The efficiency of indicator-based local search for multi-objective combinatorial optimisation problems

Matthieu Basseur, Arnaud Liefooghe, Le Khoi, Edmund K. Burke

The efficiency of indicator-based local search for multi-objective combinatorial optimisation problems

M. Basseur and A. Liefooghe and K. Le and E. K. Burke*

Abstract

In the last few years, a significant number of multi-objective metaheuristics have been proposed in the literature in order to address real-world problems. Local search methods play a major role in many of these metaheuristics procedures. In this paper, we adapt a recent and popular indicator-based selection method proposed by Zitzler and Künzli in 2004, in order to define a population-based multi-objective local search. The proposed algorithm is designed in order to be easily adaptable, parameter independent and to have a high convergence rate. In order to evaluate the capacity of our algorithm to reach these goals, a large part of the paper is dedicated to experiments. Three combinatorial optimisation problems are tested: a flow shop problem, a ring star problem and a nurse scheduling problem. The experiments show that our algorithm can be applied with success to different types of multi-objective optimisation problems and that it outperforms some classical metaheuristics. Furthermore, the parameter sensitivity analysis enables us to provide some useful guidelines about how to set the parameters.

1 Introduction

The application of metaheuristics to multi-objective combinatorial optimisation problems is a popular research area. The growing interest in these methods originates from the mid-eighties, when the first Pareto evolutionary algorithms were proposed. These evolutionary algorithms, which use the Pareto dominance concept in their selection process are very successful: a huge number of variants are proposed in the literature. The Pareto-based approaches are an alternative to aggregation based methods, which

*M. Basseur is with the Automated, Scheduling, Optimisation and Planning (ASAP) research group, University of Nottingham, Jubilee Campus, UK and the Laboratoire d'Études et de Recherche en Informatique d'Angers (LERIA), University of Angers, France; e-mail: basseur@info.univ-angers.fr. K. Le and E. K. Burke are with the ASAP research group, University of Nottingham, UK; e-mail:{kxl,ekb}@cs.nott.ac.uk. A. Liefooghe is with the INRIA Dolphin research group, Laboratoire d'Informatique Fondamentale de Lille (LIFL), CNRS, Cité Scientifique, 59655 Villeneuve d'Ascq cedex, France; e-mail:Arnaud.Liefooghe@lifl.fr.

represent a simple way to transform a multi-objective problem into a single objective one.

In this paper, we will use a slightly different type of generic method, which can include Pareto dominance based algorithms as well as scalar methods. We employ the principles proposed by Zitzler and Künzli [41], in their IBEA (*Indicator-Based Evolutionary Algorithm*). As described by the authors, *"IBEA is based on quality indicators where a function I assigns each Pareto set approximation a real value reflecting its quality [45]: Then the optimisation goal becomes the identification of a Pareto set approximation that minimizes (or maximizes) I"*. As such, they say, *"I induces a total order of the set of approximation sets in the objective space, in contrast to the classical aggregation functions like weighted sum that operate on single solutions only and gives rise to a total order of the corresponding objective vectors"* (see [41]). In [23] and [41], different indicator-based multi-objective optimizers have been proposed. The main advantage of the indicator concept is that no additional diversity preservation mechanisms are required, since it can be tackled in the binary indicator at hand. Zitzler and Künzli [41] have demonstrated that indicator-specific search can yield results which are superior to popular algorithms such as SPEA2 and NSGA-II with respect to the indicator under consideration [43, 12]. Furthermore, since the principle is simple, it could be adapted to various types of problem. For instance, the indicator-based search has been proposed also in [15], and successfully adapted to optimisation with uncertainty [6].

This paper aims to propose a generic metaheuristic, which enables the optimiser to avoid some of the drawbacks of classical methods. The design of generic metaheuristics is not an easy task, since the proposed method needs to satisfy several objectives, as described below:

- The technique should be easily scalable to different optimisation problems: to reach this goal, the method has to be as simple as possible and avoid the exploitation of problem specificities.
- There should be low levels of parameter sensitivity: the proposed method should be defined by a small number of parameters. Moreover, the number of parameters which have a major influence on the results should be minimised.
- The methodology should be effective (as much as is possible) on different problem sizes and problem types.

In [4], we presented a generic metaheuristic which aims to satisfy these objectives. The methodology proposed in this paper is slightly different to the approaches usually found in the literature, which use aggregation of the objective functions, or the Pareto dominance relation, since the binary-indicator concept is employed. In this paper, we will describe the method presented in [4] with further details, and we also provide a complete analysis of its performance and its parameter sensitivity. The contributions of this paper include:

- A description of an Indicator-Based Multi-Objective Local Search method, which could be easily reused to address different problems. The multi-objective metaheuristic proposed in this paper has two main characteristics: (1) the employment of the *binary indicator* concept [41], which allows us to avoid some of the

drawbacks of using aggregation and Pareto dominance based methods; (2) the method proposed is a population-based local search, which differs from a significant body of the multi-objective metaheuristics literature, where evolutionary algorithms are principally represented.

- The application of the proposed method to different combinatorial optimisation problems: the goal is to show the scalability and the efficiency of the proposed method on different problems. Three different multi-objective problems are considered: a flow shop scheduling problem, a ring star problem and a nurse scheduling problem. These problems are really different in terms of type, size, constraints and number of objective functions.

- A parameter sensitivity analysis: the proposed method is defined by a small number of parameters. We provide an analysis that evaluates the parameters that have a great influence on the results, and propose guidelines to set these parameters for optimisers which are interested in the application of our method to a new multi-objective problem.

The paper is organised as follows. In section 2, some definitions are introduced in order to define multi-objective optimisation as well as the binary-indicator search principle. In section 3, the Indicator-Based Multi-Objective Local Search (IBMOLS) algorithm is described, and also its iterative version where the population initialisation is realised in different ways. In sections 4 to 6, we present three different multi-objective combinatorial problems, which are solved using the IBMOLS algorithm. Then some conclusions and perspectives are discussed in section 7.

2 Multi-objective binary quality indicators

Before introducing the concept of indicator-based optimisation, let us introduce some useful notations and definitions, partially taken from [41] and [6]. Let X denote the search space of the optimisation problem under consideration and let Z denote the corresponding objective space. Without loss of generality, we assume that $Z = \Re^n$ and that all n objectives are to be minimised. Each decision vector $x \in \mathbf{X}$ is assigned exactly one objective vector $z \in Z$ on the basis of a vector function $f : \mathbf{X} \rightarrow Z$ with $z = f(x)$. The mapping f defines the evaluation of a solution $x \in \mathbf{X}$, and often one is interested in those solutions that are Pareto optimal with respect to f. A Pareto optimal solution is defined as follows:

Definition 1 *$x \in \mathbf{X}$ is said to be Pareto optimal if and only if a solution $x_i \in \mathbf{X}$ which dominates x does not exist.*

Definition 2 *A decision vector x_1 is said to dominate another decision vector x_2 (written as $x_1 \succ x_2$), if $f_i(x_1) \leq f_i(x_2)$ for all $i \in \{1, \ldots, n\}$ and $f_j(x_1) < f_j(x_2)$ for at least one $j \in \{1, \ldots, n\}$.*

The relation $x_1 \succ x_2$ means that the solution x_1 is *preferable* to x_2. The main goal is to find a high quality approximation of the Pareto optimal set. What constitutes 'high

quality' very much depends on the decision maker and the optimisation scenario. As in [41], we here assume that the optimisation goal is given in terms of a binary quality indicator I.

A binary quality indicator [45] can be thought of as a *continuous extension* of Pareto dominance on sets of objective vectors. The value $I(\mathbf{A}, \mathbf{B})$ quantifies the difference in quality between two sets of objective vectors $\mathbf{A}$ and $\mathbf{B}$. Now, if $\mathbf{R}$ denotes the set of Pareto optimal solutions (or any other reference set), then the overall optimisation goal can be formulated as:

$$\text{argmin}_{\mathbf{A} \in \mathcal{M}(X)} \; I(\mathbf{A}, \mathbf{R}) \tag{1}$$

where $\mathcal{M}(X)$ is the space of *objective vector sets*. Since $\mathbf{R}$ is fixed, I actually represents a unary function that assigns, to each Pareto set approximation, a real number; the smaller the number, the more preferable is the approximation.

The indicator could be used to compare two single solutions, or a single solution against an entire population. With such a comparison, the indicator can be used to establish the selection process of evolutionary algorithms [41]. Indeed, the solution to delete (respectively select) from the population should be the one which has the worst (respectively best) indicator value according to the rest of the population. In other words, during the selection process, the goal is to delete the solutions with the smallest degradation of the overall quality of the population, in terms of the quality indicator being used.

In order to be considered as a natural extension of the Pareto dominance concept, the defined indicator has to be compliant with the Pareto dominance relation. As defined in [41], a binary indicator I has to verify the dominance preserving property (definition 3). Let us note that throughout the paper, we will write $I(x, \mathbf{P})$ instead of $I(\{x\}, \mathbf{P})$ when a set is reduced to a single solution x, to simplify our notations. Moreover, in order to avoid confusion with singleton solutions, sets of solutions are written in boldface.

Definition 3 *A binary indicator I is denoted as dominance preserving if:*
(1) for all $x_1, x_2 \in \mathbf{X}$, $x_1 \succ x_2 \Rightarrow I(x_1, x_2) < I(x_2, x_1)$, and
(2) for all $x_1, x_2, x_3 \in \mathbf{X}$, $x_1 \succ x_2 \Rightarrow I(x_3, x_1) \geq I(x_3, x_2)$.

In the following, we first define some possible binary indicators, then we present a detailed example. We first propose to use the two indicators presented in [41]: The (additive) epsilon indicator (I_ϵ - equation 2) and the hypervolume indicator (I_{HD} - equation 3).

$$I_\epsilon(x_1, x_2) = max_{i \in \{1,\ldots,n\}}(f_i(x_1) - f_i(x_2)) \tag{2}$$

$I_\epsilon(x_1, x_2)$ (where $x_1 \in \mathbf{X}$ and $x_2 \in \mathbf{X}$) represents the minimal translation (in objective space) on which to execute x_1 so that it dominates x_2 (see figure 1). Let us note that the translation could take negative values. We assume, throughout the paper, that all the objective functions are normalised.

$$I_{HD}(x_1, x_2) = \begin{cases} H(x_2) - H(x_1) & \text{if } x_2 \succ x_1 \text{ or } x_1 \succ x_2 \\ H(x_1 + x_2) - H(x_1) & otherwise \end{cases} \tag{3}$$

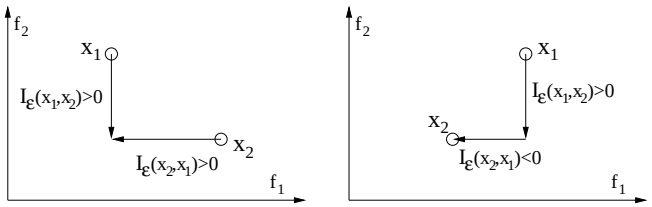

Figure 1: Illustration of the I_ϵ indicator applied to two solutions x_1 and x_2 (left hand side: no dominance relation between x_1 and x_2; right hand side: $x_2 \succ x_1$) ©[4].

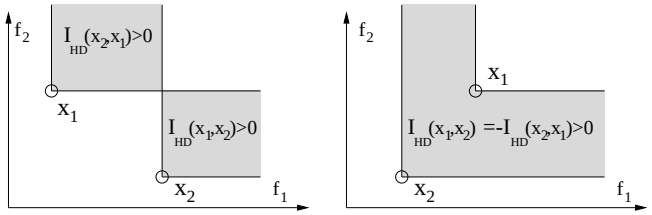

Figure 2: Illustration of the I_{HD} indicator applied to two solutions x_1 and x_2 (left hand side: no dominance relation between x_1 and x_2; right hand side: $x_2 \succ x_1$) ©[4].

$H(x_1)$ represents the volume of the space that is dominated by x_1. $I_{HD}(x_1, x_2)$ represents the volume of the space that is dominated by x_2 but not by x_1 (figure 2).

In order to evaluate the quality of solutions according to a whole population $\mathbf{P}$ and a binary indicator I, several different approaches could be defined as follows:

- One possibility is to simply sum up the indicator values for each population member with respect to the rest of the population (equation 4). The value obtained takes into account every solution in the population.

$$I(\mathbf{P} \setminus \{x\}, x) = \sum_{z \in \mathbf{P} \setminus \{x\}} I(z, x) \tag{4}$$

- One could be interested in finding the solution which obtains the minimal indicator value against x, i.e. the best solution according to x and I (equation 5). The computed value is not influenced by dominated solutions from the population.

$$I(\mathbf{P} \setminus \{x\}, x) = min_{z \in \mathbf{P} \setminus \{x\}}(I(z, x)) \tag{5}$$

- Lastly, we consider a trade off between these two approaches, which is an additive approach that amplifies the influence of dominating population members over dominated ones (equation 6, where $\kappa > 0$ represents the scaling factor). In our experiments, we will use this formulation for the I_ϵ and I_{HD} indicators.

$$I(\mathbf{P} \setminus \{x\}, x) = \sum_{z \in \mathbf{P} \setminus \{x\}} -e^{I(z,x)/\kappa} \tag{6}$$

Note that when $\kappa \to 0$, the same order relation between solutions is obtained with equations 5 and 6, with one difference: when two solutions have the same minimal indicator value with equation 5, equation 6 enables a decision between them according to the second minimal indicator value computed for these solutions. Then, values near to 0 are preferred for κ.

I_ϵ and I_{HD} are proved to be dominance preserving in [41]. Moreover, most of the classical Pareto ranking techniques could be used to define binary indicators, and use equation 5 or 4 to combine the indicator values (it is easy to show that these indicators also verify the dominance preserving relation). The expressions obtained are very simple and could be used as a comparison of the different available indicators. Some classical multi-objective ranking techniques from the literature are adapted into binary indicators below, without taking into account the diversity maintaining mechanisms of the corresponding algorithms. In the following, we propose to adapt three classical ranking methods into binary indicators.

First, the WAR ranking method proposed by Bentley and Wakefield [7] is similar to the binary indicator defined in equation 7. In this equation, we keep only the Pareto dominance relation part of the original method proposed by Bentley and Wakefield, since one goal of this paper is to evaluate the quality of I_ϵ and I_{HD} indicators according to Pareto dominance based indicators. By using the additive combination of the indicator values, we obtain equation 8, which corresponds approximatively to the ranking method of Bentley and Wakefield.

$$I_{Ben}(x_1, x_2) = \sum_{i \in \{1,\dots,n\}} -\phi(f_i(x_1), f_i(x_2)), \tag{7}$$

$$\text{with } \phi(f_i(x_1), f_i(x_2)) = \begin{cases} 1 & \text{if } f_i(x_1) < f_i(x_2) \\ \frac{1}{2} & \text{if } f_i(x_1) = f_i(x_2) \\ 0 & \text{otherwise} \end{cases}$$

$$I_{Ben}(\mathbf{P}, x) = \sum_{z \in \mathbf{P}} (I_{Ben}(z, x)) \tag{8}$$

The ranking method of Fonseca and Flemming [16] (equations 9 and 10) can be adapted in the same way. We obtain a similar formulation which, in this case, exactly corresponds to the original ranking method.

$$I_{Fon}(x_1, x_2) = \begin{cases} -1 & \text{if } x_1 \succ x_2 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$I_{Fon}(\mathbf{P}, x) = \sum_{z \in \mathbf{P}} (I_{Fon}(z, x)) \tag{10}$$

Lastly, the well known ranking method proposed by Goldberg [17] and used in NSGA and NSGA-II by Srinivas and Deb [36] is described by equations 11 and 12. Note that this indicator uses the *min* combination method (equation 5), and we consider that the fitness value of x_1 is known.

$$I_{Sri}(x_1, x_2) = \begin{cases} I_{Sri}(\mathbf{P}, x_1) - 1 & \text{if } x_1 \succ x_2 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$I_{Sri}(\mathbf{P}, x) = min_{z \in \mathbf{P}}(I_{Sri}(z, x)) \tag{12}$$

With equations 8, 10, and 12, we have formulated several classical Pareto ranking methods (from [7], [16] and [36] respectively). The ranks obtained by 10 and 12 during the selection process correspond exactly to the ranks obtained by the corresponding Pareto ranking method. Note that several ranking methods, such as those used in SPEA2 [43], are not adaptable into binary indicators.

The principle of indicator-based search, as described in [41], consists of using binary indicators to evaluate the *fitness* of multi-objective solutions. The *fitness* computed are used during the selection process of the algorithm. We will apply all the indicators presented in this section as selection operators for the IBMOLS algorithm, which is presented in the next section.

In figures 3, 4, 5, 6 and 7, examples of fitness computation using the binary indicators I_ϵ, I_{HD}, I_{Ben}, I_{Fon} and I_{Sri} respectively are provided. In these figures, the same population $\mathbf{P}$ is evaluated using the different indicators. $\mathbf{P}$ contains 8 individuals, defined as follows (f being a biobjective vector function):

$$\mathbf{P} = \{x_1, \ldots, x_8\}, \text{ with } \begin{cases} f(x_1) = (11, 2) \\ f(x_2) = (5, 3) \\ f(x_3) = (8, 4) \\ f(x_4) = (13, 4) \\ f(x_5) = (9, 6) \\ f(x_6) = (4, 7) \\ f(x_7) = (2, 8) \\ f(x_8) = (6, 10) \end{cases} \tag{13}$$

Let $\mathbf{W}$ be the set of solutions with the worst efficiency change according to the binary indicator being used. In our example, $\mathbf{W} = \{x_5\}$ using I_ϵ, $\mathbf{W} = \{x_4\}$ using I_{HD}, $\mathbf{W} = \{x_8\}$ using I_{Ben}, $\mathbf{W} = \{x_4, x_8\}$ using I_{Fon} and $\mathbf{W} = \{x_4, x_5\}$ using I_{Sri}. On this very simple example, $\mathbf{W}$ is different for each binary indicator, which allows us to expect that the choice of a good binary indicator can have a great influence on the results obtained by a binary indicator based metaheuristic.

3 Indicator-based multi-objective local search

Most multi-objective optimisation algorithms from the literature are evolutionary algorithms, since such methods are easily adaptable to a multi-objective context. Indeed, evolving a population of solutions is a natural way to find a set of compromise solutions. Note that the binary quality indicator principle was first proposed within a multi-objective evolutionary algorithm [41]. However, local search algorithms are known to be efficient for many real-world applications, and especially on large-scale problems. Several papers propose local search for multi-objective optimisation. In [24], a multi-objective local search is based on the dominance relation between the considered solution and an archive of compromise solutions, and is incorporated into an evolution strategy method; this algorithm is known as the Pareto Archived Evolution Strategy. In [20], a Multi-Objective Genetic Local Search was proposed. The local

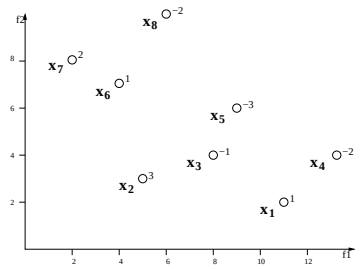

Figure 3: Fitness evaluation using the I_ϵ binary indicator. To simplify this example, the combination of indicator values is realised according to equation 5. In our experiments, we use equation 6, which is similar.

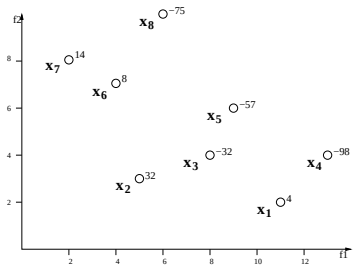

Figure 4: Fitness evaluation using the I_{HD} binary indicator. To simplify this example, the combination of indicator values is realised according to equation 5. In our experiments, we use equation 6, which is similar.

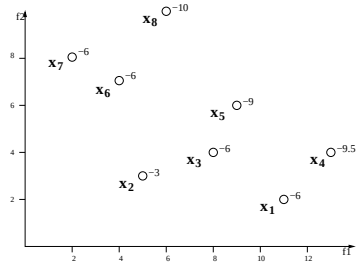

Figure 5: Fitness evaluation using the I_{Ben} binary indicator.

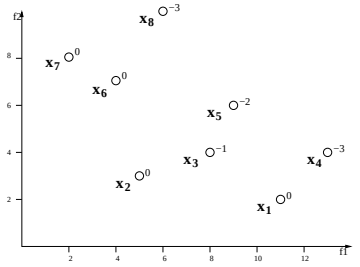

Figure 6: Fitness evaluation using the I_{Fon} binary indicator.

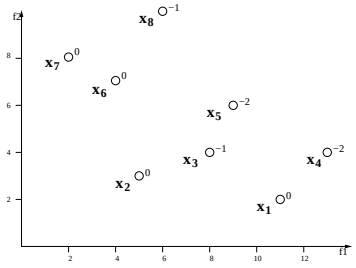

Figure 7: Fitness evaluation using the I_{Sri} binary indicator.

search method, which is based on aggregations of the objective functions, is incorporated within a multi-objective genetic algorithm. In [21], the Tabu search principle is applied to multi-objective optimisation. Several other papers propose multi-objective local search approaches, such as in [39, 31, 40, 34]. Tutorials on multi-objective techniques are presented in [42, 11]. Some application areas are given in [14], including portfolio optimisation, airline operations, railway transportation, radiation therapy planning and computer networks.

The algorithm presented in this section has been designed according to several specificities which compose an original, simple and general-purpose search method:

- This is an original algorithm, since a major part of multiobjective local searches from the literature are based either on the Pareto dominance relation between solutions or on aggregation methods.
- The proposed algorithm has only a few parameters.
- No diversity preservation mechanism is required. The diversity of the population should be contained and improved by the use of the binary indicator defined by the decision maker.
- The local search deals with a fixed population size, which enables it to find multiple non-dominated solutions in a single run, without any specific mechanism dedicated to control the number of non-dominated solutions during the local search process.

The indicator-based multi-objective local search (IBMOLS) described below is defined mainly for discrete combinatorial problems and needs some adaptation in order to be applicable to continuous problems. IBMOLS uses the basic principle of local search and is focused upon binary indicator-based fitness assignment. We first present the IBMOLS baselines, then we discuss the parameter values.

3.1 Algorithm description

The IBMOLS algorithm maintains a population $\mathbf{P}$. Then it generates the neighborhood of an individual in $\mathbf{P}$ until a good solution is found, i.e. one which is better than at least one solution of $\mathbf{P}$ in terms of the indicator being used. By iterating this principle to all the solutions in $\mathbf{P}$, we obtain a local search step. The entire local search is terminated when the archive $\mathbf{A}$ of non-dominated solutions has not received any new solution during a complete local search step. A detailed description of IBMOLS is outlined in algorithm 1, which is taken from [4].

In this paper, the neighborhood will be explored in a random order, each neighbor being generated once at most. It means that while any interesting neighbor is found, we pick randomly a new neighbor in the set of unexplored neighbors. The neighborhood generation stops when an interesting solution is found or when the entire neighborhood is explored. In this algorithm, we choose to stop the neighborhood generation once the first improving solution is found (first neighboring solution that improves the quality of P with respect to I). Then, we do not explore the entire neighborhood for keeping the best neighbor. Two main reasons guided our choice: (1) it allows to speed up

the convergence of the population, since most of the time we generate only a small part of the neighborhood; (2) The selection of the best neighbor leads to deterministic local search steps (one possible way from initial solution to a set of local optima). The selection of a random improving move allows us to reach different local optima from a single initial solution. This issue is interesting since the local search will be iterated. Lastly, a recent study show that, in many cases, this neighborhood search is more efficient that best move search in many multiobjective cases [30].

Moreover, we assume that objective values of all solutions are normalised; to achieve this, the minimum m_i and maximum M_i value of each objective function f_i in the population $\mathbf{P}$ are computed first:

$$\begin{cases} m_i = & min_{x \in \mathbf{P}}(f_i(x)) \\ M_i = & max_{x \in \mathbf{P}}(f_i(x)) \end{cases} \tag{14}$$

Then each objective function i of every individual x is normalised as follows:

$$F_i(x) = \frac{f_i(x) - m_i}{M_i - m_i} \tag{15}$$

where $F_i(x)$ is the normalised i^{th} objective function of the individual x. The extreme values of the population are computed after the initialisation process and after each local search step (see algorithm 1). Then, to compute an indicator value $I(x_1, x_2)$, normalised values of objective functions $F_i(x)$ are employed. Note that extreme values are not updated after each solution generation, but after each local search step, only when a new solution is introduced in the population. When a new maximal value NM_i replaces an old one M_i, the objective values of the solutions $x \in \mathbf{P}$ can be updated easily (note that a similar equation is used when a new minimal value is found):

$$F_i(x) = \frac{f_i(x) - m_i}{NM_i - m_i} \tag{16}$$

This change does not affect solution fitness values if we use dominance-based indicators (I_{Ben}, I_{Fon} and I_{Sri}). When using I_ϵ or I_{HD}, equation 16 leads to a modification of solution fitness values, since it is computed according to objective function difference and also equation 6. Then, each time a bound is updated, the fitness of each solution in the population has to be computed entirely. This step seems a little bit expensive in time, but it is performed only a few time during a local search, since it is applied only when a candidate solution is selected to be introduced in the population, and at least one of the normalised objective functions of this solution is out of the interval $[0, 1]$.

Local search methods are usually used in an iterative way, in order to increase the chance of finding good local optima. In our experiments, we will test an iterative IBMOLS algorithm. Iterated local search algorithms imply the use of a solution(s) initialisation function for each local search. The overall execution process, outlined in algorithm 2 works as follows: a Pareto set approximation $\mathbf{PO}$ is maintained and updated with the solutions found by IBMOLS. After each local search, a new initial population is created for the next IBMOLS execution, using the *generatePopulation* function (see algorithm 2, taken from [4]).

Algorithm 1 Baseline Algorithm: IBMOLS

Input: N (population size)
I (binary indicator)
Output: $\mathbf{A}$: (Pareto approximation set)
Step 1 - initialization: Generate an initial population $\mathbf{P}$ of size N
Step 2: $\mathbf{A} \leftarrow$ Non dominated solutions of $\mathbf{P}$
Step 3 - fitness assignment: Calculate fitness values of individual x in $\mathbf{P}$, i.e., $\mathit{Fit}(x) = I(P \setminus \{x\}, x)$.
Step 4 - local search step: For all $x \in \mathbf{P}$ do:
update, for each objective function f_i, the minimal m_i and maximal M_i values in $\mathbf{P}$ (for objective functions normalisation)
repeat
1) $x^* \leftarrow$ one unexplored neighbor of x
2) $P \leftarrow P \cup x^*$ 3) compute x^* fitness: $I(P \setminus \{x^*\}, x^*)$
4) update all $z \in \mathbf{P}$ fitness values: $\mathit{Fit}(z) += I(x^*, z)$
5) $\omega \leftarrow$ worst individual in $\mathbf{P}$
6) remove ω from $\mathbf{P}$
7) update all $z \in \mathbf{P}$ fitness values: $\mathit{Fit}(z) -= I(w, z)$
until all neighbors are explored or $\omega \neq x^*$
Step 5 - termination: $\mathbf{A} \leftarrow$ Non dominated solutions of $\mathbf{A} \bigcup \mathbf{P}$. If $\mathbf{A}$ does not change, then return $\mathbf{A}$; else perform another local search step.

Algorithm 2 iterated IBMOLS algorithm

Input: N (population size)
I (binary indicator)
Output: $\mathbf{PO}$: (Pareto approximation set)
Step 1: $\mathbf{PO} \leftarrow \emptyset$
Step 2: while running time not reached do
1) $\mathbf{P} \leftarrow generatePopulation(\mathbf{PO}, N)$
2) $\mathbf{A} \leftarrow IBMOLS$ output (initialised with $\mathbf{P}$)
3) $\mathbf{PO} \leftarrow$ non dominated solutions of $\mathbf{PO} \cup \mathbf{A}$
Step 3: Return $\mathbf{PO}$.

I_ϵ indicator and extreme solutions: The shape of the Pareto front is convex for many real world multiobjective problems. In this case, if the population contains only non-dominated solutions, the worst solution will be often an extreme solution, and deleting extreme solutions drive to a loss of diversity in the objective space. In order to avoid this problem, we needed to slightly modify the IBMOLS algorithm when the I_ϵ indicator is used. The algorithm automatically assigns the best possible fitness to extreme non-dominated solutions. This allow to avoid deleting these extreme solutions. Experiments show that this modification can have a great influence on the results.

3.2 Parameters

In order to design a general-purpose metaheuristic, the number of parameters which are sensitive to the problem treated has to be reduced as much as possible. The IBMOLS

algorithm is defined only by three main parameters. They can be defined dynamically during the search or fixed according to the problem instance under consideration. These parameters are the population size, the binary indicator, and the function which initialises the population. We do not consider the neighborhood structure, which is a problem-oriented parameter. We discuss these parameters below.

Population size:

The most intuitive and classical multi-objective local search consists of maintaining a set $\mathbf{A}$ of non-dominated solutions, then generating the neighborhood $\mathbf{A}'$ of solutions in $\mathbf{A}$, then extracting the non-dominated solutions of $\mathbf{A} \cup \mathbf{A}'$, and repeating the process while improvement is realised [14]. The main problem with this algorithm is the population size which is not fixed, and which is heavily dependent upon the problem, objective functions and space dimension being considered. In extreme cases, one can obtain, during the search, only one non-dominated solution, which implies a significant loss of diversity, or a number of non-dominated solutions which grows exponentially and which radically slows down the convergence and can also leads to storage problems. The IBMOLS algorithm deals with a fixed population size, which allows us to avoid this problem. In this paper, we will provide a performance analysis with different population sizes. Our goal is to provide guidelines for future applications of the IBMOLS algorithm to combinatorial optimisation problems.

Binary indicator:

The evaluation of different binary indicators is the main objective of the paper. We will compare the efficiency of two binary indicators (I_ϵ and I_{HD}) previously presented in [41] against other ones, which are based on the dominance relation (I_{Ben}, I_{Fon} and I_{Sri}). The indicator used in the IBMOLS algorithm has a significant influence on determining its efficiency. Experiments are used to determine the most efficient indicators.

Population generation:

In most metaheuristics, the initial population is randomly created. In our experiments, the initial population is also randomly initialised. However, once an entire local search is terminated, the function which generates a new population is very important: Even if the initial population is entirely created randomly, it seems crucial to keep information about good solutions when we iterate the local search process. This issue is often specific to the problem treated.

Naturally, we also have to define a neighborhood operator in order to apply the IBMOLS algorithm to a specific problem. These parameters are discussed in the experimental sections. In the following, we aim to analyse the efficiency of our algorithm on different combinatorial problems. The goal is firstly to evaluate the ability of IBMOLS to be adapted on different types of problems. Secondly, we aim at evaluating the efficiency of the IBMOLS algorithm on these problems and to provide guidelines on how to choose parameters properly according to the problem considered. A significant number of important issues has to be considered in order to provide a complete analysis of a generic metaheuristic. These issues include:

- Evaluating the difficulties encountered when the method is applied to different optimisation problems.
- Comparing the method with well-known methods.

- Comparing the method with specific approaches proposed for the problem under consideration.
- Evaluating the ability of the method to be efficient on large and small search spaces.
- Analysing the behavior of the method when a small or large execution time is available.
- Evaluating the parameter sensitivity of the method.

Furthermore, this list could be enhanced by issues directly related to the multi-objective aspect of the problem under consideration, such as the capacity to find a diversified set of non-dominated solution, the ability to explore different shapes of objective space (for example, convex, concave or discontinuous Pareto fronts) or the efficiency of the method when the number of objective functions increases. In our experiments, we were not able to tackle all these issues, but we provide some useful experiments in order to take into account the most critical issues.

In the three next sections, we will apply the IBMOLS algorithm to three multi-objective optimisation problems which are derived from real world situations. We propose three different case studies in order to evaluate the level of generality of the IBMOLS algorithm. Moreover, we will analyse the sensitivity of the parameters in order to extract some useful information for the application of the IBMOLS algorithm to other real world multiobjective combinatorial optimisation problems.

4 Application I: A Flow-Shop problem

In this section, we propose the application of the IBMOLS algorithm to a bi-objective Flow-shop Scheduling Problem that has appeared in the scientific literature over the years. Firstly, we give some details about this optimisation problem and the parameter values used for the experiments. Then we describe our experimental design and also how we evaluate the quality of each algorithm tested. Lastly, we give the results obtained and we extract some useful information from these results. In this section, we aim to perform a lot of experiments on this problem in order to evaluate the efficiency of the method, evaluate its parameter sensitivity, and compare the I_{HD} and I_ϵ indicators to dominance-based indicators.

4.1 Problem description

The Flow-shop Scheduling Problem (FSP) is one of the numerous scheduling problems. Scheduling problems are often studied with a multi-objective approach, since many different objective functions can be considered, such as sum or maximum completion time, sum or mean tardiness among others.

The FSP can be presented as a set of n jobs $\{J_1, J_2, \ldots, J_n\}$ to be scheduled on m machines $\{M_1, M_2, \ldots, M_m\}$. Machines are critical resources: one machine cannot be assigned to two jobs simultaneously. Each job J_i is composed of m consecutive

tasks $\{t_{i1}, \ldots, t_{im}\}$, where t_{ij} represents the j^{th} task of the job J_i requiring the machine m_j. To each task t_{ij} is associated a processing time p_{ij}. Each job J_i must be achieved before its due date d_i. In our study, we are interested in the permutation FSP where jobs must be scheduled in the same order on all the machines (see figure 8).

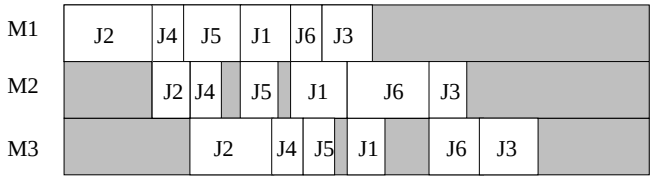

Figure 8: Permutation Flow-Shop solution: 6 jobs scheduled on 3 machines.

We aim to minimize two objectives: C_{max}, the makespan (total completion time), and T, the total tardiness. Each task t_{ij} is scheduled at the time s_{ij}. The two objectives can be computed as follows:

$$C_{max} = \max_{i \in [1 \ldots N]} \{s_{iM} + p_{iM}\}$$

$$T = \sum_{i=1}^{n} [\max(0, s_{iM} + p_{iM} - d_i)]$$

In the Graham et al. notation [18], this problem is denoted: $\mathrm{F}/\mathrm{perm}, d_i/(C_{max}, T)$.

Overviews of multi-objective methods applied to scheduling are given in [27] and [32]. In [29], C_{max} minimisation has been proved to be NP-hard for more than two machines. The total tardiness objective T has been studied only a few times for m machines [22], but total tardiness minimisation for one machine has been proved to be NP-hard [13]. The evaluation of the performances of our algorithm has been realised on some Taillard benchmarks for the FSP [37], which have been extended to the bi-objective case [38][1].

4.2 Parameter settings

Providing a complete parameter sensitivity analysis of IBMOLS, using different binary indicators, population initialisation methods and population sizes requires a huge number of experiments, especially since we aim to provide statistical analysis. In order to reduce the time of experiments, we analyse the sensitivity of each parameter separately. When the sensitivity of a parameter is not tested, its value is defined as follows:

- Generation of the initial population for local search: random mutations applied on the archived solutions. The number of mutations applied to the original solution is $0.3n$, where n is the permutation (decision vector) size. A random mutation consists of replacing a solution by a randomly chosen neighbor with respect to the neighborhood used by the local search algorithm.

[1] benchmarks available at http://www.info.univ-angers.fr/pub/basseur/bench.html

- Population size N: defined according to the size of the instance, as proposed in [4] (see table 1).
- Binary indicator: I_ϵ.

For the I_ϵ and I_{HD} indicators, the scaling factor κ has been set to 10^{-3}. I_{HD} needs a reference point, which has been set to $[2, 2]$ (normalised values), as suggested in [41]. The running time allowed to each algorithm is fixed and defined according to the instance size (see table 1).

Table 1: Parameter setting: Population size N and running time T. $ta\_\#i\_\#j\_\#k$ represents the k^{th} bi-objective instance with i jobs and j machines.

Instance	N	T	Instance	N	T
$ta\_20\_5\_01$	10	20"	$ta\_20\_20\_01$	10	2'
$ta\_20\_5\_02$	10	20"	$ta\_50\_5\_01$	10	5'
$ta\_20\_10\_01$	10	1'	$ta\_50\_10\_01$	20	10'
$ta\_20\_10\_02$	10	1'	$ta\_50\_20\_01$	20	20'

The last parameters are those which are directly related to the problem considered: the individual coding and the neighborhood operator.

- Individual coding: sequence of jobs. A solution of a problem with n jobs and m machines is represented by a permutation of size n (the jobs have to be scheduled in the same order on all machines [5]). Then, the associated desion space is of size $n!$.
- Neighborhood operator: insertion of the i^{th} job to the position j. The jobs between position i and j are shifted. In [37], this operator has been shown to be more efficient than exchange operator for C_{max} minimisation.

4.3 Experimental design

As shown in [3], genetic algorithms, and especially NSGA-II, are strongly outperformed by local search based algorithms, on this problem. However, as shown in [3], the application of a basic local search algorithm, keeping all the non-dominated solutions during the neighborhood search, implies a very long computation time on large-size instances. IBMOLS enables the combination of the local search principle and of a fixed-size population.

With this set of experiments, we do not aim to show the superiority of the IBMOLS algorithm against other approaches. This first set of experiments mainly aims at evaluating the parameter sensitivity of IBMOLS. To reach this goal, we perform three sets of experiments, using different binary indicators, population initialisation methods and population sizes. Note that we do not evaluate the IBMOLS algorithm against other algorithms from the literature, but we aim to evaluate the quality of the binary indicators derived from these studies (the classical multi-objective evolutionary algorithms, such

as NSGA-II or SPEA2 have their own specificities, for instance the diversity maintaining mechanism).

The quality assessment protocol works as follows: we first create a set a 20 runs with different initial populations for each algorithm and each benchmark instance. Runs are realised on a P4 - 2.4GHz machine, with 1Gb RAM. The performance evaluation protocol described below will be used for the three optimisation problems tested in this paper.

To evaluate the quality of k different sets $\mathbf{A_0} \ldots \mathbf{A_{k-1}}$ of non-dominated solutions obtained on a problem instance, we first compute the set $\mathbf{PO}^*$, which corresponds to the set of non-dominated solutions extracted from the union of all the solutions obtained from the different executions. Moreover, we define a reference point $z = [w_1, w_2]$, where w_1 and w_2 correspond to the worst value for each objective function in $\mathbf{A_0} \cup \cdots \cup \mathbf{A_{k-1}}$. Then, to evaluate a set $\mathbf{A_i}$ of solutions, we compute the difference between $\mathbf{A_i}$ and $\mathbf{PO}^*$ in terms of a performance indicator. In particular, we evaluate our outputs using the R metric, ϵ and enclosed hypervolume indicators, using the same experimental protocol described later in this section. Let us note that we obtain similar results using each assessment indicator and/or statistical tests, i.e. there are no significant differences obtained with each possible indicator or statistical test. Then, in this paper, we will focus on the enclosed hypervolume indicator only [44]. This hypervolume difference has to be as close as possible to zero (figure 9).

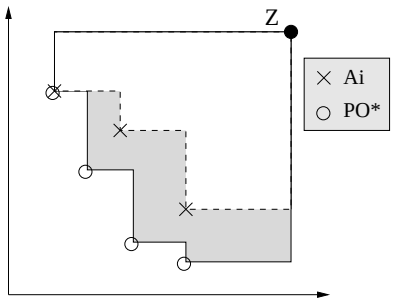

Figure 9: Illustration of hypervolume difference between a reference set $\mathbf{PO}^*$ and a set of non-dominated solutions A_i (shaded area) ©[4].

For each algorithm, we compute the 20 hypervolume differences corresponding to the 20 runs. Once the hypervolume differences are computed, there are two main ways to merge these values. The simplest way consists of computing the average hypervolume difference value for each algorithm tested. As suggested in [25], it is more representative to perform statistical tests on the sets of hypervolume differences, in order to evaluate with which confidence level an algorithm A outperforms an algorithm B (with respect to hypervolume differences).

Unfortunately, showing the entire statistical result requires a lot a space, since we have to show the statistical analysis result for each pair of runs, for each problem instance. We are not able to provide these heavy tables, especially since we want to provide a range of experiments on different problems. We propose a balance between statistical analysis and average values, which allows us to give the maximum amount

of information within a small table. Our tables will provide two different pieces of information:

- The average hypervolume difference is computed for each algorithm and each problem instance. This enables us to have a good idea of the overall performance of each approach.
- In order to refine the results, some values are given in **bold** style, which means that the corresponding algorithm is **not** statistically outperformed by the algorithm which obtains the best average result. To simplify, the results in bold corresponds to the best methods (statistically) on the problem instance considered.

We use the Mann-Whitney statistical test, as described in [25]. We obtain P values corresponding to the hypothesis *"the first algorithm performs better than the second one in terms of hypervolume difference"*. This is equal to the lowest significance level for which the null-hypothesis (the medians are drawn from the same distribution) would still be rejected. In our experiments, we say that an algorithm A outperforms an algorithm B if the Mann-Whitney test provides a high confidence level which indicates that the P-value of the hypothesis *"A performs better than B"* is lower than 5%. More details are given in [25].

In order to allow a graphical comparison of stochastic multi-objective optimisers, the concept of attainment function is described in [19]. The attainment function $\alpha_A(z)$ corresponds to the probability that at least one element of a non-dominated set $\mathbf{P}$ dominates a reference point z (refer to [19] for more details), where $\mathbf{P}$ is obtained by a single execution of an algorithm A.

The test procedure has been undertaken with the performance assessment package provided by Knowles et al. [25], which can be found at the following URL: `http://www.tik.ee.ethz.ch/pisa/assessment.html`.

4.4 Experiments

We provide a very detailed experimental analysis of this problem. Our analysis is divided into three parts which evaluate the binary indicators, the initialisation strategies and the population size impact, results being given respectively in tables 2, 3 and 4.

4.4.1 Binary indicators

The first set of experiments which are carried out are dedicated to evaluating the different binary indicators. The results obtained with five different indicators (I_ϵ, I_{HD}, I_{Ben}, I_{Sri} and I_{Fon}), are shown in Table 2. We can extract several pieces of information for each indicator:

- I_ϵ: This indicator obtains the best average results on many instances. Indeed, I_ϵ statistically outperforms all the other indicators for the 4 largest instances ($ta\_20\_20\_01$, $ta\_50\_5\_01$, $ta\_50\_10\_01$ and $ta\_50\_20\_01$). I_ϵ obtains the best average value and outperforms every indicator except I_{HD}, on the $ta\_20\_10\_01$

and $ta\_20\_10\_02$ instances. On the two smallest instances, I_ϵ does not outperform many indicators. However, I_ϵ is only shown to be statistically worse than I_{Fon} on the $ta\_20\_5\_01$ and $ta\_20\_5\_02$ instances.

- I_{HD}: This indicator obtains good overall results. Indeed, this is the only indicator which does not perform statistically worse than I_ϵ on the $ta\_20\_10\_01$ and $ta\_20\_10\_02$ instances. Moreover, even if I_ϵ outperforms I_{HD} on the two largest instances, I_{HD} obtains a good average hypervolume difference and reaches the second position within the different indicators. On the four other instances ($ta\_20\_5\_01$, $ta\_20\_5\_02$, $ta\_20\_20\_01$ and $ta\_50\_5\_01$), I_{HD} obtains good average results.

- I_{Fon}: This indicator obtains the best results within the set of indicators which are only based on the dominance relation. The overall results of this indicator are comparable to the I_{HD} indicator. However, good performances are obtained on different instances. I_{Fon} obtains the best results for the two smallest instances, and the quality of the algorithm seems to decrease when the problem size increases.

- I_{Sri}: The average results obtained are similar, but always worse, to those obtained by I_{Fon}.

- I_{Ben}: This indicator performs worse than the other indicators for all the instances.

To summarise, the I_ϵ and I_{HD} indicators tend to outperform the Pareto dominance based indicators, especially when the problem size increases. I_ϵ performs slightly better than I_{HD}. Furthermore, a reference point has to be set for I_{HD}. We can conclude that the I_ϵ indicator should be used to solve the bi-objective FSP.

To illustrate the efficiency of the different binary indicators, we have represented the empirical attainment function computed for the $ta\_50\_20\_01$ instance in figure 10, taken from [4]. This figure illustrates the minimal values in the objective space which are attained with at least 90% of the runs. This shows the superiority of I_ϵ and I_{HD} against the other indicators on the largest instance considered.

Table 2: Indicator comparison.

Indicator	I_ϵ	I_{HD}	I_{Ben}	I_{Sri}	I_{Fon}
$ta\_20\_5\_01$	0.005	0.077	0.117	0.009	**0.002**
$ta\_20\_5\_02$	0.070	0.062	0.097	0.020	**0.010**
$ta\_20\_10\_01$	**0.002**	**0.004**	0.045	0.010	0.004
$ta\_20\_10\_02$	**0.018**	**0.021**	0.075	0.024	0.022
$ta\_20\_20\_01$	**0.001**	0.011	0.045	0.007	0.004
$ta\_50\_5\_01$	**0.009**	0.059	0.271	0.076	0.034
$ta\_50\_10\_01$	**0.055**	0.089	0.341	0.151	0.099
$ta\_50\_20\_01$	**0.058**	0.077	0.349	0.182	0.111

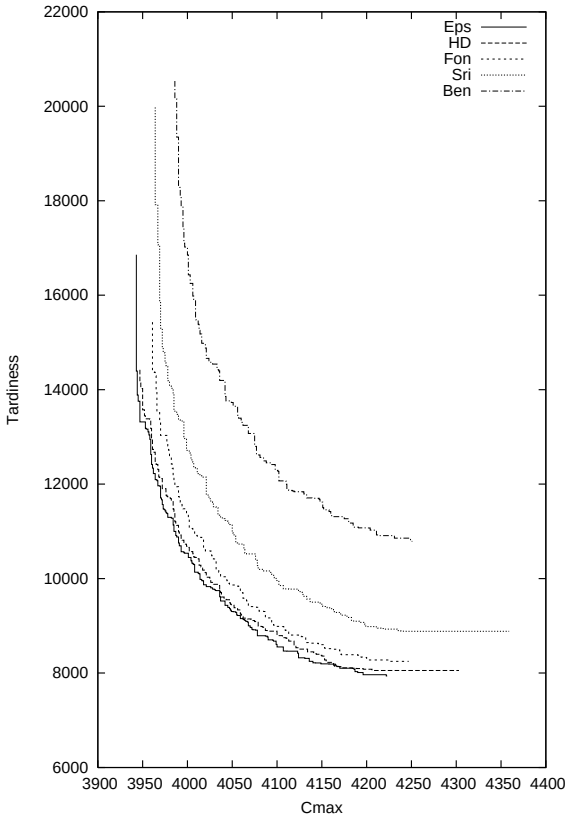

Figure 10: Empirical attainment functions obtained by the different indicators on the $ta\_50\_20\_01$ instance. The lines correspond to the limit of the objective space, which is attained by at least 90% of the runs carried out with a specific binary indicator ©[4].

4.4.2 Population generation methods

We propose three different functions corresponding to different implementations of the $generatePopulation$ function used in algorithm 2. The three functions, can be outlined as follows:

- *Rand*: generate N random individuals.
- *Cro*: generate a crossover output from $2N$ parents selected randomly from $\mathbf{PO}$. Each solution can be selected once. If $\mathbf{PO}$ size $< 2N$, then select all $\mathbf{PO}$ individuals once, and fill the missing parents pool with random individuals. The recombination operator applied is the two point crossover used in [5].
- *RM*: Apply random mutations on N randomly selected and different solutions of $\mathbf{PO}$, such as in a basic simulated annealing algorithm [1]. Each solution can be selected once. A predefined number of random moves are applied on each

solution using the neighborhood operator used by the local search algorithm. If $\mathbf{PO}$ size $< N$, then select all $\mathbf{PO}$ individuals, and fill the missing pool with random individuals (i.e. random permutations).

Table 3 presents the results obtained using three different population generation methods. We compare seven different sets of executions, since we performed five different sets of executions for *RM* method. Indeed, *RM* consists of applying random mutations to locally optimal solutions regarding to the neighborhood operator being used, so the amount of random mutations to apply had to be defined. In these experiments, we tested 5%, 10%, 20%, 30% and 50% of the problem size.

The results show that the *RM* initialisation method performs better than *Rand* and *Cro* for all the instances. The *Cro* and *Rand* initialisation methods obtain a better average hypervolume difference than *RM* with 5% mutation rate on only one instance: $ta\_20\_5\_01$. As a first conclusion, we can say that *RM* is the most efficient initialisation algorithm, even if *Cro* efficiency can be improved according to the quality of the crossover operator being used.

The results obtained with the different *RM* initialisations significantly depend upon the problem instance considered. On large-size instances ($ta\_50\_5\_01$, $ta\_50\_10\_01$ and $ta\_50\_20\_01$), applying 5% or 10% of random mutations seems to be the best choice. This result can be explained since the execution time is limited, then it is better to search around the most useful solutions (small mutation rate applied on the best solutions) rather than diversify the search (large mutation rate applied on the best solutions).

On the other instances, a 5% mutation rate performs worse than other mutation rates in many cases. The best mutation rate depends on the instance considered. On the $ta\_20\_5\_01$ instance, 10% and 20% mutation rates obtain the best results. On the $ta\_20\_5\_02$ instance, 20% and 30% rates outperform the other mutation rates. On the $ta\_20\_10\_01$ instance, 50% obtains the best average hypervolume difference, but it does not outperform results obtained with the 20% and 30% mutation rates. On this instance, it seems to be very important to diversify the search by applying a lot of random mutations on the optimal solutions. On the $ta\_20\_10\_02$ instance, a 10% mutation rate outperforms most of the other mutation rates except the 5% mutation rate which obtains good results, even when the size of this instance is small. Lastly, on $ta\_20\_20\_01$, results obtained with different mutation rates are comparable, even if the best average result is obtained with a 20% mutation rate.

As a conclusion, the *RM* population generation method seems to be an effective way to initialise populations. Furthermore, we can suggest the use of a small mutation rate to solve large-size problems. For small-size problems, it is not clear how to choose the best mutation rate. Intuitively, creating solutions by applying high mutation level corresponds to more diversification in the search. On small instances, the diversification enables the exploration of most parts of the search space, in order to find the optimal solutions. On large instances, we can only focus on small parts of the search space, then it is better to focus the search on areas of the solution space which is known to contain good solutions. Perhaps it would be interesting to define this value adaptively during the search procedure. Further experiments are provided in the next sections, in order to obtain more knowledge about how to set this parameter.

Table 3: Initialisation strategy comparison.

Init strategy	RM					Rand	Cro
RM rate	5%	10%	20%	30%	50%		
$ta\_20\_5\_01$	0.063	**0.018**	**0.007**	0.011	0.009	0.039	0.024
$ta\_20\_5\_02$	0.087	0.039	**0.011**	**0.013**	0.018	0.129	0.140
$ta\_20\_10\_01$	0.005	0.006	**0.004**	**0.004**	**0.003**	0.007	0.014
$ta\_20\_10\_02$	**0.024**	**0.021**	0.030	0.028	0.028	0.035	0.063
$ta\_20\_20\_01$	0.010	**0.003**	**0.002**	**0.003**	**0.003**	0.013	0.060
$ta\_50\_5\_01$	**0.024**	**0.029**	0.050	0.061	0.079	0.122	0.117
$ta\_50\_10\_01$	**0.080**	**0.066**	0.087	0.102	0.114	0.169	0.263
$ta\_50\_20\_01$	**0.081**	**0.081**	0.107	0.115	0.136	0.174	0.134

4.4.3 Population size

In many population-based metaheuristics, such as evolutionary algorithms, the population size is a significant parameter. Generally, the results are better when the population size increases and when enough computational time is employed. When the execution time is limited, the population size has to be set in order to allow the algorithm to converge just before the time limit.

Table 4 shows the results obtained using different population sizes. The first important result which can be extracted from this set of experiments is that the IBMOLS algorithm performs well using a small population size: the largest population size tested (50 individuals) never obtains the best result, for all problem instances. However, the best population size seems to increase according to the size of the problem. For most of the smallest instances, the best results are achieved with less than 10 individuals in the population, but for the two largest instances, $ta\_50\_10\_01$ and $ta\_50\_20\_01$, the best population sizes lay in the range of 15 to 30 individuals. Then, in order to solve new instances, it could be interesting to evaluate the average results when the population size is fixed linearly according to the number N of jobs times the number M of machines, $\frac{N.M}{10}$ for example.

Table 4: Population size comparison.

Population size	3	5	8	10	15	20	30	50
$ta\_20\_5\_01$	**0.002**	0.004	0.006	0.009	0.010	0.015	0.025	0.065
$ta\_20\_5\_02$	**0.005**	**0.004**	**0.015**	0.038	0.037	0.081	0.099	0.147
$ta\_20\_10\_01$	0.013	0.007	**0.004**	0.005	0.006	0.007	0.012	0.016
$ta\_20\_10\_02$	0.032	**0.027**	**0.028**	**0.025**	**0.024**	**0.023**	**0.026**	0.038
$ta\_20\_20\_01$	0.008	0.003	**0.002**	**0.002**	**0.002**	0.004	0.006	0.017
$ta\_50\_5\_01$	0.071	0.046	**0.034**	**0.034**	0.043	0.061	0.067	0.079
$ta\_50\_10\_01$	0.213	0.128	0.100	**0.091**	**0.087**	**0.098**	0.125	0.147
$ta\_50\_20\_01$	0.211	0.149	0.107	0.097	**0.084**	**0.078**	**0.077**	0.110

Figures 11 and 12 show the evolution of the performance of the IBMOLS algorithm, in terms of average hypervolume difference over time. Figure 11 shows that

results obtained with a population size of 30 individuals are outperformed by those obtained with a smaller population size, and it does not seem to change when the run time increases ($ta\_20\_10\_01$ instance). Figure 12 is obtained on $ta\_50\_20\_01$ instance. In this case, the results obtained with 30 individuals are outperformed by those obtained with eight individuals, but only when the run time is less than around 350 seconds. After that period, we can observe that the IBMOLS algorithm using 30 individuals starts to outperform the one using eight individuals, as we observe for many evolutionary algorithms. However, on the same figure, we can observe that the employment of 50 individuals never seems to perform well. This issue is interesting and is discussed in section 6, where a complete set of experiments is provided, using different running times.

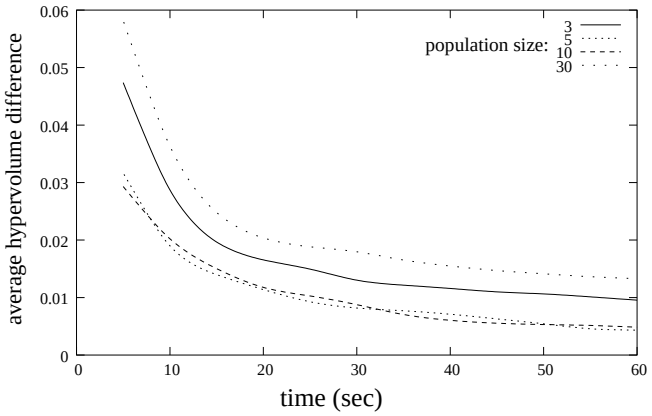

Figure 11: Evolution of the average hypervolume difference, using different population sizes: $ta\_20\_10\_01$ instance.

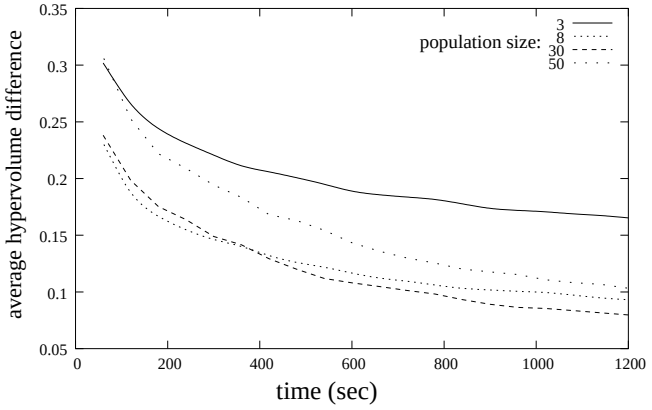

Figure 12: Evolution of the average hypervolume difference, using different population sizes: $ta\_50\_20\_01$ instance.

Note that the execution times are very short in this study (see table 1), in order to be able to provide statistical analysis of the different executions. However, the results obtained by the $IBMOLS$ algorithm are close to the best values found in [5], with very long execution times.

5 Application II: A Ring Star problem

In this section, we will consider another application: the Ring Star problem, which is an academic problem with many real world applications. First, we present some details about the bi-objective Ring Star problem. Secondly, we discuss the parameter values used for the experiments, and how to adapt IBMOLS to this new problem. Then, we describe our experimental protocol and provide an analysis of the results. In this section, we aim to compare our method with two classical MultiObjective Evolutionary Algorithms (MOEAs) of the literature, i.e. NSGA2 and IBEA, for the Ring Star problem.

5.1 Problem description

The *Ring Star Problem* (RSP) [26] can be described as follows. Let $G = (V, E, A)$ be a complete mixed graph where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices, $E = \{[v_i, v_j] | v_i, v_j \in V, i < j\}$ is a set of edges, and $A = \{(v_i, v_j) | v_i, v_j \in V\}$ is a set of arcs. Vertex v_1 is the depot. To each edge $[v_i, v_j]$, we assign a non-negative *ring cost* c_{ij}, and to each arc (v_i, v_j) is assigned a non-negative *assignment cost* d_{ij}. These costs are defined as follows: let l_{ij} denote the Euclidian distance between two nodes v_i and v_j of a TSPLIB data file. As proposed by Labbé et al. [26], the ring cost c_{ij} and the assignment cost d_{ij} are both been set to l_{ij} for every pair of nodes v_i and v_j.

The RSP consists of locating a simple cycle through a subset $V' \subset V$ (with $v_1 \in V'$) while (i) minimising the sum of the ring costs related to all edges that belong to the cycle, and (ii) minimising the sum of the assignment costs of arcs directed from every non-visited node to a visited one so that the associated cost is minimum. An example of solution is given in figure 13, where the solid lines represent the edges that belong to the ring and the dashed lines represent the arcs of the assignments.

The first objective is called the *ring cost* and is defined as:

$$\sum_{[v_i, v_j] \in E} c_{ij} x_{ij} \tag{17}$$

where x_{ij} is a binary variable equal to 1 if and only if the edge $[v_i, v_j]$ belongs to the cycle.

The second objective, the *assignment cost*, can be computed as follows:

$$\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} d_{ij} \tag{18}$$

This problem is particularly challenging because, for each given subset of nodes to visit, a classical Traveling Salesman Problem (TSP) still remains to be solved.

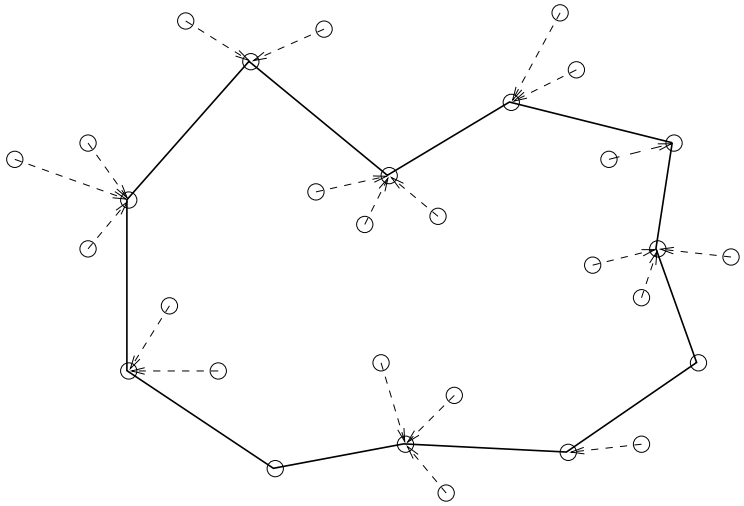

Figure 13: The bi-objective Ring Star problem: minimize the ring cost and the assignment cost (bold cycle). Each vertex which does not belong to the ring is assigned to the closest vertex in the ring (dashed lines).

5.2 Parameter setting

The parameters which are directly related to the problem considered, the individual coding and the neighborhood operator, are defined as follows:

- Individual coding: sequence of vertices. A solution of a problem with n vertices is represented by a set of k vertex indices, with $0 \leq k \leq n$. These k values corresponds to visited nodes, in the order they are visited.
- Neighborhood operator: the Ring Star problem is at the same time an assignment problem and a routing problem. In order to take into account these two specificities, the neighborhood is divided into three subsets:
  1. 2-opt exchange operator: the sequence of visited nodes between two vertices v_i and v_j is reversed.
  2. Insert vertex: add a vertex v in the ring. The position of v in the ring is chosen in order to minimise the ring cost, i.e. placed at the best position among all possible ones according to the ring cost. The assignment is updated according to the vertex inserted.
  3. Delete vertex: remove a vertex v from the ring. Re-assign the vertices which were assigned to v.

  As for the FSP problem, the neighbors are randomly generated, without considering any exploration order between the three subsets of neighbors.

The other parameters are defined as for the first application, except the binary indicator. Indeed, we will only consider here the ϵ indicator, which has obtained the best average performance on the bi-objective FSP.

5.3 Experimental design

In the previous section, the statistical analysis of the results demonstrated the efficiency of the I_ϵ indicator, which outperforms the other binary indicators. In these experiments and the experiments presented in the next section, we will not provide further results using different binary indicators, since the results obtained are very similar and the best results are achieved by the I_ϵ indicator. For the same reason, we will not provide an analysis of the results of the different $generatePopulation$ functions, since ***RM*** clearly outperforms the other approaches on the three optimisation problems treated in this paper.

Here, we provide a statistical analysis of different IBMOLS versions, using different population sizes and mutation rates in the ***RM*** initialisation. Furthermore, we provide a comparison against two well known MOEAs from the literature: NSGA II and IBEA [12, 41]. The performance assessment protocol used in these experiments is described in the previous section. For each methodology, we compute the 20 hypervolume differences corresponding to the 20 runs. Then, we compute the statistical confidence level for the affirmation *"algorithm A outperforms algorithm B"* (Mann-Whitney test); the average hypervolume values which appear in bold in the tables correspond to the algorithms which are not statistically outperformed by any other algorithm.

Eight problem instances, extracted from the traveling salesman problem benchmark instances[2], are tested: $eil51$, $st70$, $kroA100$, $bier127$, $kroA150$, $kroA200$, $pr264$ and $pr299$. The number contained in each instance name represents the number of vertices of the corresponding problem.

We experiment with nine different versions of the IBMOLS algorithm, using combinations of three different population sizes and three different mutation rates in the ***RM*** population generation strategy. The previous experiments showed that the use of a small mutation rate and population size allows us to obtain better results. In the experiments presented here, we test different values:

- Mutation rate: 5%, 10% and 20% of the problem size (i.e. the number of vertices).
- Population size: three different values (S, M and L), defined according to the instance size, i.e. respectively 5, 10 and 15 for $Eil51$ and $St70$ instances, 10, 15 and 20 for $kroA100$ and $bier127$ instances, and 15, 20 and 30 for $kroA150$, $kroA200$, $pr264$ and $pr299$ instances.

The IBEA and NSGA II algorithms are also tested. For both algorithms, the crossover probability is set to 0.25, and the mutation probability to 1.00, with a probability of 0.25, 0.25 and 0.50 for the remove, the insert and the 2-opt operator, respectively. The population size is fixed to 100 individuals. Lastly, we present results for the

[2]URL: `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

IBEA algorithm using a small population size (M values), since as for the IBMOLS algorithm, IBEA seems to perform better using a small population size. Implementation was realised using the $ParadisEO$ platform [9]. The experiments are realised on an Intel duo-core 2*2.4GHz with 2Gb RAM.

5.4 Experiments

Table 5 summarises the results obtained with the different algorithms on several instances. The difference between the algorithms, in terms of hypervolume difference, is very small. Then, in Table 5, the value 0 is given to the most efficient algorithm, and the other values correspond to the deviation of the considered algorithm with respect to the best algorithm. The table shows the efficiency of the IBMOLS algorithm, which obtains very good results on all instances except on $kroA100$ and $pr299$ instances, where some IBMOLS versions are statistically outperformed by IBEA (with 100 individuals). IBEA obtains good results, but is generally outperformed by most of the IBMOLS versions.

IBMOLS seems to perform better using the larger population size. However, the larger size corresponds to 15, 20 or 30 individuals, which is quite small in comparison to usual population size used in classical MOEAs. Concerning the mutation level, no overall conclusion can be extracted from the table, even if the rate still has an influence on the results.

As a conclusion, the tests realised on the ring star problem show some similar properties to those carried out on the flow shop problem, but with several differences: IBMOLS performs better using a small population size, even if the optimal population size is larger on the RSP. When applied on the FSP, the 'optimal' mutation rate of IBMOLS should be smaller with increasing problem size. For the RSP this seems not to be the case. Moreover, IBMOLS outperforms two state-of-the-art well known MOEAs: NSGA-II and IBEA.

Table 5: Algorithm comparison: average difference with the best algorithm (hypervolume difference deviation, $\times 10^{-3}$). The running times are 10", 20", 1', 2', 5', 10', 20' and 50' for the $eil51$, $st70$, $kroA100$, $bier127$, $kroA150$, $kroA200$, $pr264$ and $pr299$ instances respectively.$\{S, M, L\}$ equals to $\{5, 10, 15\}$ for $Eil51$ and $St70$ instances $\{10, 15, 20\}$ for $kroA100$ and $bier127$ instances, and $\{15, 20, 30\}$ for $kroA150$, $kroA200$, $pr264$ and $pr299$ instances.

Algorithms	IBMOLS									NSGA-II	IBEA	
RM rate	5%			10%			20%			-	-	
Pop size	S	M	L	S	M	L	S	M	L	100	100	M
eil51	1.144	0.630	**0.075**	1.953	0.656	**0**	3.142	0.890	**0.119**	3.068	0.995	3.572
st70	1.036	0.531	**0.070**	1.601	0.677	**0.121**	2.042	0.498	**0**	2.077	0.198	2.208
kroA100	1.224	**0.237**	**0**	1.635	0.412	**0.020**	2.491	0.584	**0.268**	3.079	**0.793**	6.577
bier127	1.514	**0.302**	**0.035**	2.132	**0.390**	**0**	3.030	1.271	0.784	3.385	0.972	10.430
kroA150	0.880	0.324	**0**	1.332	0.701	**0.311**	1.926	1.164	0.421	3.106	19.321	8.525
kroA200	0.862	0.641	**0**	1.020	0.668	**0.152**	1.292	1.085	0.650	2.608	44.943	8.189
pr264	**0.039**	0.113	0.238	0.059	0.189	0.327	0.263	**0.069**	**0**	0.614	0.163	1.430
pr299	0.111	0.204	0.092	0.041	0.155	0.436	0.054	**0.098**	**0**	0.967	**2.571**	2.743

6 Application III: A Nurse scheduling problem

Nurse rostering is an important search problem that has received significant attention from the research community [8]. In this section, we will consider a Nurse Scheduling Problem (NSP) which is directly extracted from a real world situation. This problem is slightly different from the two first problems considered, and will give us an idea of the difficulties of adapting IBMOLS to real world problems. Indeed, the problem considered here contains a lot of hard constraints and three objectives. In this section, IBMOLS is compared to SEAMO-R, which was proposed to solve the NSP problem in [28]. First, we briefly describe the nurse scheduling problem. Secondly, we discuss the parameter values used for the experiments, and issues on how to adapt IBMOLS to this real world problem. Then we describe our experimental protocol and provide an analysis of the results.

6.1 Problem description

The Nurse Scheduling Problem described in this paper, is to construct non-cyclic schedules for a ward of nurses in the *Queens Medical Centre* in Nottingham, UK. The scheduling period is a 28 days period to cover a 24-hours basis (early, late and night shift), seven days a week. Each nurse works either on a part-time or on a full-time basis. Nurses are classified in a hierarchy according to their qualifications and training. This NSP includes the most common constraints from the nurse scheduling literature, as identified in [10].

The *Queens Medical Centre* NSP is formulated as the ordered pair $\langle Nurses, C\rangle$ where $Nurses = \{N_i : 1 \leq i \leq n\}$ is a set of n nurses and C is a set of constraints. Nurses usually indicate their individual working preference (e.g. days off, preferred shifts, etc.) for each scheduling period. Constructed schedules should meet the work regulation such as one working shift a day, maximum working hours regarding nurses' contract, maximum/minimum consecutive working days, illegal shift patterns. The constructed schedules should also confront the coverage demands regarding nurse qualifications and training. Furthermore, any surplus or deficit (coverage, demand) of nurses over the scheduling period should be evenly distributed amongst shifts.

We aim to minimise three objective functions, which are briefly described below:

- Work regulation violations: Combination of three preference types, i.e. (1) *SingleNight*: Penalty is applied each time a night shift is assigned to a nurse on a specific day, and shifts different to Night are assigned on adjacent days; (2) *WeekendSplit*: Penalty is applied each time a nurse is assigned to work only on a single day of a weekend; (3) *WeekendBalance*: Penalty is applied if a nurse is assigned to work at least one day in each of the four weekends in the scheduling period.

- *Coverage* demands satisfaction in the scheduling period: if the number of nurses with specific qualifications and training assigned to a given shift is less than the coverage demand, a penalty equals to the deficit in the number of nurses assigned is applied.

- Even distribution of nurses amongst shifts (*CoverageBalance*): it consists in measuring the statistical variation on the difference between the number of qualified nurses assigned to each shift and the coverage demand for qualified nurses.

Note that we set a fixed threshold for the nurses' individual working preferences to guarantee a minimum level of staff satisfaction amongst different scheduling periods rather than trying to optimise it. For more details of the problem description see [28], and visit `http://www.cs.nott.ac.uk/~tec/NRP/` for a web repository of nurse scheduling problems.

6.2 Parameter setting

The parameters which are directly related to the problem, the individual coding and the neighborhood operator, are defined as follows:

- Individual coding: set of n schedules of 84 time slots (28 days, each day being divided in three time slots). A decoding procedure is applied on the individuals before the evaluation process, since some constraints have to be satisfied. The decoder is able to apply a small change to a solution in order to build a feasible solution. For more details, see [28].

- Neighborhood operator: in [28], the authors remark that during the evolution process, the crossover operator is able to build good solutions whereas the mutation operator is unable to improve existing good solutions. Then, they choose to use only the crossover operator in their GA. The crossover operator used is the *cycle crossover* [33], applied on each nurse schedule. The cycle crossover is described in the example below:

  Let two permutations $P1$ and $P2$ of size nine, with $P1 = (1, 2, 3, 4, 5, 6, 7, 8, 9)$ and $P2 = (4, 1, 2, 8, 7, 6, 9, 3, 5)$. The child C starts by taking the first value from $P1$, the child $C = (1, -, -, -, -, -, -, -, -)$. The next value must be from $P2$ and from the same position. This gives value 4, which is in fourth position on $P1$: $C = (1, -, -, 4, -, -, -, -, -)$. This process is iterated until a cycle is obtained ($C = (1, 2, 3, 4, -, -, -, 8, -)$ for this example). Once, the cycle is obtained, the remaining value are pasted from $P2$, then we obtain $C = (1, 2, 3, 4, 7, 6, 9, 8, 5)$.

  The *cycle crossover* is applied on each couple (schedule of nurse i of individual 1, schedule of nurse i of individual 2). In our case, the neighborhood consists of applying the cycle crossover on only one selected nurse, in order to favorise local moves instead of applying an entire crossover operation. Then, if the algorithm runs with N individuals on a problem instance containing n nurses, the neighborhood size is equal to $n * N$.

The other parameters are defined as for the two first applications. For the population generation function, the random mutation rate in the application consists of several random swap mutations on the individuals.

6.3 Experimental design

The different datasets available for this problem have similar sizes, e.g. around 15 nurses and 28 days divided into 84 time slots. The dataset corresponds to nurse schedule in different month in 2001 (from March to September), which are named accordingly ($March2001, \ldots, September2001$). In order to evaluate the effectiveness of the IBMOLS algorithm under different conditions, we perform three different test series, using different run times: 30 seconds, 5 minutes and 30 minutes. Our goal is to evaluate the IBMOLS algorithm's efficiency when using short run times as well as when using long run times. Furthermore, we will evaluate the variation of the optimal parameter values according to the run time available. In order to evaluate the efficiency of the IBMOLS algorithm, we will compare our results with the previous algorithm which was proposed for this problem: SEAMO-R [28]. The population sizes used to evaluate SEAMO-R and IBMOLS are different, since IBMOLS is efficient using small populations and SEAMO-R is efficient using large populations. Then, the population sizes tested correspond to the most efficient possible values.

6.4 Experiments

Table 6: Algorithm comparison: 30 seconds runs.

Algorithms	IBMOLS						SEAMO-R		
RM rate	5%			20%			-		
Pop size	10	20	30	10	20	30	50	100	200
March2001	**0.239**	**0.268**	0.293	**0.266**	0.300	**0.272**	0.558	0.537	0.770
April2001	**0.257**	**0.272**	0.324	0.304	**0.278**	**0.284**	0.644	0.596	0.829
May2001	**0.257**	**0.274**	**0.275**	**0.276**	0.316	**0.258**	0.683	0.580	0.777
June2001	0.529	0.519	0.426	0.483	0.565	0.424	0.427	**0.203**	**0.246**
July2001	0.346	0.304	**0.260**	**0.278**	0.313	**0.245**	0.525	0.393	0.594
August2001	**0.287**	**0.285**	0.373	**0.284**	**0.319**	0.322	0.691	0.655	0.947
September2001	0.392	**0.334**	**0.326**	**0.354**	**0.336**	**0.310**	0.612	0.498	0.723
Average	0.330	0.322	0.325	0.321	0.347	0.302	0.591	0.495	0.698

Table 7: Algorithm comparison: 5 minutes runs.

Algorithms	IBMOLS						SEAMO-R		
RM rate	5%			20%			-		
Pop size	10	20	30	10	20	30	50	100	200
March2001	**0.175**	**0.172**	0.226	0.288	0.239	0.203	0.694	0.504	0.449
April2001	**0.137**	0.199	**0.169**	0.281	0.230	0.169	0.721	0.532	0.439
May2001	**0.136**	0.168	**0.144**	0.235	0.221	0.204	0.686	0.484	0.369
June2001	0.263	0.352	0.338	0.273	0.286	0.282	0.526	0.182	**0.095**
July2001	**0.148**	0.198	**0.166**	0.191	**0.182**	**0.179**	0.494	0.284	0.221
August2001	**0.162**	**0.185**	**0.186**	0.221	**0.196**	**0.202**	0.743	0.530	0.419
September2001	0.187	0.220	0.232	0.224	0.220	0.246	0.742	0.466	0.306
Average	**0.173**	**0.213**	**0.209**	**0.245**	**0.225**	0.212	0.658	0.426	0.328

Table 8: Algorithm comparison: 30 minutes runs.

Algorithms	IBMOLS						SEAMO-R
RM rate	5%			20%			-
Pop size	10	20	30	10	20	30	300
March2001	**0.191**	0.259	0.249	0.304	0.267	0.249	0.609
April2001	0.206	**0.199**	**0.172**	0.265	0.260	0.254	0.718
May2001	0.226	0.255	**0.165**	0.363	0.300	0.315	0.584
June2001	**0.270**	0.298	0.358	**0.183**	**0.250**	**0.279**	0.246
July2001	0.261	0.255	0.254	**0.153**	**0.143**	0.172	0.333
August2001	0.247	0.287	**0.192**	0.314	0.330	0.312	0.672
September2001	**0.171**	**0.196**	0.239	0.231	0.267	0.251	0.432
Average	0.225	0.250	0.233	0.259	0.260	0.262	0.513

Results obtained with runs of 30 seconds, 5 minutes and 30 minutes are respectively described in tables 6, 7 and 8. For each problem, we have tested different combinations of population sizes (10, 20 and 30) and *RM* rates (5% and 20%).

The comparison of the different IBMOLS approaches against SEAMO-R allows us to conclude that IBMOLS approaches are statistically more efficient than the SEAMO-R approach, except on the $June2001$ instance. SEAMO-R performs slightly better on this instance, but the difference tends to be reduced when the running time increases. In particular, SEAMO-R is statistically outperformed on the $June2001$ instance on runs of 30 minutes. On the remaining problem instances, SEAMO-R is statistically outperformed, on runs of 30 seconds, 5 minutes and 30 minutes.

Now let us discuss about results obtained using different parameter values for the IBMOLS algorithm. No clear conclusion can be extracted from tables 6, 7 and 8. However, we can observe two tendencies from these tables. Firstly, many of the IBMOLS algorithm versions are incomparable on runs of 30 seconds, but when the run time increases, the number of statistically incomparable versions is reduced. Secondly, the IBMOLS version using the smallest population size and the smallest mutation level obtains the best average results on 5 and 30 minutes runs. This observation confirms our conclusion introduced in the previous experiments, i.e. the IBMOLS algorithm is more efficient using a small population size and a small mutation rate.

7 Conclusion and perspectives

In this paper, we presented a new and generic multi-objective metaheuristic using the binary quality indicator concept. We proposed the use of a binary indicator within an iterated local search algorithm. The algorithm combines a recent, popular and efficient mechanism proposed for MOEAs and the iterated local search principle, which is known to perform well on real world applications.

One advantage of the indicator-based search is its high level of generality, mainly due to the small number of parameters that are required. We have designed the IBMOLS algorithm in order to propose a methodology which is as generic as possible. We have performed a wide range of experiments, using different parameter values, to evaluate its level of generality and to find guidelines on how to set the small num-

ber of parameters needed for the IBMOLS algorithm. Furthermore, we perform some experiments on other methods in order to evaluate the effectiveness of our algorithm.

We have evaluated the IBMOLS algorithm by applying it to three different combinatorial problems. This enables us to make the following observations:

- The IBMOLS algorithm is highly generic: it has been easily and successfully applied on very different combinatorial problems. However, it does not facilitate the avoidance of a specific study of the considered problem before addressing it, especially with regard to the neighborhood structure being used in the algorithm. For instance, the neighborhood structure used for the nurse scheduling problem is very important, and the one used in this paper (based on nurse schedule mating) has a major influence on the results. The use of classical neighborhood operators, such as *swap* or *insert* operators lead to disappointing results on this problem.

- The comparison with some other algorithms from the literature allows us to say that the IBMOLS algorithm is efficient on different problems. IBMOLS outperforms some classical state-of-the-art multi-objective evolutionary algorithm, even if some exceptions exist.

- Among the binary indicators tested in this paper, we advice the use of I_ϵ which outperforms the other indicators in many cases. However, many other indicators could be defined and could outperform I_ϵ.

- The IBMOLS algorithm is more efficient using a small population size. In many cases, the best results are achieved using a population of less than 10 individuals. However, if the search space is large or if the run time available is large, we suggest the increase of this size to several tens of individuals. Following this guideline, the search methodology should obtain good results in many cases. However, there are some exceptions such as those encountered in our experiments. Then, a good solution should be to start the search with small-size population, then after several local searches the population size can be defined adaptively during the search. Such principle has been already proposed for single objective optimisation [2].

- The initialisation of the local search populations is an important parameter of the IBMOLS algorithm. In our experiments, the method of applying random mutations to some non-dominated solutions from the archive is highly efficient. The efficiency of this initialisation method depends on the amount of random mutations to be applied. We observe that the same general tendencies when observing population size i.e. good results can be obtained with a small number of random mutations, except on large-size problems and with a large computation time. However, these results are not very clear, and we strongly suggest to adapt the mutation rate during the search in order to find a high quality rate, also because the most-suited parameter value probably changes during the algorithm execution.

The binary-indicator search principle has been successfully proposed for evolutionary algorithms in [41] and for local search in this paper. This general principle could

be adapted to every type of (meta)heuristic search, such as ant colony optimisation and Tabu search [1]. Once we have different indicator-based search strategies, it will be interesting to propose an adaptive version of IBMOLS, which will be efficient on a new problem without preliminary studies. To achieve this goal, the exploration of hyper-heuristics [35] could also lead to significant results. This will help us to search for the most appropriate population size or initialisation function dynamically during the search, by evolving different indicator-based searches. With such an approach, we could obtain a modified IBMOLS algorithm which will be applicable across a large range of multi-objective problems.

Another perspective is to explore the possible definition and evaluation of other indicators. In particular, it should be interesting to use the unary hypervolume indicator within IBMOLS algorithm, since this is the most commonly accepted performance indicator in the community. However, this would not be an easy task since the enclosed hypervolume computation is an #P-hard problem according to the number of objectives.

References

[1] E. Aarts, J. Korst, and W. Michiels. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 7: Simulated annealing. Springer, 2005. ISBN: 978-0-387-23460-1.

[2] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Congress on Evolutionary Computation (CEC 2005)*, volume 2, pages 1769–1776, 2005.

[3] M. Basseur. *Design of cooperative algorithms for multi-objective optimization: Application to the Flow-shop scheduling problem.* PhD thesis, University of Sciences and Technology of Lille, France, June 2005.

[4] M. Basseur and E. K. Burke. Indicator-based multiobjective local search. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3100–3107, Singapore, September 2007.

[5] M. Basseur, F. Seynhaeve, and E.-G. Talbi. Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In *IEEE Congress on Evolutionary Computation (CEC 2002)*, volume 2, pages 1151–1156, Honolulu, USA, 2002.

[6] M. Basseur and E. Zitzler. Handling uncertainty in indicator-based multiobjective optimization. *International Journal of Computational Intelligence Research*, 2(3):255–272, 2006.

[7] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. *Soft Computing in Engineering Design and Manufacturing*, 5:231–340, 1997.

[8] E. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

[9] S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.

[10] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems - a bibliographic survey. *European Journal of Operational Research*, 151:447–460, 2003.

[11] K. Deb. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 10: Multi-objective optimization, pages 273–316. Springer, 2006. ISBN: 978-0-387-23460-1.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transaction on Evolutionary Computation*, 6(2):181–197, 2002.

[13] J. Du and J. Y.-T. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of operations research*, 15:483–495, 1990.

[14] M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *TOP (Trabajos de Investigacion Operativa)*, 12(1):1–63, June 2004.

[15] M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In *3rd international conference on evolutionary multi-criterion optimization (EMO 2005)*, volume 3410 of *Lecture Note in Computer Science*, pages 62–76. Springer, 2005.

[16] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In *Fifth International Conference on Genetic Algorithms (ICGA'93)*, pages 416–423, San Mateo, USA, 1993.

[17] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[18] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[19] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In *1st international conference on evolutionary multi-criterion optimization (EMO 2001)*, volume 1993 of *Lecture Note in Computer Science*, pages 213–225. Springer, 2001.

[20] T. H. Ishibuchi. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans. on Systems, Man., and Cybernetics -Part C: Applications and Reviews*, 28(3):1998, Aug 1998.

[21] M. P. Hansen. Tabu search for multiobjective optimization: MOTS. In *MCDM'97 conference*, Cap town, South Africa, Jan 1997.

[22] Y.-D. Kim. Minimizing total tardiness in permutation flowshops. *European Journal of Operational Research*, 33:541–551, 1995.

[23] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, 2002.

[24] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[25] J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastive multiobjective optimizers. Technical Report TIK-Report No. 214, Computer Engineering and Networks Laboratory, ETH Zurich, July 2005.

[26] M. Labbé, G. Laporte, I. Rodríguez Martín, and J. J. Salazar González. The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, 43:177–189, 2004.

[27] D. Landa-Silva, E. K. Burke, and S. Petrovic. *Metaheuristic for Multiobjective Optimisation*, chapter 4: An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling, pages 91–129. Springer, 2004.

[28] D. Landa-Silva and K. N. Le. *Adaptive and Multilevel Metaheuristics*, volume 136, chapter 7: A Simple Evolutionary Algorithm with Self-Adaptation for Multi-Objective Nurse Scheduling, pages 133–155. Springer-Verlag, 2008.

[29] J. K. Lenstra, A. H. G. R. Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[30] A. Liefooghe, S. Mesmoudi, J. Humeau, L. Jourdan, and E.-G. Talbi. A study on dominance-based local search approaches for multiobjective combinatorial optimization. In *Second International Workshop on Engineering Stochastic Local Search Algorithms (SLS 2009)*, volume 5752 of *Lecture Notes in Computer Science*, pages 120–124, Brussels, Belgium, 2009.

[31] T. Murata, H. Nozawa, H. Ishibuchi, and M. Gen. Modification of local search directions for non-dominated solutions in cellular multiobjective genetic algorithms for pattern classification problems. In *2nd international conference on evolutionary multi-criterion optimization (EMO 2003)*, volume 2632 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2003.

[32] A. Nagar, J. Haddock, and S. Heragu. Multiple and bicriteria scheduling: A litterature survey. *European Journal of Operational Research*, 81:88–104, 1995.

[33] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic algorithms and their application*, pages 224–230, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[34] L. Paquete and T. Stützle. A study of local search algorithms for the biobjective QAP with correlated flow matrices. *European Journal of Operational Research*, 169(3):943–959, 2006.

[35] P. Ross. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 17: Hyper-Heuristics, pages 527–556. Springer, 2006. ISBN: 978-0-387-23460-1.

[36] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[37] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operations Research*, 64:278–285, 1993.

[38] E. G. Talbi, M. Rahoual, M. H. Mabed, and C. Dhaenens. A hybrid evolutionary approach for multicriteria optimization problems : Application to the flow shop. In *Evolutionary Multi-Criterion Optimization (EMO'01)*, volume 1993 of *Lecture Notes in Computer Science*, pages 416–428, 2001.

[39] K. C. Tan, T. H. Lee, and E. F. Khor. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *Evolutionary Computation*, 5(6):565–588, 2001.

[40] A. Viana, J. Pinho de Sousa, and M. A. Matos. Multiobjective constraint oriented neighbourhoods. In *6th Metaheuristics International Conference (MIC 2005)*, pages 896–903, Vienna, Austria, 2005.

[41] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, Birmingham, UK, September 2004.

[42] E. Zitzler, M. Laumanns, and S. Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. In X. Gandibleux et al., editors, *Metaheuristics for Multiobjective Optimisation*, Lecture Notes in Economics and Mathematical Systems, pages 3–37. Springer, 2004.

[43] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[44] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation*, 3:257–271, 1999.

[45] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. on Evolutionary Computation*, 7(2):117–132, 2003.