Finding Optimal Nuclear Reactor Core Reload Patterns Using Nonlinear Optimization and Search Heuristics

A.J. Quist, E. de Klerk, C. Roos, T. Terlaky
Faculty of Information Technology and Systems,
Delft University of Technology, Delft, The Netherlands
[a.j.quist,e.deklerk,c.roos,t.terlaky]@twi.tudelft.nl

R. van Geemert, J.E. Hoogenboom,
Interfacultary Reactor Institute,
Delft University of Technology, Delft, The Netherlands
[r.vangeemert, j.e.hoogenboom]@iri.tudelft.nl

T. Illés*
Department of Operations Research,
Eötvös Loránd University, Budapest, Hungary
illes@konig.elte.hu

January 19, 1998

Abstract

We use a simplified but still quite realistic model to the nuclear reactor fuel management problem to search for optimal fuel loading schemes. Several adaptations and model adjustments are described that make the model tractable for a general nonlinear mixed-integer solver. Results are compared with results from pairwise interchange optimization. Use of solutions from nonlinear mixed-integer optimization as starting values for local search heuristics leads to powerful optimization methods.

1 Introduction

An important issue in the operation of a nuclear reactor is the design of a reloading pattern for each cycle. Usually, a number of fuel bundles is discharged at the end of a cycle (EoC), and the same number of fresh bundles is inserted in the core, while all bundles are reshuffled to a configuration that is optimal with respect to some performance criterion. There are several strategies for reloading. We consider the situation in which at each reloading, the number of discharged bundles is the same, being one third or one fourth of the total number of bundles.

Having fixed the number of discharged bundles, it is possible to simply measure the remaining reactivity of the bundles at each EoC, and then find an optimal pattern using

*On leave from the Eötvös Loránd University, Budapest, and partially supported by OTKA No. T 14302 grant.

a subset of these bundles together with some fresh bundles. Another approach is to apply the same reloading pattern every year. After repeating this for a number of years, the reactor will reach an equilibrium state, in which (ideally) each cycle has the same characteristics. In this paper, we will consider such an equilibrium cycle strategy. A recent study comparing this equilibrium strategy to the more usual strategy of successive re-optimization at each EoC is made by Yamamoto [17].

Several optimization criteria can be chosen. One way is to search a loading pattern for which the cycle length will be maximal before the reactor becomes *sub-critical*, i.e. the number of neutrons produced is less than the number of neutrons lost by absorption or leakage from the reactor, which stops the reactor operation. Another approach is to minimize the leakage out of the core. In this paper, we fix the cycle length and search for a pattern for which the reactivity of the core at EoC is maximal, measured by a property called the uncontrolled effective multiplication factor. All the stated objectives are in some sense equivalent in an equilibrium cycle [17]: a core with maximal EoC reactivity, also would have the longest possible cycle length before becoming sub-critical. Furthermore, the leakage factor is correlated with the effective multiplication factor.

In the literature different methods to solve the reloading problem can be found. Several papers describe solution methods using genetic algorithms, simulated annealing [6], pairwise interchange and other neighborhood search heuristics [2]. Since the problem can be stated as an assignment problem with nonlinear side-constraints, it can also be viewed as a mixed integer nonlinear optimization problem. Variants using linear and nonlinear programming to solve subproblems have been applied in [16, 15]. Recent results have been published on a mixed-integer linear programming (MILP) approach [12], and also on a mixed-integer nonlinear programming (MINLP) approach [13, 18].

In our paper, we compare a pairwise interchange heuristic to algorithms using MINLP solvers. Some years ago it would have been impossible to solve the whole model as one big nonlinear discrete optimization problem. At present, NLP solvers have been improved to the extent that the results for realistic models compare favorably with combinatorial search heuristics. In addition, with this approach it is well possible to include continuous optimization problems in a natural manner. These include for example the addition of burnable poisons and the insertion of control rods.

This paper is organized as follows. In Section 2, we will give some general physical background and review neutron diffusion theory. In Section 3, this physical description is used to develop a mathematical optimization model and some special characteristics of the model are discussed. Section 4 describes the implemented MINLP procedure. Section 5 deals with computational results on some test problems.

2 Physical description

In a nuclear reactor core, the desired energy is obtained by a controlled fission chain reaction. The fission of a nucleus into two smaller nuclei is induced by the capture of a free neutron, and produces a large amount of energy, together with some new free neutrons. The state of the reactor depends on the number of free neutrons in the core. The neutron flux therefore plays a major role in models describing the reactor processes.
A popular way to describe the neutron flux in the reactor core is to consider the production, absorption and transport of neutrons as a diffusion process. This approach forms the basis of the model used in this paper, and will briefly be discussed here. More detailed descriptions can be found in e.g. [4, 10, 11].

2.1 Diffusion equations

The diffusion model describes how neutrons are produced, absorbed, and how they diffuse everywhere in the core. The diffusion coefficients and production and absorption rates depend on the material in the core, and also on the 'energy-level' of the neutrons. Neutrons at a higher energy have a higher diffusion coefficient, and have different probabilities to undergo certain reactions such as collisions with nuclei. Although the energy of the neutrons in the core is spread over a very wide interval (from about 0.001 eV up to 10 MeV), we may divide them roughly in two groups. Group 1, the *fast* group, contains neutrons at high energy (> 1 eV). Neutrons that are released during fission belong to this group. Group 2 contains the neutrons at lower energy. It is called the *thermal* group because the energy is comparable to the thermal energy of the surrounding nuclei. Thermal neutrons arise when fast neutrons collide with nuclei, thereby losing much of their energy. This process is called *scattering.*

The rates at which neutrons are absorbed by nuclei to cause different reactions are in nuclear theory represented by *macroscopic cross-sections* Ω (dimension cm^{-1})[1]. These can be interpreted as the probabilities that a neutron will take part in certain reactions per unit path length when moving through the core. The cross-sections depend on the composition of the core. Therefore they are position-dependent. Since the fuel is burning up during the operation of the plant, the cross-sections are in fact also time-dependent, but we first consider an idealistic core in which the core-composition is independent of time. (Note that this does not mean that the neutron flux is invariant in time.) Let X be the space of the core and the surrounding water, in a coordinate system that has yet to be specified, then for any position $x \in X$ we distinguish the following types of cross-sections:

- $\Omega_f^1(x)$, $\Omega_f^2(x)$ (dim. cm^{-1}): *Fission cross-section* of the fast and thermal group, i.e. the probability per unit path length that a neutron in the fast or thermal group will be absorbed by a nucleus to cause a fission reaction.
- $\Omega_a^1(x)$, $\Omega_a^2(x)$: *Absorption cross-section* of the fast and thermal group, i.e. the probability per unit path length that a neutron in the fast or thermal group will be absorbed by a nucleus, either to cause a fission reaction or due to another capture process (except for scattering, see below).
- $\Omega_s(x)$: *Downscattering (or slowing down) cross-section* from the fast to the thermal group, i.e. the probability per unit path length that a fast neutron is absorbed by a nucleus and immediately re-emitted at an energy level in the thermal group. It is assumed that we can neglect the scattering in the opposite way.

We further introduce the following parameters:

- $D^1(x)$, $D^2(x)$ (dim. cm): The position-dependent diffusion coefficients for the fast and thermal group.
- ν_1, ν_2: The average number of fast neutrons, produced by fission reactions that are induced by neutrons in the fast and thermal group.
- v_1, v_2 (dim. $cm\,s^{-1}$): The average neutron speed in the fast and thermal group.

Given those parameters, the neutron flux in the core can be described by a set of diffusion equations. This neutron flux is time-dependent, since it can increase or decrease when the reactor is not in a stable ('critical') state. We define

[1] In physical papers and textbooks, the notation Σ is used. Since this can lead to confusion with the symbol of summation, we use Ω instead.

- $\phi^1(x,t)$, $\phi^2(x,t)$ (dim. $cm^{-2}s^{-1}$): The neutron flux for the fast and thermal group. The neutron flux is defined as the total rate at which neutrons pass a sphere with unit cross-section area.[2]

Now, the set of time-dependent diffusion equations for the fast and thermal group can be written as (see [4]):

$$\frac{1}{v_1}\frac{\partial\phi^1(x,t)}{\partial t} - \nabla_x \cdot D^1(x)\nabla_x\phi^1(x,t) + \Omega_a^1(x)\phi^1(x,t) + \Omega_s(x)\phi^1(x,t) = \nu_1\Omega_f^1(x)\phi^1(x,t) + \nu_2\Omega_f^2(x)\phi^2(x,t) \quad (1)$$

$$\frac{1}{v_2}\frac{\partial\phi^2(x,t)}{\partial t} - \nabla_x \cdot D^2(x)\nabla_x\phi^2(x,t) + \Omega_a^2(x)\phi^2(x,t) = \Omega_s(x)\phi^1(x,t). \quad (2)$$

These equations are stated in the general form

$$\text{rate of change} - \text{diffusion term} + \text{neutron loss} = \text{neutron gain}.$$

When the neutron gain by production and neutron loss by absorption and diffusion are exactly equal, then the flux becomes independent of time, and the reactor is said to be *critical*. This will in general not be the case. However, under certain conditions, we may assume that $\phi(x,t)$ can be separated into $T(t)\phi(x)$, giving time-dependent relations, that are not discussed here, and space-dependent equations. These space-dependent equations can only have solutions if we introduce an additional degree of freedom in the system. This is obtained by introducing an eigenvalue λ to the system:

$$-\nabla_x \cdot D^1(x)\nabla_x\phi^1(x) + \left[\Omega_a^1(x) + \Omega_s(x)\right]\phi^1(x) = \lambda\left[\nu_1\Omega_f^1(x)\phi^1(x) + \nu_2\Omega_f^2(x)\phi^2(x)\right]; \quad (3)$$

$$-\nabla_x \cdot D^2(x)\nabla_x\phi^2(x,t) + \Omega_a^2(x)\phi^2(x,t) = \Omega_s(x)\phi^1(x,t). \quad (4)$$

The eigenvalue is placed such that it has a nice physical interpretation. We define the *effective multiplication factor* as follows:

$$k^{eff} = \frac{\text{total neutron production rate}}{\text{total neutron loss rate}}. \quad (5)$$

Then

$$\lambda = \frac{1}{k^{eff}}.$$

The thermal group diffusion equation (4) is eliminated by neglecting the diffusion term. This is possible since the neutron leakage in the thermal group is relatively small [4]. Equation (4) is then rewritten as

$$\phi^2(x) = \frac{\Omega_s(x)}{\Omega_a^2(x)}\phi^1(x),$$

which can be substituted in (3) to finally obtain the eigenvalue differential equation

$$-\nabla_x \cdot D^1(x)\nabla_x\phi^1(x) + \Omega_a^1(x)\phi^1(x) + \Omega_s(x)\phi^1(x) = \frac{1}{k^{eff}}\left[\nu_1\Omega_f^1(x) + \nu_2\Omega_f^2(x)\frac{\Omega_s(x)}{\Omega_a^2(x)}\right]\phi^1(x). \quad (6)$$

The thermal flux $\phi^2(x)$ is eliminated, so we only need to continue with $\phi^1(x)$. For ease of notation we will simply define

$$\phi^1(x) \rightarrow \phi(x).$$

[2] This is unlike the flux in other areas of physics, which is defined as the *net* flow rate per unit area in a given direction.

Since (6) is a one-group approximation of the two-group diffusion equations, it is known as the $1\frac{1}{2}$ group diffusion equation.

The introduction of an eigenvalue in (6) introduces the existence of infinitely many eigenfunctions, and thus infinitely many solutions. However, the eigenfunctions are orthogonal. For a one-dimensional core, (6) is a Sturm-Liouville equation, which has the property that the eigenfunction corresponding to the largest value of k^{eff} is nonnegative over the whole space X. For the more-dimensional case, the same property can be motivated by physical reasons. The eigenfunctions corresponding to the other eigenvalues are irrelevant for our study; they are not dominant in the system.

2.2 Some simplifications

We need to simplify (6) in order to derive our nodal model. First, we will assume that the diffusion coefficient D^1, as well as some cross-sections are position-independent. The fission cross-section Ω_f strongly depends on burnup and is very position-dependent. The absorption cross-sections, especially for the thermal group, also depend on burnup. However, in the $1^1/_2$ group approximation, the downscattering of neutrons is the most important mechanism of removing neutrons since we neglect thermal diffusion. This downscattering is almost independent of burnup since it is mainly determined by the reactor coolant. So, we redefine:

$$D^1(x) \to D^1, \quad \Omega_s(x) \to \Omega_s, \quad \Omega_a^1(x) \to \Omega_a^1, \quad \Omega_a^2(x) \to \Omega_a^2.$$

We now use $\Omega_f^1(x)$ and $\Omega_f^2(x)$ to define the new variable $k^\infty(x)$ in the following way:

$$k^\infty(x) \;=\; \left(\nu_1\Omega_f^1(x) \,+\, \nu_2\Omega_f^2(x)\frac{\Omega_s}{\Omega_a^2}\right) \;/\; (\Omega_a^1 \,+\, \Omega_s). \tag{7}$$

The variable k^∞, *the infinite multiplication factor,* is interpreted as the *ratio* of local neutron production and absorption. The eigenvalue differential equation (6) then transforms to

$$-\,L_f^2\nabla_x^2\phi(x) \;+\; \phi(x) \;=\; \frac{1}{k^{eff}}k^\infty(x)\phi(x), \tag{8}$$

with

$$L_f \;=\; \sqrt{D^1/(\Omega_a^1 \,+\, \Omega_s)},$$

known as the *fast diffusion length.*

2.3 Boundary values

The eigenvalue differential equation (8) requires boundary conditions. The simplest boundary condition is on the outer region of the core; we require that

$$\phi(x) \;=\; 0, \quad x \in \text{boundary of } X. \tag{9}$$

We sometimes apply symmetry of the core and consider only a quarter or an octant of the core; in that case, the derivative of the flux should vanish at symmetry edges:

$$\frac{\partial\phi(x)}{\partial x} \;=\; 0, \quad x \text{ on symmetry-edge.} \tag{10}$$

With these boundary conditions, $\phi(x)$ is still not fully determined, since we may multiply it with a constant. This freedom is used to fix the power, which is proportional to the product $k^\infty(x)\phi(x)$

$$\int_X \phi(x)k^\infty(x)\,dx \;=\; \frac{P_c}{\mu_f(\Omega_a^1 + \Omega_s)} \tag{11}$$

where

- P_c (eVs^{-1}) is the required total power over the whole core;
- μ_f (eV) is the amount of energy released per fission.

2.4 Fuel burnup

In Section 2.1, the cross-sections were assumed time-independent. After that, we used separation of variables to find a steady-state estimation for the flux, being the solution of a time-independent eigenvalue differential equation. In this section, we will study time-dependence on the long term due to fuel burnup, that is, due to the changes in cross-sections.
During reactor operation, the fission reactions cause burnup of the fuel. The amount of fissionable material, and therefore the fission cross-sections Ω_f^1 and Ω_f^2 decrease. The absorption cross-sections also change. On the one hand they decrease because of the decreasing atom density of fissionable nuclides. On the other hand they increase because the fission products absorb neutrons, without causing a fission or scattering reaction. Moreover, most of the material is not fissionable and will not change in time. We assume for the moment that we may neglect the changes in absorption cross-sections and only concentrate on changes in the fission cross-sections. So, we introduce time dependence on the following variables:

$$\Omega_f^1(x) \to \Omega_f^1(x,t),\ \Omega_f^2(x) \to \Omega_f^2(x,t),\ k^{eff} \to k^{eff}(t),\ k^\infty(x) \to k^\infty(x,t),\ \phi(x) \to \phi(x,t),$$

where t ranges from the begin of a cycle (BoC) to EoC. The decays of $\Omega_f^1(x,t)$ and $\Omega_f^2(x,t)$ are described by

$$\frac{\partial}{\partial t}\Omega_f^1(x,t) = -\sigma_a \Omega_f^1(x,t)\phi(x,t), \tag{12}$$

$$\frac{\partial}{\partial t}\Omega_f^2(x,t) = -\sigma_a \Omega_f^2(x,t)\phi(x,t), \tag{13}$$

where

- σ_a (dim. cm^2) is the so-called *microscopic cross-section.*

From (12) and (13) we can derive a description for the decay of $k^\infty(x,t)$ by using (7):

$$\begin{aligned} \tfrac{\partial}{\partial t}k^\infty(x,t) &= \left(\nu_1 \tfrac{\partial}{\partial t}\Omega_f^1(x,t) + \nu_2 \tfrac{\partial}{\partial t}\Omega_f^2(x,t)\tfrac{\Omega_s}{\Omega_a^2}\right) \,/\, (\Omega_a^1 + \Omega_s) \\ &= -\sigma_a \left[\left(\nu_1\Omega_f^1(x,t) + \nu_2\Omega_f^2(x,t)\tfrac{\Omega_s}{\Omega_a^2}\right) \,/\, (\Omega_a^1 + \Omega_s)\right] \phi(x,t) \\ &= -\sigma_a k^\infty(x,t)\phi(x,t). \end{aligned} \tag{14}$$

This differential equation is supplied with a boundary condition in time. If the initial configuration of the fuel is known, we can specify the initial value of $k^\infty(x,t)$, which is specified by the initial fuel distribution and denoted by k^{fresh}:

$$k^\infty(x,0) = k^{fresh}(x). \tag{15}$$

In this case no bounds on the end time t_{EoC} are needed. In case equilibrium cycles are studied, the boundary conditions become more complex. In that case, part of the core is initiated with fresh fuel, but $k^\infty(x,0)$ in the remaining bundles depends on $k^\infty(y(x), t_{EoC})$, with $y(x)$ the place of the fuel in the previous cycle:

$$k^\infty(x,0) = \begin{cases} k^{fresh}(x) & \text{when the fuel at position } x \text{ is fresh;} \\ k^\infty(y(x), t_{EoC}) & \text{when fuel is moved from } y(x) \text{ to } x. \end{cases} \tag{16}$$

Obviously, since k^∞ now is time-dependent, the flux distribution also is time-dependent, so that we have to redefine the eigenvalue differential equation (8), the boundary conditions (9), (10) and the normalization constraint (11) over the whole time-interval $t = [0, t_{EoC}]$:

$$-L_f^2 \nabla_x^2 \phi(x,t) + \phi(x,t) = \frac{1}{k_t^{eff}} k^\infty(x,t)\phi(x,t), \tag{17}$$

$$\phi(x,t) = 0, \quad x \in \text{boundary of } X, \tag{18}$$

$$\frac{\partial \phi(x,t)}{\partial x} = 0, \quad x \text{ on symmetry-edge}, \tag{19}$$

$$\int_X \phi(x,t) k^\infty(x,t)\, dx = \frac{P_c}{\mu_f(\Omega_a^1 + \Omega_s)}. \tag{20}$$

2.5 Derivation of the simplified model

The model as stated in (14) and (17) with boundary and normalization conditions (16), (18), (19) and (20) for a fixed pattern can be discretized and solved using finite difference algorithms. The starting value of $k^\infty(x,t)$ depends on the fuel configuration at the beginning of the period, which in turn depends on the loading pattern. An extra difficulty is that we are performing a search for the best of all possible reloading patterns, so the loading pattern is unknown in advance. This makes all coefficients $k^\infty(x,0)$ variable, resulting in a system of equations with a large number of nonlinear and nonconvex terms. In this paper, we make use of a model based on the theory of Green functions [4, 9, 11]. The key idea of this approach is that we can write the solution of differential equation (17) with boundary conditions (18), (19) in implicit form:

$$\phi(x,t) = \frac{1}{k_t^{eff}} \int_X G(x,x') k^\infty(x',t)(\Omega_a^1 + \Omega_s)\phi(x',t)\, \mathrm{d}x' \tag{21}$$

where the *Green function* $G(x,x')$ (dim cm^{-2}) can be interpreted as the probability that a neutron produced at x' will be absorbed at x. The function G depends on the left hand side operator of (17) and on the boundary conditions, and can be numerically approximated beforehand using eigenvalue expansion methods.
Equation (21) will be discretized. The function $G(x,x')$ is then replaced by a neutron interaction matrix. However, since there is some freedom in how to approximate $G(x,x')$ and how to derive from $G(x,x')$ the discrete version $G_{i,j}$, a much rougher discretization is possible than with direct discretization of (17), provided that some effort is spent in finding a good approximation of $G_{i,j}$. For convenience, we chose the discretization to coincide with the nodes in the core. Suppose the core consists of N nodes. We define the probabilities $G_{i,j}$, $i,j = 1, ..., N$:

- $G_{i,j}$: the probability that a neutron produced in node j will be absorbed in node i.

The infinite multiplication factor k^∞ and flux ϕ are averaged in a nodal way. Furthermore, the time-axis is discretized; we divide the core cycle into $T-1$ equal time periods of length Δt, and calculate k^∞ and ϕ only at T different times. The discrete versions of k^{eff}, k^∞ and ϕ are then defined as:

- k_t^{eff}: The effective multiplication factor at time t.
- $k_{i,t}^\infty$: The average infinite multiplication factor in node i at time t.
- $\phi_{i,t}$: The average neutron flux in node i at time t.

The integral equation (21) will then result in the set of equations

$$\phi_{i,t} = \frac{1}{k_t^{eff}} \sum_{j=1}^{N} G_{i,j} k_{j,t}^\infty \phi_{j,t}, \quad i = 1, ..., N, \; t = 1, ..., T. \tag{22}$$

The power fixing constraint (20) now becomes

$$\sum_{i=1}^{N} k_{i,t}^{\infty} \phi_{i,t} = \frac{P_c}{\mu_f(\Omega_a^1 + \Omega_s)}, \quad t = 1, ..., T \tag{23}$$

The decay equation (14) is replaced by the forward discretization

$$k_{i,t+1}^{\infty} - k_{i,t}^{\infty} = -\sigma_a \Delta t k_{i,t}^{\infty} \phi_{i,t}, \quad i = 1, ..., N, \; t = 1, ..., T-1. \tag{24}$$

Resuming, the crucial issue in the model is the derivation of the matrix $G_{i,j}$. There are several ways to compute G (see e.g. remarks in [4, 14]). Different approaches are described in e.g. [6] and [11]. In our paper, the method described in [6] is used.

2.6 Eigenvalues and eigenvectors

The nodalized integral equation (22) has solutions corresponding to a spectrum of eigenvalues. When we define, for fixed t, the diagonal matrix K containing the elements k_i^{∞}, and denote with ϕ the vector containing ϕ_i, $i = 1, ..., N$, then equation (22) can be written in eigenvalue equation form (index t is omitted):

$$k^{eff} \phi = GK\phi.$$

As already noted at the end of Section 2.1, it is proven for the one-dimensional case and expected for the more-dimensional cases that the eigenfunction corresponding to the largest k^{eff} is everywhere nonnegative in the continuous differential equation (17). This therefore also holds for the integral equation (21), and we assume that it is also valid for the discretized equation.
The consequence is that, using the restriction that $\phi_{i,t}$ is nonnegative everywhere, we will find the principal eigenmode. This is the only one in which we are interested.

3 Mathematical model

In this section, we will develop a nodal core model that describes the reloading patterns and the corresponding evolution of the core from BoC to EoC, where the physical characteristics are calculated for each fuel bundle separately. The evolution of the core during a cycle is computed in a fixed number of discrete time steps. We do calculations on equilibrium cycle reload patterns, which means that after each cycle, the same reloading pattern is applied. At each EoC, the same fixed number of old fuel bundles is discharged, (typically one fourth), which are all of the same age. The advantage of this is approach is that, ideally, the behavior of the reactor will be the same during all subsequent cycles. The objective of our optimization is to maximize the effective multiplication factor at EoC, while operational constraints are satisfied. In our case, these consist only of safety-limitations, limiting the maximal power density in each node.

The necessary equations can be divided into four different types. The *first* class of equations describes the evolution of the core during the cycle, given the infinite multiplication factors at BoC. The *second* class specifies a loading pattern. The *third* class describes the reloading operation itself; here the equilibrium cycle property is specified. The *fourth* class are the operational constraints. We will discuss the four types in sequence.

Before proceeding, we have to specify the problem dimensions. The following size-constants are used:

- N – the number of nodes in the core;
- M – the number of discharged bundles at EoC;
- L – the number of age groups in the core; note that $L = N/M$;
- T – the number of time steps. A cycle is divided in $T-1$ time-intervals of equal length. The first time step describes the state of the core at BoC, the next time steps describe the core at the end of the subsequent time intervals.

3.1 Core evolution equations

The physical state of the reactor core is computed using the model as derived in the previous section, that is based on $1\frac{1}{2}$ group diffusion theory. From Section 2.5 we have the variables

- $k_{i,t}^{\infty}$, the average infinite multiplication factor of the bundle in node i,
- $\phi_{i,t}$, the average fast neutron flux at node i, and
- k_t^{eff}, the effective multiplication factor of the core.

To simplify the model, we normalize the power by introducing the variable

$$R_{i,t} = \phi_{i,t}\frac{\mu_f(\Omega_a^1+\Omega_s)}{P_c}.$$

We further introduce

$$\alpha = \frac{\sigma_a}{\mu_f(\Omega_a^1+\Omega_s)}.$$

Because of our assumption that $(\Omega_a^1+\Omega_s)$ is constant, we can replace ϕ with R in (22), and we get the core evolution equations from (22), (23) and (24):

$$\begin{aligned}
k_t^{eff}R_{i,t} &= \sum_{j=1}^{N} G_{i,j}k_{j,t}^{\infty}R_{j,t}, \quad i=1,...,N,\ t=1,...,T,\\
k_{i,t+1}^{\infty} - k_{i,t}^{\infty} &= -\alpha P_c \Delta t k_{i,t}^{\infty}R(i,t), \quad i=1,...,N,\ t=1,...,T-1,\\
\sum_{i=1}^{N} k_{i,t}^{\infty}R_{i,t} &= 1, \quad t=1,...,T.
\end{aligned}$$

3.2 Loading pattern specification

We now turn to the variables and equations specifying the equilibrium cycle reload pattern. At each EoC, the same number of old fuel bundles is discharged, and all those removed bundles are of the same age. When, for example, a quarter of the bundles is removed at each EoC, all those bundles have spent 4 cycles in the core. Furthermore, the initial configuration of the core will be the same at each BoC. A handy way to describe the life of the bundles in the core during L cycles is the trajectory notation [7]. This notation describes how the bundles are moved during reloading. For example, suppose we have 12 nodes in the core, and at each EoC, three bundles are discharged. We then may have the following trajectory representation:

$$\begin{array}{ccccccc}
4 & \rightarrow & 6 & \rightarrow & 8 & \rightarrow & 10\\
2 & \rightarrow & 3 & \rightarrow & 1 & \rightarrow & 5\\
7 & \rightarrow & 11 & \rightarrow & 9 & \rightarrow & 12
\end{array}.$$

From this representation, we can for example read that nodes 4, 2 and 7 always contain fresh bundles. During a reloading, the bundle from node 10 is discharged, the one from

node 8 is moved to node 10, a fresh bundle is inserted at position 4, etc.
To be able to formulate the problem in optimization context, we introduce binary variables, which may take values either zero or one and describe the trajectories. We need the set of variables

$$x_{i,\ell,m}, \quad i = 1, ..., N, \ \ell = 1, ..., L, \ m = 1, ..., M,$$

that are defined in the following way:

$$x_{i,\ell,m} = \begin{cases} 1 & \text{if node } i \text{ will contain the bundle of age } \ell \text{ from trajectory } m, \\ 0 & \text{otherwise.} \end{cases}$$

In the above example, among the nonzero variables are for example $x_{4,1,1}$, $x_{6,2,1}$ and $x_{12,4,3}$. We denote the index m as the 'bundle-number' of the bundle. Of course, we need to describe the necessary properties of the variables $x_{i,\ell,m}$ in a set of equations. To do this, it suffices to specify that each node contains exactly one bundle, and that each bundle is located in one node:

$$\sum_{\ell=1}^{L} \sum_{m=1}^{M} x_{i,\ell,m} = 1, \quad i = 1, ..., N;$$

$$\sum_{i=1}^{N} x_{i,\ell,m} = 1, \quad \ell = 1, ..., L, \ m = 1, ..., M;$$

$$x_{i,\ell,m} \in \{0, 1\}.$$

3.3 Reloading operation

After specifying the variables and restrictions that describe the core evolution and the loading pattern respectively, we have to merge them by equations that specify the reloading operation. That is, we have to specify how the infinite multiplication factors at BoC depend on the EoC state of the previous cycle.
The reloading equation below consists of two terms. In the first term, we use the variables $x_{i,1,m}$ to determine whether node i contains a fresh bundle after reloading, and assign to $k_{i,1}^{\infty}$ the value k^{fresh} if this is the case. If not, then the second term determines from $x_{i,\ell,m}$ which bundle (ℓ, m) is in node i. It then finds the node where this bundle was located at age $\ell - 1$ and assigns the $k_{j,T}^{\infty}$ of that node to $k_{i,1}^{\infty}$ of the current node. The total reloading equation is given by

$$k_{i,1}^{\infty} = \sum_{m=1}^{M} x_{i,1,m} k^{fresh} + \sum_{\ell=2}^{L} \sum_{m=1}^{M} x_{i,\ell,m} \sum_{j=1}^{N} x_{j,\ell-1,m} k_{j,T}^{\infty}, \quad i = 1, ..., N$$

As an alternative, one might write the reloading equation in the simpler format

$$\sum_{i=1}^{N} x_{i,\ell,m} k_{i,1}^{\infty} = \begin{cases} \sum_{j=1}^{N} x_{j,\ell-1,m} k_{j,T}^{\infty}, & \ell = 2, ..., L, \ m = 1, ..., M, \\ k^{fresh}, & \ell = 1, \ m = 1, ..., M. \end{cases}$$

However, although the integer solutions of the two formulation coincide, the second formulation allows fractional solutions with very unrealistic objective values.

3.4 Safety limitation

Not all loading patterns are allowed. For safety reasons, it is required that the maximal power in one bundle is not too large. This power peaking constraint is applied for every node and at any time, and is stated as follows:

$$k_{i,t}^{\infty} R_{i,t} \leq \frac{f^{lim}}{N}, \quad i = 1, ..., N, \; t = 1, ..., T,$$

where f^{lim} is a value greater than 1.

3.5 Objective

The objective of the optimization is to maximize the effective multiplication factor at EoC:

$$\max k_T^{eff}.$$

3.6 Octant symmetry

We can merge all the equations that are obtained in the above sections into one model. When doing so for a whole core, we may get a huge problem. However, the geometric structure of the core suggests that we may assume symmetry, so that we only need to model a part of the core. It is usual in literature to assume at least quadrant symmetry, and often octant symmetry is assumed. In our implementation, we restrict ourselves to an octant core. This has some consequences for the model.
The most important difference concerns the bundles on diagonal positions (cf. Figure 1).

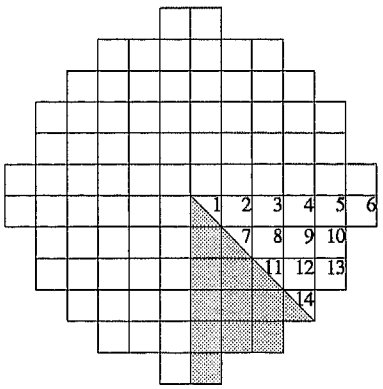

Figure 1: Octant symmetric core.

Suppose a bundle from a non-diagonal position has to be reloaded into a diagonal position. Because this diagonal position belongs to two octants, there are in fact two candidates from different octants for placing at the diagonal position. To solve this problem, we require that a bundle is assigned to two diagonal positions simultaneously, with half the volume in one position, and half the volume in the other position. When in a next cycle these two half bundles are placed in one new node, the composed bundle has at BoC average properties of the two diagonal bundles at EoC. This is modeled by introducing a volume vector $V = V_1, ..., V_N$, where V_i is $1/2$ if i is a diagonal node, and V_i is 1 for all other nodes. Note that now the number of different ages in the core L times the number of

fresh bundles M is no longer equal to N, the number of nodes in the octant core. Instead,

$$LM = \sum_{i=1}^{N} V_i.$$

This has some consequences for the model equations. The restriction that each bundle is in exactly one node, now is replaced by the restriction that each bundle is in one or two nodes with total volume 1:

$$\sum_{i=1}^{N} V_i x_{i,\ell,m} = 1.$$

After reloading, a bundle is given the average properties of its predecessors:

$$k_{i,1}^{\infty} = \sum_{m=1}^{M} x_{i,1,m} k^{fresh} + \sum_{\ell=2}^{L} \sum_{m=1}^{M} x_{i,\ell,m} \sum_{j=1}^{N} V_j x_{j,\ell-1,m} k_{j,T}^{\infty}, \quad i = 1, ..., N$$

Moreover the power normalization has to be performed on the octant core:

$$\sum_{i=1}^{N} V_i k_{i,t}^{\infty} R_{i,t} = 1.$$

The way of dealing with diagonal elements as described here introduces small errors in the solution. Suppose that a fresh bundle is placed in node 2 of the core in Figure 1. For symmetry reasons, the corresponding node in the gray octant also contains a fresh bundle. Both bundles have exactly the same characteristics at the first EoC. Now suppose they have to be replaced to diagonal position 1 and 14. Since bundles cannot be split, one bundle is placed in 1 and the other in 14. At the second EoC, they are replaced to node 9 and its gray counterpart. For symmetry reasons, it is assumed that the bundles in node 9 and its gray counterpart are exactly the same, which is in our model achieved by averaging the bundles from node 1 and from node 14 at EoC. In reality however, the bundle that has been at node 1 will have a burnup different from the bundle that has been in node 14. To reduce the error as much as possible, we require that the two parts of a diagonal bundle are in two adjacent nodes; in Figure 1, this means that node 1,7 and 11,14 respectively contain the same fuel element.

3.7 Model summary

The complete model for the octant core is listed below. For a larger part of the core, the model structure is the same, except that then all volumes are equal to 1.

$$\max_{x,k^{\infty},R,k^{eff}} k_T^{eff}$$

subject to:

$$
\begin{aligned}
\textstyle\sum_i V_i x_{i,\ell,m} &= 1, && \ell = 1,...,L,\ m = 1,...,M \\
\textstyle\sum_l \sum_m x_{i,\ell,m} &= 1, && i = 1,...,N \\
k^{\infty}_{i,1} &= \textstyle\sum_{\ell>1} \sum_m x_{i,\ell,m} \sum_j V_j\, x_{j,\ell-1,m}\, k^{\infty}_{j,T} \\
&\quad +(\textstyle\sum_m x_{i,1,m}) k^{fresh}, && i = 1,...,N \\
k^{eff}_t R_{i,t} &= \textstyle\sum_j G_{i,j} k^{\infty}_{j,t} R_{j,t}, && i = 1,...,N,\ t = 1,...,T \\
k^{\infty}_{i,t+1} &= k^{\infty}_{i,t} - (\alpha P_c \Delta t) k^{\infty}_{i,t} R_{i,t}, && i = 1,...,N,\ t = 1,...,T-1 \\
\textstyle\sum_i V_i k^{\infty}_{i,t} R_{i,t} &= 1, && t = 1,...,T \\
k^{\infty}_{i,t} R_{i,t} &\leq \frac{f^{lim}}{\sum_j V_j}, && i = 1,...,N,\ t = 1,...,T \\
k^{eff}_t &\geq 0, && t = 1,...,T \\
k^{\infty}_{i,t} &\geq 0, && i = 1,...,N,\ t = 1,...,T \\
R_{i,t} &\geq 0, && i = 1,...,N,\ t = 1,...,T \\
x_{i,\ell,m} &\in \{0,1\}, && i = 1,...,N,\ \ell = 1,...,L,\ m = 1,...,M.
\end{aligned}
$$

The above model differs from the model as used in [13] in mainly 3 ways. First of all, the model in [13] had k^{eff} fixed to 1, and instead the burnup of the bundles was introduced as an optimization variable. Secondly, there was no distinction between the bundles of the same age. However, it is clear that a fresh bundle in the centre has much different characteristics at EoC than a fresh bundle at the boundary, so this was really a rough estimation. Thirdly, there was no division of the cycle into time-steps, but the power distribution was assumed to be constant during the whole cycle.

4 Mathematical Optimization strategy

Mathematical optimization is a powerful tool for optimization of a given objective function, subject to a number of constraints. A mathematical optimization algorithm searches for an assignment of the variables such that the constraints are satisfied and the objective function value is (close to) optimal. Common mathematical optimization strategies in loading pattern optimization are simulated annealing and other neighborhood search heuristics. Those methods jump from pattern to pattern, and evaluate the physical variables in each fixed pattern.

In our approach, the problem is initially solved as a system of equations, where all the equations listed in the model (Section 3.7) are regarded as one large system, except for the integrality constraints $x_{i,\ell,m} \in \{0,1\}$, which are relaxed to the range constraints $x_{i,\ell,m} \in [0,1]$. During this process, one cannot speak about 'evaluating a pattern', but both the physical and assignment variables are varied simultaneously in one system to find a solution that satisfies the equations and has an optimal objective value. This solution, called the *relaxed solution*, need not be a valid loading pattern, as some x-variables may be fractional. In physical terms, this may be interpreted in some sense as a 'partial loading pattern', in which parts of bundles are mixed up in several nodes. We need some additional work to end up with an *integer solution*, where all x-variables are 0 or 1.

When an optimization model satisfies some conditions regarding convexity and continuity, it can be shown that the relaxed solution is a global optimum. Using sophisticated rounding and branching techniques, one then ends up with the global optimal loading pattern with respect to the used model.
Unfortunately, our problem is not convex, and it seems to be extremely difficult to guarantee that a given solution is the global optimum. Nevertheless, rather good solutions may be found within reasonable time. In the next section, we will discuss the method that has been used in this paper.

4.1 Mixed integer nonlinear programming using DICOPT

Our model is a mixed integer nonlinear optimization model. To solve it, we used the solver package DICOPT, running within the high-level modeling language GAMS [1]. In GAMS, an optimization problem can easily be stated, without concerns from the user's side about memory storage, computation of function derivatives, and so on. The package DICOPT solves a sequence of nonlinear programming problems and mixed integer linear programming problems. Let y denote the physical variables, and x denote the binary variables. The DICOPT algorithm can be stated as follows [5].

begin
  read (x^0, y^0); {*read the user-supplied starting guess*}
  Solve the relaxed (NLP) problem $\rightarrow$ (x^1, y^1).
  if x^1 is integer **then stop:** Optimal solution found;
  else
    $i := 1$;
    $z := -\infty$; *Initialize lower bound on maximization*
    repeat
      $\overline{z} := z$;
      linearize around (x^i, y^i) and add constraints to the linearized (MILP) problem;
      Add integer cuts to the MILP problem;
      Solve the MILP problem to the integer optimal solution (x^{i+1}, w^{i+1});
      Solve the NLP problem with $x = x^{i+1}$ being fixed.
      Let the solution be (x^{i+1}, y^{i+1}), and the optimum value z;
      $i := i + 1$;
    until $(\overline{z} \geq z)$;
  end {*else*};
end.

If the relaxed solution happens to be integral, then the algorithm is terminated. If not, a linearization is made around the current solution point. The MILP consists of the union of all linearizations of iteration 0 to i. It ends up with an integer solution, which is in our case a loading pattern. However, the physical variables may not be accurate due to the linearization. Therefore, the NLP problem is solved to find the physical variables for this loading pattern. For our problem, also dedicated pattern evaluation methods could be used. From the new point, a new linearization is made, until the objective function of the NLP starts worsening.
Different solvers can be attached to DICOPT to solver the NLP subproblems and the MILP subproblems. For the solution of the NLP subproblems, we used the nonlinear solver CONOPT [3], which gave more robust solutions than the competing solver MINOS5. It makes use of the Generalized Reduced Gradient method [8]. The MILP-solutions are obtained by the solver CPLEX, one of the most advanced MILP-solvers today. It internally

uses the linear simplex method combined with a Branch and Bound approach to reach an integer solution.

Because our problem is nonconvex, the algorithm does not necessarily yield a global optimum. Even worse, no guarantee about the quality of the solution can be given. This holds also for the local search heuristics, however. In the next section, we describe the adaptations of our model that made DICOPT working, and the specific problems we encountered with our model.

4.2 Pairwise Interchange

We compared our MINLP results to the results from a pairwise interchange (PI) algorithm. That algorithms starts by evaluating an arbitrary pattern. Then with this pattern as a parent, all neighbor patterns are evaluated, that result from interchanging two bundles in the core. From all neighbors that are feasible (i.e. satisfy the operational constraints), we select the one with best objective value. If this value is better than the current best objective, the neighbor is chosen as the new parent pattern. To prevent PI from getting stuck in a local optimum too early, we also proceed if no improving feasible neighbor exists, but infeasible neighbors with better objective function value do exist. Then the best infeasible neighbor is chosen as a parent. So, we have the following algorithm.

begin

read the starting pattern x^0.
evaluate $x^0 \rightarrow f(x^0)$
If x^0 is feasible **then**
    $f_B := f(x^0)$, $f_I := 0$.
    $x_B := x^0$.
else
    $x_I := x^0$.
$i = 0$; $Stop := False$;
repeat
    $f_B^{old} := f_B$, $f_I^{old} := f_I$;
    do $x \in Neighbors(x^i)$
      evaluate $x \rightarrow f(x)$;
      if $f(x) > f_B$ **and** x is feasible **then**$f_B := f(x)$, $x_B := x$;
      elseif $f(x) > \max(f_B, f_I)$ **then**$f_I := f(x)$, $x_I := x$;
    enddo
    if $f_B > f_B^{old}$ **then** $x^{i+1} = x_B$;
    elseif $f_I > \max(f_B^{old}, f_I^{old})$ **then** $x^{i+1} = x_I$;
    else $Stop := True$;
    $i := i + 1$;
until $Stop = True$;
if $f_B > 0$ **then** Feasible Local Optimum found
else No feasible solution found

end.

From the 30 test cases that are listed in Table 2, the restart from the best infeasible neighbor turned out to be helpful in almost half of the cases, as is shown in Table 1. On the other hand it shows that in a few cases it caused a longer search without improving the objective function.

Of course, there are several other options possible to take advantage of infeasible patterns. Another approach is for example to simply select the neighbor with best objective function value, regardless of its feasibility. When iterations proceed, the infeasibility al-

	Pattern with no feasible improvement		Best pattern after exploring infeasible neighbors	
Prob.	Obj. value	Iterations	Obj. value	Iterations
small				
4	1.04205	8	1.04275	12
5	1.03696	9	1.03975	17
6	1.03336	9	1.03407	12
7	1.02574	13	1.02753	21
10	1.02475	15	1.02475	21
medium				
1	1.03524	19	1.03537	23
3	1.03414	15	1.03492	24
4	1.03347	16	1.03429	27
6	1.03558	21	1.03558	23
8	1.03477	18	1.03548	27
large				
1	1.04753	44	1.04758	74
3	1.04718	43	1.04722	59
6	1.04758	44	1.04762	56
7	1.04673	48	1.04676	76
9	1.04667	54	1.04670	62

Table 1: Test problems from Table 2 where infeasible neighbors were explored.

lowed is then gradually decreased, until only feasible patterns can be selected as new parents. This further can be combined with a penalty in the objective function.

4.3 Rounding procedure

Instead of using the DICOPT scheme for finding an integer value, we also developed a simple rounding procedure. As already noted, many integer solutions turn out to be local optima, and in several cases, the NLP solver immediately ends up with an integer value. When the returned optimum is non integer, this is often due to a power peaking limit (see Figure 2), and the number of nonintegers is usually small. In order to quickly find an integer solution, we start with the relaxed solution and enumerate all possible integer solutions such that

- ones in the relaxed solution remain ones, and
- zeros in the relaxed solution remain zeros.

Suppose we have the following part of the assignment matrix:

$$\begin{pmatrix} 0.2 & 0.8 & & & & \\ 0.8 & 0.2 & & & & \\ & & & 0.5 & 0.5 \\ & & 0.7 & & 0.3 \\ & & 0.3 & 0.5 & 0.2 \end{pmatrix}$$

There are two possibilities to round the first two rows and columns, and independently there are three possibilities to round the last three rows and columns, so there are six possible rounded solutions in total.
The feasible rounding with best objective value is returned, or, if none of the rounded solutions is feasible, the best infeasible rounding is returned.

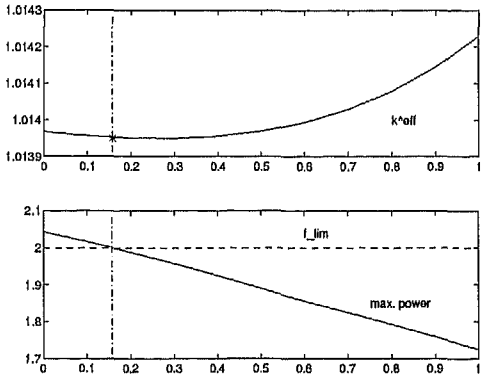

Figure 2: Fractional solution due to the power peak constraint. In the first plot, it is shown that the two integer solutions at $x = 0$ and $x = 1$ are local maxima. In the second plot, it is shown that in $x = 0$ the power peak is exceeded. The solution found for the relaxed problem is indicated at $x \approx 0.16$. Rounding leads to the (in this case even better) solution $x = 1$.

4.4 Setup of the problem

We can make several variations and reformulations of our model in order to make it more appropriate to solve with DICOPT. Also a good initialization of the variables is very important. In this section, the most successful adaptations and starting values will be discussed.

4.4.1 Starting values

The choice of the starting values can have a big influence on the behavior of the algorithms. If, for example, all variables in our model are initialized to 0, the optimization solvers often get stuck in a local infeasible solution. We have to supply starting values that have some sensible physical meaning.
We can get help from physical knowledge. It is known that fresh bundles in the middle lead to a large value of the power peak. On the other hand, the bundles on the boundaries of the core should be sufficiently burned, since this causes loss of neutrons in the surrounding water. It is known that good patterns therefore have a sort of a 'ring structure', in which old bundles are put on the boundaries, and fresh bundles somewhere in the middle, (like Figure 3). Although we want to use a starting point that is based on such known good

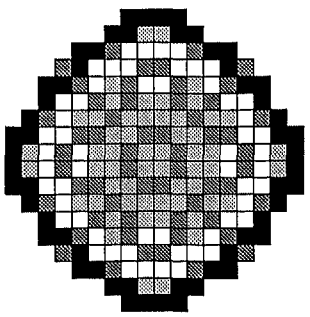

Figure 3: Ring structure of a good loading pattern.

patterns, we cannot use one such a pattern as a starting point, since in the nonlinear

optimization approach, these integer assignments tend to be local optimal solutions in the continuous space. It is therefore nearly impossible to escape from such a starting point. Instead, we assigned for each age a number of nodes where it is likely that the bundles of this age are located, as in Figure 4. Using these assignments, a fractional

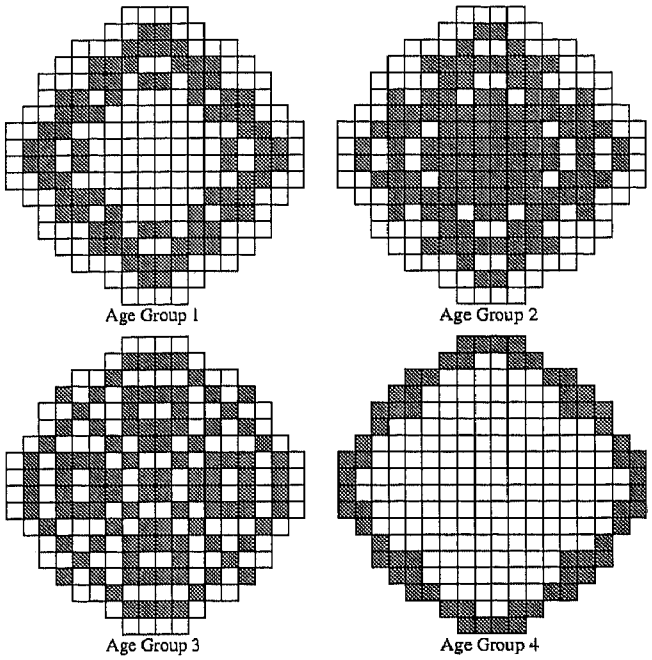

Figure 4: Most likely locations for bundles in the different age groups

starting pattern is constructed, in which it is specified how many percent of each bundle is initially in each node. This assignment is made in such a way that 90% of the volume of the bundles of age i is spread in the nodes assigned for this age, and 10% is spread over the other nodes. An evenly distributed assignment is obtained by choosing that pattern for which the sum of squares of the percentages is minimal.
Also, for the physical variables, an initial guess is needed. This is supplied by applying the power method to solve the physical equations for the given fractional assignment.

4.4.2 Decreasing k^{eff}

During the cycle, the value of the eigenvalue k^{eff} (which is a measure for the reactivity of the core) decreases. Since the problem is very sensitive to the value of k^{eff}, we added the set of constraints

$$k_t^{eff} \geq k_{t+1}^{eff}, \quad t = 1, ..., T-1.$$

These constraints, which are intuitively clear, can be derived from (22) and (24).

4.4.3 Relaxation of the Power peaking constraint

For a fixed pattern (i.e. fixed binary x-variables), the physical equations describe the evolution of the core, and there always exists a feasible solution with respect to those equations. The power peaking restrictions however, really restrict the number of feasible

reloading patterns. Motivated by this observation, we initially relax the power peaking constraint with some perturbation variable ε: instead of

$$k_{i,t}^{\infty}\phi_{i,t} \;\leq\; \frac{f^{lim}}{\sum_{j=1}^{N} V_j}$$

we implemented

$$k_{i,t}^{\infty}\phi_{i,t} \;\leq\; (1+\varepsilon)\frac{f^{lim}}{\sum_{j=1}^{N} V_j}.$$

The variable ε is nonnegative, and after some experiments given an upper bound of 0.01. It is penalized in the objective function with penalty parameter ω, which is set to 1. So, the objective now becomes

$$\max_{x,k^{\infty},\phi,k^{eff},\varepsilon} \; k_T^{eff} \;-\; \omega\varepsilon.$$

During the optimization, the value of ε should vanish. If not, we have to repeat the optimization with the previous solution as starting point, but with a higher ω or with ε fixed to 0.

5 Computational results

We compared the nonlinear programming approach to pairwise interchange. We considered three different core geometries, and for each geometry we used 10 different parameter sets. The small problems have a core geometry as in Figure 1. One octant contains 14 nodes, of which 4 are diagonal nodes, so there are effectively 12 nodes. The medium-sized core geometry is given in Figure 3, where one octant contains 31 nodes (effectively 28). The large core has 52 nodes in an octant (effectively 48), as in Figure 5. In all problems, all bundles remain for 4 years in the core before they are discharged. The cycle-length is divided into 6 time-steps. We compared 4 algorithms:

1. DC: DICOPT running under the modeling language GAMS;
2. CR: CONOPT, followed by the simple rounding procedure;
3. CRP: As CR, but then followed by Pairwise Interchange;
4. PI: Pairwise interchange from an arbitrary starting point.

The results are listed in Table 2. Computation times are in seconds on an HP 9000/720 workstation.

It should be noted, that using DC, only an older release of the nonlinear solver CONOPT could be used to solve the nonlinear subproblems. For CR and CRP we could use a newer release, and it turned out that especially the relaxed NLP problem took much less time with this new release. For example, in large problem 7, the old release in DC took about 66000 seconds to solve the first NLP problem, while the new release in CR finished this in about 500 seconds. This suggests that the DC running times can be much smaller. Since for all but one of the small problems, the new release of CONOPT gave immediately an integer solution, using this new version with DC would have lead to the same answers as CR for these problems.

Apart from these remarks, it is striking that the advanced and expensive method of DC on the one hand and the simple and fast rounding procedure on the other hand do not beat out each other. For the medium problems, DC returns 5 times a better integer solution than CR, while CR wins 4 times. For the large problems, CR wins even 7 times,

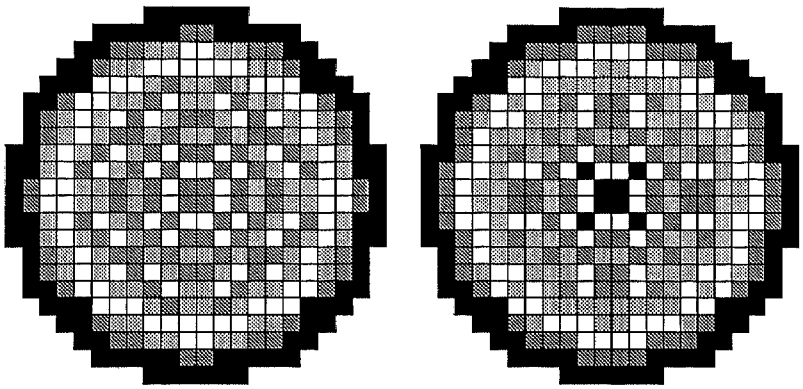

Figure 5: CRP solution (left) and PI solution (right) of Large 2, with the same objective value.

and DC 2 times, but the differences are very small.

Comparing the two PI implementations, it becomes clear that starting from the CR solution is often helpful. For the small problems, CRP gives 5 times a better answer, the direct PI method 1 time. For the medium problems this ratio is 7:3, for the large problems it is 4:5. For all the cases it holds that starting at the CR solution on the average improves solution time. Equal objective values do not always imply that the pattern is the same, as is shown in Figure 5.

6 Conclusions and further research

Direct nonlinear optimization, combined with a procedure for finding integer solutions, is a good candidate approach for the reactor core optimization problem. When the solution, obtained with this approach, is subjected to a local search procedure, it produces for our test problems results that are competitive to the direct pairwise interchange algorithm. The computation time for the nonlinear optimization is very favorable.
Good modeling is essential, especially when using nonlinear optimization. Due to the very nonconvex structure of the problem, relaxation of some constraints, or addition of artificial constraints, is of much help in the optimization. It is also very important to supply a good starting point.
There are several possibilities to find an integer solution from the relaxed solution, obtained by nonlinear optimization. Besides subsequent linearization as in DICOPT, simple rounding also behaves surprisingly good. We further may apply a Branch & Bound scheme to refine the rounding method.
A strong advantage of nonlinear optimization is its capability to handle other, by nature continuous optimization problems such as burnable poison distributions. It is interesting to study the behaviour of a one-stage optimization procedure optimizing both the reloading pattern and a Burnable Poison distribution over the fresh bundles.
Another advantage is that implicit formulations of the problem are allowed, since inside such an optimization procedure a Newton-like method is used to find a feasible solution. This makes it possible to apply for example central difference techniques for the time-discretization, while in the direct iterative methods this is more cumbersome because it needs some predictor-corrector techniques.
Due to all these possibilities, and regarding the results of this paper, we conclude that the use of nonlinear optimization in core fuel management is worth to be studied in more

detail.

References

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

[2] J.N. Carter. Genetic algorithms for incore fuel management and other recent developments in optimisation. *Advances in Nuclear Science and Technology*, 25:113–154, 1997.

[3] A.S. Drud. A large-scale grg code. *ORSA Journal on Computing*, 6(2), 1994.

[4] J.J. Duderstadt and L.J. Hamilton. *Nuclear Reactor Analysis.* John Wiley & Sons, Inc., 1976.

[5] GAMS Development Corporation, Washington, DC. *GAMS - The Solver Manuals.*

[6] R. van Geemert. Evaluation and optimization of fuel assembly shuffling schemes for a nuclear reactor core. Technical Report IRI-131-95-016, Delft University of Technology, 1995.

[7] R. van Geemert, A.J. Quist, and J.E. Hoogenboom. Reload pattern optimization by application of multiple cyclic interchange algorithms. In *Proceedings of PHYSOR96*, 1996.

[8] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

[9] I. Gohberg and S. Goldberg. *Basic Operator Theory.* Birkhäuser, 1981.

[10] J.E. Hoogenboom and H. van Dam. Reactor physics (in dutch). Lecture Notes Delft University of Technology, 1994.

[11] A.J. de Jong. Reloading pattern design for batch refuelled nuclear reactors. Technical Report IRI 131-95-010, Delft University of Technology, 1995.

[12] T.K. Kim and C.H. Kim. Determination of optimized PWR fuel loading pattern by mixed integer programming. In *Proceedings of PHYSOR '96*, pages I76–I85, 1996.

[13] E. de Klerk, T. Illés, A.J. de Jong, C. Roos, T. Terlaky, J. Valkó, and J.E. Hoogenboom. Optimization of nuclear reactor reloading patterns. *Annals of OR*, 69:65–84, 1997.

[14] W.M. Stacey, Jr. *Space-Time Nuclear Reactor Kinetics.* Academic Press, 1969.

[15] J.S. Suh and S.H. Levine. Optimized automatic reload program for pressurized water reactors using simple direct optimization techniques. *Nuclear Science and Engineering*, 105:371–382, 1990.

[16] S. Wei and C. Pingdong. A new approach for low-leakage reload core multi-cycle optimization design. In *Proceedings of PHYSOR '96*, pages I86–I95, 1996.

[17] A. Yamamoto. Comparison between equilibrium cycle and successive multicycle optimization methods for in-core fuel management of pressurized water reactors. In *Proceedings of the Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications*, pages 769–781, Saratoga Springs, New York, 1997. American Nuclear Society, Inc.

[18] N. Zavaljevski. A model for fuel shuffling and burnable absorbers optimization in low leakage PWR's. *Annals of Nuclear Energy*, 17(4):217–220, 1990.

	Objective Values				Comp. Times			
Prob.	DC	CR	CRP	PI	DC	CR	CRP	PI
small								
1	INFEAS.	1.00961	1.01130	1.01114	7.7	3.3	72.8	74.5
2	1.04361	1.04823	1.04844	1.04848	41.1	4.4	27.0	73.0
3	1.02023	1.02011	1.02069	1.02051	13.3	2.7	30.6	80.8
4	1.04035	1.04186	1.04275	1.04275	54.6	5.1	49.5	94.4
5	1.03917	1.03827	1.03975	1.03975	53.3	3.2	67.1	135.8
6	1.03321	1.03316	1.03407	1.03407	19.1	5.1	48.9	92.7
7	1.02541	1.02726	1.02753	1.02753	89.0	3.6	49.5	150.6
8	1.02765	1.02948	1.02970	1.02793	87.8	4.3	31.4	79.5
9	1.02612*	1.02645*	1.02605	1.02425	162.6	3.1	35.5	68.0
10	1.02509	1.02504	1.02545	1.02475	103.4	4.2	54.3	136.5
medium								
1	1.03451*	1.03536	1.03538	1.03537	2660.6	53.0	544.2	3852.9
2	1.03486	1.03476	1.03520	1.03510	517.3	61.9	3112.6	3348.2
3	1.03457	1.03475*	1.03489	1.03492	1246.2	64.8	3057.8	3506.1
4	1.03347	1.03362	1.03432	1.03429	6604.7	75.8	2517.9	4125.9
5	1.03312	1.03345*	1.03381	1.03380	588.0	61.1	2668.8	3476.4
6	1.03534	1.03518	1.03545	1.03558	850.9	70.7	2121.6	3855.6
7	1.03374*	1.03332*	1.03404	1.03349	1584.9	63.9	2231.5	3002.7
8	1.03490	1.03447	1.03532	1.03548	6656.1	57.8	2260.3	4630.7
9	1.03350	1.03387	1.03413	1.03402	1040.8	85.6	3446.7	4084.6
10	1.03300*	1.03304	1.03340	1.03329	1795.8	53.5	2645.9	3562.1
large								
1	1.04718	1.04716	1.04761	1.04756	8480.5	475.6	38523.6	81468.1
2	1.04686	1.04724	1.04738	1.04738	8429.8	443.8	12720.2	61920.2
3	1.04687	1.04692	1.04717	1.04724	6335.1	387.2	20210.5	80259.1
4	1.04650	1.04645	1.04682	1.04658	7545.1	396.7	18669.3	39161.3
5	0.96213*	1.04627*	1.04648	1.04646	322.3	400.0	32159.7	60335.1
6	1.04719	1.04721	1.04756	1.04757	1.1E+4	435.1	23060.3	53541.8
7	1.04656	1.04655*	1.04663	1.04675	6.9E+4	562.3	12320.3	64471.6
8	1.04708	1.04727	1.04754	1.04739	5680.6	438.7	21513.7	38945.7
9	1.04652	1.04654	1.04664	1.04665	4554.5	458.5	18176.8	52791.8
10	1.04281*	1.04605	1.04618	1.04623	468.4	493.8	15823.7	50714.4

*=Fractional solution found, but no feasible integer solution found.

Table 2: Results for 30 different test problems.