$$\|\phi\|_1 \equiv \sum_{i=1}^{N} \phi_i = \|GW(X,\phi)k\|_1 \leq \|G\|_1 \|W(X,\phi)k\|_1 = \|G\|_1 \sum_{j=1}^{n} p_j = \|G\|_1,$$

by (10), (12) and (13), where $\|G\|_1 = \max_{1\leq j\leq N} \sum_{i=1}^{N} g_{ij}$. This implies

$$L \geq 1 - \|G\|_1. \tag{18}$$

From the definition of G in terms of probabilities it follows that $\|G\|_1$ is strictly smaller than one, which shows that $L > 0$. Hence, (18) and $p_1 = \ldots = p_n = 1/n$ determine the following upper bound in (17):

$$\alpha B_d \leq \frac{k_1(0) - (2 - \|G\|_1)}{\frac{1}{2}(1 + 1/n)}. \tag{19}$$

The weaker bound

$$\alpha B_d \leq \frac{k_1(0) - 1}{\frac{1}{2}(1 + 1/n)} \tag{20}$$

follows by taking $L = 0$ in (17).

By using a similar argument as above and utilizing the infinity norm $\|G\|_\infty = \max_{1\leq i\leq N} \sum_{j=1}^{N} g_{ij}$, one obtains the upper bound $\phi_i \leq f_{lim}/N$, $i = 1,\ldots,N$ on the neutron fluxes from (4) and (16).

The relaxation described in the previous section does not guarantee an upper bound on the objective, as the resulting problem is nonconvex. The Shor relaxation [17] of the problem may alternatively be used, with a guaranteed upper bound on the optimal burn-up value. The Shor relaxation was introduced as a way to obtain bounds for indefinite quadratic problems. The problem presented here can be rewritten as a quadratic problem as all the nonlinear constraints are multilinear. The Shor relaxation can be formulated as a semidefinite programming problem, for which a new class of efficient polynomial time algorithms is available [21].

Unfortunately, no guarantee can be given that the Shor relaxation yields a useful bound for this problem, but it remains a promising option for future research.

5. Computational results

We consider a specific numerical instance of the optimization problem with $N = 12$ bundles and $n = 2$, 3 or 4 different types of fuel to be placed in the bundles in some loading pattern. As before, the bundles must contain equal amounts of each fuel type. The total number of possible patterns is given by $N!/((N/n)!)^n$ and is shown in table 1 for this instance. The number of possible patterns is too large in general to allow complete enumeration in a reasonable time. Moreover, the possible patterns increase exponentially with increasing problem size.

The results presented here were obtained by using the GAMS software package [1]. GAMS (General Algebraic Modelling System) is a high level language,

Table 1

Number of possible loading patterns for different batch sizes.

N = 12 bundles	
Batches	Possible patterns
2	924
3	34650
4	369600

which enables easy statement of an optimization problem. Symbolic differentiation of nonlinear functions, efficient storage, etc. are done automatically, and state-of-the-art optimization solvers are incorporated in the software. These include well-known solvers like the *sequential linearly constrained programming* (SLC) solver Minos5 [13] and the *generalized reduced gradient method* (GRG) solver Conopt [6]. The results for the relaxed problems presented here were obtained by using the solver Conopt. Conopt proved to be more robust on these problems than Minos5. Minos5 could not obtain a feasible solution from the same range of starting points as Conopt could. Moreover, problems in maintaining feasibility were experienced with Minos5 for some problems. These remarks are consistent with the general view that Conopt often outperforms Minos5 on problems with mostly nonlinear constraints, as is the case here.

The GAMS results are compared to *local search heuristic* results of de Jong [9], where an integer solution is found by performing all improving 2-bundle pairwise interchanges (PI). These heuristic solutions are the best patterns obtained after a number of trails from different starting patterns. As such, it is difficult to assign a computational time value to the PI procedure which can reasonably be compared to optimization solvers. We therefore omit such a comparison and focus on the quality of obtained solutions, although CPU times for the solvers are given as an indication of the computational effort involved.

Table 2 shows the best heuristic PI solutions in [9] obtained from different starting points. The loading patterns are given as rows of 12 numbers, where value j of the ith element in the row indicates that fuel from batch j is placed in bundle i. Note that the heuristic solutions are dependent on the starting patterns in general.

Table 3 gives the GAMS solutions by the NLMIP solver Dicopt [23], and the best relaxed solution in each instance is also shown. The mixed integer (MIP) subproblems generated by Dicopt were solved by the solver OSL [24], while the relaxed nonlinear problems were solved by Conopt. The MIP solver OSL was chosen over the MIP solver ZOOM [18], as the default branching rule of the latter solver required too much memory for storage of the node table for the four batch problems.

Table 2

Reloading patterns obtained by the PI heuristic from different starting patterns. The discharge burn-up corresponding to each pattern is shown.

Batches	Starting pattern	Final pattern	Burn-up
2	1 1 1 1 1 1 2 2 2 2 2 2	2 1 2 1 1 1 1 2 2 2 1 2	0.2115
2	2 2 2 2 2 2 1 1 1 1 1 1	2 2 1 1 1 1 2 2 1 2 2 1	0.2119
3	1 1 1 1 2 2 2 2 3 3 3 3	3 1 3 1 2 1 2 2 1 3 3 2	0.2336
3	3 3 3 3 2 2 2 2 1 1 1 1	2 3 2 1 1 1 3 3 3 2 2 1	0.2394
4	1 1 1 2 2 2 3 3 3 4 4 4	3 4 1 2 1 2 1 3 4 4 3 2	0.2503
4	4 4 4 3 3 3 2 2 2 1 1 1	3 4 1 3 1 2 2 2 4 3 4 1	0.2496

Table 3

Reloading patterns obtained by the solver Dicopt with corresponding burn-up values and burn-up values for the relaxed problems.

Batches	Reloading pattern	Burn-up	Relaxation
2	2 2 1 2 1 1 2 2 1 2 1 1	0.2072 (6.86s)	0.2165 (0.65s)
3	2 3 2 1 1 1 3 2 3 3 2 1	0.2383 (14.67s)	0.2430 (2.60s)
4	3 2 3 1 1 4 1 4 2 3 4 2	0.2512 (41.53s)	0.2586 (3.38s)

Table 4

Reloading patterns obtained by branch and bound with corresponding burn-up values. The number of nodes evaluated in the b&b tree is shown.

Batches	Reloading pattern	Burn-up	Relaxation	b&b nodes
2	2 2 1 1 1 1 2 2 1 2 2 1	0.2119 (9.85s)	0.2165	20
3	2 2 3 1 1 1 3 2 3 3 2 1	0.2394 (33.88s)	0.2430	26
4	3 2 4 1 1 2 3 4 4 3 2 1	0.2557 (193.9s)	0.2586	81

The CPU times for the solvers are given for execution on a HP-9000 workstation with 48MB memory. Since the Dicopt solutions do not compare favourably with the heuristic solutions, a convex branch and bound implementation in the GAMS language was also done. Table 4 gives the convex branch and bound solutions. The solver Conopt was used to solve the subproblems. The branching was done over all solutions of the transport equations, by using the following simple algorithm:

Step 1. For $i = 1,\ldots,N$ do

Step 2. Live nodes correspond to fixed 0–1 values for ξ_{ij}, $\hat{i} < i$. In other words, a live node corresponds to some (potentially optimal) allocation of fuel types to bundles $1,\ldots,i-1$. Now loop over all live nodes:

(a) For $j = 1,\ldots,n$ do
  (i) Fix $\xi_{ij} = 1$ and $\xi_{ij} = 0$, $k \neq j$. In other words, place fuel j in bundle i.
  (ii) Solve the optimization problem. If the objective is worse than the best known integer solution, then "kill" the current node. Otherwise add the node to the live node table.

The best heuristic solution was used as an initial lower bound for the branch and bound procedure. The number of nodes in the 0–1 tree which were evaluated before obtaining the optimal solution is also shown in the table.

Table 5 summarizes the above results. The column marked "Bound" gives the upper bound from (19), and the column "Relaxed" gives the relaxed solution.

Table 5

Comparison of burn-up values given by different solution strategies. The best known relaxed solution and best known upper bound on the optimal burn-up is given.

Batches	PI1	PI2	Dicopt	b&b	Relaxed	Bound
2	0.21154	0.21193	0.20719	0.21193	0.21649	0.2556
3	0.23362	0.23936	0.23833	0.23938	0.24296	0.2875
4	0.25026	0.24961	0.25116	0.25565	0.25864	0.3067

Remarks on tables 2 to 5:

1. For two batches ($n = 2$), both (PI) solutions have better objective values than the Dicopt solution. The (PI) objectives are also within 1% of the best relaxed solution obtained by GAMS. The branch and bound gives the same pattern as the best (PI) solution.
2. For three batches ($n = 3$), the GAMS nonlinear mixed integer solver Dicopt produced a pattern identical to the result in table 4 except that the fuel types in bundles 8 and 10 were switched. The Dicopt solution can therefore be improved by PI to obtain the best PI solution. Note, however, that the Dicopt solution is superior to the second best PI solution, illustrating again that the heuristic solutions are dependent on the starting configurations. The best heuristic solution differs from the relaxed solution by less than 2%. The branch and bound yielded a slightly superior solution to the best PI solution. It is of interest to note that two completely different patterns correspond to virtually the same objective value.
3. The branch and bound solution for four batches is roughly 2% better than the best PI solution. It seems, therefore, that the branch and bound procedure may be more efficient than PI for larger problems, but larger problem instances will have to be evaluated to confirm this. An improvement of 2% may seem small, but is financially significant in view of the remarks in section 3.

4. The branch and bound solution is always within 2% of the corresponding relaxed solution. If global optimality of the relaxed solutions is assumed, a measure of goodness of the loading patterns is obtained. In other words, the assumption implies that no patterns exist with burn-up more than 2% higher than the branch and bound solutions. In computational trails, the same optimal value of the relaxed problem is always obtained, but for different local minima. This may suggest global optimality of the relaxed solutions, but attempts to prove this have so far been unsuccessful.

   Unfortunately, the bounds from (19) are not tight, as is clear from table 5. This further stresses the importance of pursuing new strategies to find globally optimal relaxed solutions, as suggested in section 4.

6. A second class of problem instances

We now consider the case where the bundles in the core have different volumes. In particular, full and half sized bundles may be used. In figure 2, a model of the *Borssele* reactor core with 96 bundles is shown.

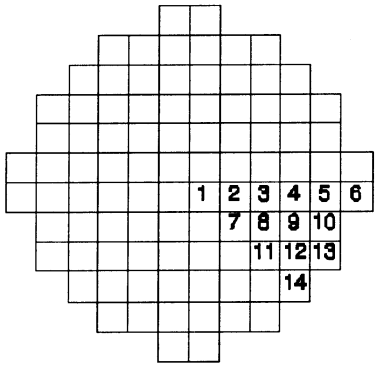

Figure 2. 96-bundle nuclear core.

Octant symmetry of the reloading pattern is required, and only the bundles marked 1,…,14 are therefore considered in this case. The bundles 1, 7, 11, 14 are "half-bundles", divided in two by the diagonal and with equal halves in two octants. The problem remains to distribute equal amounts of fuel from the different batches among the bundles. If a given fuel type is assigned to a half bundle, the same fuel type must also be assigned to some other half bundle.

The optimization model must be modified only slightly to allow for the half-bundles. The transport equations (9) become

$$\sum_{i=1}^{N} v_i \xi_{ij} = N/n, \quad j = 1, \ldots, n,$$

$$\sum_{j=1}^{n} \xi_{ij} = 1, \qquad i = 1, \ldots N,$$

where v_i is the volume (one or one half) of bundle i.

The nodal power densities P_i must also be multiplied by the nodal volume to obtain the nodal power. This also holds for the *batch power densities* p_j, $j = 1,\ldots, n$. The new equations are

$$p_j = k_j(T) \sum_{i=1}^{N} v_i \phi_i \xi_{ij}, \qquad j = 1, \ldots, n.$$

$$\sum_{j=1}^{N} p_j = 1,$$

$$P_i = v_i \phi_i \sum_{j=1}^{n} \xi_{ij} k_j(T), \qquad i = 1, \ldots N,$$

$$P_i \leq f_{lim}/N, \qquad i = 1, \ldots, N.$$

The other equations in the previous optimization formulation remain unchanged. The matrix G is, however, now defined differently [9], and is no longer bounded by $\|G\|_\infty < 1$ and $\|G\|_1 < 1$. The bound in (20) still holds, if $L \geq 0$ is added as a constraint to the optimization problem.

7. Results for the second class of instances

We consider loading patterns with two, three and four different fuel types (batches). Figures 3, 4 and 5 reproduce the PI heuristic solution for two fuel batches as obtained in [9]. In the figures, the bundles are shaded according to the fuel type – the fresh fuel is the lightest and the most burnt fuel the darkest. (For two-fuel batches, all the bundles are therefore black or white). The three different starting points therefore yield the same PI solution. The GAMS solver Dicopt and the branch and bound strategy obtain the same solution.

The heuristic solution for three batches depends on the starting pattern, as can be seen in figures 6, 7 and 8. The pattern produced by Dicopt for three-fuel batches is shown in figure 9. To give some idea of the associated physical characteristics of the loading pattern, the corresponding nodal power graph and after-cycle fuel density graph are also shown. The power graph shows that the highest power corresponds to the freshest fuel (as may be intuitively expected). For example, there is a central region of relatively low power values, as the pattern has older fuel in the central bundles. (Patterns with fresh fuel in the middle violate the nodal power peaking

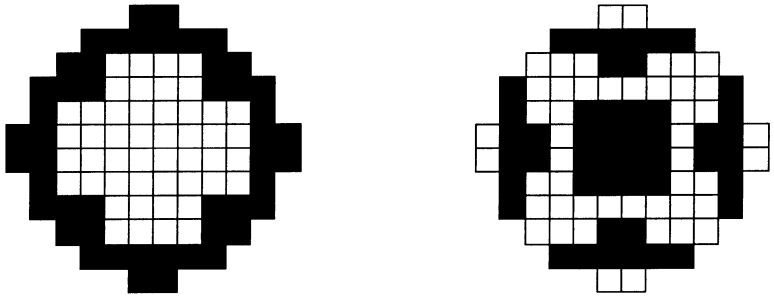

Figure 3. First heuristic solution: 2-batch starting pattern with final solution.

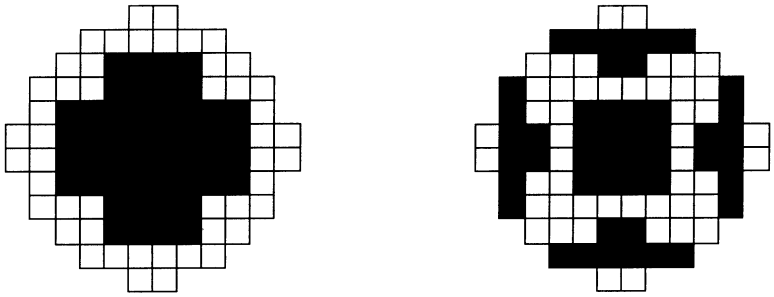

Figure 4. Second heuristic solution: 2-batch starting pattern with final solution.

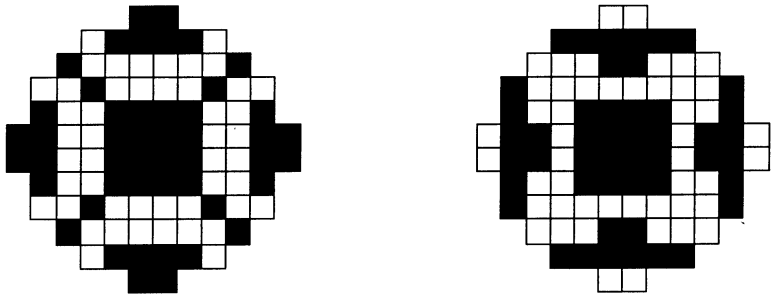

Figure 5. Third heuristic solution: 2-batch starting pattern with final solution.

constraints.) The fuel density graph shows that lowest density is associated with the oldest fuel.

The branch and bound procedure for three fuel batches again yielded the same pattern as the best (PI) solution.

Before we quantitatively compare the different solutions, we give the results for four batches (figures 10 to 14).

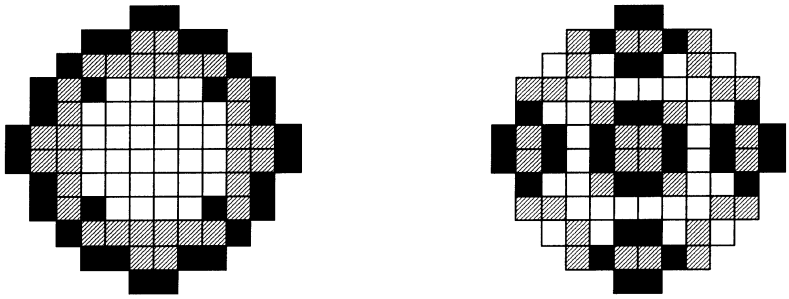

Figure 6. First heuristic solution: 3-batch starting pattern with final solution.

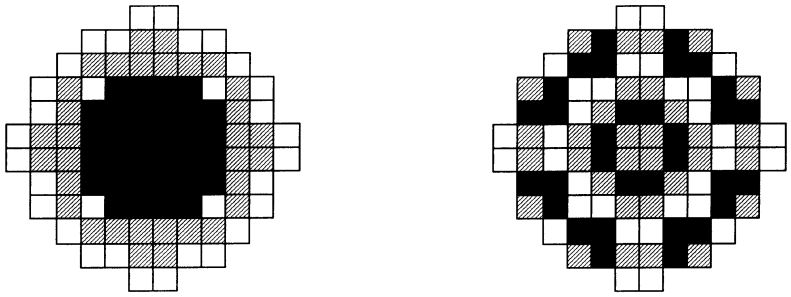

Figure 7. Second heuristic solution: 3-batch starting pattern with final solution.

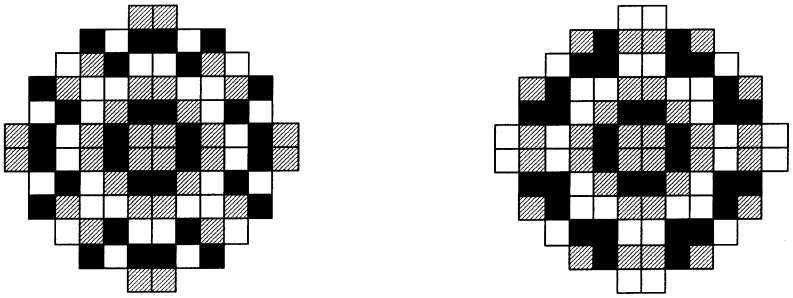

Figure 8. Third heuristic solution: 3-batch starting pattern with final solution.

More details on the branch and bound solutions are given in table 6. Once again, only a small percentage of the total nodes had to be evaluated.

Table 7 now compares the heuristic, Dicopt, and branch and bound solutions. The solutions times as reported by GAMS are included in brackets where relevant. The upper bound from (20) is given in the column marked “Bound”.

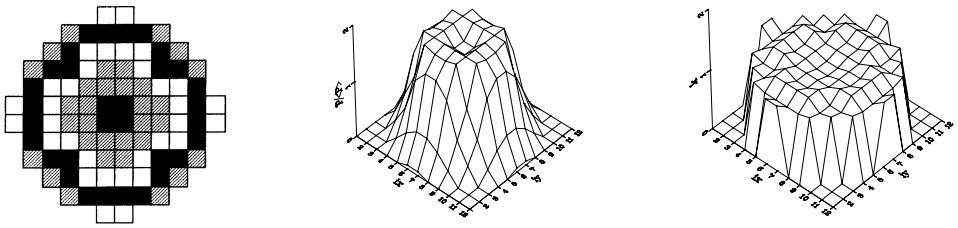

Figure 9. Reloading pattern obtained by the solver Dicopt for the 3-batch problem, with nodal power and fuel density graphs.

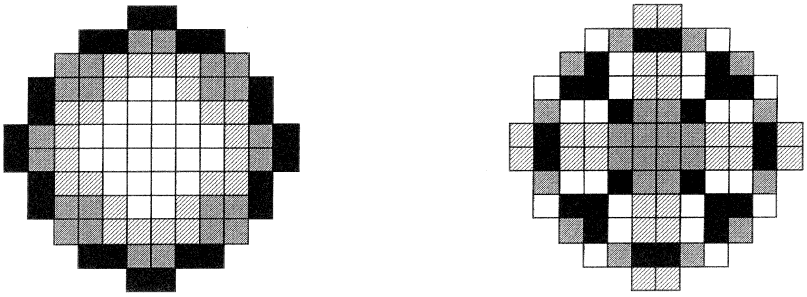

Figure 10. First heuristic solution: 4-batch starting pattern with final solution.

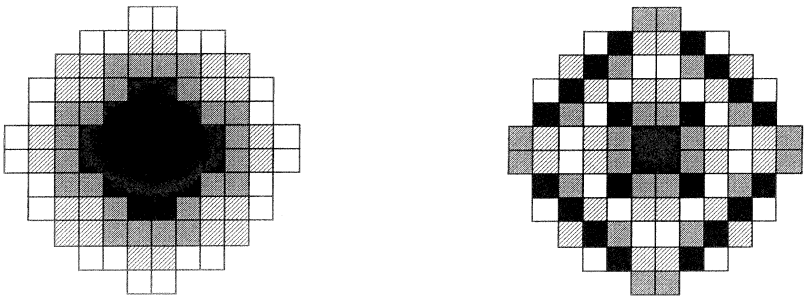

Figure 11. Second heuristic solution: 4-batch starting pattern with final solution.

Some remarks on table 7:

1. For 2 batches, the same solution was obtained by all three solution methods.
2. For 3 batches, the Dicopt solution was different from the PI solutions, but almost equally as good. Moreover, it could not be further improved by the heuristic. The branch and bound yielded the best PI solution.

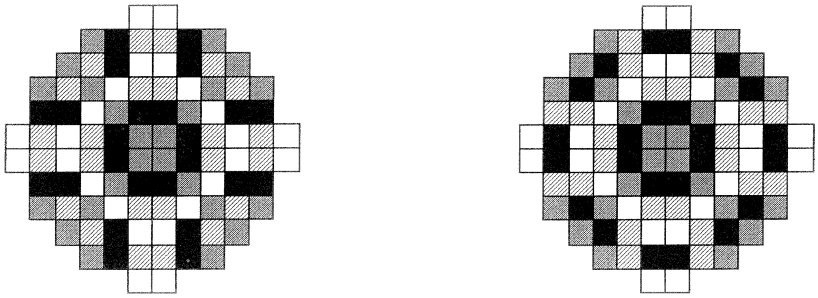

Figure 12. Third heuristic solution: 4-batch starting pattern with final solution.

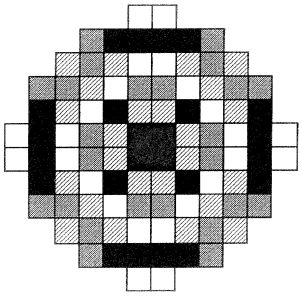

Figure 13. Reloading pattern obtained by branch and bound with the Conopt solver for the 4-batch problem.

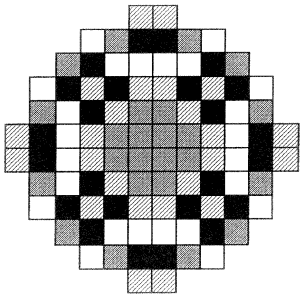

Figure 14. Reloading pattern obtained by the solver Dicopt for the 4-batch problem.

Table 6

Burn-up values for reloading patterns obtained by convex branch and bound with the solver Conopt. The relaxed solution, number of possible patterns, and number of nodes evaluated in the b&b tree are shown.

Batches	Burn-up	Relaxation	Possible patterns	b&b nodes
2	0.2101 (29.68s)	0.2141 (0.70s)	1932	55
3	0.2369 (123.8s)	0.2399 (1.350s)	85050	114
4	0.2518 (795.9s)	0.2553 (2.74s)	974400	271

Table 7

Burn-up values for reloading patterns obtained via different solution strategies. For each problem the best known relaxed solution and upper bound on the optimal value are shown.

Batches	PI1	PI2	PI3	Dicop	b&b	Relaxation	Bound
2	0.21008	0.21008	0.21008	0.21008 (13.84s)	0.21008	0.21406	0.2579
3	0.23421	0.23687	0.23687	0.23683 (43.41s)	0.23687	0.23991	0.2901
4	0.25141	0.24937	0.25047	0.25006 (311.4s)	0.25178	0.25526	0.3094

3. For 4 batches, the Dicopt algorithm yielded the pattern in figure 14. This result was obtained by Dicopt using the nonlinear solver Conopt and the mixed integer programming solver OSL [24]. The associated burn-up is slightly worse than the best obtained, which again shows that several patterns with similar burn-up values are possible.

   The solution obtained by branch and bound was superior to the best PI result, by about one percent. As discussed earlier, an improvement of this size can be viewed as significant.

8. Conclusion

A simplified mathematical model for describing reloading pattern design in a nuclear reactor has been formulated here as a nonlinear mixed integer optimization problem (NLMIP). The objective is to maximize the length of operating cycle by maximizing the burn-up of the fuel to be discharged. Two optimization strategies have been compared to known heuristic solutions obtained by pairwise interchange (PI) of fuel bundles.

The two optimization methods are a new NLMIP solver Dicopt, and a nonlinear branch and bound (b&b) algorithm, both employing state-of-the-art nonlinear programming (NLP) solvers. The solvers Dicopt (NLMIP) and Conopt (NLP) (used in b&b) proved sufficiently robust to generate integer solutions.

Notable is that the different methods often generate different loading patterns with only small differences in objective values. This aids the goal of finding a set of "good loading patterns", from which one can be chosen from other considerations or used as trial solutions in more complicated mathematical models.

For the largest problems, our implementation of the b&b with Conopt produces the best solutions. Surprisingly, the heuristic (PI) solutions proved competitive with the b&b solutions, and in general better than the Dicopt solutions. Nevertheless, the improved b&b solutions correspond to increases in operating cycle length of up to 2%, which would result in large yearly monetary savings if it could be realized in practice.

The relaxed solutions by Conopt give an upper bound on the optimum under the assumption of global optimality of the relaxed solution. The best integer solutions obtained are within 2% of this bound in each case.

It remains an open problem to prove global optimality for the best obtained integer solutions, or to obtain a tight upper bound on the optimal value. A promising future option for obtaining a good upper bound is to solve the Shor dual of the optimization problem.

References

[1] A. Brooke, D. Kendrick and A. Meeraus, *GAMS User's Guide, Release 2.25*, The Scientific Press, San Francisco, USA, 1992

[2] Y.A. Chao, C.W. Hu and C.A. Suo, A theory of fuel management via backward diffusion calculation, Nuclear Science and Engineering 93(1986)78–87.
[3] K. Chitcara and J. Weisman, An equilibrium approach to optimal in-core fuel management for pressurized water reactors, Nuclear Technology 24(1974)33–49.
[4] D.J. Cropaczek and P.J. Turinski, In-core nuclear fuel management optimization for pressurized water reactors utilizing simulated annealing, Nuclear Technology 95(1991)9–32.
[5] M.J. Driscoll, T.J. Downar and E.E. Pilat, The linear reactivity model for nuclear fuel management, American Nuclear Society, La Grange Park, Illinois, USA, 1990.
[6] A. Drud, CONOPT – A CRG code for large sparse dynamic nonlinear optimization problems, Mathematical Programming 31(1985)153.
[7] J.J. Duderstadt and L.J. Hamilton, *Nuclear Reactor Analysis*, Wiley, 1976.
[8] L.W. Ho and A.F. Rohach, Perturbation theory in nuclear fuel management optimization, Nuclear Science and Engineering 82(1982)151–161.
[9] A.J. de Jong, Reloading pattern design for batch refueled nuclear reactors, Report IRI-131-95-010, Interfacultary Reactor Institute, Delft University of Technology, Delft, The Netherlands, 1995.
[10] J. Kim, T.J. Downar and A. Sesonske, Optimization of core reload design for low leakage fuel management in pressurized water reactors, Nuclear Science and Engineering 96(1982)85–101.
[11] S.H. Levine, In-core fuel management educational module, Nuclear Technology 53(1981)303–325.
[12] B.I. Lin, B. Zolotar and B. Weisman, An automated procedure for selection of optimal refueling policies for light water reactors, Nuclear Technology 44(1979)258–275.
[13] B.A. Murtagh and M.A. Saunders, *MINOS 5.1 User's Guide*, Report SOL 83-20R, Stanford University, December 1983; revised January 1987.
[14] B.N. Naft and A Sesonske, Pressurized water reactor optimal fuel management, Nuclear Technology 14(1972)123–132.
[15] G.T. Parks, An intelligent stochastic optimization routine for in-core fuel cycle design, Transactions of the American Nuclear Society 57(1988)259–260.
[16] G.T. Parks, Advances in optimization and their applicability to problems in the field of nuclear science and technology, Advances in Nuclear Science and Technology 21(1990)195–253.
[17] N.Z. Shor, Quadratic optimization problems., Soviet Journal of Circuits and Systems Sciences 25(1987)1–11.
[18] J.A. Singhal, R.E. Marsten and T.L. Morin, Fixed order branch-and-bound methods for mixed-integer programming: The ZOOM System, Department of Management Information Systems, University of Arizona, Tucson, AZ 85721, Dec. 1987.
[19] J.A. Stillman and Y.A. Chao, The optimum fuel and power distribution for pressurized water reactors, Nuclear Science and Engineering 103(1989)321–333.
[20] R. Stout and A.H. Robinson, Determination of optimal fuel loadings in pressurized reactors using dynamic programming, Nuclear Technology 20(1973)86–102.
[21] L. Vanderberghe and S. Boyd, Positive definite programming, SIAM Review, to appear.
[22] F.C.M. Verhagen and M. van der Schaar, Simulated annealing in LWR fuel management, *TOPNUX'93 Proceedings 2*, 1993, pp. 37–39.
[23] J. Visvanathan and I.E. Grossman, A combined penalty function and outer approximation method for MINLP optimization, Computers and Chemical Engineering 14(1990)769–782.
[24] *Optimization Subroutine Library: Guide and Reference, Release 2*, 4th ed., IBM Corporation, New York, June 1992.