UNIVERSITÀ POLITECNICA DELLE MARCHE
Repository ISTITUZIONALE

Data-driven predictive maintenance policy based on multi-objective optimization approaches for the component repairing problem

This is the peer reviewd version of the followng article:

Original
Data-driven predictive maintenance policy based on multi-objective optimization approaches for the component repairing problem / Pisacane, Ornella; Potena, Domenico; Antomarioni, Sara; Bevilacqua, Maurizio; Emanuele Ciarapica, Filippo; Diamantini, Claudia. - In: ENGINEERING OPTIMIZATION. - ISSN 0305-215X. - 53:10(2021), pp. 1752-1771. [10.1080/0305215X.2020.1823381]

Availability:
This version is available at: 11566/284711 since: 2024-04-19T13:49:13Z

Publisher:

Published
DOI:10.1080/0305215X.2020.1823381

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. The use of copyrighted works requires the consent of the rights' holder (author or publisher). Works made available under a Creative Commons license or a Publisher's custom-made license can be used according to the terms and conditions contained therein. See editor's website for further information and terms and conditions.
This item was downloaded from IRIS Università Politecnica delle Marche (https://iris.univpm.it). When citing, please refer to the published version.

(Article begins on next page)

12 October 2024

Data-Driven Predictive Maintenance Policy based on Multi-objective Optimization Approaches for the Component Repairing Problem

Ornella Pisacane[a], Domenico Potena[a], Sara Antomarioni[b], Maurizio Bevilacqua[b], Filippo Emanuele Ciarapica[b] and Claudia Diamantini[a]

[a]Dipartimento di Ingegneria dell'Informazione, Universitá Politecnica delle Marche, Italy; [b]Dipartimento Ingegneria Industriale e Scienze Matematiche, Universitá Politecnica delle Marche, Italy

ARTICLE HISTORY
Compiled September 9, 2020

Abstract
In systems with many components that are required to be constantly active, like refinery, predicting the components that will break in a time interval after a stoppage may significantly increase their reliability. However, predicting the set of components to be repaired is a challenging task, especially when several conditions (e.g., breakage probability, repair time and cost) have to be considered simultaneously. A data-driven predictive maintenance policy is proposed for maximizing the system reliability and minimizing the maximum repair time, considering both budget and human resources constraints. Therefore, a data-driven algorithm is designed for extracting component breakage probabilities. Then, two bi-objective optimization approaches are proposed for determining the set of components to repair. The former is based on the formulation of a bi-objective MILP model solved through the AUGMECON method. The latter implements a bi-objective Large Neighborhood Search, outperforming the first approach.

KEYWORDS
Augmented ϵ-Constraint; Large Neighborhood Search; Predictive Maintenance; Mathematical Programming

1. Introduction

In the context of industrial plants, achieving high reliability is fundamental and can be pursued by implementing effective maintenance policies, even considering its significant impact on companies' competitiveness and cost-effectiveness (Peng, Dong, and Zuo 2010). In industrial sectors characterized by high operational risk, the occurrence of components' breakage is particularly critical since it may affect not only the plant's operation but also the integrity of the environment and people safety (Bevilacqua and Ciarapica 2018). As reported by the European Committee for Standardization in EN 13306:2017 (EN 2017), the following approaches should be taken into account in defining the most appropriate maintenance policy for an industrial environment:

- *corrective maintenance*: an intervention is carried out after a breakage occurrence, in order to restore the normal system functioning;

CONTACT D. Potena. Email: d.potena@univpm.it

- *preventive maintenance*: maintenance is carried out at predefined intervals or conditions;
- *predictive maintenance*: maintenance is carried out according to significant characteristics like the breakage forecast, suggested by the estimations of the degradation state, or the component breakage probability.

Indeed, predictive maintenance policies offer the opportunity of anticipating components' breakage through the analysis of historical data and the implementation of proper algorithms. Specifically, the increasing data availability, enabled by Industry 4.0 infrastructures, facilitates the development of data-driven algorithms for assets' availability and reliability maximization that allow a continuous process monitoring (Kumar, Chinnam, and Tseng 2019). Selecting the optimal set of components to be predictively repaired (namely, solving a Component Repairing Problem (CRP)) represents a challenging goal when several conditions (e.g., breakage probability, repair time, repair cost) have to be taken into account. This work is specifically focused on process industries whose plants are characterized by a high number of components and are constantly active (e.g. oil refinery, steel mill, highly automated plants). The probability of having a plant stoppage is high due to the presence of several components that can (also simultaneously) break. Furthermore, the component breakage can have serious effects, even leading to a plant stoppage to carry out the repair. Since the plant works constantly, both a stoppage and the subsequent restart (in particular, the transient before returning to regime) impact on the components, reducing their life time and increasing the probability that they may break after the restart (Antomarioni et al. 2019a); in turn, increasing the probability that a stoppage occurs.

Moreover, in the considered scenario, the status of the components cannot be often monitored due to a lack of sensors or because such sensors would be too complex to install or too expensive (especially when the installation requires stopping the production plant).

Therefore, the goal of this work is to define a data-driven approach, based on the available data related to past failures, to predict the components that will break in a time interval after a stoppage occurs. Preventively repairing these components during the stoppage period increases the reliability of the whole plant. Components to be repaired must be carefully chosen in order to reduce the impact on both the time to recover from a stoppage and the overall maintenance costs (e.g., the maintainer hourly cost and the component repair costs).

The main contributions of this work consist in:

- designing a new data-driven predictive maintenance policy, considering conflicting criteria: the plant reliability has to be maximized, predictively maintaining the components with highest breakage probability and the maintenance interventions has to be as short as possible, selecting for maintenance the components with shortest repair times;
- formulating a bi-objective Mixed Integer Linear Programming (MILP) model for the bi-objective CRP (b-CRP), i.e., selecting the best set of components to repair, maximizing the plant realiability and minimizing the maximum repair time, simultaneously, under budget and human resources constraints;
- solving the proposed model through the Augmented ϵ-constraint (AUGMECON) approach;
- designing a bi-objective Large Neighborhood Search (b-LNS) meta-heuristic for efficiently addressing medium and large sized instances;
- carrying out an experimental campaign on real-life alike case studies, inspired

by an oil refinery plant;
- performing a data-driven analysis on the effectiveness of the b-LNS moves.

The rest of the paper is structured as follows: Section 2 presents a literature review concerning recent trends on multi-objective optimization based maintenance approaches. Section 3 describes the problem statement and the notation used, while the bi-objective MILP model formulated for the b-CRP is detailed in Section 4. Sections 5 and 6 outline the AUGMECON and the b-LNS, respectively. In Section 7, the experimental results obtained on real-life alike instances taken from a real oil refinery are discussed, introducing some significant performance metrics. Useful insights are also derived on the effectiveness of the b-LNS moves through a data-driven analysis. Finally, Section 8 concludes the work and highlights future research directions worth of investigation.

2. Literature Review

This section describes the main literature contributions proposed in the maintenance field, with focus on multi-objective mathematical programming models and solution methods. The readers are referred to Garg and Deshmukh (2006) for a comprehensive literature review on maintenance management.

Maintenance is usually a very time consuming activity from the production objectives point of view, since it typically requires a system stoppage. However, delaying maintenance interventions to avoid interruptions of the production flow may significantly increase the failure probability (Ruiz, García-Díaz, and Maroto 2007). Hence, due to these contrasting criteria, several literature contributions apply multi-objective optimization techniques for designing effective maintenance policies. For example, Ruiz, García-Díaz, and Maroto (2007) propose Ant Colony Optimization (ACO) and Genetic Algorithms (GAs) for a preventive machine maintenance, minimizing the completion time of the last job in the production schedule. Marseguerra, Zio, and Podofillini (2002) propose a GA for a condition based maintenance policy, determining the optimal degradation level for a preventive maintenance policy, maximizing the profit and the availability simultaneously. They describe the model predicting the evolution of the degrading system through Monte Carlo simulation (MCS). In Kumar, Jain, and Gandhi (2018), a predictive tool aimed at deciding the optimum condition-based maintenance policy is designed. Specifically, the authors propose a semi-Markov process in order to both model the steady state availability analysis of mechanical systems and evaluate the optimal condition monitoring interval. This interval is then used for maximizing the system availability through a GA approach. The problem of the best maintenance inspection interval identification is tackled by Marseguerra, Zio, and Podofillini (2004) considering both the maximum system reliability and the minimum variance of the model parameters, proposing a multi-objective GA. A similar approach is proposed in Huang, Qu, and Zuo (2009), together with a mathematical programming model for maximizing the system reliability and minimizing its costs, simultaneously. Okasha and Frangopol (2009) formulate the problem of selecting the best maintenance actions by mathematical programming, maximizing the system reliability, minimizing its redundancy and the life-cycle cost. A Non-dominated Sorting GA II (NSGA-II) is also designed.

Mathematical programming is used in Min et al. (2009) for defining the best preventive maintenance policy both minimizing maintenance costs and maximizing the

reliability of a high-speed railway power system. A Chaos Self-Adaptive Evolutionary Algorithm (CSEA) is also proposed. Considering the same objectives, Raad, Sinske, and Van Vuuren (2009) define a maintenance policy for water distribution systems, proposing a Genetically Adaptive MultiMethod Search (AMALGAM) outperforming existing approaches like NSGA-II. Similarly, Berrichi et al. (2010) design an ACO algorithm for both maximizing the production system availability and minimizing the makespan. Carlos et al. (2012) address a maintenance problem in a nuclear power plant, solved by a Particle Swarm Optimization (PSO) algorithm. In the same application context, Gjorgiev, Kančev, and Čepin (2012) compare four different versions of a GA (weight-based classical GA, weight-based steady state GA, weighted sum GA and NSGA-II). Tian, Lin, and Wu (2012) use the Physical Programming (PP) for simultaneously maximizing the reliability and minimizing the maintenance costs in condition based maintenance. In Loganathan and Gandhi (2016), a PSO algorithm under reliability constraints for minimizing the maintenance cost is designed.

Moghaddam and Usher (2011) formulate a multi-objective optimization model to determine the optimal preventive maintenance and replacement schedules in a multi-component system. Both a generational GA and a Simulated Annealing (SA) algorithm are also designed. A preventive maintenance optimization based approach is also proposed by Moghaddam (2013) with the aim of determining the optimal maintenance schedules in production systems. The formulated model is then solved through a procedure that combines MCS and Goal Programming (GP). Ebrahimipour, Najjarbashi, and Sheikhalishahi (2015) design an exact approach, based on Weighted Sum (WS) method, to schedule preventive maintenance achieving minimum cost and maximum reliability. An interactive fuzzy multi-objective linear model for the minimization of the maintenance costs and of the scheduling tardiness is formulated in Seif, Yu, and Rahmanniyay (2018).

Lexicographic Goal Programming (LGP) and ϵ-constraint approaches are applied in Certa et al. (2012) instead, to define the optimal scheduling for components replacement and inspection, minimizing the total cost and the unavailability time. The selection of the most appropriate maintenance policy can also be addressed through the Analytic Hierarchy Process (AHP), since it requires several criteria to be simultaneously evaluated. For example, Bertolini and Bevilacqua (2006) apply AHP for evaluating the maintenance alternative policies (i.e., corrective, preventive and predictive) considering three specific criteria, i.e., the occurrence, the severity, and the detectability. Then, a GP model is formulated for selecting the best maintenance policy for each centrifugal pump under budget and human resources constraints. However, they do not focus on the CRP. AHP can be also combined with GP (Arunraj and Maiti 2010): firstly, AHP is applied to prioritize the possible maintenance policies comparing them in terms of cost and risk; then, the selection of the best maintenance policy is performed through GP.

Moghaddam (2015) compare the performances of five GAs, for optimizing the operational costs and the overall reliability of a Computer Numerical Control (CNC) machine. Fan and Xia (2017) apply a GA instead, to solve a multi-objective optimization problem related to an energy-efficiency building envelope retrofitting plan, for maximizing the energy savings and the net present value of the investment, while minimizing its payback period. A maintenance plan in building retrofit is addressed in the multi-objective model proposed by Wu, Xia, and Wang (2015), solved through Multi-Objective Neighborhood Field Optimization (MONFO) where the retrofit cost, the energy saving and the net present value are optimized simultaneously.

Recently, imperfect maintenance policies have been also proposed. For example, in

Table 1. Main literature contributions on optimization approaches for system maintenance. Papers have been classified on the basis of maintenance policy proposed: predictive (PM), data-driven (DD) and multi-objective (MO).

Paper	**PM**	**DD**	**MO**	**Objectives**	**Approaches**
Marseguerra et al. (2002)			✓	Profit, availability	GA
Marseguerra et al. (2004)			✓	Reliability, Failure Uncertanity	GA
Ruiz et al. (2007)				Makespan	ACO, GAs
Huang et al. (2009)			✓	Reliability, Cost	GA
Min et al. (2009)			✓	Reliability, Cost	CSEA
Okasha & Frangopol (2009)			✓	Reliability, Cost	GA
Raad et al. (2009)			✓	Reliability, Cost	AMALGAM
Arunraj and Maiti (2010)			✓	Cost, Risk	AHP-GP
Berrichi et al. (2010)			✓	Unavailability, Makespan	ACO
Moghaddam & Usher (2011)			✓	Reliability, Cost	GA, SA
Carlos et al. (2012)			✓	Unavailability, Cost	PSO
Certa et al. (2012)			✓	Unavailability, Cost	LGP ϵ-constraint
Gjorgiev et al. (2012)			✓	Unavailability, Ageing, Cost, Uncertainty	GAs
Tian et al. (2012)			✓	Reliability, Cost	PP
Moghaddam (2013)			✓	Costs, Reliability, Availability	GP+MCS
Ebrahimipour et al. (2015)			✓	Reliability, Cost	WS
Moghaddam (2015)			✓	Reliability, Cost	GAs
Wu et al. (2015)			✓	Retrofit cost, Energy saving, Net present value	MONFO
Fan and Xia (2017)			✓	Energy saving, Payback period, Net present value	GA
Seif et al. (2018)			✓	Cost, Tardiness	GA
Su and Liu (2019)			✓	Availability, Cost rate	NSGA-II
Antomarioni et al. (2019b)	✓	✓		Breakages Probability	ILP
This work	✓	✓	✓	**Reliability** **Max Repair Time**	**AUGMECON** **b-LNS**

Su and Liu (2019), a NSGA-II is applied to solve a multi-objective imperfect preventive maintenance optimization problem in the context of the electromechanical products. It is worth noting that, in the present work, attention is focused only on significant failures and therefore, the possibility to have also imperfect repairs is not taken into account. Indeed, the domain of the present work is such that the cost of a plant stoppage is so high that replacing components or repairing them as-good-as-new is preferable.

Table 1 summarizes the main literature contributions. Specifically, for each paper, it is shown whether the maintenance policy proposed is predictive (PM) and/or data-driven (DD) and/or multi-objective (MO). The third column specifies the objective/s considered for optimization whereas the last column reports the approaches used. It is worth noting that only Antomarioni et al. (2019b) proposing optimization approaches for system maintenance actually shows predictive features. In fact, they propose a predictive maintenance approach where the CRP is mathematically formulated through Integer Linear Programming (ILP) but for only minimizing the breakages' probability (single-objective), under the limiting assumption that a breakage of a specific component occurred.

Table 2. Notation used

Set	**Meaning**
C	Set of components
E_S	Ordered set of stoppage events
E_b	Ordered set of component breakage events
Parameter	**Meaning**
bp_{c_j}	Breakage probability of component c_j
rc_{c_j}	Repair cost of component c_j
rt_{c_j}	Repair time of component c_j
n_{c_j}	Number of operators required for repairing component c_j
ΔL_{c_j}	Life span of component c_j
$\overline{\Delta F}_{c_j}$	Mean time between failure of component c_j
T_{max}	Maximum time allowed for maintenance planning
B	Maximum budget allowed for maintenance planning
C_{work}	A fixed hourly cost for employing an operator
ΔT	Time interval of observation

To the best of our knowledge, this work is the first contribution in which a data-driven predictive maintenance policy is proposed and the CRP is modelled through multi-objective mathematical programming for maximizing the system reliability and minimizing the maximum repair time, simultaneously.

3. Problem Statement and Notation

This section introduces both the problem statement and the notation used, the latter also summarized in Table 2.

The b-CRP aims at finding the optimal set of components to repair for simultaneously maximizing the overall system reliability and minimizing the maximum repair time required, under constraints on both the total budget B and the total repair time T_{max}.

The set C contains the components which the optimal set has to be selected from. For each $c_j \in C$, a repair cost rc_{c_j}, a repair time rt_{c_j} and the number of maintainers n_{c_j} required for repairing it are given.

The parameter C_{work} denotes the fixed hourly cost of employing a maintainer whereas bp_{c_j} is the breakage probability of the j-th component in a given time interval, ΔT, of observation, after a system stoppage. The procedure followed for estimating the breakage probability bp_{c_j} of each $c_j \in C$ is detailed in Algorithm 1.

Since the phenomenon under investigation is binary (component failure/working), the algorithm implements a maximum likelihood estimation to derive the set of bp_{c_j}. This approach returns good probability estimations when working conditions do not change considerably over time and a lot of data on past breakages is available. These two assumptions hold in the scenario considered in this work: complex and constantly active process industries (e.g. oil refinery, steel mill, highly automated plants).

The first part of the algorithm concerns the data pre-processing aimed at estimating the set of breakage probabilities bp_{c_j} from input data. To this end, the algorithm combines information regarding component breakages with system stoppages occurred in the past. In detail, the algorithm scrolls the list E_b of k component breakage events (step 5), computing how many times each component breaks within a time window

Algorithm 1 The data-driven optimization-based approach

Input:
Set of components C;
Component's life span ΔL_{c_j} and component MTBF $\overline{\Delta F}_{c_j}$, $\forall c_j \in C$;
Time interval ΔT;
List of stoppage events $E_S = \{\langle t'_{S_1}, t''_{S_1}\rangle \langle t'_{S_2}, t''_{S_2}\rangle \ldots, \langle t'_{S_m}, t''_{S_m}\rangle\}$, where:
t'_{S_i} is the timestamp when the stoppage S_i occurs;
t''_{S_i} is the timestamp when the plant restarts after a stoppage;
such that $t'_{S_i} < t''_{S_i},\ i = 1, \ldots, m$ and $t''_{S_i} < t'_{S_{i+1}},\ i = 1, \ldots, m-1$;
List of component breakage events $E_b = \{\langle \gamma_1, t_{\gamma_1}\rangle, \langle \gamma_2, t_{\gamma_2}\rangle, \ldots, \langle \gamma_k, t_{\gamma_k}\rangle\}$, where
t_{γ_σ} is the timestamp which the component $\gamma_\sigma \in C$ breakage occurs at, and
$t_{\gamma_j} < t_{\gamma_{j+1}},\ j = 1, \ldots, k-1$.
Output: Set of components to repair $\bar{C} \subseteq C$

$bp_{c_j} := 0, \forall c_j \in C$;
$i := 1$;
$t_{start} = t''_{S_i}$;
$t_{end} = t'_{S_{i+1}}$;
for $\sigma := 1$ to k **do**
  if $t_{\gamma_\sigma} > t_{start}$ **then**
    if $(t_{\gamma_\sigma} \leq t_{end} \wedge t_{\gamma_\sigma} \leq (t_{start} + \Delta T))$ **then**
      $bp_{\gamma_\sigma} := bp_{\gamma_\sigma} + \frac{1}{m}$
    else
      if $i < m - 1$ **then**
        $i := i + 1$;
        $t_{start} = t''_{S_i}$;
        $t_{end} = t'_{S_{i+1}}$;
      end if
    end if
  end if
end for
for $j := 1$ to $|C|$ **do**
  if $\Delta L_{c_j} < \overline{\Delta F}_{c_j}$ **then**
    $bp_{c_j} = \left(1 - \frac{\overline{\Delta F}_{c_j} - \Delta L_{c_j}}{\overline{\Delta F}_{c_j}}\right) bp_{c_j}$
  end if
end for
$\bar{C}$:=**selectBestComponents**$(C, T_{max}, B, bp_c, \forall j \in C)$

of length ΔT starting from the restart of the system after a stoppage (steps 7-9). If a breakage event falls outside the time window, the next stoppage is used to define a new time window (steps 10-14).

In order to take into account the life span of the component c_j, namely the time since the last replacement or repair of c_j, its breakage probability is adjusted based on the *Mean Time Between Failure* (MTBF) of c_j. If the component has been recently repaired, that is the component life span is less than a given percentage of its MTBF (i.e., $\overline{\Delta F}_{c_j}$), then the breakage probability will be decreased proportionally to the difference between its life span and $\overline{\Delta F}_{c_j}$ (step 20). If its life span is longer than its MTBF, then bp_{c_j} can be set to a very large value to increase the chances of the component being selected for repair; or one can force its replacement, thus setting its breakage probability to zero in order to prevent it being selected.

Finally, step 23 (i.e., routine *selectBestComponents*) aims at selecting the sub-set of components $\bar{C} \subseteq C$ that is more convenient to repair with regards to the minimization of two criteria, under budget and time constraints. More specifically, $\bar{C}$ represents the set of components with a total repair cost and time, both not exceeding their maximum availability (B and T_{max} respectively), minimizing the total breakage probability and the maximum repair time, simultaneously. To this purpose, one of the two solution approaches proposed, respectively, in Sections 4-5 and in Section 6 is invoked.

All parameters in Table 2 are derived from data on past failure events (E_S, E_b, ΔL_{c_j}, ΔT), are extracted from data on components characteristics (rc_{c_j}, rt_{c_j}, n_{c_j}, $\overline{\Delta F}_{c_j}$), are constants provided by domain experts (T_{max}, B, C_{work}) or are estimated by Algorithm 1 (bp_{c_j}). It is worth noting that the value of ΔT should not be chosen too small otherwise only a few breakages would be considered and the maximum likelihood estimation criterion implemented in Algorithm 1 would produce inaccurate results. Similarly, ΔT should not be too wide, otherwise the breakages would be poorly correlated to the stoppage of the plant, still returning inaccurate results. For these reasons, the value of ΔT is set on the basis of past knowledge on plant stoppages. In particular, ΔT is set to the average time between two consecutive stoppages.

4. A Multi-objective Mixed Integer Linear Programming model for the Component Repairing Problem

This section describes the bi-objective Mixed Integer Non-Linear Programming formulation, aimed at selecting the sub-set $\tilde{C} \subseteq C$ of components, optimizing two criteria, simultaneously. The former, to minimize, denotes the total breakage probability of the non-selected components (i.e., the maximization of the total breakage probability of the selected components and thus, of the system reliability); the latter, to minimize, represents the maximum repair time (i.e., the makespan). It is worth remarking that all the costs involved as well as the budget and the maximum time available for repairs are those known at the moment of the system stoppage and therefore, they are deterministic in the proposed approach.

Then, the following decision variable x_{c_j} is introduced, $\forall c_j \in C$, equal to 1 if the component c_j is selected for being repaired, 0 otherwise.

$$\min BP = \sum_{c_j \in C} (1 - x_{c_j}) bp_{c_j} \tag{1}$$

$$\min RT^{max} = \max_{c_j \in C} rt_{c_j} x_{c_j} \tag{2}$$

s.t.

$$C_{work} \sum_{c_j \in C} n_{c_j} x_{c_j} rt_{c_j} + \sum_{c_j \in C} rc_{c_j} x_{c_j} \leq B \tag{3}$$

$$\sum_{c_j \in C} rt_{c_j} x_{c_j} \leq T_{max} \tag{4}$$

$$x_{c_j} \in \{0; 1\} \quad \forall c_j \in C \tag{5}$$

The objective function BP (1) to minimize represents the total breakage probability associated with the non-selected components whereas RT^{max} (2) to minimize denotes the maximum maintenance time, i.e., the maximum time devoted to repair a selected component. In fact, since the present work does not consider the scheduling of the maintenance activities but only the planning, the companies may be very interested in having a maintenance plan allowing them to save time.

The total repair cost has not to exceed B (3). It is worth noting that the total cost is due to two parts. The former considers the effective time cost of the maintainers used, known the time required by each component to be repaired. The latter refers to the repair cost of each component that takes into account also the material used for repairing. It is worth remarking that the costs do not include those due to the components shortage. In fact, the predictive maintenance activities, especially in complex systems, are usually performed in a medium/long term and therefore, a shortage of the components to be repaired is highly unlikely. These facts have been confirmed by the company which the problem and the data were derived from. The total repair time does not exceed T_{max} (4). Finally, the binary nature of the decision variables are defined in (5).

The proposed model is linearized by introducing an additional continuous non-negative variable y equal to $\max_{c_j \in C} rt_{c_j} x_{c_j}$ and the related additional constraint (6):

$$y \geq rt_{c_j} x_{c_j} \quad \forall c_j \in C \tag{6}$$

The value of RT^{max} is normalized by dividing the value of the variable y by T_{max}. Therefore, the Mixed Integer Linear Programming (MILP) formulation proposed for the b-CRP is in the following:

$$\min BP = \sum_{c_j \in C} (1 - x_{c_j}) bp_{c_j} \tag{7}$$

$$\min RT^{max} = \frac{y}{T_{max}} \tag{8}$$

s.t.

$$C_{work} \sum_{c_j \in C} n_{c_j} x_{c_j} rt_{c_j} + \sum_{c_j \in C} rc_{c_j} x_{c_j} \leq B \tag{9}$$

$$\sum_{c_j \in C} rt_{c_j} x_{c_j} \leq T_{max} \tag{10}$$

$$y \geq rt_{c_j} x_{c_j} \quad \forall c_j \in C \tag{11}$$

$$x_{c_j} \in \{0; 1\} \quad \forall c_j \in C \tag{12}$$

5. The Augmented ϵ-Constraint Approach

This section describes the AUGMEnted $\epsilon-$CONstraint (AUGMECON) approach, introduced by Mavrotas (2009), for solving the bi-objective MILP model (7), (8)–(12). Indeed, this approach has been already applied successfully for solving several other decision problems (e.g., Yu and Solvang (2016))

In a bi-objective optimization problem, the objective function $f(x)$, supposed to minimize, can be expressed through a bi-dimensional vector $z = f(x) = (z_1 = f_1(x), z_2 = f_2(x))$, being the $n-$dimensional vector x a feasible solution in the feasible region $X \subseteq \mathbb{R}^n$. Therefore, the following two definitions hold:

Definition 1 ***Dominance condition***:
A solution $x^{'}$ with $(z_1^{'}, z_2^{'})$ dominates a solution $x^{''}$ with$(z_1^{''}, z_2^{''})$ if and only if $z_1^{'} \leq z_1^{''}$ and $z_2^{'} \leq z_2^{''}$ and at least one inequality is strictly satisfied.

Definition 2 ***Pareto Efficiency***:
A solution $x \in X$ is Pareto Efficient if and only if $\nexists x^{'} \in X$ that dominates it.

The Pareto Front contains all the Pareto Efficient solutions. The methods proposed for solving multi-objective optimization problems can be classified into three classes: *a priori*, *interactive* and *a posteriori* (Ehrgott and Gandibleux 2000). The *a priori* methods (e.g., GP methods) assume to know all the preferences before starting the decision making process, finding solutions that respect all of them. Instead, in the *interactive* approaches, it is assumed that all the preferences are introduced by the decision maker during the decision making process and therefore, these methods require several interactions with him/her. Finally, in *a posteriori* approaches, all the efficient solutions are firstly generated and then, analyzed according to the decision maker preferences. The *Weighted Sum* and the ϵ*-Constraint method* are the most widely used a posteriori approaches.

Regarding the ϵ-Constraint approach, assuming that two objectives (z_1 and z_2) have

to be minimized, the following optimization problem is defined (13)-(15):

$$min z_1 \tag{13}$$

subject to

$$z_2 \leq \epsilon_2 \tag{14}$$

$$x \in X \tag{15}$$

By properly varying the ϵ_2 parameter, right hand side of the introduced constraint (14), the efficient solutions can be determined. However, one issue is related to how setting the variation range of ϵ_2. One way is to build a square *payoff table*, with a number of columns (rows) equal to that of the objective functions, through *lexicographic* optimization. In case of bi-objective optimization, the first row of such a 2×2 payoff matrix contains z_1^* and $\bar{z_2}$, denoting respectively the optimal value of z_1 when only it is optimized and the optimal value of z_2 when only it is optimized under the constraint $z_1 = z_1^*$. Instead, the second row reports $\bar{z_1}$ and z_2^*, where the latter is the optimal value of z_2 when only it is optimized and the former is the optimal value of z_1 when only it is optimized under the constraint $z_2 = z_2^*$.

In order to avoid generating weakly Pareto efficient solutions, the AUGMECON is applied. First, the payoff table is derived through the lexicographic optimization and then, the variation range of ϵ_2 is determined as $[z_2^*, \bar{z_2}]$. In addition, it is required that constraint (14) has to be binding. Therefore, it is transformed into an equality by adding a non-negative auxiliary variable s. Such a variable is also introduced in (13) with lower priority, multiplied by $\frac{eps}{\delta}$, where *eps* represents a user defined constant and δ is computed as $\delta = \bar{z_2} - z_2^*$ (i.e., it is the width of the variation range). The ϵ_2 parameter is then varied in the range $[z_2^*, \bar{z_2}]$ by a step $\delta_{\epsilon_2} = \frac{\delta}{\alpha}$ where α is a user input value.

It is worth remarking that, in order to avoid the *trivial* solution (i.e., the one in which no component is repaired), the following constraint is added to the formulation (7)–(12):

$$\sum_{c_j \in C} x_{c_j} \geq 1 \tag{16}$$

6. A Multi-objective Large Neighborhood Search

In this section, a bi-objective Large Neighborhood Search (hereafter, denoted as b-LNS) is described to efficiently address both medium and large sized instances. The Large Neighborhood Search (LNS), proposed by Shaw (1998), is a meta-heuristic successfully used in several application contexts. Its main advantage is searching larger and complex neighborhood in order to find better quality solutions (Pisinger and Ropke 2010). Specifically, when the decision problem presents very tight constraints, it could be very easy to get stuck at a local minimum.

The main idea is that, starting from an initial solution (e.g., randomly generated), applying both *destroy* and *repair* moves at each iteration, a better solution can be detected. In order to reduce the computational time, it usually starts with a small-sized

Algorithm 2 b-LNS outline

Input: TL,γ, num_R, num_A;
Output: Set of non-dominated feasible solutions S_{best}

1: $S_{best} := \emptyset$;
2: sol :=**InitialSolution**();
3: $S_{best} := S_{best} \cup \{sol\}$;
4: **while !stop**(TL) **do**
5: RM :=**SelectRandom**(num_R);
6: IM :=**SelectRandom**(num_A);
7: **if** $RM \neq 0 \vee IM \neq 0$ **then**
8: sol :=**Neigh**(RM, IM, sol,γ);
9: **else**
10: IM :=**SelectRandom**($num_A - 1$)+1;
11: sol :=**Neigh**(RM, IM, sol,γ);
12: **end if**
13: **if !Dominated**(sol) **then**
14: $S_{best} := S_{best} \cup \{sol\}$;
15: **end if**
16: sol :=**SelectRandom**($|S_{best}|$);
17: **end while**

neighborhood, gradually increased during the search. To the best of our knowledge, a very few literature contributions already propose a multi-objective LNS (Schaus and Hartert 2013; Oddi, Rasconi, and Cesta 2015). In the proposed b-LNS, designed to solve the b-CRP, the destroy moves aim at removing components from the current solution *sol* (i.e., *remove moves*) whereas the repair ones aim at adding new components in the current solution (i.e., *add moves*). Algorithm 2 outlines the proposed b-LNS.

The b-LNS receives a Time Limit (TL), a user selected parameter γ and two integer number (num_R and num_A) denoting, respectively, the number of remove and add moves implemented. The parameter γ, used only in two of the moves, denotes the percentage of either the components already repaired in *sol* to remove or the components not repaired in *sol* to select in the new solution. It returns the set S_{best} of non-dominated solutions.

The routine *InitialSolution* generates an initial solution in which the components to repair are selected by decreasing repair times; equal, by increasing breakage probability; equal, by decreasing repair cost. In any case, the selection is performed respecting both B and T_{max}. The routine *stop* returns TRUE if TL is reached; FALSE, otherwise.

SelectRandom receives an integer number (i.e., num_R or num_A) and returns an integer number, randomly generated, between 0 and the input number, representing the id of either a remove or an add move to apply (steps $5-6$). In steps $7-12$, a new solution is found by starting from the current one and by applying the moves selected in the previous steps. The routine *Neigh* applies the remove and the add move to *sol*. In particular, the *Remove* moves are detailed in the following. From the set $\bar{C}(sol)$ of components repaired in *sol*:

(1) *RemoveRandom*: randomly removes a component;
(2) *RemoveBP*: removes the component with the lowest breakage probability;
(3) *RemoveTime*:removes the component with the highest repair time;

(4) *RemoveBPT*: removes the component c_j such that:

$$c_j := \underset{c_k \in \bar{C}(sol)}{\text{argmin}} \{bp_{c_k} + (1 - \frac{rt_{c_k}}{T_{max}^{sol}})\} \tag{17}$$

where T_{max}^{sol} represents the maximum repair time over all the repair times of the components in $\bar{C}(sol)$;

(5) *RemoveGamma*: randomly removes γ components.

It worth remarking that more than one component may be individuated by the *RemoveBP*, *RemoveTime* and *RemoveBPT* moves to be potentially removed. This is due to the fact that, especially in the application context of this work, more than one component may satisfy the requirements imposed by each of these moves.

The *Add* moves are described in the following. From the set $C \backslash \bar{C}(sol)$ of components not repaired in *sol*:

(1) *AddRandom*: randomly selects a component;
(2) *AddBP*: adds the component with the highest breakage probability;
(3) *AddTime*: adds the component with the lowest repair time;
(4) *AddBPT*: adds the component c_j such that:

$$c_j := \underset{c_k \in C \backslash \bar{C}(sol)}{\text{argmax}} \{bp_{c_k} + (1 - \frac{rt_{c_k}}{T'_{max}})\} \tag{18}$$

where T'_{max} denotes the maximum repair time of the components not repaired in the solution *sol*;

(5) *AddGamma*: randomly selects γ components.

Each component to add is checked respecting the constraints on both B and T_{max}. Also in this case, more than one component may be individuated by the *AddBP*, *AddTime* and *AddBPT* moves to be potentially added.
The *RemoveBPT* aims at selecting the components to remove that are a trade-off between a low breakage probability and a high repair time. Similarly, the *AddBPT* adds the components that are a good compromise between a high breakage probability and a low repair time.

In addition, the moves *Switch* and *NoMove*, shared between the remove and the add moves, are applied when the move id is, respectively, equal to 6 and 0. In particular, the *Switch* move randomly selects one component repaired in *sol* and one component not repaired in *sol* such that the former is removed and the latter is added in the new solution, considering the respect of the problem constraints. While, the *NoMove* move does not apply any remove/add move to *sol*. However, the situation in which both the ids are equal to 0, i.e., no components are removed and added from/to *sol* (steps $10 - 11$), is avoided.

The routine *Dominated* (step 13) returns TRUE if *sol* is dominated; FALSE, otherwise. For this purpose, *sol* is compared with all the non-dominated ones already found and stored in S_{best}. Once a new non-dominated solution *sol* is added to S_{best}, the non-dominated solutions already found but dominated by *sol* are consequently removed from S_{best}.

In step 16, the routine *SelectRandom* returns a randomly generated number that is the position in S_{best} of the new starting solution to select at the next iteration. This step is introduced for shaking the algorithm, trying to explore different areas of the

searching space.

In order to speed-up the b-LNS, once a new solution is found, the value of the objective (7) is computed as the gap with that of the current solution sol. More specifically, let $BP(sol)$ and $BP(sol')$ be respectively the values of (7) with reference to sol and to the solution sol' found after applying a remove and an add move. Let $C_{remove} \subseteq \bar{C}(sol)$ and $C_{add} \subseteq C \setminus \bar{C}(sol)$ be the list of components to remove and to add, respectively. Then, $BP(sol')$ can be computed as $BP(sol') = BP(sol) + \sum_{j \in C_{remove}} bp_{c_j} - \sum_{i \in C_{add}} bp_{c_i}$.

Concerning RT^{max}, let $RT^{max}(sol)$ and $RT^{max}(sol')$ be the maximum repair time of sol and sol', respectively. Let $\overline{rt} = \max_{c_j \in C_{add}} \{rt_{c_j}\}$ and $\hat{rt} = \max_{c_j \in C_{remove}} \{rt_{c_j}\}$ be the maximum repair time of the components belonging to C_{add} and C_{remove} respectively. If $RT^{max}(sol) \leq \overline{rt}$ then $RT^{max}(sol') = \overline{rt}$, else if $RT^{max}(sol) > \hat{rt}$ then $RT^{max}(sol') = RT^{max}(sol)$, otherwise $RT^{max}(sol') = \max_{c_j \in (\bar{C}(sol) \setminus C_{remove}) \cup C_{add}} \{rt_{c_j}\}$. Since the number of components to add and remove at each iteration is less than $|\bar{C}(sol)|$ (at most $\gamma|\bar{C}(sol)|$), the time to compute $RT^{max}(sol')$ is on average less than the time to compute the maximum on the whole set $\bar{C}(sol)$.

In the worst case, the complexity of each iteration of the b-LNS is $O(\Gamma log(\Gamma))$, where $\Gamma = max\{|\bar{C}|, |C \setminus \bar{C}|\}$. Indeed, in the worst case, the complexity of $RemoveBP$ and $RemoveTime$ is $O(|\bar{C}|log(|\bar{C}|))$, the complexity of $AddBP$ and $AddTime$ is $O(|C \setminus \bar{C}|log(|C \setminus \bar{C}|)$.

7. Computational results

Both the AUGMECON and the b-LNS were implemented by using Java as programming language. ILOGs CPLEX Concert Technology (version 12.9) was used for solving the MILPs generated by the AUGMECON. The experiments were carried out on a 4-core processor with at 3.40GHz with 32GB RAM.

7.1. *Case study*

The experimental campaign was carried out on a set of real-life alike instances inspired by an oil refinery.

To this end, a three year-time interval, from January 2001 to December 2003, was analyzed. Data came from two databases: one contains information on the process cycle, recording the hourly amount of product entering each sub-plant and, if any, the stoppage detail; the other stores information on both component breakages and the related maintenance interventions. Such a refinery is characterized by a total number of 715 components. From data, it turns out that in the period considered, the number of component breakages occurred was 6160 and the total amount of stoppages was 1164. The distributions of both repair costs and times of the components are provided in Figures 1 and 2. In particular, the repair times distribution is Gaussian-like while one can note that the most of components have a low cost.

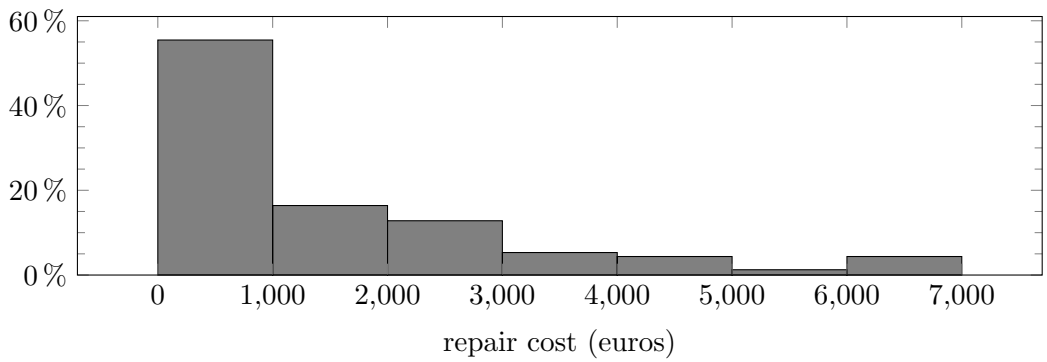

Figure 1. Distribution of the components' repair costs.

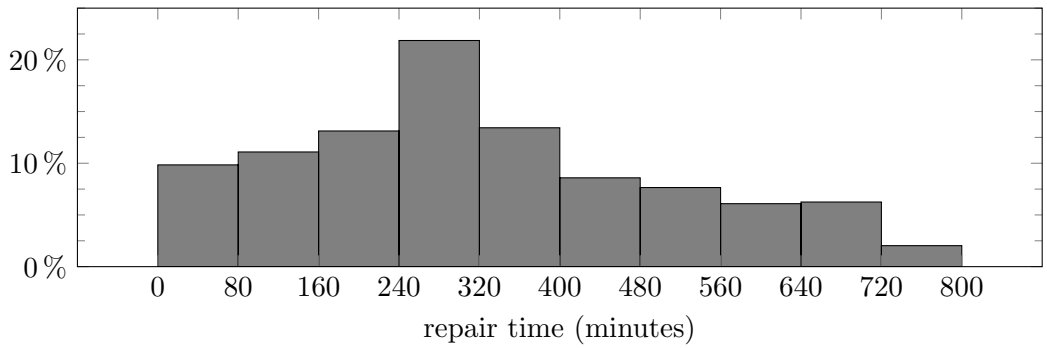

Figure 2. Distribution of the components' repair times.

7.2. *Experimental Setting*

The parameters setting, shown in Table 3, is fixed in collaboration with a domain expert. Moreover, 7 instances are generated and grouped into three sets, namely *small* (i.e. $I20, I40, I80$), *medium* (i.e., $I160, I320$) and *large* (i.e., $I640, I1280$) as reported in Table 3.

Each instance was randomly generated from the case study distribution. It is worth remarking that the largest number of components is 1280 since it is by far greater than the number of components of a very complex case study like the one considered, hence it is not reasonable to have more than that.

The AUGMECON was run with an increasing step equal to 1 while, *eps* and the total time limit have been set to 10^4 and 3600 seconds, respectively. Furthermore, each MILP, at each iteration of the AUGMECON, was solved with a CPU time limit of 3600 seconds too.

Concerning the b-LNS, TL was set to 3600 seconds. Its results were collected in specific time instants, namely 0.1, 0.2, 0.5, 1, 2, 3, 5, 10, 20, 30, 50 seconds, then from 100 to 1000 seconds with step 100 and from 1000 to 3600 seconds with step 200. However, for the sake of simplicity, the best results obtained are reported. The γ parameter was set to 0.20 for both *small* and *medium* sets. Instead, for *large* instances, it was varied from 0.05 to 0.20 and the best experimentally found value was taken. Specifically, for $I640$ and $I1280$, it was fixed to 0.19 and 0.09, respectively. Readers are referred to Subsection 7.3 for a detailed discussion on this parameter.

The results of both AUGMECON and b-LNS were evaluated considering the following multi-objective metrics:

Table 3. Parameters setting and instances generation.

Parameter	**Value**
B	170000 euros
T_{max}	1440 minutes
C_{work}	30 euros per hour
ΔT	1 month
Instance name	$\lvert C \rvert$
$I20$	20
$I40$	40
$I80$	80
$I160$	160
$I320$	320
$I640$	640
$I1280$	1280

- number of non-dominated solutions (η);
- number of non-dominated solutions of one of the two approaches that are actually dominated by the other ($\hat{\eta}$);
- *Spacing* (S), i.e., a metric introduced in Schott (1995) that measures the range variance of neighboring solutions in the front. In other words, it measures the distribution of the solutions along the front and it is defined as follows:

$$S = \sqrt{\frac{1}{\eta}\sum_{i=1}^{\eta}(d_i - \bar{d})^2} \tag{19}$$

where $\bar{d}$ is the average of all the distances $d_i, \forall i = 1, \ldots, \eta$ and the i-th distance d_i is computed as:

$$d_i = \min_{j \in F: j \neq i} \left\{ \left| \widetilde{BP}_i - \widetilde{BP}_j \right| + \left| \widetilde{RT}_i^{max} - \widetilde{RT}_j^{max} \right| \right\} \tag{20}$$

where F denotes the front whereas $\widetilde{BP}_i$ and $\widetilde{RT}_i^{max}$ represent respectively the normalized value of the two objective functions of the i-th non-dominated solution. For example, $\widetilde{BP}_i$ is computed as follows:

$$\widetilde{BP}_i = \frac{BP_i - \min_{j \in F} BP_j}{\max_{j \in F} BP_j - \min_{j \in F} BP_j}$$

It is worth remarking that the smaller the S value, the higher the diversification of F is.

7.3. *Numerical Results*

Table 4 reports the numerical results of the experiments.Considering the *small* and the *medium* sized instances, both the approaches always give the same front and, then, the same S value. In particular, on the *small* instances, the computational time required by the AUGMECON (i.e., 48.51, 92.55, 139.74, respectively) is by far higher than that of the b-LNS (i.e., 0.5, 0.5, 0.5 seconds, respectively).

Table 4. Numerical results of experiments on *small*, *medium* and *large* sets.

		AUGMECON			b-LNS		
	Instance	η	$\hat{\eta}$	S	η	$\hat{\eta}$	S
Small	I20	9	0	0.069	9	0	0.069
	I40	13	0	0.055	13	0	0.055
	I80	12	0	0.100	12	0	0.100
Medium	I160	12	0	0.083	12	0	0.083
	I320	8	0	0.155	8	0	0.155
Large	I640	8	0	0.038	8	0	0.038
	I1280	6	0	0.057	9	1	0.042

On $I160$, the percentage increment of the computational time required by the AUGMECON (i.e., 223.39 seconds) with regard that of the b-LNS (i.e., 3 seconds) is 7346%. Finally, on $I320$, the computational time required by the AUGMECON is by far higher than that of the b-LNS, i.e., 640 seconds against 100 seconds.

As for the *large* set of instances, on $I640$, both the approaches give the same front. The computational time required by the AUGMECON is higher than that of the b-LNS, i.e., 1409.45 seconds against 500 seconds, respectively. Instead, on $I1280$, the AUGMECON, that also reaches the time limit, returns 6 non-dominated solutions. For the b-LNS, η is 9 but 1 solution is actually dominated by those of the AUGMECON.

In order to obtain a better front, the AUGMECON was run without time limit, but after 12 hours of computation, it failed to finish, fully saturating the memory. The time required by the b-LNS to return the best front on $I1280$ is 2600 seconds. Comparing the fronts obtained by the two approaches, the AUGMECON is not able to find the three solutions with lower values of BP. Instead, the solution with higher value of BP for the b-LNS is actually dominated by one solution found by the AUGMECON.

In general, as expected, the computational times of the AUGMECON are on average about 2 order of magnitude greater than those of the b-LNS for obtaining the same front (see Figure 3), although both the solution methods are comparable from the non-dominated solutions point of view. Indeed, the AUGMECON on average solves 3157, 8624 and 20098 MILPs on *small*, *medium* and *large* sets, respectively.

Furthermore, the b-LNS never reaches the time limit whereas the AUGMECON reaches one hour of computation on $I1280$. For *small* and *medium* sets, the execution time of the b-LNS is lower than 100 seconds whereas for *large* set, whatever the value of γ, it always requires less than 2600 seconds. It is noteworthy that AUGMECON uses CPLEX for solving each MILP, which runs in multi-threaded mode exploiting all the 8 threads of the processor. Instead, the b-LNS runs sequentially.

Regarding the γ parameter, it is worth noting that the b-LNS returns the best results with high values up to $I640$ whereas small values are used for $I1280$, as already anticipated in Section 7.1. In fact, as defined in Section 6, the parameter γ denotes the percentage of components that in a solution has to be removed/added. This way, it represents a perturbation (i.e., *shaking*) applied to the search space. Numerical results suggest that for small/medium instances, shaking a lot the current solution may help the algorithm finding new non-dominated solutions. Instead, for large instances, shaking a bit at a time is preferable. This phenomenon mainly depends on the time

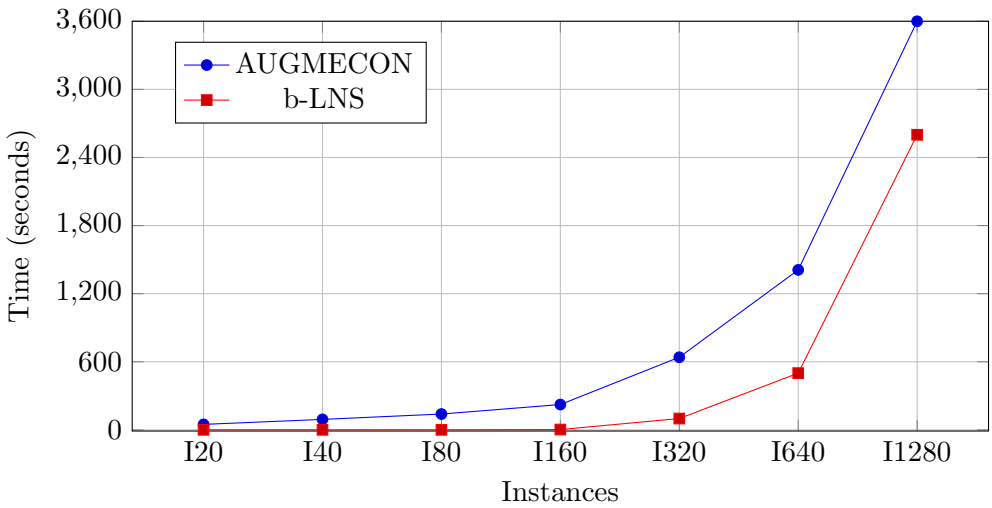

Figure 3. Computational time of AUGMECON and time for the best front of b-LNS.

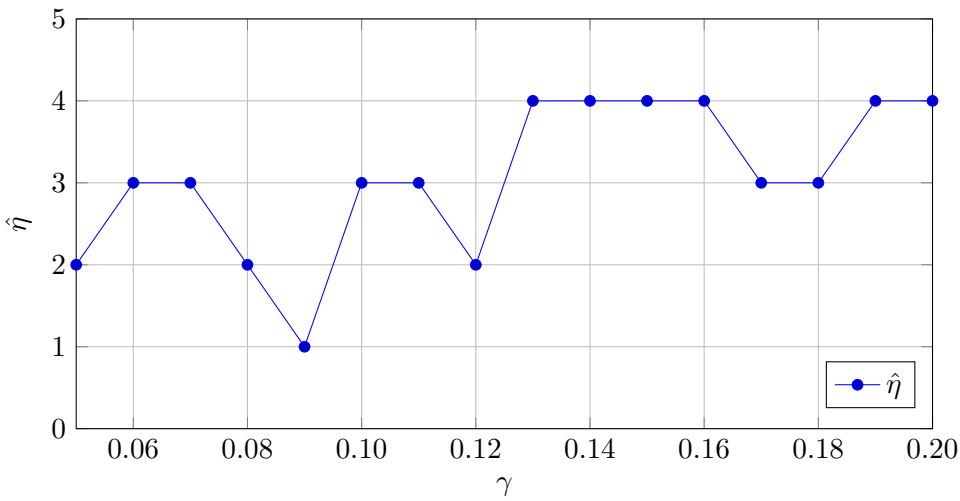

Figure 4. Values of $\hat{\eta}$ on $I1280$ varying the parameter γ

needed to execute the *AddGamma* and the *RemoveGamma* moves. Indeed, with high values of γ, both the moves require computational times to be executed higher than those needed with small values. Therefore, a few moves can be applied in the CPU time limit of 3600 seconds. This may have a negative impact on the large instances where the solutions space increases. Figure 4 shows the trend of $\hat{\eta}$ on $I1280$, by varying the value of γ from 0.05 to 0.20. In particular, with higher values ($\gamma \in [0.15, 0.20]$), the b-LNS performances deteriorate since its results are worse than those obtained with lower values of this parameter.

7.4. *A data-driven analysis on the moves effectiveness*

In this section, the effectiveness of the moves of the b-LNS is studied through a data-driven analysis performed on $I1280$. On it, in fact, the b-LNS provides more non-dominated solutions than those detected by the AUGMECON (that reaches the one

Table 5. Data-driven analysis on the moves effectiveness.

	AddRandom	AddBP	AddTime	NoMove	AddBPT	AddGamma	Switch
RemoveRandom	0.49%	0.49%	0.00%	0.00%	0.98%	9.8%	0.49%
RemoveBP	0.98%	0.49%	0.00%	0.00%	0.00%	13.73%	0.00%
RemoveTime	2.45%	4.41%	1.47%	5.39%	0.98%	13.73%	2.94%
NoMove	4.41%	1.47%	10.29%	0.00%	1.47%	3.92%	0.00%
RemoveBPT	0.00%	0.49%	0.00%	1.47%	0.00%	2.94%	0.00%
RemoveGamma	1.47%	0.49%	2.45%	0.00%	1.96%	3.43%	0.00%
Switch	0.00%	0.00%	0.00%	0.00%	0.00%	4.9%	0.00%

hour time limit) but with a value of $\hat{\eta}$ equal to 1. This means that one of the non-dominated solutions of the b-LNS is actually dominated by the AUGMECON.

The data-driven analysis is carried out by running the b-LNS on the $I1280$ for 100 seconds, gathering information on a total number of 6491782 pairs of moves. Each pair is made up by both a *Remove* and an *Add* move (see Section 6) and has been classified as either ND or D pair in the case in which it returned either a Non-Dominated or a Dominated solution, respectively. Table 5 reports the percentage of times in which a given pair of moves has been selected and it has been classified as ND pair (i.e., it has been effective). Among the 6491782 pairs of moves used in 100 seconds, 204 of them have been effective giving a non-dominated solution. In particular, the most profitable remove move is the *RemoveTime*, i.e., in almost 31.37% of the cases while the second most profitable one is *NoMove* (i.e., 21.57% of cases). The most profitable add move is the *AddGamma*, in about 52.45% of the cases. It is worth noting that each solution has 13.89 components on average and then, *AddGamma* adds 3 components on average. The second profitable add move is *AddTime*, in about 14.22% of the cases. The pairs of moves (*RemoveBP*, *AddGamma*), (*NoMove*, *AddTime*), (*RemoveTime*, *AddGamma*) and (*RemoveRandom*, *AddGamma*) perform the best. It is worth noting that all of them add more components than those removed (at most one at time).

The *Switch* move is effective only in 8.33% of the cases and it is also the most time consuming move. In addition, the *RemoveBPT* and the *AddBPT* are effective in only 4.90% and 4.58% of the cases, respectively. For this reason, all of these three moves may be candidate to be removed from the set of available moves.

Therefore, the b-LNS is run on the $I1280$ without the *Switch*, the *RemoveBPT* and the *AddBPT* move, obtaining $\eta = 9$ (greater than that of the AUGMECON equal to 6) but with $\hat{\eta}$ equal to 0 meaning that no solution is dominated by those of the AUGMECON. Moreover, in this case, the best front is obtained by the b-LNS in only 1200 seconds.

Figure 5 shows the b-LNS execution on the $I160$. In particular, each point represents a new non-dominated solution generated during a specific iteration. The arc going from point p to point p' indicates that the non-dominated solution p' has been generated starting from the non-dominated solution p. Points with only ingoing arcs are those that, although have been further explored, have not generated new non-dominated solutions (e.g., the point with $BP = 2$ and $RT^{max} = 0.5$). It is worth noting that the solutions concentrate in the neighbour of the solutions belonging to the front (denoted by blue points). In particular, it is interesting observing that the higher concentration of those solutions is in the right-hand side of the plot, where there is

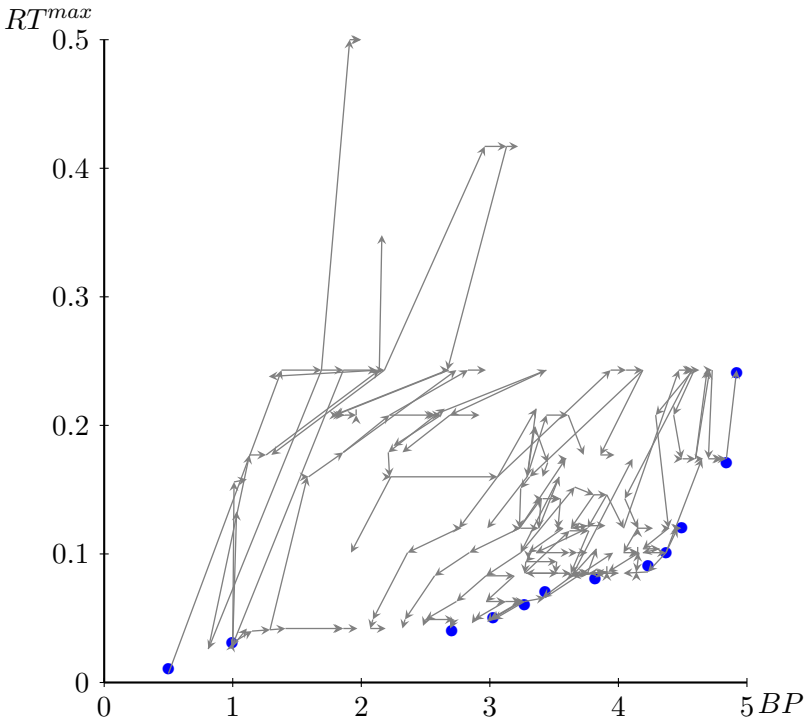

Figure 5. Solutions generated by the b-LNS on the I160

the highest density of solutions. This indicates that around those solutions the b-LNS introduces only small adjustments. In fact, the average length of arcs linking points having $BP > 2.7$ and $RT^{max} \leq 0.17$ (i.e., the region with 9 of 12 solutions) is 0.153 while the average length of other arcs is 0.263.

7.5. *Practical implications*

The proposed approach aims to develop a predictive maintenance policy to identify the optimal set of components to replace for maximizing the plant reliability and minimizing the maximum time spent for repairs. This problem, modeled through bi-objective MILP, supports the maintenance managers in implementing an effective maintenance policy, considering not only the breakage probabilities but also the resource constraints. The present approach is entirely data-driven since all parameters are derived from data on the past failure events, are extracted from data on the component characteristics, are constants provided by the domain experts or are the breakage probabilities estimated by Algorithm 1. This way, the proposed approach 1) can be applied to different application contexts and 2) does not require the parameters tuning. Moreover, the input parameters are those known at the moment of the system stoppage. Then, the proposed approach adapts well to changes in the working conditions (e.g. B, T_{max}, repair costs and the number of operators needed for repairing each component). If the working conditions do not change, one can use all available historical data on the breakages to better estimate the probabilities. Otherwise, one can weight less the oldest information with regard to the newest one or to use a sliding time window of observation.

Additionally, depending on the plant characteristics, the decision makers should de-

fine whether considering the whole plant, limiting the implementation of the predictive maintenance policy to its portion or to the most critical components.
Considering the impact of the approach on a wider perspective, its implementation also involves other departments. In fact, one can run the proposed approach before a stoppage occurs. This way, it is possible to know in advance which components could be used for maintenance purposes and hence, to implement the appropriate supplying policy.

8. Conclusions

The increasing huge amount of data, enabled by Industry 4.0 infrastructures, needs designing data-driven algorithms for assets availability and reliability maximization for a continuous process monitoring. In particular, determining the optimal set of components to be predictively repaired (namely, solving the Component Repairing Problem) represents a challenging task especially when several conditions (e.g., breakage probability, repair time and cost) have to be considered simultaneously.

Since the systems with many components, requiring also to be constantly active, have to be accurately monitored, in this paper, a new data-driven predictive maintenance policy is proposed. Firstly, receiving a set of components, a time interval for the observation, a list of both stoppage and breakage events, the breakage probability of each component is determined. Then, according to the breakage probabilities, a multi-objective optimization approach is designed for individuating the sub-set of components to repair, maximizing the overall system reliability and minimizing the maximum time spent for repairing them, under constraints on both the maximum budget and the the maximum repairing time. Two alternative optimization solution methods are described: one is based on the formulation of a bi-objective Mixed Integer Linear Programming model solved through the AUGMEnted ϵ-CONstraint (AUGMECON) approach; the other implements a bi-objective Large Neighborhood Search (b-LNS) meta-heuristic.

Three sets of real-life alike instances (of increasing size) were derived from a real oil refinery. Numerical results show that the b-LNS significantly outperforms the AUGMECON approach concerning the computational time required. Concerning the solution quality, the front of the b-LNS is almost the same of that provided by the AUGMECON approach except when the AUGMECON reaches the CPU time limit of one hour. A data-driven analysis was also performed for assessing the effectiveness of the b-LNS moves and for deriving insights useful to those researchers who are going to design a b-LNS to address decision problems arising in similar application contexts.

Future research may focus on applying the proposed approaches for solving other problems arising in similar application contexts where the decisions mainly concern systems reliability.

References

Antomarioni, Sara, Maurizio Bevilacqua, Domenico Potena, and Claudia Diamantini. 2019a. "Defining a data-driven maintenance policy: an application to an oil refinery plant." *International Journal of Quality & Reliability Management* 36 (1): 77–97.

Antomarioni, Sara, Ornella Pisacane, Domenico Potena, Maurizio Bevilacqua, Filippo Emanuele Ciarapica, and Claudia Diamantini. 2019b. "A predictive association rule-

based maintenance policy to minimize the probability of breakages: application to an oil refinery." *The International Journal of Advanced Manufacturing Technology* 105 (9): 3661–3675.

Arunraj, NS, and J Maiti. 2010. "Risk-based maintenance policy selection using AHP and goal programming." *Safety science* 48 (2): 238–247.

Berrichi, Ali, Farouk Yalaoui, Lionel Amodeo, and M Mezghiche. 2010. "Bi-objective ant colony optimization approach to optimize production and maintenance scheduling." *Computers & Operations Research* 37 (9): 1584–1596.

Bertolini, Massimo, and Maurizio Bevilacqua. 2006. "A combined goal programmingAHP approach to maintenance selection problem." *Reliability Engineering & System Safety* 91 (7): 839–848.

Bevilacqua, Maurizio, and Filippo Emanuele Ciarapica. 2018. "Human factor risk management in the process industry: A case study." *Reliability Engineering & System Safety* 169: 149–159.

Carlos, Sofía, Ana Sánchez, Sebastián Martorell, and J-F Villanueva. 2012. "Particle Swarm Optimization of safety components and systems of nuclear power plants under uncertain maintenance planning." *Advances in Engineering Software* 50: 12–18.

Certa, Antonella, Mario Enea, Giacomo Galante, and Toni Lupo. 2012. "A multi-objective approach to optimize a periodic maintenance policy." *International Journal of Reliability, Quality and Safety Engineering* 19 (06): 1240002.

Ebrahimipour, V, A Najjarbashi, and Mohammad Sheikhalishahi. 2015. "Multi-objective modeling for preventive maintenance scheduling in a multiple production line." *Journal of Intelligent Manufacturing* 26 (1): 111–122.

Ehrgott, Matthias, and Xavier Gandibleux. 2000. "A survey and annotated bibliography of multiobjective combinatorial optimization." *OR-Spektrum* 22 (4): 425–460.

EN, BS. 2017. "13306: 2017: Maintenance–Maintenance terminology." *BSI Standards Publication* .

Fan, Yuling, and Xiaohua Xia. 2017. "A multi-objective optimization model for energy-efficiency building envelope retrofitting plan with rooftop PV system installation and maintenance." *Applied Energy* 189: 327–335.

Garg, Amik, and SG Deshmukh. 2006. "Maintenance management: literature review and directions." *Journal of quality in maintenance engineering* 12 (3): 205–238.

Gjorgiev, Blaže, Duško Kančev, and Marko Čepin. 2012. "Risk-informed decision making in the nuclear industry: Application and effectiveness comparison of different genetic algorithm techniques." *Nuclear Engineering and Design* 250: 701–712.

Huang, Hong-Zhong, Jian Qu, and Ming J Zuo. 2009. "Genetic-algorithm-based optimal apportionment of reliability and redundancy under multiple objectives." *IIE Transactions* 41 (4): 287–298.

Kumar, Akhilesh, Ratna Babu Chinnam, and Finn Tseng. 2019. "An HMM and polynomial regression based approach for remaining useful life and health state estimation of cutting tools." *Computers & Industrial Engineering* 128: 1008–1014.

Kumar, Girish, Vipul Jain, and OP Gandhi. 2018. "Availability analysis of mechanical systems with condition-based maintenance using semi-Markov and evaluation of optimal condition monitoring interval." *Journal of Industrial Engineering International* 14 (1): 119–131.

Loganathan, MK, and OP Gandhi. 2016. "Maintenance cost minimization of manufacturing systems using PSO under reliability constraint." *International Journal of System Assurance Engineering and Management* 7 (1): 47–61.

Marseguerra, Marzio, Enrico Zio, and Luca Podofillini. 2002. "Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation." *Reliability Engineering & System Safety* 77 (2): 151–165.

Marseguerra, Marzio, Enrico Zio, and Luca Podofillini. 2004. "Optimal reliability/availability of uncertain systems via multi-objective genetic algorithms." *IEEE Transactions on Reliability* 53 (3): 424–434.

Mavrotas, George. 2009. "Effective implementation of the ε-constraint method in multi-

objective mathematical programming problems." *Applied Mathematics and Computation* 213 (2): 455–465.

Min, Liu Xiao, Wu Jun Yong, Yang Yuan, and Xu Wei Yan. 2009. "Multiobjective optimization of preventive maintenance schedule on traction power system in high-speed railway." In *2009 Annual Reliability and Maintainability Symposium*, 365–370. IEEE.

Moghaddam, Kamran S. 2013. "Multi-objective preventive maintenance and replacement scheduling in a manufacturing system using goal programming." *International Journal of Production Economics* 146 (2): 704–716.

Moghaddam, Kamran S. 2015. "Preventive maintenance and replacement optimization on CNC machine using multiobjective evolutionary algorithms." *The International Journal of Advanced Manufacturing Technology* 76 (9-12): 2131–2146.

Moghaddam, Kamran S, and John S Usher. 2011. "A new multi-objective optimization model for preventive maintenance and replacement scheduling of multi-component systems." *Engineering Optimization* 43 (7): 701–719.

Oddi, Angelo, Riccardo Rasconi, and Amedeo Cesta. 2015. "A multi-objective large neighborhood search methodology for scheduling problems with energy costs." In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 453–460. IEEE.

Okasha, Nader M, and Dan M Frangopol. 2009. "Lifetime-oriented multi-objective optimization of structural maintenance considering system reliability, redundancy and life-cycle cost using GA." *Structural Safety* 31 (6): 460–474.

Peng, Ying, Ming Dong, and Ming Jian Zuo. 2010. "Current status of machine prognostics in condition-based maintenance: a review." *The International Journal of Advanced Manufacturing Technology* 50 (1-4): 297–313.

Pisinger, David, and Stefan Ropke. 2010. "Large neighborhood search." In *Handbook of metaheuristics*, 399–419. Springer.

Raad, Darian, Alexander Sinske, and Jan Van Vuuren. 2009. "Robust multi-objective optimization for water distribution system design using a meta-metaheuristic." *International Transactions in Operational Research* 16 (5): 595–626.

Ruiz, Rubén, J Carlos García-Díaz, and Concepción Maroto. 2007. "Considering scheduling and preventive maintenance in the flowshop sequencing problem." *Computers & Operations Research* 34 (11): 3314–3330.

Schaus, Pierre, and Renaud Hartert. 2013. "Multi-objective large neighborhood search." In *Proceedings of International Conference on Principles and Practice of Constraint Programming*, 611–627. Springer.

Schott, Jason R. 1995. "Fault tolerant design using single and multicriteria genetic algorithm optimization." M.S. Thesis, Massachusetts Institute of Technology.

Seif, Javad, Andrew Junfang Yu, and Fahimeh Rahmanniyay. 2018. "Modelling and optimization of a bi-objective flow shop scheduling with diverse maintenance requirements." *International Journal of Production Research* 56 (9): 3204–3225.

Shaw, Paul. 1998. "Using constraint programming and local search methods to solve vehicle routing problems." In *Proceedings of International Conference on Principles and Practice of Constraint Programming*, 417–431. Springer.

Su, Chun, and Yang Liu. 2019. "Multi-objective imperfect preventive maintenance optimisation with NSGA-II." *International Journal of Production Research* 58 (13): 4033–4049.

Tian, Zhigang, Daming Lin, and Bairong Wu. 2012. "Condition based maintenance optimization considering multiple objectives." *Journal of Intelligent Manufacturing* 23 (2): 333–340.

Wu, Zhou, Xiaohua Xia, and Bo Wang. 2015. "Improving building energy efficiency by multi-objective neighborhood field optimization." *Energy and Buildings* 87: 45–56.

Yu, Hao, and Wei Deng Solvang. 2016. "An improved multi-objective programming with augmented ε-constraint method for hazardous waste location-routing problems." *International Journal of Environmental Research and Public Health* 13 (6): 548.