Multiobjective in-core nuclear fuel management optimisation by means of a hyperheuristic

E.B. Schlünza,b,c,∗, P.M. Bokova, J.H. van Vuurenc

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
cStellenbosch Unit for Operations Research in Engineering, Department of Industrial Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa

Abstract

This paper is concerned with the problem of constrained multiobjective in-core fuel management optimisation (MICFMO) using, for the first time, a hyperheuristic technique as solution approach. A multiobjective hyperheuristic called the AMALGAM method (an evolutionary-based technique incorporating multiple sub-algorithms simultaneously) is compared to three previously-studied metaheuristics, namely the nondominated sorting genetic algorithm II, the Pareto ant colony optimisation algorithm and the multiobjective optimisation using cross-entropy method, in an attempt to improve upon the level of generality at which MICFMO may be conducted. This solution approach was motivated by a lack of consistent performance by the aforementioned metaheuristics when applied in isolation. Comparisons are conducted in the context of a test suite of several problem instances based on the SAFARI-1 nuclear research reactor. Nonparametric statistical analyses in respect of the optimisation results reveal that the AMALGAM method significantly outperforms the three metaheuristics in the majority of problem instances within the test suite. The practical relevance of the hyperheuristic results is further demonstrated by comparing the solutions thus obtained to a reload configuration designed according to the current fuel assembly reload design approach followed at the SAFARI-1 reactor.

Keywords: Hyperheuristic, Multiobjective optimisation, In-core fuel management optimisation, Nuclear reactor

1. Introduction

The in-core fuel management optimisation (ICFMO) problem refers to the task of finding an optimal assignment of fuel assemblies to loading positions in a nuclear reactor core, subject to certain constraints. Such an assignment of assemblies is referred to as a fuel reload configuration (or a loading pattern). The ICFMO problem is well known in the field of nuclear engineering and has been a subject of research for many years [1–4]. Characteristics associated with this problem include its large combinatorial decision space, multiple conflicting, nonlinear objective functions and constraints that generally cannot be expressed in closed form, and computationally expensive function evaluations using a reactor core simulator [5, 6].

A number of solution techniques within the realms of mathematical programming, expert- or knowledge-based systems and metaheuristics, have been proposed for solving the ICFMO problem [2, 5]. Metaheuristics have, in particular, emerged as the most prominent solution techniques applicable to the problem. Examples thereof include simulated
annealing [4], genetic algorithms [7], particle swarm optimisation [8], ant colony optimisation [1] and tabu search [9]. Apart from solution techniques, research efforts have also been aimed toward reducing the computational cost associated with function evaluations in the ICFMO problem. In this regard, artificial neural network (ANN) surrogate models have been applied with good effect for predicting objective function and constraint function values, as opposed to calculating them explicitly using reactor core simulators [10, 11].

It has been pointed out that, very often, multiple objective functions have to be optimised simultaneously in instances of the ICFMO problem [5, 6]. The overwhelming majority of research concerning ICFMO has, however, been performed in the context of single-objective optimisation. In multiobjective optimisation, the aim is to identify a set of trade-off solutions to a problem instance, based on the fundamental notion of Pareto optimality. Accordingly, the multiobjective in-core fuel management optimisation (MICFMO) problem is the problem of finding a so-called Pareto optimal set of fuel reload configurations for a nuclear reactor core.

Until recently, only a few multiobjective optimisation algorithms (MOAs) had been applied to the MICFMO problem in the literature. Multiobjective genetic algorithm approaches were considered by Parks [3], Do and Nguyen [12], and Hedaya et al. [13], whereas multiobjective simulated annealing approaches were considered by Park et al. [14] and Engrand [15]. Improvements on Engrand’s simulated annealing algorithm were also proposed by Parks and Suppakitnarm [16] as well as by Kellar [17]. Finally, a multiobjective particle swarm optimisation algorithm was applied to the MICFMO problem by Babazadeh et al. [18], while Schlünz et al. [19] considered a multiobjective cross-entropy method. In each case, however, the aforementioned MOAs were employed in isolation — their efficacy and comparative performance in respect of solving the MICFMO problem therefore remained unknown. Then, in 2016, Schlünz et al. [20] conducted an investigation in which eight modern state-of-the-art MOAs were compared for solving several constrained MICFMO problem instances in a test suite based on the SAFARI-1 research reactor. In that study, it was found that the nondominated sorting genetic algorithm II (NSGA-II) [21], the Pareto ant colony optimisation (P-ACO) algorithm [22] and the multiobjective optimisation using the cross-entropy method (MOOCEM) [23] were generally the best-performing metaheuristics for solving constrained instances of the MICFMO problem. No single MOA was, however, able to consistently outperform the other metaheuristics with respect to all, or most of, the MICFMO problem instances in the test suite.

This lack of consistent MOA performance in the context of MICFMO therefore motivates further studies into the suitability of different solution techniques for the problem. Accordingly, this paper builds upon the work conducted by Schlünz et al. [20] by attempting to improve the level of generality at which constrained MICFMO may be performed. This is achieved by investigating, for the first time, the efficacy of a multiobjective hyperheuristic solution technique known as the AMALGAM method (which is an acronym for a multi-algorithm, genetically adaptive multiobjective) [24]. Hyperheuristics have been shown not only to raise the level of general applicability, but also to achieve improved quality and/or efficiency in optimisation results [25]. In the case of the AMALGAM method, which incorporates multiple metaheuristics simultaneously in an adaptive manner, testing of the method by Vrugt and Robinson [24] in respect of benchmark problem instances in the continuous domain revealed that improvements in computational efficiency of up to a factor of ten may be achieved over the individual sub-algorithm metaheuristics.

The same test suite of constrained MICFMO problem instances considered in [20] is employed in this paper, along with corresponding ANN surrogate models for faster function evaluations [26]. Using this test suite, the AMALGAM method is compared to
the NSGA-II, the P-ACO algorithm and the MOOCEM in a structured and statistically sound manner, as advocated in the literature [27, 28]. These three metaheuristics are also employed as sub-algorithms in the AMALGAM method. Furthermore, the nonparametric Friedman test and Nemenyi, Wilcoxon-Wilcox, Miller post hoc procedure, in particular, are employed within the statistical comparative analysis [29].

The paper is organised as follows. The constrained MICFMO problem, the test suite of problem instances and the corresponding ANN surrogate models are described in §2. A brief introduction to hyperheuristics is then presented in §3, along with a description of the AMALGAM method and its implementation for constrained MICFMO. Thereafter, the performance assessment and statistical testing procedures adopted in this study are discussed in §4. The comparative results obtained in respect of the test suite are then presented and analysed in §5. The practical relevance of the AMALGAM method is also demonstrated in §6 by comparing a subset of the results obtained with that rendered by the current SAFARI-1 reload configuration design approach. The paper finally closes in §7 with a brief set of conclusions.

2. The constrained MICFMO problem

The MICFMO problem may generally be regarded as a nonlinear assignment problem in which available fuel assemblies are to be assigned in an optimal manner to loading positions in a nuclear reactor core for a particular operational cycle of the reactor. Although the problem may be extended so as to include multiple operational cycles, only the single-cycle version thereof is considered in this paper. The optimisation model for the constrained MICFMO problem proposed in [20] is also adopted in this work. Accordingly, it is assumed that the number, $n$, of fuel assemblies (labelled $1,\ldots,n$) is equal to the number of loading positions in the core (also labelled $1,\ldots,n$). In the model, a reload configuration is represented by a permutation vector $x = [x_1,\ldots,x_n]$ where the decision variable $x_i = j$ denotes that fuel assembly $j \in \{1,\ldots,n\}$ is assigned to loading position $i \in \{1,\ldots,n\}$. Furthermore, $\mathcal{X}$ is the set of all possible reload configurations (i.e. permutation decision vectors). All objective functions are also assumed, without loss of generality, to require maximisation. The general model of the constrained MICFMO problem with $q$ objective functions $f_1(x), f_2(x),\ldots,f_q(x)$ is therefore formulated in [20] as

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

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

2.1. The test suite for constrained MICFMO

The same test suite of constrained MICFMO problem instances proposed by Schlünz et al. [20] is employed in this paper. These instances are based on the SAFARI-1 nuclear research reactor in South Africa, which is primarily utilised for commercial irradiation services, along with nuclear and materials research. The commercial services rendered by the reactor revolve around the production of radioisotopes (primarily molybdenum-99,
which is used for medical diagnostic purposes) and the neutron transmutation doping of silicon (to produce silicon semiconductors used in electronic equipment). Several neutron beam tubes surrounding the reactor core, are utilised for neutron scattering, radiography and diffraction experiments in respect of the nuclear and materials research.

The core layout of the SAFARI-1 reactor is presented in Figure 1. Currently, the reactor core consists of a $9 \times 8$ lattice which houses twenty-six fuel assemblies, six control rods, seven dedicated molybdenum production rig facilities, two general-purpose isotope production rig (IPR) facilities, as well as other core components which are not specified in detail. The locations of the neutron beam tubes and the silicon doping facility are also indicated in Figure 1.

The test suite of problem instances was constructed to include seven typical objectives associated with the MICFMO problem for the SAFARI-1 reactor in various combinations. A fixed set of constraints corresponding to the most prominent safety and utilisation requirements for the reactor was also imposed. As mentioned in the introduction, function evaluations for the ICFMO problem are typically performed by a reactor core simulator. Accordingly, objectives and constraints have to be translated into core parameters whose values may be returned by such a simulator. The objectives and constraints adopted in the test suite are listed in Tables 1(a) and 1(b), respectively, along with their translated core parameters. Note that the specific limiting values of the constraints are proprietary knowledge and, as such, are not divulged here but have, instead, been recorded within a confidential technical report kept at the South African Nuclear Energy Corporation [30]. The resulting test suite consists of three classes of multiobjective optimisation problem instances. Each class corresponds to a different number of objectives (between two and four) present in the problem instances. These instances are identified and presented in Tables 1(c)–1(e). The reader is referred to [20] for further details regarding the test suite.

2.2. ANNs for function evaluations

As was the case in [20], several ANN surrogate models, capable of predicting SAFARI-1 objective function and constraint values, are employed for function evaluations in this paper. These neural networks were constructed in previous work for the purpose of reducing
Table 1: An MICFMO test suite based on the SAFARI-1 reactor. Each problem instance is denoted by “P#.#” in which the first number corresponds to the class it belongs to, and the second number is an enumeration over the instances. A check mark “✓” represents the inclusion of an objective in a particular problem instance.

(a) The objectives forming part of the test suite

<table>
<thead>
<tr>
<th>Number</th>
<th>Goal</th>
<th>Objective</th>
<th>Core parameter evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Maximise</td>
<td>Fuel cycle length</td>
<td>Excess reactivity</td>
</tr>
<tr>
<td>2</td>
<td>Minimise</td>
<td>Power peaking factor</td>
<td>Power peaking factor</td>
</tr>
<tr>
<td>3</td>
<td>Maximise</td>
<td>Total molybdenum-99 production</td>
<td>Assembly-averaged power levels in all molybdenum rigs</td>
</tr>
<tr>
<td>4</td>
<td>Maximise</td>
<td>Utilisation of the silicon doping facility</td>
<td>Average thermal neutron flux over the facility</td>
</tr>
<tr>
<td>5</td>
<td>Maximise</td>
<td>Research capability at beam tubes 1 &amp; 2</td>
<td>Average thermal neutron flux over the beam tube faces</td>
</tr>
<tr>
<td>6</td>
<td>Maximise</td>
<td>Research capability at beam tube 5</td>
<td>Average thermal neutron flux over the beam tube face</td>
</tr>
<tr>
<td>7</td>
<td>Maximise</td>
<td>Isotope production in the IPR facilities</td>
<td>Maximum axial thermal neutron flux in both facilities</td>
</tr>
</tbody>
</table>

(b) The constraints imposed on all problem instances in the test suite

<table>
<thead>
<tr>
<th>Number</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The total molybdenum-99 production must be greater than the demand (i.e. assembly-averaged power levels in all molybdenum rigs)</td>
</tr>
<tr>
<td>2</td>
<td>The molybdenum-99 yield for each rig must be above a minimum limit (i.e. assembly-averaged power levels in each molybdenum rig)</td>
</tr>
<tr>
<td>3</td>
<td>The production capability in each IPR must be above a specified threshold (i.e. maximum axial thermal neutron flux in each IPR)</td>
</tr>
<tr>
<td>4</td>
<td>The core power peaking factor must be below the safety limit</td>
</tr>
<tr>
<td>5</td>
<td>The total control bank worth must be above the safety limit</td>
</tr>
<tr>
<td>6</td>
<td>The shutdown margin must be above the safety limit</td>
</tr>
</tbody>
</table>

(c) Class 1 bi-objective test problem instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective numbers</th>
</tr>
</thead>
<tbody>
<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>

(d) Class 2 tri-objective test problem instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective numbers</th>
</tr>
</thead>
<tbody>
<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>

(e) Class 3 tetra-objective test problem instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Objective numbers</th>
</tr>
</thead>
<tbody>
<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>

the computation time required to perform MOA comparative studies involving the aforementioned constrained MICFMO test suite [26]. The OSCAR-4 reactor core simulator, which is used to support the operation of the SAFARI-1 reactor [31], requires approxi-
mately four minutes of computation time on an Intel® Core™ i5-2500 CPU with 4 GB RAM operating at 3.30 GHz within a 32-bit operating system to evaluate a single reload configuration. A brief overview of the three-dimensional simulation model of the SAFARI-1 reactor employed in the OSCAR-4 simulator may be found in [32].

Usage of the ANNs reduces the aforementioned computation time by four orders of magnitude, while yielding average absolute relative prediction errors of less than 2% on test data sets when compared to the OSCAR-4 values. The reader is referred to [26] for details on these ANN surrogate models.

3. Hyperheuristic optimisation techniques

As is the case for metaheuristics, there is no universally accepted definition of what a hyperheuristic is. The general idea, which dates back to the early 1960s, however, is that a hyperheuristic approach should in some way automate the design of a search methodology so as to be more generally applicable than existing algorithms [25]. As part of their classification of hyperheuristic techniques, Burke et al. [33] recently proposed the definition that “a hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems.” As such, hyperheuristics operate on a space of heuristics, attempting to find a good solution method achieving general applicability, rather than attempting to solve an optimisation problem instance directly [25].

A hyperheuristic has traditionally been characterised by a domain/problem layer and a hyperheuristic layer, separated by a so-called domain barrier [34]. The domain layer typically consists of a low-level heuristic repository, candidate solutions and their corresponding representations/encodings, the objective functions and constraints associated with a particular problem instance, together with any other problem-specific information. The idea behind the domain barrier is then to “shield” the hyperheuristic layer from the aforementioned information so as to maintain a certain level of generality for the hyperheuristic. It has, however, been argued that this strict notion of a domain barrier may be relaxed without seriously compromising the effectiveness and generality of a hyperheuristic [34, 35]. Typically, then, the hyperheuristic layer consists of a high-level strategy that guides the low-level heuristics using limited problem-independent information available to it from the domain barrier, e.g. the objective function value(s) associated with a given solution, the low-level heuristics, logical flags indicating whether a solution is new or not, whether the search has stagnated or not, and so forth [33].

Burke et al. [33] proposed a classification structure in which hyperheuristics are partitioned primarily into two general categories, namely heuristic selection methodologies (in which a higher level strategy selects existing low-level heuristics) and heuristic generation methodologies (in which a higher level strategy generates new low-level heuristics from existing heuristic components). Depending on the manner in which the low-level heuristics create new candidate solutions, a secondary layer of classification within these two hyperheuristic categories may also be considered. Low-level heuristics that create a candidate solution incrementally, starting from an empty solution, are referred to as construction heuristics. Similarly, low-level heuristics that commence with a complete candidate solution and then iteratively modify that solution, are referred to as perturbation heuristics. Finally, if a hyperheuristic technique employs some feedback mechanism during the search process, it is considered to be a learning algorithm, as opposed to a no-learning algorithm which does not incorporate feedback. The so-called source of information feedback may then impose an additional dimension to the classification of hyperheuristics in which a distinction is made between online and offline learning [33]. Online learning, as the term
suggests, occurs while the hyperheuristic is solving an optimisation problem instance. During offline learning, however, the hyperheuristic solves a set of training problem instances in order to gather knowledge (e.g. in the form of rules) that may hopefully benefit the hyperheuristic when it is used to solve unseen problem instances [33]. The aforementioned classification structure is illustrated in Figure 2.

![Figure 2: The hyperheuristic classification structure proposed by Burke et al. [33] which takes the nature of the space of heuristics into account, as well as the source of information feedback.](image-url)

An authoritative survey in respect of the state of the art in hyperheuristic development and application was recently published by Burke et al. [25]. Although the majority of hyperheuristics discussed in that paper are concerned with single-objective optimisation, it should be noted that a number of multiobjective hyperheuristic techniques have also recently been developed. Maashi et al. [36], for example, designed an online learning selection hyperheuristic based on a so-called choice function, and applied it to the popular Walking Fish Group multiobjective test suite [37], as well as a real-world problem involving vehicle crashworthiness design. Charan Kumari and Srinivas [38], on the other hand, applied the multi-objective hyper-heuristic evolutionary algorithm (MHypEA) to a software module clustering problem. Other examples of multiobjective hyperheuristics may be found in [39, 40], as well as a many-objective hyperheuristic example in [41]. The multiobjective hyperheuristic adopted in this paper is described in the next section.

### 3.1. The AMALGAM method as a multiobjective hyperheuristic

Proposed by Vrugt and Robinson [24] in 2007, the AMALGAM method is an evolutionary-based multiobjective hyperheuristic technique in which the concepts of simultaneous multi-algorithm search and self-adaptive offspring creation are combined in order to solve multiobjective optimisation problem instances. The method employs $k$ sub-algorithms (i.e. low-level heuristics) simultaneously, with each sub-algorithm adaptively creating a number of offspring solutions proportional to its so-called reproductive success during the previous generation. The notion of global information sharing is also important in the method, with each sub-algorithm having access to the entire population in order to create its offspring. According to the classification structure proposed by Burke et al. [33], the AMALGAM method may be viewed as an online learning selection hyperheuristic.

The AMALGAM method borrows largely from the NSGA-II developed by Deb et al. [21]. The same two attributes, namely a Pareto rank and a crowding distance, constitute the fitness of any solution within a population in both algorithms. A nondominated sorting procedure, which partitions a population into different nondominated fronts, may be employed to determine the Pareto rank of each solution. This is achieved by means of the fast nondominated sorting algorithm (FNSA) proposed in [21]. Finally, the same elitist selection procedure adopted in the NSGA-II is also employed in the AMALGAM method.
in order to obtain a new parent population of size $N$ for the next generation. According to this procedure, the parent and offspring populations ($P_t$ and $Q_t$, respectively) are combined to form a new population $R_t$ during iteration $t$. Thereafter, the members of $R_t$ are ranked and sorted into nondominated fronts $F_1, F_2, \ldots$ using the FNSA. A population $P_{t+1}$ is created by including all solutions from the first front $F_1$, then all solutions from the second front $F_2$, and so forth, until inclusion of all solutions from the next front would yield a population size greater than $N$. The solutions in that next front are then sorted in decreasing order of crowding distance value, after which they are included one-by-one in $P_{t+1}$ until the population size equals $N$. This procedure is illustrated in Figure 3.

![Figure 3: The elitist selection procedure employed in the NSGA-II and the AMALGAM method in order to determine the next population (adapted from [21]).](image)

The reproductive success of a sub-algorithm which, as mentioned earlier, guides the self-adaptive offspring creation mechanism in the AMALGAM method, is determined as follows. Let $N^i_t$ be the number of offspring solutions that sub-algorithm $i \in \{1, \ldots, k\}$ created during generation $t$. Of those offspring solutions, let $S^i_{t+1}$ be the number of solutions included in the next parent population $P_{t+1}$ during the elitist selection procedure described above. These solutions are hereafter referred to as successful solutions. Then, the reproductive success of sub-algorithm $i$ is the ratio of the number of included solutions to the total number of offspring solutions it created, that is $S^i_{t+1}/N^i_t$. In order to reward the sub-algorithms according to their reproductive success, the number of offspring solutions that sub-algorithm $i$ should create during generation $t + 1$ is calculated as approximately

$$N^i_{t+1} = N \frac{(S^i_{t+1}/N^i_t)}{\sum_{i=1}^{k} (S^i_{t+1}/N^i_t)},$$

where $N$ again denotes the population size. Furthermore, it is recommended that a minimum value for $N^i_{t+1}$ be enforced in order to avoid de-activating any sub-algorithm [24].

The step-by-step procedure of the AMALGAM method is then as follows. The method is initialised with a randomly generated parent population $P_0$ of size $N$, which is then ranked and sorted according to the FNSA. An offspring population $Q_0$, also of size $N$, is generated next. Each sub-algorithm $i \in \{1, \ldots, k\}$ creates $N^i_0 = N/k$ offspring solutions, while having access to the entire parent population. The generation counter $t$ is then set to zero, and the following steps are iterated until some stopping criterion, such as a maximum number of generations reached, is met:
1. Combine the parent and offspring populations to create a new population $R_t \leftarrow P_t \cup Q_t$.
2. Rank and sort the solutions in $R_t$ into nondominated fronts $F_1, F_2, \ldots$ by means of the FNSA.
3. Determine the next population $P_{t+1}$ by following the elitist selection procedure described in Figure 3. This step may require the calculation of a crowding distance for each solution within a certain nondominated front.
4. Count the number of successful solutions $S^i_{t+1}$ created by each sub-algorithm $i$. Determine the new number $N^i_{t+1}$ of offspring solutions that each sub-algorithm should create according to the expression in (2).
5. Using each sub-algorithm $i$, generate $N^i_{t+1}$ new offspring solutions in order to create the next offspring population $Q_{t+1}$ of size $N$.
6. Increment the value of the generation counter $t \leftarrow t + 1$.

Upon termination of the method, the final population of solutions is returned as an approximately Pareto optimal set.

In the original implementation of the AMALGAM method by Vrugt and Robinson [24], four sub-algorithms were employed, namely the NSGA-II, a particle swarm optimisation algorithm, an adaptive Metropolis search algorithm, and differential evolution. The sub-algorithm selection was motivated by the outcome of numerical experiments conducted by the authors. Furthermore, the initial parent population of solutions was generated using Latin hypercube sampling [42], while a minimum value for $N^i_{t+1}$ was enforced at 5. Finally, no constraint handling technique was implemented in the method because the authors used it to solve unconstrained benchmark problem instances.

### 3.2. The AMALGAM method for constrained MICFMO

Given that the NSGA-II, the P-ACO algorithm and the MOOCEM were found to be the best-performing metaheuristics for solving the constrained MICFMO problem in [20], it was decided that these three MOAs, employing a permutation-based solution encoding, should constitute the sub-algorithms in the AMALGAM method investigated in this paper. Accordingly, the method employs both construction and perturbation heuristics according to the hyperheuristic classification structure proposed by Burke et al. [33]. Furthermore, a multiplicative penalty function constraint handling technique, considered in [20], is employed here in each sub-algorithm to allow for constrained multiobjective optimisation.

Although the NSGA-II may be employed within the AMALGAM method as is, specific decisions are required as to the manner in which the remaining two sub-algorithms are to be incorporated into the framework. The P-ACO algorithm and the MOOCEM require additional information other than the current population (in which the fitness of a solution is determined by its Pareto rank and crowding distance) in order to generate new solutions. In the description below concerning these specific decisions, the terminology employed corresponds to that adopted within the source material [20, 22, 23].

Consider first the P-ACO algorithm, whose single heuristic information matrix and $q$ different pheromone matrices (one for each objective) are now introduced into the AMALGAM method. Since a fixed heuristic information matrix, denoted by $\eta$, is utilised in the P-ACO algorithm, the same matrix adopted in [20] is implemented directly in the AMALGAM method. The pheromone matrices $\tau^\ell = [\tau^\ell_{ij}]$, on the other hand, have to be updated continually based on the solutions created by the method for $\ell = 1, \ldots, q$. As such, the manner in which the updating is performed has to be decided upon.

In order to facilitate the concept of global information sharing, a so-called global pheromone update is applied using the entire offspring population generated by all the
different sub-algorithms in the method. As stated in [22], only the best and second-best solutions obtained in the population (in respect of each objective \( \ell \)) are utilised during this update. Let \( x^\ell_b \) and \( x^\ell_{sb} \) denote these best and second-best solutions, respectively, and let \( \varrho \) denote the evaporation rate. The global pheromone update is then performed according to

\[
\tau^\ell_{ij} = (1 - \varrho)\tau^\ell_{ij} + \varrho \Delta \tau^\ell_{ij},
\]

(3)

where \( \Delta \tau^\ell_{ij} \) is related to \( x^\ell_b \) and \( x^\ell_{sb} \), and is calculated using the formula

\[
\Delta \tau^\ell_{ij} = \begin{cases} 
10, & \text{if } x_i = j \text{ in } x^\ell_b, \\
5, & \text{if } x_i = j \text{ in } x^\ell_{sb}, \\
0, & \text{otherwise}.
\end{cases}
\]

(4)

So-called *local pheromone updates* are, however, only performed once an offspring solution is created using the P-ACO sub-algorithm. Suppose that fuel assembly \( j \) has been assigned to loading position \( i \) in the offspring solution, i.e. \( x_i = j \). Each pheromone matrix is then (locally) updated according to

\[
\tau^\ell_{ij} = (1 - \varrho)\tau^\ell_{ij} + \varrho \tau_0,
\]

(5)

where \( \tau_0 \) is the initial pheromone level.

The MOOCEM is considered next, together with its elite set \( \mathcal{E} \) and corresponding probability matrix \( P = [p_{ij}] \) introduced here into the AMALGAM method. The original formulation of the AMALGAM method provides only for a primary population of solutions to be kept (as described in the previous section). In order to employ the MOOCEM as a sub-algorithm, however, it was decided that an elite set should be maintained separately as a secondary population. Also, the entire offspring population generated by all the different sub-algorithms is considered for possible inclusion in the elite set during each generation, before the probability matrix is updated (thus also facilitating global information sharing). Using the newly-determined elite set for a given generation, the entry \( p_{ij} \) in row \( i \) and column \( j \) of the probability matrix may then be updated according to

\[
p_{ij} = \frac{\sum_{\mathcal{E}} |\mathcal{X}_{ij}| I_{\{x \in \mathcal{X}_{ij}\}}}{|\mathcal{E}|},
\]

(6)

where \( |\mathcal{E}| \) denotes the cardinality of the elite set, where \( \mathcal{X}_{ij} \) is the set of all reload configurations \( x \in \mathcal{X} \) for which \( x_i = j \), and where \( I_{\{x \in \mathcal{X}_{ij}\}} \) is an indicator function defined as

\[
I_{\{x \in \mathcal{X}_{ij}\}} = \begin{cases} 
1 & \text{if } x \in \mathcal{X}_{ij}, \\
0 & \text{otherwise}.
\end{cases}
\]

Note that (6) is a modified version of the updating rule derived in [19].

The complexity of the AMALGAM method is increased by the introduction of a separate elite set for the MOOCEM sub-algorithm, and necessitates additional computer memory during the algorithmic execution. A different approach could have been to employ the current population as a proxy for the elite set. It is, however, envisaged that this complicating factor is an acceptable trade-off for improved generality and/or performance of the method.

A pseudo-code listing of the AMALGAM method for constrained MICFMO, as described above, is presented in Algorithm 1. This version of the method was implemented
in the Matlab software suite [43]. To the best knowledge of the authors, this work constitutes the first application of a multiobjective hyperheuristic to the MICFMO problem. It also represents the first AMALGAM variation which incorporates the P-ACO algorithm and the MOOCEM as sub-algorithms, along with the multiplicative penalty function constraint handling technique for multiobjective optimisation considered in [19, 20].

Algorithm 1: The AMALGAM method for constrained MICFMO using NSGA-II, P-ACO and MOOCEM as sub-algorithms.

Input : An MICFMO problem instance (which may or may not be constrained), a population size $N$, a maximum number of generations $t_{\text{max}}$ and tuning parameter values for each of the $k = 3$ sub-algorithms.

Output : An approximate Pareto set, $\tilde{P}_S$.

1. Initialise a pheromone matrix $\tau^\ell$ for each objective $\ell = 1, \ldots, q$ using $\tau_0$
2. Initialise the probability matrix $P$
3. Randomly generate an initial population $P_0$ of size $N$
4. Initialise the elite set $E \leftarrow \{ \text{nondominated solutions in } P_0 \}$
5. Set $N_i^0 \leftarrow N/k$
6. Use each sub-algorithm $i \in \{1, \ldots, k\}$ to generate $N_i^0$ new offspring solutions, and create offspring population $Q_0$ of size $N$
7. Apply the local pheromone update to $\tau^\ell$ for each objective $\ell = 1, \ldots, q$ according to (5)
8. $t \leftarrow 0$
9. while $t < t_{\text{max}}$ do
10. \hspace{1em} $R_t \leftarrow P_t \cup Q_t$
11. \hspace{1em} Rank and sort $R_t$ into nondominated fronts $F_1, F_2, \ldots$ using the FNSA in [21].
12. \hspace{1em} Construct $P_{t+1}$ by following the elitist selection procedure described in Figure 3.
13. \hspace{1em} Determine, for each $i \in \{1, \ldots, k\}$, the number of successful solutions, $S_i^{t+1}$, contributed by sub-algorithm $i$ to $P_{t+1}$
14. \hspace{1em} Calculate $N_i^{t+1}$ according to (2) for each $i \in \{1, \ldots, k\}$ and enforce $N_i^{t+1} = 4$ as a minimum value, if necessary
15. \hspace{1em} Apply the global pheromone update to $\tau^\ell$ for each objective $\ell = 1, \ldots, q$ using updating rules (3) and (4)
16. \hspace{1em} Update the elite set $E$ by considering the solutions in $Q_t$
17. \hspace{1em} Update the probability matrix $P$ according to (6)
18. \hspace{1em} Use each sub-algorithm $i \in \{1, \ldots, k\}$ to generate $N_i^{t+1}$ new offspring solutions, and create offspring population $Q_{t+1}$ of size $N$
19. \hspace{1em} Apply the local pheromone update to $\tau^\ell$ for each objective $\ell = 1, \ldots, q$ according to (5)
20. \hspace{1em} $t \leftarrow t + 1$
21. end
22. $\tilde{P}_S \leftarrow P_{t_{\text{max}}}$

Note that, for the NSGA-II sub-algorithm, the partially matched/mapped crossover operator [44] and scramble mutation operator [45] are employed for generating offspring solutions. The P-ACO sub-algorithm uses decision transition rules based on those of the well-known ant colony system [46], along with a random aggregation of the pheromone matrices. An ordered mapping is then employed during the generation of an offspring solution (i.e. an ant traversing a graph) to assign a fuel assembly (selected according to the transition rule) to a loading position [47]. Finally, an offspring solution is generated within the MOOCEM sub-algorithm by sampling a fuel assembly for each loading position independently according to the distribution formed by the relevant row of the probability matrix. After each sample, the distributions are re-normalised accordingly due to the permutation-based nature of a solution. Further details concerning the offspring generation by each sub-algorithm may be found in [19, 20, 22, 47].
Furthermore, in this version of the AMALGAM method, a minimum value for $N_{t+1}^i$ is enforced at 4 as follows. Once $N_{t+1}^i$ has been calculated according to (2), any fractions are rounded to the nearest integer. If any value is smaller than the minimum of 4, that value is increased up to the minimum. In order to offset such increases, a corresponding value is subtracted from the maximum value of $N_{t+1}^i$ attained so that $\sum_{i=1}^k N_{t+1}^i = N$.

Finally, the tuning parameter values within each sub-algorithm are set to the same values adopted in [20] for each individual MOA, while the population size is also selected as $N = 30$. Accordingly, for the NSGA-II, the crossover probability is taken as 0.9 and the mutation probability as $1/n$, where $n$ denotes the permutation vector length. Following the same notation in [20], the parameter values within the P-ACO algorithm are $\alpha = 1.75$, $\beta = 0.75$ and $q_0 = 0.8$, while the evaporation rate is selected as 0.2 and the initial pheromone level as 1. Similarly, the smoothing parameter and initial rank threshold value within the MOOCEM are set to 0.9 and 0, respectively. As explained by Schlüinz et al. [20], these values were determined following qualitative pilot studies in which parameter sensitivity analyses were conducted based on an experimental design approach.

4. Performance assessment procedure

As mentioned in §1, the aim of multiobjective optimisation is to identify the Pareto optimal set of solutions for a given problem instance. Since the MOAs considered in this paper are so-called approximate solution techniques, only a Pareto set approximation is yielded by each algorithm as its output (along with the corresponding nondominated front). No claim is, however, made in respect of closeness to optimality when referring to an approximation.

4.1. Performance indicators

In order to assess the performance of different MOAs, a comparison between the nondominated fronts returned by the algorithms has to be performed. Accordingly, a number of so-called performance indicators have been proposed in the literature that assign a scalar quantity to a nondominated front as a measure of its quality [48]. Two indicators that are often recommended are the unary hypervolume indicator [49] and the unary $R^2$ indicator [50], because they fulfill the desirable property of monotonicity (or Pareto compliance) [28, 48].

The hypervolume indicator essentially measures the portion of objective space enclosed by a given nondominated front and a specified reference point. Objective function values are typically scaled to the same order of magnitude, or to a (0, 1) range, before calculating the hypervolume indicator so as to avoid biasing it in favour of larger-valued objectives. The $R^2$ indicator, on the other hand, compares different nondominated fronts on the basis of a set of parameterised utility functions. A popular choice is to adopt the augmented weighted Chebyshev metric within the utility function.

A variation of the hypervolume indicator, called the hypervolume difference to reference set, was adopted in [20], along with the $R^2$ indicator. For the sake of consistency, these two indicators, denoted by $I_{HVD}$ and $I_{R^2}$, respectively, are also employed as performance measures in this paper. Note that smaller values of $I_{HVD}$ and $I_{R^2}$ are preferred.

4.2. Experimental design

A similar experimental design to the one followed in [20] is employed in this paper, thus allowing for a fair comparison of the optimisation results. As such, a stopping criterion of 1,050 function evaluations was imposed for the AMALGAM method. This corresponds
to the practical limitation encountered at the SAFARI-1 reactor should the OSCAR-4 reactor core simulator be used for optimisation. The reader is referred to [20, 51] for detailed discussions on this limitation.

The AMALGAM method, like the NSGA-II, the P-ACO algorithm and the MOOCEM, is also a stochastic algorithm and may therefore yield different nondominated fronts if it is applied multiple times to the same optimisation problem instance. In order to obtain a reasonably representative indication of its average performance, fifty optimisation runs of the AMALGAM method (employing different random number generator seeds) were executed for each problem instance in the constrained MICFMO test suite. Furthermore, the same fixed set of different random number generator seeds utilised by Schlüinz et al. [20] was employed in this study, along with the same fixed set of random initial solutions. The results obtained in this study using the AMALGAM method, and the existing results obtained by the three metaheuristics within each test problem instance therefore correspond to so-called matched samples. The selection of an appropriate statistical test for analysing the optimisation results is informed by this property of the samples.

As was the case in [20], the function values obtained for minimisation objective 2 were linearly transformed so as to correspond to values in a maximisation paradigm. Thereafter, the objective function values within all the nondominated fronts were scaled to the range (0, 1) using the maximum and minimum values attained for each objective. By doing so, all the values are of approximately the same magnitude.

Finally, reference nondominated fronts required for \( I_{HVD} \) and \( I_{R2} \) calculation in any problem instance were determined as follows. For each test problem instance, the nondominated fronts returned by the AMALGAM method, as well as those returned by the NSGA-II, the P-ACO algorithm and the MOOCEM in the study by Schlüinz et al. [20], were pooled together. A corresponding “overall” reference nondominated front was then determined by isolating the nondominated points from this pool. In addition, the reference and ideal points required for \( I_{HVD} \) and \( I_{R2} \) calculation were selected to be the same as those adopted in [20].

4.3. Statistical analysis

As suggested in the literature [28, 52], a hypothesis testing approach from the field of inferential statistics, employing nonparametric procedures, is followed in this paper for analysing the MOA results. Such an approach may be employed since the results obtained by the different algorithms may be considered as samples from an underlying distribution. Furthermore, two types of analyses that are typically performed in algorithmic comparative studies, namely single-problem and multi-problem analyses, are conducted in this study [52]. In a single-problem analysis, different problem instances are considered independently from one another. The results obtained over the fifty optimisation runs by the MOAs in respect of a given problem instance are therefore compared. In a multi-problem analysis, on the other hand, different problem instances are considered together. In such an analysis, a single result per algorithm/problem instance pair is required (e.g. an average value over the fifty optimisation runs) and those results are then compared over multiple instances.

In this paper, the nonparametric Friedman test and the Nemenyi, Wilcoxon-Wilcoxon, Miller (NWWM) procedure are employed [29]. The Friedman test may be applied to a set of two or more matched samples and is able to detect whether a statistically significant difference exists between at least two of the samples. Once such a difference has been detected, an appropriate post hoc procedure has to be performed in order to isolate the individual differences between pairs of samples. Since the aim in this paper is to compare
the NSGA-II, the P-ACO algorithm and the MOOCEM only to the AMALGAM method, and not to one another, the NWWM post hoc procedure is appropriate. Three one-tailed significance tests are performed in the NWWM procedure — one for each metaheuristic/AMALGAM pair of samples.

The Friedman test was applied utilising the Statistics and Machine Learning Toolbox [53] within the Matlab software suite, whereas the NSM3 package [54] within the programming language R was utilised to apply the NWWM procedure. A significance level of $\alpha = 0.05$ was adopted for all the results reported in this paper.

5. Numerical results and discussion

The comparative study results obtained by following the experimental design discussed in the previous section are presented and analysed in this section. As mentioned above, both single-problem and multi-problem analyses are performed in the comparison between the AMALGAM method, and the NSGA-II, the P-ACO algorithm and the MOOCEM.

5.1. Single-problem analysis

Within the context of the single-problem analysis, there are four samples of data for each test problem instance in the test suite. These samples correspond to matched indicator values obtained from the fifty optimisation runs for the AMALGAM method and its three constituent sub-algorithm metaheuristics (whose values were obtained from the work reported in [20]). The results (or samples) for problem classes 1, 2 and 3 are presented in the form of box plots in Figures 4, 5 and 6, respectively. The average value of each sample is also included in the graphs as black diamond points. First, a visual exploratory data analysis in respect of each problem instance is performed for both the $I_{HVD}$ and $I_{R^2}$ indicators by comparing the box plots of the sub-algorithm metaheuristics with that of the AMALGAM method within each graph. The reader is reminded that smaller indicator values are preferred.

It may be observed in Figures 4–6 that the AMALGAM method performs well compared to the NSGA-II, the P-ACO algorithm and the MOOCEM with respect to both performance indicators and across the problem instances in all three classes of the test suite. The performance of the hyperheuristic therefore seems to scale well as the number of objectives in a problem instance increases. Also, the robustness of the AMALGAM method in terms of its sample variability appears to be similar to that of the competing metaheuristics.

As may be seen in Figure 4, it appears that problem instance P1.5 is the only one in which the AMALGAM method is outperformed (with respect to both performance indicators) by one of its sub-algorithm metaheuristics, namely the P-ACO algorithm. Although such behaviour, in which an individual low-level heuristic outperforms the hyperheuristic, may seem unexpected, it is not uncommon in the literature [36, 40, 55]. Upon further investigation, it was observed that both techniques yielded similar quality results (on average with respect to both indicators) during the first few generations when solving problem instance P1.5. Thereafter, the AMALGAM method yielded superior results, until the P-ACO algorithm surpassed it in quality after approximately twenty generations. The particular cause of the behaviour could not, however, be identified satisfactorily, nor could a definitive explanation be found in the literature.

It may also be observed in Figure 5 that the performance of the MOOCEM is of similar quality than that of the AMALGAM method in the context of problem instances in class 2 of the test suite, especially with respect to the indicator $I_{HVD}$. 
Figure 4: Box plots of the $I_{HVD}$ samples (on the left-hand side) and $I_{R2}$ samples (on the right-hand side) obtained by the AMALGAM method, the NSGA-II, the P-ACO algorithm and the MOOCEM for each problem instance in class 1 of the test suite. A "⊤"-mark in the top-left corner of a box plot indicates that the AMALGAM method performs significantly better than the corresponding sub-algorithm.
Figure 5: Box plots of the $I_{HVD}$ samples (on the left-hand side) and $I_{R2}$ samples (on the right-hand side) obtained by the AMALGAM method, the NSGA-II, the P-ACO algorithm and the MOOCEM for each problem instance in class 2 of the test suite. A "⊤"-mark in the top-left corner of a box plot indicates that the AMALGAM method performs significantly better than the corresponding sub-algorithm.
The next step in this single-problem analysis is to determine whether a statistically significant difference exists between the results returned by the AMALGAM method and those returned by the corresponding sub-algorithm metaheuristics. Accordingly, the Friedman test is applied to the samples for each problem instance. Recall from the discussion in Section 4.3 that the aim here is to compare each separate metaheuristic to the AMALGAM method, and not to one another. As such, if the Friedman test detects a statistically significant difference, the NWWM post hoc procedure is applied in order to identify for which metaheuristic the results differ from those of the AMALGAM method. Due to the one-tailed significance tests that are performed in the NWWM procedure, if a statistically significant difference is detected by the post hoc procedure, it may be inferred that the AMALGAM method performs significantly better than the associated metaheuristic.

Due to space limitations, the details of the statistical test results obtained in this single-problem analysis are not reproduced here. Instead, only a summary of the findings

Figure 6: Box plots of the $I_{HV}$ samples (on the left-hand side) and $I_{R2}$ samples (on the right-hand side) obtained by the AMALGAM method, the NSGA-II, the P-ACO algorithm and the MOOCEM for each problem instance in class 3 of the test suite. A "\^"-mark in the top-left corner of a box plot indicates that the AMALGAM method performs significantly better than the corresponding sub-algorithm.
is presented. The Friedman test detected a statistically significant difference (at $\alpha = 0.05$) for every problem instance in the test suite, with respect to both indicators. This outcome is not surprising, given the good results observed for the AMALGAM method during the visual exploratory analysis. Recall that there are sixteen problem instances in the constrained MICFMO test suite. During the post hoc analysis, the NWWM procedure revealed that the AMALGAM method performs significantly better than the NSGA-II in fourteen problem instances with respect to the indicator $I_{\text{HVD}}$, and in another fourteen instances with respect to the indicator $I_{\text{R2}}$. Similarly, the hyperheuristic significantly outperforms the P-ACO algorithm in thirteen of the problem instances with respect to the indicator $I_{\text{HVD}}$, and in another fourteen instances with respect to the indicator $I_{\text{R2}}$. Finally, the post hoc procedure also revealed that the AMALGAM method performs significantly better than the MOOCEM in eight of the problem instances with respect to the indicator $I_{\text{HVD}}$, and in another fourteen instances with respect to the indicator $I_{\text{R2}}$. The specific instances in which the AMALGAM method performs significantly better than a sub-algorithm are indicated in Figures 4–6 by means of a “⊤”-mark in the top-left corner of the corresponding box plot.

5.2. Multi-problem analysis

Next, a multi-problem analysis is performed within each class of problem instances in the test suite. Accordingly, an average indicator value (calculated over the fifty optimisation runs) is determined for each MOA/problem instance pair. These average indicator values for $I_{\text{HVD}}$ and $I_{\text{R2}}$ correspond to the black diamond points in Figures 4–6 and constitute the samples to be used in the statistical tests.

The Friedman test is applied to the average indicator value samples for each class of problem instances in order to determine whether a statistically significant difference exists between the results returned by the AMALGAM method and those of the corresponding sub-algorithm metaheuristics in this multi-problem analysis. The resulting $p$-values are presented in Table 2, with bold-faced entries corresponding to a detection of a statistically significant difference (for $\alpha = 0.05$).

Table 2: Multi-problem analysis results when comparing the AMALGAM method with the NSGA-II, the P-ACO algorithm and the MOOCEM. The table contains the $p$-values obtained by Friedman tests applied to the average indicator value samples for each problem instance class. Bold-faced entries represent a statistically significant difference (at $\alpha = 0.05$).

<table>
<thead>
<tr>
<th>Sample</th>
<th>Friedman test $p$-values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{\text{HVD}}$</td>
<td>0.0141</td>
</tr>
<tr>
<td>$I_{\text{R2}}$</td>
<td>0.0141</td>
</tr>
</tbody>
</table>

Although the sample sizes are small, the Friedman test detected a statistically significant difference for every problem instance class in the test suite, with respect to both performance indicators. The $p$-values obtained by the subsequent application of the NWWM post hoc procedure in respect of all three problem classes (i.e. from the pair wise one-tailed significance tests) are presented in Table 3.

It may be observed in Table 3 that the AMALGAM method performs significantly better than the NSGA-II in all three problem classes with respect to the indicator $I_{\text{HVD}}$, and also in classes 1 and 3 with respect to the indicator $I_{\text{R2}}$. Furthermore, the method significantly outperforms the P-ACO algorithm in problem class 2 with respect to both performance indicators, as well as in class 1 with respect to the indicator $I_{\text{HVD}}$. Finally, it
Table 3: Multi-problem analysis results when comparing the AMALGAM method with the NSGA-II, the P-ACO algorithm and the MOOCEM. The table contains the p-values obtained by the NWWM post hoc procedure in which three pairwise significance tests are performed for each problem instance class. Bold-faced entries represent a statistically significant difference (at $\alpha = 0.05$).

<table>
<thead>
<tr>
<th>Problem</th>
<th>NWWM procedure</th>
<th>p-values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NSGA-II</td>
</tr>
<tr>
<td>Class 1</td>
<td>$I_{HV}$</td>
<td>0.0085</td>
</tr>
<tr>
<td></td>
<td>$I_{R2}$</td>
<td>0.0114</td>
</tr>
<tr>
<td>Class 2</td>
<td>$I_{HV}$</td>
<td>0.0111</td>
</tr>
<tr>
<td></td>
<td>$I_{R2}$</td>
<td>0.0726</td>
</tr>
<tr>
<td>Class 3</td>
<td>$I_{HV}$</td>
<td>0.0003</td>
</tr>
<tr>
<td></td>
<td>$I_{R2}$</td>
<td>0.0484</td>
</tr>
</tbody>
</table>

is found that the AMALGAM method performs significantly better than the MOOCEM in all three problem classes with respect to the indicator $I_{R2}$.

5.3. Summary of findings

Recall that Schlünz et al. [20] found the NSGA-II, the P-ACO and the MOOCEM to be generally the best-performing MOAs with respect to both the $I_{HV}$ and $I_{R2}$ indicators across the constrained MICFMO problem instances in all three classes of the test suite. No single metaheuristic was, however, able to consistently outperform the other MOAs with respect to all, or most of, the problem instances. Following a rigorous comparative study, it has been shown in this paper that the AMALGAM method significantly outperforms the three aforementioned MOAs in the majority of instances within the test suite. This inference is supported by both the single-problem and multi-problem statistical analyses. The hyperheuristic therefore achieves the dual goal of raising the level of generality at which constrained MICFMO may be performed, and of yielding improved quality in optimisation results.

5.4. Comparing the proposed AMALGAM method to a randomised approach

It has been suggested in the hyperheuristic literature that a learning algorithm should also be compared to a randomised version thereof (i.e. a no-learning algorithm equivalent) in order to gauge the efficacy of the learning mechanism [56]. It was decided that, in the context of the proposed AMALGAM method, its randomised version would still incorporate the global information sharing aspects, but exclude the self-adaptive offspring creation based on reproductive success. As such, instead of calculating $N_{i+1}^{t}$ according to (2), random values no less than the minimum of 4 are sampled uniformly during each generation in the randomised AMALGAM version such that $\sum_{i=1}^{k} N_{i+1}^{t} = N$.

Following the same experimental design as in §4.2, a separate head-to-head comparative study between the proposed AMALGAM method and its randomised version was performed. For the statistical analysis, however, the nonparametric Wilcoxon signed rank test [29] is employed. As before, due to space limitations, only a summary of the findings are presented here.

During the single-problem analysis, statistical tests revealed that there is no significant difference between the proposed and randomised versions of the AMALGAM method in fifteen of the problem instances with respect to the indicator $I_{HV}$, as well as in fourteen instances with respect to the indicator $I_{R2}$. In the remaining three instances (P1.1 for both indicators, and P3.2 for the indicator $I_{R2}$), the proposed AMALGAM method significantly outperformed the randomised version. A similar outcome was also observed during the
multi-problem analysis — the Wilcoxon signed rank test detected no significant difference between the two hyperheuristic versions in all three problem classes, with respect to both performance indicators.

It may therefore be concluded that an online learning and a no-learning version of the proposed AMALGAM method are equally effective (in a statistical sense) at raising the level of generality at which constrained MICFMO may be performed, and at yielding improved quality in optimisation results. Accordingly, the global information sharing aspect of the hyperheuristic is the primary factor contributing to its success, as opposed to the learning mechanism, within the context of the test suite of MICFMO problem instances. Due to the absence of similar head-to-head comparisons in other published studies employing the AMALGAM method [24, 36, 57, 58], the extent to which this observation may hold true for the method, in general, remains unknown. Darian et al. [55] did, however, show that modifications to the expression in (2) may lead to improved results.

6. Practical usage of AMALGAM results

In the comparative study, the performance of the AMALGAM method was determined relative to that of the NSGA-II, the P-ACO and the MOOCEM due to the absence of true Pareto fronts. As a result, the study did not necessarily reveal whether the solutions (i.e. reload configurations) obtained by the AMALGAM method are of adequate quality for practical usage. In this section, the practical relevance of the hyperheuristic solutions are demonstrated by comparing them with a reload configuration designed according to the current manual approach followed at the SAFARI-1 reactor. This design approach is based on a combination of several years of operating experience at the reactor and the well-known highest-mass to lowest-flux (or out-in) heuristic [59]. A safe reload configuration for any given operational cycle is typically returned by the approach, and it also generally meets the utilisation requirements of the reactor, especially in terms of molybdenum-99 production.

The problem instances in class 1 of the test suite are selected for the demonstration in this section. Accordingly, the fifty nondominated fronts yielded by the AMALGAM method, for each of those problem instances, are used to determine the respective empirical attainment functions (EAFs). As discussed by López-Ibáñez et al. [60], the EAF is an estimator of “the probability of an algorithm finding at least a solution whose objective vector dominates or is equal to an arbitrary vector in the objective space in a single run.” Using the EAF, a so-called $k$%-attainment surface may then be determined which corresponds to a boundary separating the objective space attained (i.e. dominated) by $k$% of the multiple runs of a MOA.

The attainment surfaces for $k = \{0, 25, 50, 75, 100\}$ are illustrated in Figure 7 for each problem instance in class 1. These illustrations were generated utilising the “eaf” package [61] within the programming language $R$. Note that $k = 0$ corresponds to the best attainment surface and separates the objective space attained by at least one of the fifty AMALGAM runs from the objective space not attained by any of the runs. Similarly, $k = 50$ corresponds the median attainment surface and may be used to examine the objective function vectors attained by at least half of the fifty AMALGAM runs. Finally, in order to gauge the worst-case results obtained by the hyperheuristic, $k = 100$ corresponds to the worst attainment surface. All the values in Figure 7 have been scaled according to the percentage improvement in each objective function value over that of the SAFARI-1 design approach solution. As such, the coordinate $(0, 0)$ within each graph corresponds to the objective function vector attained by the SAFARI-1 solution. Two straight lines intersecting at this coordinate are included in each graph for ease of reference.
Figure 7: The attainment surfaces for the fifty nondominated fronts obtained by the AMALGAM method for each problem instance in class 1 of the test suite, scaled according to the percentage improvement in each objective function value over that of the SAFARI-1 design approach solution.
It may be observed in Figures 7(a) and 7(b) that the best attainment surfaces obtained by the AMALGAM method yield improvements of up to approximately 25% in the value of the fuel cycle length objective over that of the SAFARI-1 approach, at the cost of a 2% deterioration in relative power peaking factor and a 5.3% deterioration in the value of the total molybdenum-99 production objective for problem instances P1.1 and P1.2, respectively. Simultaneous improvements in the values of both objectives for problem instances P1.1 and P1.2 have also been found. The worst attainment surface in Figure 7(a), in particular, also shows that the AMALGAM method is expected to be capable of always finding a solution that dominates the SAFARI-1 approach in problem instance P1.1.

Problem instance P1.3 closely resembles the current operating scenario faced at the SAFARI-1 reactor. It is observed in Figure 7(c) that the SAFARI-1 approach solution performs well and essentially lies on the best attainment surface obtained by the AMALGAM method. This is not surprising, since years of experience (on which the SAFARI-1 approach is based) can never be discounted when real-world optimisation problems are considered. Still, an entire spread of solutions ranging from an improvement of approximately 19% in the value of the silicon doping utilisation objective (at the cost of a 6% deterioration in the value of the total molybdenum-99 production objective) to an improvement of 1% in total molybdenum-99 production (at the cost of a deterioration of approximately 13% in silicon doping utilisation) have been uncovered by the hyperheuristic. The closeness of the different attainment surfaces also indicates the robustness of the AMALGAM method for this problem instance.

In problem instances P1.4 and P1.5, which are illustrated in Figures 7(d) and 7(c), respectively, numerous solutions obtained by the AMALGAM method are able to achieve a simultaneous improvement in the values of both objectives over that of the SAFARI-1 approach. As before, the worst attainment surfaces indicate that the hyperheuristic is expected to always find a solution that dominates the SAFARI-1 approach in these two instances.

Finally, as was the case for problem instance P1.3, it is observed in Figure 7(f) that the SAFARI-1 approach solution in problem instance P1.6 lies, again, on the best attainment surface. An entire spread of solutions was also obtained using the AMALGAM method. Improvements of up to 19% in the value of the silicon doping utilisation objective have been achieved, at the cost of a 24% deterioration in the value of the beam tube 5 utilisation objective. Conversely, an improvement of up to 6.7% in beam tube 5 utilisation over that of the SAFARI-1 approach has been uncovered, at the cost of a 14.6% deterioration in silicon doping utilisation.

Although the demonstration in this section was restricted to class 1 problem instances, solutions of similar quality (i.e. of similar improvement/deterioration trade-offs) have also been obtained by the AMALGAM method for the problem instances in classes 2 and 3 of the test suite.

7. Conclusions

In this paper, a multiobjective hyperheuristic, known as the AMALGAM method, was compared to three previously-studied metaheuristics (the NSGA-II, the P-ACO algorithm and the MOOCEM) in the context of the constrained, single-cycle MICFMO problem. A test suite of sixteen problem instances based on the SAFARI-1 research reactor was utilised for the comparative study. Two multiobjective optimisation performance indicators were adopted in this paper, namely the hypervolume difference to reference set and $R^2$ indicators. Furthermore, comprehensive single-problem and multi-problem statistical analyses were conducted on the optimisation results using nonparametric procedures.
This work constitutes the first application of a multiobjective hyperheuristic to the MICFMO problem. It was found that the AMALGAM method significantly (in a statistical sense) outperforms the NSGA-II, the P-ACO algorithm and the MOOCEM in the majority of problem instances within the test suite. The hyperheuristic is able to raise the level of generality at which MICFMO may be performed, and it is capable of yielding improved quality in optimisation results (compared to the preferred metaheuristics). A separate, head-to-head comparison between the proposed AMALGAM method and a randomised version thereof also revealed that the success of the hyperheuristic, within the context of the MICFMO test suite, may be attributed to its global information sharing aspects. Further studies in respect of the efficacy of the self-adaptive learning mechanism within the AMALGAM method are therefore recommended.

A subset of the results returned by the hyperheuristic was also compared with a solution obtained using the current reload configuration design approach followed at the SAFARI-1 reactor. It was found that the AMALGAM method yields solutions of adequate quality for practical usage, but that decision makers at SAFARI-1 may be afforded improved flexibility in their choice of reload configurations based on the good spread of solutions obtained in the hyperheuristic attainment fronts.

As such, the AMALGAM method (whose sub-algorithms consist of the NSGA-II, the P-ACO algorithm and the MOOCEM) is recommended as a state-of-the-art technique for solving the constrained, single-cycle MICFMO problem.

Acknowledgements

The first author (E.B. Schlünz) was financially supported in part by Necsa, 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. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which led to an improvement in the quality of the paper.

References


URL http://www.mathworks.com/products/matlab/


URL https://cran.r-project.org/package=NSM3


URL https://cran.r-project.org/package=eaf