A comparative study on multiobjective metaheuristics for solving constrained in-core fuel management optimisation problems

E.B. Schlünza,b,c,∗, P.M. Bokovd, J.H. van Vuurene

aRadiation and Reactor Theory, The South African Nuclear Energy Corporation SOC Ltd, PO Box 582, Pretoria, 0001, South Africa
bDepartment of Logistics, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa
cDepartment of Industrial Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa

Abstract

In this paper, the topic of constrained multiobjective in-core fuel management optimisation (MICFMO) using metaheuristics is considered. Several modern and state-of-the-art metaheuristics from different classes, including evolutionary algorithms, local search algorithms, swarm intelligence algorithms, a probabilistic model-based algorithm and a harmony search algorithm, are compared in order to determine which approach is most suitable in the context of constrained MICFMO. A test suite of sixteen optimisation problem instances, based on the SAFARI-1 nuclear research reactor, has been established for the comparative study. The suite is partitioned into three classes, each consisting of problem instances having a different number of objectives, but subject to the same stringent constraint set. The effectiveness of a multiplicative penalty function constraint handling technique is also compared with the constrained-domination technique from the literature. The different optimisation approaches are compared in a nonparametric statistical analysis. The analysis reveals that multiplicative penalty function constraint handling is a competitive alternative to constrained-domination, and seems to be particularly effective in the context of bi-objective optimisation problems. In terms of the metaheuristic solution comparison, it is found that the nondominated sorting genetic algorithm II (NSGA-II), the Pareto ant colony optimisation (P-ACO) algorithm and the multiobjective optimisation using cross-entropy method (MOOCEM) are generally the best-performing metaheuristics across all three problem classes, along with the multiobjective variable neighbourhood search (MOVNS) in the bi-objective problem class.

∗Corresponding author. Tel.: +27 (0) 12 305 5724
Email addresses: bernard.schlunz@necsa.co.za (E.B. Schlünz), pavel.bokov@necsa.co.za (P.M. Bokov), vuuren@sun.ac.za (J.H. van Vuuren)
Keywords: Metaheuristics, Multiobjective optimisation, Constraint handling, In-core fuel management optimisation, Nuclear reactor

List of acronyms

- AMOSA: Archived Multiobjective Simulated Annealing
- ANN: Artificial Neural Network
- CDP: Constrained-Domination Principle
- ICFMO: In-Core Fuel Management Optimisation
- MICFMO: Multiobjective In-Core Fuel Management Optimisation
- MOHS: Multiobjective Harmony Search
- MOOCEM: Multiobjective Optimisation Using Cross-Entropy Method
- MOVNS: Multiobjective Variable Neighbourhood Search
- MPF: Multiplicative Penalty Function
- NSGA-II: Nondominated Sorting Genetic Algorithm II
- OMOPSO: Optimised Multiobjective Particle Swarm Optimisation
- OSCAR-4: Overall System for the Calculation of Reactors, Version 4
- P-ACO: Pareto Ant Colony Optimisation
- SAFARI-1: South Africa Fundamental Atomic Research Installation 1
- SPEA2: Strength Pareto Evolutionary Algorithm 2

1. Introduction

The in-core fuel management optimisation (ICFMO) problem is a nonlinear assignment problem in which an optimal fuel reload configuration (or a loading pattern) is sought for a nuclear reactor core. It is a classical problem in the field of nuclear engineering and has been studied for several decades [1]. Fresh and partially-burnt fuel assemblies from an available set have to be assigned to loading positions in a reactor core in such a way that the resulting configuration optimises reactor performance, while also ensuring that operational constraints are satisfied.

Several characteristics make the ICFMO problem difficult to solve, such as a large disjoint feasible decision space, nonlinear objectives and constraints that lack derivative information, and computationally demanding function evaluations [2, 3]. These challenging characteristics have led to various solution approaches being proposed over the years for solving the ICFMO problem. Techniques involving mathematical programming and expert/knowledge-based systems were utilised during the early years of research [2]. More recently, however, metaheuristic techniques, such as simulated annealing, tabu search, evolutionary- and swarm intelligence algorithms, have been applied to ICFMO [1]. Research has also been aimed towards reducing the computational cost of function evaluations associated...
with ICFMO. In this regard, surrogate models, such as artificial neural networks, have been utilised for predicting function values instead of calculating them explicitly using a reactor core simulator [4, 5].

The majority of the above-mentioned research has been performed in the context of single-objective optimisation. The ICFMO problem is, however, inherently multiobjective in nature [2, 3]. In this paper, we consider the multiobjective in-core fuel management optimisation (MICFMO) problem in terms of the fundamental notion of Pareto optimality (colloquially known as the paradigm “true” multiobjective optimisation). Accordingly, a Pareto optimal set of fuel reload configurations is sought for a nuclear reactor core. Currently, the appropriateness of several modern, Pareto-based metaheuristics for solving MICFMO problems is not known, especially in terms of their comparative performance. The aim of this paper is to provide a baseline and guide for future developments in terms of solving constrained MICFMO problems using metaheuristics, and potentially hybrid approaches and hyperheuristics.

In this paper, we compare eight modern, state-of-the-art metaheuristics for solving constrained MICFMO problems. The metaheuristics considered include two popular evolutionary algorithms, namely the nondominated sorting genetic algorithm II (NSGA-II) [6] and the strength Pareto evolutionary algorithm 2 (SPEA2) [7]; two swarm intelligence algorithms, namely optimised multiobjective particle swarm optimisation (OMOPSO) [8] and Pareto ant colony optimisation (P-ACO) [9]; two local search algorithms, namely archived multiobjective simulated annealing (AMOSA) [10] and multiobjective variable neighbourhood search (MOVNS) [11]; a probabilistic model-based algorithm called the multiobjective optimisation using cross-entropy method (MOOCEM) [12]; and finally, a multiobjective harmony search (MOHS) algorithm [13]. By sourcing the metaheuristics from different classes of algorithms, we attempt to encompass a diversity of metaheuristics available in the literature. We also compare a multiplicative penalty function constraint handling technique of Schlimé et al. [14] for multiobjective optimisation to the constrained-domination technique proposed by Deb et al. [6].

A test suite of sixteen constrained MICFMO problem instances, based on the SAFARI-1 nuclear research reactor, is established for the comparative study. The suite is partitioned into three classes, each consisting of test problem instances having a different number of objectives (between two and four), but subject to the same stringent constraint set. This allows us to investigate how the performance of the metaheuristics scales with the number of objectives.
It has been advocated in the literature that structured and statistically sound procedures should be used when comparing metaheuristics [15, 16]. Accordingly, we conduct a nonparametric statistical analysis on our results. In the constraint handling technique comparison, we employ the Wilcoxon signed rank test, whereas the Friedman test with the Nemenyi post-hoc procedure is employed in the metaheuristic comparison [17, 18].

The paper is organised as follows. A brief overview of MICFMO solution approaches in the literature is presented first. Then, the constrained MICFMO problem is discussed in detail, including a description of our test suite of problem instances and the artificial neural networks that we employ for function evaluations. The constraint handling techniques and the eight metaheuristics forming part of the comparative study are described next. The section thereafter contains a description of performance assessment procedures followed during our study. The results obtained for the test suite are then presented and discussed, followed by the conclusions of the paper.

2. A brief overview of MICFMO solution approaches in the literature

In most research papers claiming to perform MICFMO, scalarising approaches involving linear weighted sum aggregations of the objectives are adopted, which only yield a single solution [19–21]. Apart from the serious shortcomings associated with these approaches (e.g. the inability to uncover a Pareto optimal solution if the problem is nonconvex and the misleading role of weights [22]), the manner in which they were applied does not solve the MICFMO problem in terms of finding a Pareto optimal set of solutions. For this reason, we restrict this overview to MICFMO approaches in the literature involving “true” multiobjective optimisation.

Parks [23] designed a multiobjective genetic algorithm which utilises nondominated ranks in its selection procedure. The algorithm also maintains an archive of nondominated solutions and it employs a dissimilarity measure (in decision space) to distinguish between two nondominated solutions.

A multiobjective simulated annealing algorithm was designed by Engrand [24] in which an aggregated energy function is utilised during the calculation of the acceptance probability. The algorithm maintains a nondominated archive and periodically replaces the current solution with one present in the archive. Improvements on Engrand’s algorithm [24] were suggested by Parks & Suppapitnarm [25], with Kellar [26] suggesting further refinements on the improved algorithm.
Do & Nguyen [27] applied a single-objective genetic algorithm to an MICFMO problem instance in which a linear weighted sum aggregation of objectives is employed as the fitness function. In the algorithm, an archive of nondominated solutions (distinguished by their fitness values) is maintained which forms part of the selection pool during each generation.

A single-objective simulated annealing algorithm by Park et al. [28] has also been applied to an MICFMO problem instance. In their algorithm, which also maintains an archive of nondominated solutions, objectives and constraints are aggregated into a so-called discontinuous penalty function to be minimised.

The first application of a state-of-the-art multiobjective metaheuristic to an MICFMO problem instance that we could find appeared in 2009 in a paper by Hedayat et al. [29] in which the authors applied the NSGA-II. A parameter-tuning study for their application of the NSGA-II was performed in the paper, and the final optimisation results were compared to a reference solution.

Several years later, in 2014, Schlünz et al. [14] applied the MOOCEM to a constrained MICFMO problem instance and compared the optimisation results to historical and heuristic solutions. Shortly thereafter, Schlünz et al. [30] also used the MOOCEM within the context of validating a scalarising approach for MICFMO. To the best of our knowledge, no other modern multiobjective metaheuristics have been applied to MICFMO problems.

3. The constrained MICFMO problem

As stated in Section 1, the MICFMO problem is a nonlinear assignment problem in which fuel assemblies are assigned to loading positions in a nuclear reactor core. We assume that the number of available fuel assemblies \( n \) is equal to the number of loading positions in the core, and has been determined as part of the out-of-core fuel management decision process [3].

Let the loading positions in the core be labelled \( 1, \ldots, n \), and let the available fuel assemblies also be labelled \( 1, \ldots, n \). A reload configuration may then be represented by a permutation decision vector \( \mathbf{x} = [x_1, \ldots, x_n] \) where \( x_i = j \) denotes that fuel assembly \( j \in \{1, \ldots, n\} \) is assigned to loading position \( i \in \{1, \ldots, n\} \). Let \( \mathcal{X} \) be the set of all possible reload configurations (i.e., permutation decision vectors) and suppose, without loss of generality, that all objective functions are to be maximised. Then, the general formulation of a constrained MICFMO problem with \( q \) objective functions
\[ f_1(x), f_2(x), \ldots, f_q(x) \text{ may be formulated as} \]

\[
\begin{aligned}
\text{maximise} & \quad f(x) = [f_1(x), f_2(x), \ldots, f_q(x)], \\
\text{subject to} & \quad g_i(x) \leq g_i^{\lim}, \quad i = 1, \ldots, r, \\
& \quad h_j(x) = h_j^{\lim}, \quad j = 1, \ldots, s, \\
& \quad x \in \mathcal{X},
\end{aligned}
\]

(1)

where \( g_i(x) \) and \( g_i^{\lim} \) for \( i = 1, \ldots, r \) are the inequality constraint functions and their corresponding (non-zero) limiting values, respectively. Similarly, \( h_j(x) \) and \( h_j^{\lim} \) for \( j = 1, \ldots, s \) are the equality constraint functions and their corresponding (non-zero) limiting values, respectively.

3.1. A test suite for constrained MICFMO

Since no standard benchmark problem instances for MICFMO exist in the literature, a test suite of sixteen constrained MICFMO problems was created for this study. The reactor that we considered for the test suite is the SAFARI-1 nuclear research reactor in South Africa. The reactor operates in cycles typically lasting from 21 to 30 days, with 5-day shutdown periods inbetween. A fuel assembly usually remains in the core for 8 to 9 cycles (typically in a different loading position during each cycle) before being discharged.

SAFARI-1 is utilised for nuclear and materials research, as well as commercial irradiation services. Neutron scattering, radiography and diffraction experiments are performed at the reactor, utilising a number of neutron beam tubes located around the core. Several in-core irradiation positions are utilised for radioisotope production, primarily molybdenum-99 (\(^{99}\text{Mo}\)). Many of these radioisotopes are used for diagnostic purposes and therapeutic treatment of cancer. Moreover, an ex-core facility is utilised for silicon transmutation doping so as to produce silicon semiconductors for use in electronic equipment.

The SAFARI-1 core consists of a 9 \times 8 lattice which houses twenty-six fuel assemblies, six control rods, seven dedicated \(^{99}\text{Mo}\) production rig facilities, two general-purpose isotope production rig (IPR) facilities, as well as other core components which we do not specify in detail. The core layout of the reactor, depicting all these components, is presented in Figure 1.

Since the SAFARI-1 reactor is utilised for multiple purposes, there are several objectives that may be pursued simultaneously in fuel assembly loading decisions. This is a major benefit for the test suite because a diversity of objective space landscapes can be considered for MICFMO — something...
that has not received much attention in the literature. Similarly, a more comprehensive constraint set can also be imposed. A large and generalised decision space is also affected by the SAFARI-1 reactor because fuel assemblies are considered to be distinct from one another. This is due to the different burnup levels and profiles that they accrue when moving through the core. Therefore, the decision space cannot be reduced by considering batches of assemblies (which is typically done for power reactors).

A total of seven typical objectives associated with SAFARI-1 were considered to form part of the test suite and are listed in Table 1(a). Furthermore, the constraint set imposed on all the problems in the test suite consists of a fixed set of safety and utilisation requirements for SAFARI-1. The specific limiting values of these requirements are proprietary knowledge and are therefore not divulged in this paper. The set of inequality constraints are:

- The total $^{99}$Mo production must be greater than the demand;
- The $^{99}$Mo yield for each rig must be above a minimum limit;
- The peak axial production capability in each IPR must be above a specified threshold;
- The relative core power peaking factor must be below the safety limit;
- The total control bank worth must be above the safety limit;
- The shutdown margin must be above the safety limit.
Using the objectives in various combinations, along with the constraint set, a test suite of three classes of multiobjective optimisation problem instances was constructed for our comparative study. Class 1 contains six bi-objective problem instances, class 2 contains six tri-objective problem instances and class 3 contains four tetra-objective problem instances. These problems are identified and presented in Table 1(b).

Table 1: The MICFMO test suite based on the SAFARI-1 reactor. Each test problem is denoted by “P#. #” in which the first number corresponds to the class it belongs to, and the second number is the enumeration over the problems.

(a) The objectives forming part of the test suite

<table>
<thead>
<tr>
<th>Number</th>
<th>Goal</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Maximise</td>
<td>Fuel cycle length</td>
</tr>
<tr>
<td>2</td>
<td>Minimise</td>
<td>Relative core power peaking factor</td>
</tr>
<tr>
<td>3</td>
<td>Maximise</td>
<td>Total $^{99}$Mo production</td>
</tr>
<tr>
<td>4</td>
<td>Maximise</td>
<td>Utilisation of the silicon doping facility</td>
</tr>
<tr>
<td>5</td>
<td>Maximise</td>
<td>Utilisation at beam tubes 1&amp;2</td>
</tr>
<tr>
<td>6</td>
<td>Maximise</td>
<td>Utilisation at beam tube 5</td>
</tr>
<tr>
<td>7</td>
<td>Maximise</td>
<td>Isotope production in the IPR facilities</td>
</tr>
</tbody>
</table>

(b) The test problems classified into three classes

<table>
<thead>
<tr>
<th>Problem</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>P1.1</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>P1.2</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>P1.3</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>P1.4</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>P1.5</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>P1.6</td>
<td>✓ ✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>P2.1</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>P2.2</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>P2.3</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>P2.4</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>P2.5</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>P2.6</td>
<td>✓ ✓ ✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>P3.1</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>P3.2</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>P3.3</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>P3.4</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>
3.2. Artificial neural networks for ICFMO

Recall that one of the challenging characteristics of ICFMO is computationally demanding function evaluations. In order to evaluate the suitability of any configuration in terms of its objective and constraint function values, a reactor core simulator is usually employed. The simulator used to support the operation of SAFARI-1 is called OSCAR-4, which is an acronym for *overall system for the calculation of reactors*, version 4 [31]. Evaluating a single reload configuration for SAFARI-1 using OSCAR-4 requires approximately four minutes of computation time on a personal computer, given the objectives and constraints forming part of our test suite. The computational cost thus introduced by the simulator severely hinders efforts towards performing comparative studies that involve MICFMO because several thousands (even millions) of function evaluations are typically required. It would therefore be highly desirable if the computational cost of function evaluations using OSCAR-4 could be reduced.

In earlier work [32], we constructed *artificial neural network* (ANN) surrogate models for the prediction of the SAFARI-1 objectives and constraints mentioned in Section 3.1. Those ANNs were specifically created (using the Neural Network Toolbox [33] within the MATLAB software suite [34]) to fulfil the need for reducing the computational time required within our comparative study. We therefore employ those ANNs for function evaluations within all the metaheuristics in this paper instead of using the OSCAR-4 simulator. The reader is referred to the original paper [32] for details on the ANNs. Only a summary of the most important aspects thereof are reproduced here.

One multilayer feedforward neural network per objective or constraint was constructed. Each network consists of an input layer with twenty-six neurons (corresponding to the fuel loading positions), an output layer with one neuron (corresponding to the objective or constraint), and one hidden layer with an empirically-determined number of neurons (typically 200 or 250 neurons). Inputs to the network are the uranium-235 isotope mass of each fuel assembly assigned to each loading position. The activation functions in each network consists of a hyperbolic tangent sigmoidal function for neurons in the hidden layer and a linear function for neurons in the output layer. In order to achieve good generalisation, the Bayesian regularisation backpropagation algorithm was used as the training algorithm for the networks [35]. The resulting ANNs yielded average absolute relative errors of less than 2% on test data sets while reducing the computation time by four orders of magnitude (when compared to the true OSCAR-4 values and computation times) [32].
4. Constraint handling techniques in multiobjective optimisation

Very little literature is available on constraint handling in the context of multiobjective optimisation. The effectiveness of a multiplicative penalty function constraint handling technique is investigated in this study (in the context of MICFMO) by comparing it to the constrained-domination technique from the literature. Brief descriptions of these two techniques are presented in this section.

4.1. The constrained-domination principle

Along with the NSGA-II algorithm, Deb et al. [6] proposed a constraint handling technique using their so-called constrained-domination principle (CDP). The authors modified the definition of domination between two solutions $x$ and $y$ as follows. Solution $x$ is said to constrained-dominate solution $y$ if any of the following three conditions hold:

1. Solution $x$ is feasible and solution $y$ is infeasible;
2. Both solutions $x$ and $y$ are infeasible, but $x$ exhibits a smaller overall constraint violation;
3. Both solutions $x$ and $y$ are feasible, and $x$ dominates $y$.

The constraint handling technique then uses this CDP during binary tournament selection instead of traditional domination. In addition, an infeasible solution with a larger overall constraint violation is sorted as a member of the next nondominated front during nondominated sorting [6]. This technique is hereafter referred to as the CDP technique and, being parameter-free, is one of its major advantages.

4.2. A multiplicative penalty function

The CDP technique cannot be adopted within any multiobjective metaheuristic (e.g. it is not suitable for MOOCEM). This led Schlünz et al. [14] to develop a new constraint handling technique based on a multiplicative penalty function (MPF). In their MPF technique, an exponential function $\phi(x)$ is used to calculate a scalar penalty value related to the magnitude of the constraint violations. A free parameter $\gamma$ within $\phi(x)$ may be chosen so as to amplify or reduce the severity of the constraint violations. Then, a penalised objective vector $f_\phi(x)$ is determined by multiplying the objective function vector in (1) by this scalar value, i.e. $f_\phi(x) = \phi(x)f(x)$. This effectively transforms the constrained multiobjective optimisation problem to an unconstrained problem.
The advantage of the MPF technique is that a single value is used to penalise all $q$ objective functions, irrespective of their orders of magnitude, allowing traditional domination principles to be applied thereafter using $f_\phi(x)$. In this paper, we amplified the severity of the constraint violations by selecting $\gamma = 3.0$ whenever the MPF technique was applied.

5. Multiobjective metaheuristics for MICFMO

In this section, we briefly describe the eight multiobjective metaheuristics, introduced to the reader in Section 1, that form part of our comparative study. Each metaheuristic is described in two paragraphs: the first contains a discussion on the general workings of the metaheuristic, whereas the second contains the problem-specific modifications we had to make. Unless specifically stated otherwise, both constraint handling techniques described in Section 4 have been implemented within the metaheuristics. As a final note, the terminology employed within the descriptions of this section corresponds to that adopted within the source-material for each metaheuristic. The reader is therefore referred to those papers for any clarification.

5.1. NSGA-II

The highly-popular NSGA-II was developed by Deb et al. [6] as an improvement over its predecessor. Within the NSGA-II, the fitness of a solution is determined by the nondominated front it belongs to, as well as its crowding distance (a measure of the density of solutions surrounding a particular solution). Accordingly, the so-called crowded comparison operator is used to distinguish superiority between two solutions. The NSGA-II generates an offspring population of solutions during each generation by means of traditional crossover and mutation operators, using crowded comparison binary tournament selection. The offspring and parent populations are then pooled into a combined set of solutions. This set is partitioned into different nondominated fronts by means of a fast nondominated sorting algorithm. Within each front, solutions are ranked (and sorted again) according to their crowding distance, which promotes diversity. A new population is then selected in an elitist fashion by truncating the sorted, combined set of solutions to the required size.

In order to apply the NSGA-II to MICFMO problems, we have to specify an encoding scheme for the solutions, as well as the corresponding crossover and mutation operators associated with any genetic algorithm. Since a reload configuration may be represented by a permutation decision vector, we naturally opted for a permutation-based encoding within NSGA-II.
There are a number of dedicated permutation-based crossover and mutation operators available in the evolutionary computation literature. As such, we performed a pilot study involving the partially matched/mapped crossover, the position-based crossover and the cycle crossover operators [36], as well as the swap and scramble mutation operators [37]. It was found that partially matched/mapped crossover and scramble mutation yielded the most promising results and, as such, we implemented them within our NSGA-II implementation.

5.2. SPEA2

Zitzler et al. [7] developed the SPEA2, also as in improvement over its predecessor. The SPEA2 employs a regular population as well as an external archive containing nondominated solutions previously found. During each generation, fitness values are calculated for all the solutions in the combined set formed by the population and archive. The fitness of a solution is determined by its so-called strength value (which takes into account the number of solutions that it dominates, and by which it is dominated) and a density measure (which is based on the \(k\)-th nearest neighbour method). The nondominated solutions in the combined set are then transferred to the next generation’s archive. If the number of nondominated solutions exceeds the archive size, a truncation rule, also based on the \(k\)-th nearest neighbour method, is applied. If, however, the number of nondominated solutions is smaller than the archive size, the remainder of the archive is filled with dominated solutions having the best fitness values. A new offspring population is then created by means of traditional crossover and mutation operators, using binary tournament selection (according to fitness) on the archived solutions.

As with the NSGA-II, we opted for a permutation-based encoding within SPEA2 and also performed a pilot study involving the same three crossover operators, and the same two mutation operators. It was found that partially matched/mapped crossover and scramble mutation yielded the most promising results and therefore we implemented them within our SPEA2 implementation.

5.3. OMOPSO

In an experimental comparison between six representative, state-of-the-art multiobjective particle swarm optimisers, it was found that the OMOPSO algorithm (developed by Reyes Sierra & Coello Coello [8]) was the most salient [38]. The OMOPSO algorithm employs a global-best leader swarm consisting of nondominated solutions, a regular swarm and an external
archive. Nondominated solutions in the leader swarm are differentiated from one another by means of a crowding distance (as defined for the NSGA-II). For each solution in the regular swarm, during each generation, a global-best solution is then selected by means of a binary tournament (according to crowding distance) applied to the leader swarm. Following the creation of a new swarm by flight operators (i.e. velocity and position updates), the swarm is partitioned into three groups so as to apply different mutation operators (borrowed from the literature on evolutionary algorithms). The personal-best of each solution is then updated if the newly-created solution dominates its personal-best solution, or if they are nondominated with respect to each other. Similarly, the global-best leader swarm is updated in order to contain only nondominated solutions.

As with any particle swarm algorithm, OMOPSO was proposed originally for continuous optimisation problems and we therefore have to modify it in order to solve MICFMO problems. We performed a pilot study in which two combinatorial approaches available in the literature were compared for this purpose. In the first approach, the random keys method \[39\] is used to decode real-valued vectors into permutations. In the second approach, proposed by Hu et al. \[40\], solutions are explicitly represented as permutations while the velocity-update flight operator is redefined based on the similarity of two solutions. It was found that the permutation-based approach by Hu et al. \[40\] yielded the most promising results and, as such, we implemented it in our OMOPSO implementation. For mutation, we partition the swarm into two groups (instead of three). No mutation occurs in the first group, while the second group undergoes swap mutation. Finally, we opted not to maintain an external archive as originally proposed for OMOPSO. Instead, our implementation yields the leader swarm as the final result (as was the case in the experimental comparison in \[38\]).

5.4. P-ACO

The P-ACO algorithm was developed by Doerner et al. \[9\] for solving the multiobjective portfolio selection problem. P-ACO utilises a single colony of ants, a separate pheromone matrix per objective, and a single heuristic information matrix. For each ant, the pheromone matrices are aggregated together as a linear weighted sum using different randomly generated objective weight vectors. This aggregated value, along with the heuristic information, is then used within the transition decision rule. Local pheromone updates are performed whenever an ant has traversed an edge, whereas global pheromone updates are performed using only the best and second-best solutions (according to each objective) after all solutions have been created.
Furthermore, an external archive of nondominated solutions is maintained during the algorithm’s execution.

In order to apply P-ACO to our MICFMO problem instances, we have to specify the heuristic information matrix and a constraint handling technique. Since a single heuristic information matrix cannot cater simultaneously to multiple conflicting objectives, we opted instead for safety-related constraint information. A well-known fuel management philosophy, called highest-mass to lowest-flux, attempts to flatten the neutron flux profile over the core, thus typically yielding a safe configuration. We incorporated this philosophy into a heuristic information matrix for the SAFARI-1 reactor as follows. A normalised thermal neutron flux profile in the fuel loading positions was determined for a SAFARI-1 core loaded with fresh fuel in order to obtain a typical profile. Then, we calculated the outer product between this flux profile and the normalised (with respect to uranium-235 isotope mass) fuel assembly distribution. The resulting matrix is then inverted (and normalised again) in order to yield a heuristic information matrix in which larger values represent preferred assignments of fuel assemblies to loading positions according to the highest-mass to lowest-flux philosophy. In respect of constraint handling, the CDP technique is not appropriate for P-ACO since domination per se is not utilised within the algorithm. As such, we only incorporated the MPF technique in our P-ACO implementation.

5.5. AMOSA

The AMOSA algorithm, developed by Bandyopadhyay et al. [10], is one of the latest multiobjective simulated annealers. As the name suggests, AMOSA incorporates an archive of nondominated solutions obtained during its execution. Nondominated solutions are added to the archive as they are obtained. Whenever the size of the archive reaches a soft limit, the well-known single-link clustering algorithm [41] is applied in order to reduce the archive size again to a hard limit. AMOSA incorporates the concept of amount of domination when calculating the acceptance probability of a new solution. This amount depends on the dominance between the current solution, the new solution and the archive solutions.

In order to apply AMOSA to our MICFMO problems, we have to specify a solution representation and corresponding neighbourhood move operator. We opted for a permutation-based representation, and performed a pilot study between the swap and scramble mutation operators as our neighbourhood move operators. It was found that the scramble operator yielded the most promising results and therefore it was incorporated in our AMOSA.
implementation. Finally, instead of using a fixed minimum (final) temperature as stopping criterion in AMOSA, we rather opted for a fixed number of evaluations so as to control the duration of the algorithm explicitly.

5.6. MOVNS

Geiger [11] developed the MOVNS algorithm for solving the permutation flow shop scheduling problem. In MOVNS, a set of nondominated solutions is maintained. During each iteration, a solution from this set, whose neighbourhood has not been explored yet, is randomly selected. Similarly, a neighbourhood move operator from a predefined set of neighbourhood operators is randomly selected. Then, the neighbourhood of the solution is generated and the nondominated set of solutions is updated. If the solution is still present in the nondominated set, it is marked as explored. This procedure is repeated until all solutions in the nondominated set have been explored using all the neighbourhood move operators. Three variants of MOVNS have been proposed by Liang & Chuang [42], namely basic, perturbation and perturbation + base solution. The variants involve different strategies for selecting a solution to be explored, as well as the marking of selected solutions in the nondominated set.

We opted for a permutation-based representation of solutions in our implementation, and defined the swap and scramble mutation operators as our set of neighbourhood move operators. Due to the limited number of function evaluations available for MICFMO and the large neighbourhood sizes associated with the operators, we only generate a fixed number of neighbouring solutions during each iteration of MOVNS instead of considering the entire neighbourhood. A pilot study was performed between the three variants proposed by Liang & Chuang [42] and it was found that the perturbation variant yielded the most promising results in the context of MICFMO. We therefore employed this variant of MOVNS within our comparative study.

5.7. MOOCEM

The MOOCEM was recently developed by Bekker & Aldrich [12] as a multiobjective adaptation of the cross-entropy method for optimisation. It was specifically designed for reducing the number of function evaluations of computationally expensive simulation-based optimisation problems and is therefore a natural choice for application to the MICFMO problem. In the MOOCEM, solutions are sampled (i.e. generated) from a parameterised sampling distribution during each iteration, while an archive of “best-found” solutions (in terms of Pareto rank) is maintained. The Kullback-Leibler
**divergence** is used as a measure of “distance” between two sampling distributions. The MOOCEM re-estimates the current sampling distribution according to a problem-specific updating rule (which incorporates the solutions in the archive) until it converges to a distribution which concentrates its probability mass in the vicinity of Pareto optimal solutions.

In order to apply the MOOCEM to our MICFMO problem instances, we have to specify how solutions are sampled, which family of sampling distributions is used, and determine the problem-specific updating rule. Also, since the MOOCEM was proposed for unconstrained optimisation problems, we have to specify an appropriate constraint handling technique. Schlinz *et al.* [14] first applied the MOOCEM to the MICFMO problem. In that paper, a permutation-based solution representation was adopted, the corresponding updating rule for MICFMO was derived, and the MPF technique for handling constraints was proposed. Accordingly, in this paper, we adopted the MOOCEM algorithm for MICFMO using (only) the MPF technique exactly as described in [14].

### 5.8. MOHS

Sivasubramani & Swarup [13] proposed an MOHS algorithm which borrows largely from the NSGA-II in order to extend its single-objective variant to solve multiobjective optimisation problems. In the MOHS algorithm, a set of new solutions (equal in number to the harmony memory size) is generated during each iteration according to the same improvisation guidelines used in the single-objective algorithm. Then, the new set is combined with the current harmony memory and the fast nondominated sorting algorithm of the NSGA-II is used to partition the combined set into different nondominated fronts. Within each front, solutions are ranked (and sorted again) according to their crowding distance (also borrowed from the NSGA-II). The new harmony memory is then selected by truncating the sorted, combined set of solutions to the required size.

In order to apply the MOHS algorithm to our MICFMO problem instances, we have to specify a solution representation and possibly adapt the improvisation guidelines for generating new solutions accordingly. The single-objective harmony search algorithm has already been adapted for application to the ICFMO problem using a permutation-based representation (see *e.g.* [30]). We therefore adopted the same representation and corresponding improvisation guidelines specified in [30] for our MOHS implementation.
5.9. Tuning parameters within the metaheuristics

Qualitative pilot studies were performed in order to determine reasonable values for each metaheuristic’s tuning parameters. An emphasis was also placed on using comparable population sizes so as to facilitate fair comparisons between the metaheuristics. We acknowledge that the values are not optimal for each metaheuristic-problem instance combination. Since the aim of this work, however, is to provide a baseline and guide for future developments, we did not place emphasis on very detailed parameter tuning. The outcomes of these pilot studies are summarised below and the parameter values were found to be robust over the different test problems in all three classes. For all subsequent comparative experiments in this study, we adopted the parameter values reported in this section.

For both the NSGA-II and the SPEA2, the population size was selected as 30, the crossover probability as 0.9 and the mutation probability as $1/n$ where $n$ denotes the length of the permutation vector. Furthermore, the archive size in the SPEA2 was selected as 30. For the OMOPSO method, the swarm size and maximum number of leaders were selected as 30, and the mutation probability as $1/n$. The number of ants in the P-ACO algorithm was selected as 30 and the remaining parameters, following the notation in [9], were selected as $\alpha = 1.75$, $\beta = 0.75$, $q_0 = 0.8$, $\rho = 0.2$ and $\tau_0 = 1$. In the AMOSA algorithm, the hard and soft archive limits were selected as 30 and 45, respectively, the cooling rate as 0.85, the initial temperature as 100, and the number of iterations for each temperature step as 15. The fixed number of neighbouring solutions that are generated during each iteration of the MOVNS algorithm was selected as 15. In the MOOCEM, the sample size was selected as 30, the smoothing parameter as 0.8 and the initial rank threshold as 0. Finally, the harmony memory size in the MOHS algorithm was selected as 30, the memory consideration rate as 0.9 and the pitch adjustment rate as 0.25. Note that, wherever the scramble operator is used, a subset of size 4 was selected which does not necessarily contain contiguous positions.

5.10. Summary of our contributions in respect of metaheuristic implementation and application

A number of contributions to the existing body of knowledge are made by our modifications and application of the metaheuristics in this paper. To our knowledge, this work represents the first application of the SPEA2, the OMOPSO algorithm, the P-ACO algorithm, the AMOSA algorithm, the MOVNS algorithm and the MOHS algorithm to the MICFMO problem.
In addition, it also represents the first incorporation (to our knowledge) of multiobjective constraint handling technique(s) within the OMOPSO, P-ACO, MOVNS and MOHS algorithms.

Recall that Hedayat et al. [29] were the first to apply the NSGA-II to the MICFMO problem. In their work, they also adopted a permutation-based encoding; however, they implemented the traditional two-point crossover operator followed by a repair procedure in order to create valid permutations. They also implemented the swap mutation operator, and an additive penalty function approach for constraint handling. Our implementation of the NSGA-II, using partially matched/mapped crossover and scramble operators, and the CDP an MPF constraint handling techniques, therefore represents an alternative to that already available in the MICFMO literature.

Our implementation of the OMOPSO algorithm also demonstrates the applicability of the permutation-based approach by Hu et al. [40] as an alternative to the random keys approach used in the single-objective particle swarm optimisation ICFMO literature [43]. Similarly, the manner in which we generated our heuristic information for the P-ACO algorithm represents an alternative to that which has been adopted in the ICFMO literature, see e.g. [44].

6. Performance assessment for multiobjective metaheuristics

The aim of multiobjective optimisation, as stated in [16], is to find a good approximation of the set of Pareto optimal solutions for a given problem. Each of the metaheuristics considered in this study yields a Pareto set approximation as output. Since the true Pareto set for a real-world problem (like the ICFMO problem) is, however, typically not available, we do not make any claim in respect of closeness to optimality when referring to an approximation. In the discussion to follow, the nondominated front corresponding to a Pareto set approximation is referred to as an approximation set.

In order to assess the performance of each metaheuristic, we necessarily have to perform a comparative analysis between the approximation sets yielded by each algorithm. Several indicators have been proposed in the literature that assign a scalar value to an approximation set as a measure of its quality [45]. The unary hypervolume indicator (suggested by Zitzler & Thiele [46]) and the unary R2 indicator (proposed by Hansen & Jaszkiewicz [47]) are two recommended indicators which fulfil the desirable property of monotonicity (or Pareto compliance) [16, 45, 48].
The hypervolume indicator measures the portion of objective space that is dominated by an approximation set, relative to a reference point (which has to be dominated by the entire approximation set). In this paper, a variant of the hypervolume indicator, called the hypervolume difference to reference set (denoted by $I_{HVD}$) [16], is adopted as our first performance measure. Note that smaller values of $I_{HVD}$ are preferred.

The $R^2$ indicator (denoted by $I_{R^2}$) is based on a set of utility functions and is adopted as our second performance measure. As with $I_{HVD}$, smaller values of $I_{R^2}$ are preferred. We employed a set of augmented weighted Chebyshev utility functions for the $R^2$ indicator, as recommended in [16, 47]. Sets of “uniformly dispersed” weight vectors are required in the calculation of the $R^2$ indicator. We therefore employed the approach proposed in [47] for generating these sets. Normalised weight vector sets of size 500, 741 and 969 were thus created for class 1, class 2 and class 3 problems (as defined in Section 3.1), respectively.

6.1. Experimental design

The eight metaheuristics were implemented within the MATLAB software suite [34]. All calculations in this study were performed on an Intel® Core™ i7-2720QM computer (2.20 GHz) with 8.00 GB of RAM, running a 64-bit Windows 7 Professional operating system.

Since the SAFARI-1 reactor experiences shutdown periods lasting five days, only three days of computation time during those periods are realistically available for optimisation. This corresponds to approximately 1000 evaluations using the OSCAR-4 simulator. Although many more evaluations are possible in that time using the ANNs, we impose a limit (i.e. stopping criterion) of 1050 evaluations for each metaheuristic in this study, thereby corresponding to the practical limitation.

Each of the metaheuristics considered in this paper is a stochastic algorithm, and may therefore yield different approximation sets if they are applied multiple times to the same problem. In order to obtain a fairly representative indication of average performance, fifty optimisation runs of each metaheuristic (and its constraint handling technique variants) were executed in respect of each test problem. The outcome of each run is an approximation set consisting of feasible solutions only. An attempt was made to reduce variability due to initial conditions and random processes by employing a

---

1 A utility function represents a decision maker’s preferences in explicit mathematical form such that the function provides a complete ordering in the objective space [22].
fixed set of different random number generator seeds in each case, along
with a fixed set of uniformly sampled random initial solutions. Accordingly,
so-called matched samples (i.e. results) have been obtained within each test
problem over the different metaheuristics.

For ease of comparison, the function values obtained for objective 2
(which is the only minimisation objective) were linearly transformed so as
to correspond to values in a maximisation paradigm. Thereafter, objective
function values within all the approximation sets were scaled to the range
(0, 1) so that the values are of approximately the same magnitude.

In order to calculate the indicator values $I_{HVD}$ and $I_{R^2}$ for any approxi-
mation set in a given test problem, we also require a reference approximation
set. For each test problem, the approximation sets yielded by all of the runs
(for all the metaheuristics) were pooled together. A corresponding reference
approximation set was then determined by isolating the nondominated solu-
tions from this pool. In addition, the reference points used in the calculation
of $I_{HVD}$ were chosen as $[0, 0], [0, 0, 0]$ and $[0, 0, 0, 0]$ for problem classes 1,
2 and 3, respectively. Similarly, the ideal points used in the calculation of
$I_{R^2}$ were chosen as $[1.01, 1.01], [1.01, 1.01, 1.01]$ and $[1.01, 1.01, 1.01, 1.01]$ for
problem classes 1, 2 and 3, respectively.

6.2. Statistical analysis

Derrac et al. [49] discuss two types of analyses that are typically per-
formed in comparative studies, namely single-problem and multi-problem
analyses. In a single-problem analysis, results obtained over several runs of
metaheuristics on a given problem are considered, whereas a multi-problem
analysis considers a result per metaheuristic/problem pair [49]. The out-
comes of both these types of analyses are presented in this paper in order
to demonstrate the inferences that we may draw from them.

As suggested in [16, 49], we adopt a hypothesis testing approach from
the field of inferential statistics in order to analyse our comparative results.
The null hypothesis, denoted by $H_0$, is typically a statement of no effect or
no difference in solution quality, while the alternative hypothesis, denoted
by $H_1$, corresponds to the presence of an effect or difference [49]. A sta-
tistical test is applied using samples of data in order to determine whether
the assumed $H_0$ should be rejected in favour of $H_1$, or not. This rejection
is determined by a user-defined significance level, denoted by $\alpha$. The sta-
tistical test can yield a so-called $p$-value which represents the probability
of obtaining an effect at least as extreme as that obtained in the samples,
assuming $H_0$ is true [49]. If the $p$-value is smaller than $\alpha$, we can reject $H_0$
in favour of $H_1$ at a significance level of $\alpha$. 20
Nonparametric tests [17] are recommended when comparing metaheuristics due to the distribution-free property of these tests [16, 49]. Accordingly, we employed the nonparametric Wilcoxon signed rank test in our constraint handling technique comparison, and the nonparametric Friedman test with the Nemenyi post-hoc procedure in our metaheuristic comparison [17, 18]. In all cases, we adopted a significance level of $\alpha = 0.05$.

7. Results and discussion

The comparative study results that were obtained according to the experimental design discussed in Section 6.1 are analysed in two stages. In the first stage, we consider each of the six metaheuristics in which both constraint handling techniques have been implemented, separately. The aim of this stage is to determine whether the new MPF technique is a competitive alternative to the CDP technique. The best-performing variants of these six metaheuristics are carried forward into the second stage, along with the two remaining metaheuristics. In this second stage then, we consider the eight metaheuristics together in order to determine their capabilities for solving constrained MICFMO problems and gauge their comparative performances.

7.1. Constraint handling techniques

We first perform a single-problem analysis for the two NSGA-II variants. Therefore, we have two samples of data for each test problem (matched pairs of indicator values corresponding to the fifty runs, obtained by the MPF and CDP variants). In the Wilcoxon signed rank test, two samples are first converted into a single sample (i.e. the samples’ differences) in order to test whether this new sample has a median of zero [17]. Let $\Delta I_{HVD}$ denote the converted sample of $I_{HVD}$ values, i.e. the difference in $I_{HVD}$ between using the MPF and CDP techniques. Similarly, let $\Delta I_{R2}$ denote the converted sample of $I_{R2}$ values. Negative values in the converted samples therefore correspond to superior performance by the MPF technique, whereas positive values indicate superior performance by the CDP technique.

Box plots are often used in exploratory data analyses and provide a comprehensive view of the central tendency and spread of samples. Their use is generally more informative than the use of averages and standard deviations of samples. We present the converted samples $\Delta I_{HVD}$ and $\Delta I_{R2}$ obtained for each test problem in the form of box plots in Figure 2. As an extension to the traditional box plot, we also included the average value of each sample as a black diamond point in the graphs.
Figure 2: Box plots of the converted samples $\Delta I_{HVD}$ (on the left) and $\Delta I_{R2}$ (on the right) obtained for each test problem using the NSGA-II.

Note that we do not explicitly wish to compare the box plots to one another in Figure 2. The aim is rather to investigate whether the samples are symmetrically distributed about a median of zero, or not (so as to determine whether one constraint handling technique outperforms the other). We observe in Figure 2 that the samples are, in fact, generally well-distributed about zero, thus indicating that both constraint handling techniques yield similar performances in respect of the $I_{HVD}$ and $I_{R2}$ performance indicators. There are, however, indications that the MPF technique outperforms
the CDP technique (with respect to both $I_{HVD}$ and $I_{R2}$) for some problems in class 1 (e.g. P1.3 and P1.4) and in class 2 (e.g. P2.1, P2.6).

In order to determine whether there are statistically significant differences between the MPF and CDP technique, we apply a two-tailed Wilcoxon signed rank test to each test problem. The resulting $p$-values are presented in Table 2, where a bold-faced entry denotes a significant difference (for $\alpha = 0.05$). In the event of a significant difference, we can refer back to the box plots in Figure 2 to pronounce which technique outperformed the other.

In the majority of instances, according to Table 2, there is no statistically significant difference between the two constraint handling techniques (with respect to both $I_{HVD}$ and $I_{R2}$). The MPF technique, however, outperforms the CDP technique in the five instances where significant differences do occur. It can therefore be inferred, based on the single-problem analysis, that the new MPF technique is a competitive alternative to the CDP technique within the context of the NSGA-II.

Let us consider next a multi-problem analysis, within each class of problems, for the two NSGA-II variants. Accordingly, we have an average indicator value (calculated over the fifty runs) per variant/problem pair. As before, we can convert two samples (now consisting of average indicator values) into a single sample by calculating their difference, i.e. $\Delta I_{HVD}$ and $\Delta I_{R2}$. These converted samples are presented in Table 3. Note that the values correspond to the diamond points in Figure 2.

We observe in Table 3 that all values in the samples for class 1 are negative, thus suggesting that the MPF technique outperforms the CDP

---

Table 2: Single-problem analysis for comparing constraint handling techniques within the NSGA-II. The table contains the $p$-values obtained by two-tailed Wilcoxon signed rank tests applied to the samples $\Delta I_{HVD}$ and $\Delta I_{R2}$ for each problem. Bold-faced entries represent a statistically significant difference (for $\alpha = 0.05$).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Wilcoxon signed rank test $p$-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta I_{HVD}$</td>
<td>P1.1 0.0849, P1.2 0.7102, P1.3 0.0733, P1.4 0.0733, P1.5 0.2994, P1.6 0.0604</td>
</tr>
<tr>
<td>$\Delta I_{R2}$</td>
<td>P1.1 0.0688, P1.2 0.6675, P1.3 0.0254, P1.4 0.1080, P1.5 0.2994, P1.6 0.0452</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample</th>
<th>Wilcoxon signed rank test $p$-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta I_{HVD}$</td>
<td>P2.1 0.0578, P2.2 0.0659, P2.3 0.4780, P2.4 0.3667, P2.5 0.8281, P2.6 0.0020</td>
</tr>
<tr>
<td>$\Delta I_{R2}$</td>
<td>P2.1 0.0126, P2.2 0.6887, P2.3 0.6958, P2.4 0.7684, P2.5 0.5921, P2.6 0.0020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample</th>
<th>Wilcoxon signed rank test $p$-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta I_{HVD}$</td>
<td>P3.1 0.3719, P3.2 0.3566, P3.3 0.7102, P3.4 0.9730</td>
</tr>
<tr>
<td>$\Delta I_{R2}$</td>
<td>P3.1 0.4258, P3.2 0.9193, P3.3 0.3876, P3.4 0.9040</td>
</tr>
</tbody>
</table>
Table 3: Converted samples, consisting of the average difference between indicator values, to be used in a multi-problem analysis within each class of problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>$\Delta I_{HVD}$</th>
<th>$\Delta I_{R2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.1</td>
<td>$-0.0144$</td>
<td>$-0.0103$</td>
</tr>
<tr>
<td>P1.2</td>
<td>$-0.0045$</td>
<td>$-0.0030$</td>
</tr>
<tr>
<td>P1.3</td>
<td>$-0.0104$</td>
<td>$-0.0065$</td>
</tr>
<tr>
<td>P1.4</td>
<td>$-0.0249$</td>
<td>$-0.0123$</td>
</tr>
<tr>
<td>P1.5</td>
<td>$-0.0134$</td>
<td>$-0.0061$</td>
</tr>
<tr>
<td>P1.6</td>
<td>$-0.0103$</td>
<td>$-0.0056$</td>
</tr>
<tr>
<td>Class 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.1</td>
<td>$-0.0094$</td>
<td>$-0.0067$</td>
</tr>
<tr>
<td>P2.2</td>
<td>$0.0038$</td>
<td>$0.0005$</td>
</tr>
<tr>
<td>P2.3</td>
<td>$0.0015$</td>
<td>$-0.0003$</td>
</tr>
<tr>
<td>P2.4</td>
<td>$0.0022$</td>
<td>$0.0005$</td>
</tr>
<tr>
<td>P2.5</td>
<td>$-0.0038$</td>
<td>$-0.0025$</td>
</tr>
<tr>
<td>P2.6</td>
<td>$-0.0214$</td>
<td>$-0.0076$</td>
</tr>
<tr>
<td>Class 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.1</td>
<td>$0.0022$</td>
<td>$0.0017$</td>
</tr>
<tr>
<td>P3.2</td>
<td>$0.0026$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>P3.3</td>
<td>$-0.0013$</td>
<td>$-0.0014$</td>
</tr>
<tr>
<td>P3.4</td>
<td>$-0.0013$</td>
<td>$-0.0005$</td>
</tr>
</tbody>
</table>

The mixture of positive and negative values in the class 2 and class 3 samples suggest that there is little difference between the two techniques there.

In order to determine whether there are statistically significant differences between the two constraint handling techniques in a multi-problem analysis, we apply a two-tailed Wilcoxon signed rank test to the average samples for each class of problems. The resulting $p$-values are presented in Table 4. As before, a bold-faced entry denotes a significant difference (for $\alpha = 0.05$).

Table 4: Multi-problem analysis for comparing constraint handling techniques within the NSGA-II. The table contains the $p$-values obtained by two-tailed Wilcoxon signed rank tests applied to the samples $\Delta I_{HVD}$ and $\Delta I_{R2}$ for each problem class. Bold-faced entries represent a statistically significant difference (for $\alpha = 0.05$).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Wilcoxon signed rank test $p$-values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
</tr>
<tr>
<td>$\Delta I_{HVD}$</td>
<td>$0.03125$</td>
</tr>
<tr>
<td>$\Delta I_{R2}$</td>
<td>$0.03125$</td>
</tr>
</tbody>
</table>

Although the sample size is small, the multi-problem analysis shows that there is a statistically significant difference between the two constraint
handling techniques within problem class 1. Based on the values in Table 3, it is the MPF technique which outperforms the CDP technique in that class (with respect to both $I_{HVD}$ and $I_{R2}$). In classes 2 and 3, however, there are again no significant differences. As was the case in the single-problem analysis, it can also be inferred, based on the multi-problem analysis, that the new MPF technique is a competitive alternative to the CDP technique within the NSGA-II.

The above-mentioned single-problem and multi-problem analyses were performed not only for the NSGA-II, but also for the SPEA2 and the OMOPSO, AMOSA, MOVNS and MOHS algorithms. Due to space limitations, the details of those analyses are not reproduced here. Instead, we only present a summary thereof.

**SPEA2:** The single-problem analysis for the SPEA2 yielded very similar results to that of the NSGA-II. There were four instances in which significant differences were detected. Based on the box plots, the MPF technique outperformed the CDP technique in three of those instances. The multi-problem analysis revealed that there is only a significant difference in class 1, with the sample values indicating that it is the MPF technique which outperforms the CDP technique in that class.

**OMOPSO:** A significant difference was detected for only one instance in the single-problem analysis and it was in favour of the MPF technique. The multi-problem analysis did not reveal any significant differences in any of the three classes.

**AMOSA:** The single-problem analysis for the AMOSA algorithm yielded very interesting results. Highly-significant differences were detected in all the instances in class 1, with most $p$-values less than 0.0005. According to the box plots, the MPF technique outperformed the CDP technique in all those instances. AMOSA, therefore, strongly benefits from the MPF technique with respect to both the $I_{HVD}$ and $I_{R2}$ indicators when solving the bi-objective test problems. In five additional instances, significant differences were also detected of which four were in favour of the MPF technique. Unsurprisingly, the multi-problem analysis revealed a significant difference in class 1, with the sample values indicating that it is the MPF technique which outperforms the CDP technique. There were no statistically significant differences in the remaining two problem classes.

**MOVNS:** There were five instances in the single-problem analysis in which significant differences were detected. Four of those were in favour of the MPF technique according to the box plots. The multi-problem analysis did not reveal any significant differences in any of the three classes.

**MOHS:** In the single-problem analysis, significant differences were de-
ected in seven instances. Based on the box plots, the MPF technique outperformed the CDP technique in six of those instances. The multi-problem analysis did not reveal any statistically significant differences in any of the three classes.

The new MPF technique is therefore not only a competitive alternative to the CDP technique within NSGA-II, but also within SPEA2, OMOPSO, AMOSA, MOVNS and MOHS. The analyses further indicate that the MPF technique performs particularly well in bi-objective problems.

7.2. Metaheuristic solution comparison

Following the outcome of the constraint handling comparison, we selected the MPF technique within all the metaheuristics for solving class 1 and class 2 problems. The CDP technique was, however, selected in conjunction with the relevant metaheuristics for solving class 3 problems.

We first consider a single-problem analysis for the eight metaheuristics. Accordingly, we have eight samples of data for each test problem (matched indicator values corresponding to the fifty runs). The results/samples obtained for problem classes 1, 2 and 3 are presented in the form of box plots in Figures 3, 4 and 5, respectively. As before, we include the average value of each sample as black diamond points in the box plots. The aim, this time, is to compare the box plots within each graph to one another i.e. a visual analysis for each test problem with respect to both the $I_{HVD}$ and $I_{R2}$ indicators.

We observe in Figures 3–5 that the OMOPSO algorithm performs poorly with respect to both indicators across all three problem classes. The performance of the OMOPSO algorithm is, however, particularly poor in class 1, whereas the other seven metaheuristics perform similarly, as may be seen in Figure 3. Although the two local search metaheuristics, the AMOSA algorithm and the MOVNS algorithm, perform well for the bi-objective problems in class 1, they do not appear to scale well to the problems in classes 2 and 3 which have more objectives, as may be seen in Figures 4 and 5. This behaviour is apparent for both indicators.

In general, it appears that the MOHS algorithm is the most robust in terms of sample variability, although its performance in the majority of problems are only average. The NSGA-II, the P-ACO algorithm and the MOOCEM, on the other hand, appear to perform consistently well across all three problem classes for both indicators. The SPEA2 also performs well in most cases, but sporadically suffers from poor performance — e.g. in problems P2.3, P3.1 and P3.3, as may be seen in Figures 4 and 5.
Figure 3: Box plots of the samples $I_{HVD}$ (on the left) and $I_{R2}$ (on the right) obtained by all eight metaheuristics for problem class 1.

In order to determine whether there is a statistically significant difference between the metaheuristics in a single-problem analysis, we apply the Friedman test to each test problem. The test, however, only tells us whether
Figure 4: Box plots of the samples $I_{HVD}$ (on the left) and $I_{R_2}$ (on the right) obtained by all eight metaheuristics for problem class 2.

Figure 4: Box plots of the samples $I_{HVD}$ (on the left) and $I_{R_2}$ (on the right) obtained by all eight metaheuristics for problem class 2.

a significant difference exists between at least two samples. It does not pronounce in terms of individual differences between pairs of samples. Accordingly, an appropriate post-hoc multiple comparisons procedure has to be
employed for the purpose of isolating the individual difference(s). It does so by performing (two-tailed) pairwise significance tests between all pairs of samples, correcting for the multiple inferences it makes \cite{17, 18}. As mentioned in Section 6.2, we employ the Nemenyi procedure for this purpose and it performs \(8\) such pairwise tests for our eight metaheuristics.

Due to space limitations, and the multitude of results obtained in these statistical tests, the details of the single-problem analysis are not reproduced here. Instead, we only present a summary thereof. The Friedman test yielded a statistically significant difference (for \(\alpha = 0.05\)) for every problem instance within the test suite with respect to both indicators. In fact, the \(p\)-value was numerically zero in each test. This is not surprising, given the poor results that were obtained by the OMOPSO algorithm. In the post-hoc analyses, the Nemenyi procedure revealed that the OMOPSO algorithm is
significantly different from \(i.e.\) worse than\) every other metaheuristic across all class 1 problem instances with respect to both indicators. Furthermore, the OMOPSO, AMOSA and MOVNS algorithms were also significantly different from \(i.e.\) worse than\) every other metaheuristic in the majority of class 2 and class 3 problem instances, again with respect to both indicators. Several other significant differences were also revealed in the post-hoc analysis, but these are scattered throughout the results which make meaningful inferences difficult. As such, we do not mention them individually here.

Let us consider next a multi-problem analysis, \textit{within each class} of problem instances, for the eight metaheuristics. Therefore, we have an average indicator value (calculated over the fifty runs) per variant/problem instance pair. The average \(I_{\text{HVD}}\) and \(I_{\text{R}2}\) values correspond to the black diamond points in Figures 3–5.

Suppose we were to rank the metaheuristics according to these average indicator values, within each problem instance. Then, the average values would be replaced by integers (ranks) from 1 through to 8, where 1 corresponds to the best average value, and 8 to the worst. We may thus calculate an average rank \(R_{\text{avg}}\) for every metaheuristic over each class of problem instances. These average ranks may then be used to compare how well the metaheuristics perform in a multi-problem context. Interestingly, this intuitive approach of comparison corresponds in part to the Friedman test. The average ranks described above were employed within the Friedman test as well as the Nemenyi post-hoc procedure. We return to these ranks later.

We applied the Friedman test to the average indicator value samples for each class of problems. The resulting \(p\)-values are presented in Table 5 and, as before, a bold-faced entry denotes a significant difference (for \(\alpha = 0.05\)). Although the sample size was small, the Friedman test detected a significant difference between the metaheuristics in every problem class, for both indicators.

Table 5: Multi-problem analysis for comparing metaheuristics. The table contains the \(p\)-values obtained by Friedman tests applied to the average indicator value samples for each problem class. Bold-faced entries represent a statistically significant difference (for \(\alpha = 0.05\)).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Friedman test (p)-values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
</tr>
<tr>
<td>(I_{\text{HVD}})</td>
<td>8.1218E-04</td>
</tr>
<tr>
<td>(I_{\text{R}2})</td>
<td>0.0028</td>
</tr>
</tbody>
</table>
The outcomes of the Nemenyi post-hoc analyses are presented in tabular form, along with the average ranks mentioned above so as to assist in the interpretation. Within each class of problem instances, we sorted the metaheuristics according to their average ranks (for both $I_{HVD}$ and $I_{R_2}$). The sorted results for classes 1, 2 and 3 are presented in Tables 6, 7 and 8, respectively. Whenever the post-hoc analysis detected a significant difference between two metaheuristics, we denoted this with matching alphabetic letters in the column labelled “S.Diff”.

Table 6: Multi-problem analysis results for class 1. Metaheuristics are sorted according to their average ranks. Significant differences between any two metaheuristics are denoted by matching alphabetic letters in the column labelled “S.Diff”.

<table>
<thead>
<tr>
<th>$R_{avg}$</th>
<th>$I_{HVD}$ results</th>
<th>Metaheuristic</th>
<th>S.Diff</th>
<th>$R_{avg}$</th>
<th>$I_{R_2}$ results</th>
<th>Metaheuristic</th>
<th>S.Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>MOVNS</td>
<td>a</td>
<td></td>
<td>2.167</td>
<td>P-ACO</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>MOOCEM</td>
<td>b</td>
<td></td>
<td>3.333</td>
<td>MOVNS</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>3.333</td>
<td>NSGA-II</td>
<td>c</td>
<td></td>
<td>3.667</td>
<td>NSGA-II</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>3.833</td>
<td>SPEA2</td>
<td>-</td>
<td></td>
<td>3.667</td>
<td>MOOCEM</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>P-ACO</td>
<td>-</td>
<td></td>
<td>4.667</td>
<td>SPEA2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>MOHS</td>
<td>-</td>
<td></td>
<td>5</td>
<td>AMOSA</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>5.833</td>
<td>AMOSA</td>
<td>-</td>
<td></td>
<td>5.5</td>
<td>MOHS</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>OMOPSO</td>
<td>abc</td>
<td></td>
<td>8</td>
<td>OMOPSO</td>
<td>abcd</td>
<td></td>
</tr>
</tbody>
</table>

We observe in Table 6 that, for problem class 1, the OMOPSO algorithm is significantly different from the MOVNS algorithm, the MOOCEM and the NSGA-II with respect to both indicators, and also from the P-ACO algorithm with respect to $I_{R_2}$. Based on the average ranks, the OMOPSO algorithm is clearly the worst-performing metaheuristic in class 1, which is consistent with the finding of the single-problem analysis. The MOVNS algorithm and the MOOCEM are jointly the best-performing metaheuristics with respect to $I_{HVD}$, whereas the P-ACO algorithm is the best with respect to $I_{R_2}$ in class 1. It is worth noting that, for the P-ACO algorithm, the rankings are quite different for the two indicators. This demonstrates that the performance indicator may influence the choice of metaheuristic to employ. On the contrary, the MOVNS algorithm and the NSGA-II seem to yield consistently good results over both indicators in class 1.

According to the rankings in Table 7, the MOVNS, AMOSA and OMOPSO algorithms are clearly the worst-performing metaheuristics for class 2, which is again consistent with the finding of the single-problem analysis. The OMOPSO algorithm is significantly different from the MOOCEM, the NSGA-II and the SPEA2 with respect to both indicators, and also from the P-ACO
Table 7: Multi-problem analysis results for class 2. Metaheuristics are sorted according to their average ranks. Significant differences between any two metaheuristics are denoted by matching alphabetic letters in the column labelled “S.Diff”.

<table>
<thead>
<tr>
<th></th>
<th>$R_{avg}$</th>
<th>$I_{HVD}$ results</th>
<th>S.Diff</th>
<th></th>
<th>$R_{avg}$</th>
<th>$I_{R2}$ results</th>
<th>S.Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>MOOCED</td>
<td>a</td>
<td>1.667</td>
<td>NSGA-II</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.833</td>
<td>NSGA-II</td>
<td>b</td>
<td>2.333</td>
<td>P-ACO</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SPEA2</td>
<td>c</td>
<td>3.167</td>
<td>MOOCED</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.833</td>
<td>P-ACO</td>
<td>—</td>
<td>3.333</td>
<td>SPEA2</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.833</td>
<td>MOHS</td>
<td>—</td>
<td>4.5</td>
<td>MOHS</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MOVNS</td>
<td>—</td>
<td>6.167</td>
<td>MOVNS</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.167</td>
<td>AMOSA</td>
<td>—</td>
<td>7</td>
<td>AMOSA</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.833</td>
<td>OMOPSO</td>
<td>abc</td>
<td>7.833</td>
<td>OMOPSO</td>
<td>abcd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The P-ACO algorithm, again, performs well with respect to $I_{R2}$ and less so with respect to $I_{HVD}$. The MOOCED and the NSGA-II are the two best-performing metaheuristics over class 2 for $I_{HVD}$ and $I_{R2}$, respectively.

Table 8: Multi-problem analysis results for class 3. Metaheuristics are sorted according to their average ranks. Significant differences between any two metaheuristics are denoted by matching alphabetic letters in the column labelled “S.Diff”.

<table>
<thead>
<tr>
<th></th>
<th>$R_{avg}$</th>
<th>$I_{HVD}$ results</th>
<th>S.Diff</th>
<th></th>
<th>$R_{avg}$</th>
<th>$I_{R2}$ results</th>
<th>S.Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>P-ACO</td>
<td>—</td>
<td>1</td>
<td>P-ACO</td>
<td>abc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MOOCED</td>
<td>—</td>
<td>2</td>
<td>NSGA-II</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.25</td>
<td>MOHS</td>
<td>—</td>
<td>3.75</td>
<td>MOOCED</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>SPEA2</td>
<td>—</td>
<td>4.25</td>
<td>MOHS</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.75</td>
<td>NSGA-II</td>
<td>—</td>
<td>4.5</td>
<td>SPEA2</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>MOVNS</td>
<td>—</td>
<td>6.25</td>
<td>MOVNS</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>OMOPSO</td>
<td>—</td>
<td>7</td>
<td>AMOSA</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>AMOSA</td>
<td>—</td>
<td>7.25</td>
<td>OMOPSO</td>
<td>ad</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Table 8, we observe that there are no significant differences between any two metaheuristics with respect to $I_{HVD}$, even though the Friedman test yielded a significant $p$-value. This may be attributed to the Nemenyi procedure’s conservatism and the fact that we have so many metaheuristics to compare. However, with respect to $I_{R2}$, the P-ACO algorithm is significantly different from the MOVNS, AMOSA and OMOPSO algorithms. In addition, the NSGA-II is significantly different from the OMOPSO algorithm. Based on the average rankings in Table 8, the MOVNS, AMOSA
and OMOPSO algorithms are also the worst-performing metaheuristics for class 3, which is consistent with the finding of the single-problem analysis. The P-ACO algorithm is the best-performing metaheuristic over class 3 with respect to both indicators. In contrast this time, for the NSGA-II, the rankings are quite different between the two indicators. Also of note is the MOHS algorithm, which performs better on class 3 problems (having four objectives) than on class 1 and 2 problems (less than four objectives).

In summary, the single-problem and multi-problem analyses suggest that the OMOPSO algorithm may be discarded from any future research in the context of constrained MICFMO. Similarly, the two local search metaheuristics, the AMOSA algorithm and the MOVNS algorithm, may also be discarded for problems with three or more objectives. The MOVNS algorithm, however, performs excellently when there are only two objectives. The SPEA2 and the MOHS algorithm may likely also be discarded due to their average performance in the majority of instances, even though the MOHS algorithm seems to be the most robust metaheuristic in terms of variability. The NSGA-II, the P-ACO algorithm and the MOOCEM are generally the best-performing metaheuristics with respect to both indicators, across all three problem classes. As such, they may form an integral part of any future research, along with the bi-objective MOVNS algorithm.

8. Conclusions

In this paper, eight modern and state-of-the-art multiobjective metaheuristics were compared in respect of their performance in the context of sixteen constrained MICFMO problem instances based on the SAFARI-1 nuclear research reactor. The effectiveness of a multiplicative penalty function constraint handling technique for multiobjective optimisation was also investigated and compared to the constrained-domination technique from the literature. To the best of our knowledge, this was the first comparative study of its kind in the context of constrained MICFMO. Furthermore, six of the metaheuristics were applied to MICFMO problem instances for the first time. The sixteen test problem instances varied in the number of objectives (two, three and four) so as to investigate the scalability of the metaheuristics. Nonparametric statistical procedures, in terms of both single-problem and multi-problem analyses, were applied in the comparative study. Two indicators were adopted as performance measures in the study, namely the hypervolume difference to reference set, and the $R^2$ indicator.

It was found that the new MPF constraint handling technique for multiobjective optimisation is a competitive alternative to the CDP technique.
from the literature. In the majority of test instances, there was no statistically significant difference between the two techniques, with respect to both performance indicators. In those instances where a significant difference was detected, it was mostly the MPF technique which outperformed the CDP technique. The majority of these significant instances occurred for the bi-objective problems, and in particular when using the AMOSA algorithm.

In terms of the metaheuristic solution comparison, there was no specific metaheuristic which consistently outperformed the remaining ones across the MICFMO test problems. Conversely, the OMOPSO algorithm was consistently outperformed by the other metaheuristics in all the class 1 problems. Similarly, the OMOPSO, AMOSA and MOVNS algorithms were consistently outperformed by the remaining metaheuristics in the majority of class 2 and class 3 problems. This was the case for both performance indicators. The SPEA2 and the MOHS algorithm typically yielded only average-quality results across the test problems. It was, however, found that the NSGA-II, the P-ACO algorithm and the MOOCEM were generally the best-performing metaheuristics with respect to both indicators, across all three problem classes. Furthermore, the MOVNS algorithm performed well when solving class 1 problem instances.

Recall that the aim of this paper was to provide a baseline and guide for future developments in terms of multiobjective optimisation for solving constrained MICFMO problems. In this regard, we suggest that the NSGA-II, the P-ACO algorithm and the MOOCEM, along with the MOVNS algorithm for bi-objective problems, be considered in future research activities. These metaheuristics (and either of the constraint handling techniques) may prove useful in the development of hybrid algorithms and effective hyperheuristics for MICFMO.

**Acknowledgements**

The first author (E.B. Schlünz) was financially supported in part by Necsa via their study assistance scheme, and by the National Research Foundation (NRF) of South Africa (Grant 88003). The second and third authors (P.M. Bokov and J.H. van Vuuren) were financially supported by the NRF (Grant 96281 and Grant 70593, respectively). Any opinion, finding and conclusion or recommendation expressed in this material is that of the author(s) and the NRF does not accept any liability in this regard.
References


