Nested general variable neighborhood search for the periodic maintenance problem

Raca Todosijević, Rachid Benmansour, Said Hanafi, Nenad Mladenovic, Abdelhakim Artiba

European Journal of Operational Research 252 (2016) 385–396

Nested general variable neighborhood search for the periodic maintenance problem

Raca Todosijević [a, b, *], Rachid Benmansour [a], Saïd Hanafi [a], Nenad Mladenović [a, b], Abdelhakim Artiba [a]

[a] Université de Valenciennes et du Hainaut Cambrésis, LAMIH – UMR CNRS 8201, France
[b] Mathematical Institute, Serbian Academy of Science and Arts, Belgrade, Serbia

ARTICLE INFO

Keywords:
Scheduling
Preventive maintenance
Mixed-integer linear programming
Variable neighborhood search
Nested general VNS

ABSTRACT

In this paper we study the periodic maintenance problem: given a set of m machines and a horizon of T periods, find indefinitely repeating itself maintenance schedule such that at most one machine can be serviced at each period. In addition, all the machines must be serviced at least once for any cycle. In each period the machine i generates a servicing cost b_i or an operating cost which depends on the last period in which i was serviced. The operating cost of each machine i in a period equals a_i times the number of periods since the last servicing of that machine. The main objective is to find a cyclic maintenance schedule of a periodicity T that minimizes total cost. To solve this problem we propose a new Mixed Integer programming formulation and a new heuristic method based on general Variable neighborhood search called Nested general variable neighborhood search. The performance of this heuristic is shown through an extensive experimentation on a diverse set of problem instances.

1. Introduction

Companies usually have developed long term strategies to remain competitive, innovative and profitable. At an operational level, an efficient use of their resources is crucial to remain successful. As a consequence, the human organization and the manufacturing systems in these companies have to be adapted consequently to support managerial decisions. To achieve this goal, the production tools must be available almost all the time.

Given that manufacturing systems constitute the vast majority of company's investment and constitute their production tools, they must be in perfect conditions whenever needed. Unfortunately, these systems are subject to random failures and to deterioration and therefore have to undergo corrective maintenance (Benmansour, Allaoui, Artiba, Iassinovski, & Pellerin, 2011). Systems can also be stopped for preventive maintenance reasons to avoid (or minimize) the consequences of failures. This kind of maintenance is designed to preserve and restore equipment reliability by replacing worn components before they actually fail. Preventive maintenance activities take time that could otherwise be used for production, but delaying preventive maintenance for production may increase the probability of machine failures (Cassady & Kutanoglu, 2005). Hence, there are conflicts between maintenance planning, and production scheduling and consequently there are good reasons to try minimizing the cost of the two functions (Weinstein & Chung, 1999).

The preventive maintenance problem arises especially in large manufacturing companies. Since the capacity of the maintenance department is limited, we are trying to schedule the maintenance of a number of machines throughout the year (e.g. 52 weeks). Each machine must be serviced at least once throughout the year; otherwise the operating cost of that machine will continue to increase and the reliability of the machine will decrease to such an extent that will affect the quality of the product. In addition, during each week, the maintenance service can not service more than one machine. Finally, the desired schedule aims to determine, for each week, which machine has to be serviced (if any).

Any interruption of the line caused by any equipment malfunction or failure will result in a major disruption of output or even line or factory shutdown. Thus, an effective maintenance program should be designed to provide the required availability of machinery and output quality. Furthermore, analysis of maintenance costs indicates that a repair performed in the reactive or run to failure mode is, on average, about three times higher than the same

E-mail addresses: raca.todosijevic@univ-valenciennes.fr, racatodosijevic@gmail.com (R. Todosijević), rachid.benmansour@univ-valenciennes.fr (R. Benmansour), said.hanafi@univ-valenciennes.fr (S. Hanafi), nenad.mladenovic@univ-valenciennes.fr (N. Mladenović), abdelhakim.artiba@univ-valenciennes.fr (A. Artiba).

DOI : 10.1016/j.ejor.2016.01.014

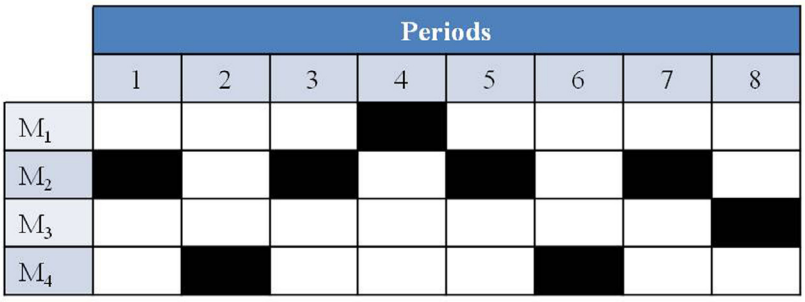

Fig. 1. Example with $m = 4$ and $T = 8$, with optimal solution $\pi^* = (2, 4, 2, 1, 2, 4, 2, 3)$ and $f(\pi^*) = 176$.

repair made within a scheduled or preventive mode (Mobley, 2002). Scheduling the repair minimizes the repair time and associated labor costs. It also reduces the negative impact of expedited shipments and lost production.

In this paper we consider the *periodic maintenance problem* (PMP). PMP is NP-hard problem stated in the following way. There is a set of machines $M = \{1, 2, \ldots, m\}$, and there is a set of periods $U = \{1, 2, \ldots, T\}$ with $T \geq m$. The PMP consists of finding an optimal cyclic maintenance schedule of length T that is indefinitely repeated. At most one machine is serviced at each period and all the machines must be serviced at least once for any cycle. When machine $i \in M$ is serviced, a given non-negative servicing cost of b_i is incurred, regardless of the period. At period $t \in U$, a machine $i \in M$ that is not serviced during some period is in operation and incurs an operation cost of $n_i(t) \times a_i$ where a_i is a given positive integer, and where $n_i(t)$ is the number of periods elapsed since last servicing of machine i. The main objective of this problem is to determine a feasible maintenance schedule with a minimum cost, i.e. to decide for each period $t \leq T$ which machine to service (if any), such that the total servicing costs and operating costs are minimized. We give here an example to well understand the present problem. If cycle length T is a decision variable then the problem is called the *Free periodic maintenance problem*. Here we consider T as an input parameter.

Illustrative example. Let the length of the maintenance cycle $T = 8$ and the total number of machines $m = 4$. We suppose furthermore that the servicing costs are $b_1 = 1$, $b_2 = 2$, $b_3 = 3$, and $b_4 = 4$ and the operation costs are $a_1 = 1$, $a_2 = 10$, $a_3 = 1$ and $a_4 = 5$.

We explain the problem using the optimal solution $\pi^* = (2, 4, 2, 1, 2, 4, 2, 3)$ obtained by a MIP formulation of the problem (that will be given later in Section 2). Solution π^* indicates that machine 2 is serviced in the first period, 4 in the second period, etc. The total corresponding cost is as follows: *Servicing cost* $= b_2 + b_4 + b_2 + b_1 + b_2 + b_4 + b_2 + b_3 = 2 + 4 + 2 + 1 + 2 + 4 + 2 + 3 = 20$. The *operating costs* are computed as follows. For Machine 1, costs are incurred in periods 1, 2, 3, 5, 6, 7 and 8. In periods 5, 6, 7, 8, 1, 2 and 3 these costs are equal respectively to a_1, $2a_1$, $3a_1$, $4a_1$, $5a_1$, $6a_1$ and $7a_1$. Thus the total cost equals 28 for machine 1. Similarly we compute the total cost for the other machines. The total cost of machine 2 is equal to $0 + a_2 + 0 + a_2 + 0 + a_2 + 0 + a_2 = 40$. The total cost of machine 3 is equal to $a_3 + 2a_3 + 3a_3 + 4a_3 + 5a_3 + 6a_3 + 7a_3 + 0 = 28a_3 = 28$. For machine 4, the total cost is $a_4 + 2a_4 + 3a_4 + 0 + a_4 + 2a_4 + 3a_4 + 0 = 12a_4 = 60$. Therefore, the objective value of this optimal solution is $f(\pi^*) = 176$. Fig. 1, also presents it.

The contributions of the paper are:

(i) New variant of Variable Neighborhood Search (VNS) called Nested General VNS (NGVNS) is proposed;
(ii) New mathematical programming formulation of periodic maintenance problem (PMP) is proposed;
(iii) New NGVNS heuristic solves exactly all 110 instances from the literature;
(iv) Comparative study of 4 exact solution methods for PMP is conducted.

The rest of the paper is organized as follows. In the next section we give four existing mathematical programming formulations of PMP and then our new formulation. In Section 3 we describe our new variant of VNS called NGVNS. Section 4 contains computational results obtained on 110 instances while Section 5 concludes the paper.

2. Problem formulations

Grigoriev, Van De Klundert, and Spieksma (2006) presented several mathematical models to solve the PMP. Hereafter we first give four models from the literature and then we present a new mathematical model.

2.1. A quadratic programming formulation

In this model, it is assumed that servicing cost equals to zero, i.e., b_i's are assumed to be null. The quadratic model uses the integer variable $x_{i,t}$ that corresponds to the number of periods between the current period $t \in U$ and the last period before t when machine $i \in M$ has been serviced. So, the quadratic formulation of PMP is stated as follows:

$$\min_x \sum_{i \in M} \sum_{t \in U} a_i x_{i,t} \tag{1}$$

$$s.t.\ x_{i,t+1}(x_{i,t+1} - x_{i,t} - 1) = 0, i \in M, t \in U, \tag{2}$$

$$x_{i,1}(x_{i,1} - x_{i,T} - 1) = 0, i \in M, \tag{3}$$

$$x_{i,t} + x_{k,t} \geq 1, i \neq k, i \in M, k \in M, t \in U, \tag{4}$$

$$x_{i,t} \in \mathbb{Z}^+, i \in M, t \in U. \tag{5}$$

The objective function (1) minimizes the total operating cost. Eqs. (2) and (3) ensure the required behavior of the $x_{i,t}$ variables. Eq. (4) imply that each pair of machines cannot be serviced simultaneously.

The authors in Grigoriev et al. (2006) also gave a linearization of this model where the servicing costs b_i are taken into account. This linearization is given in the next section.

2.2. A linearization of the quadratic programming formulation

Aforementioned quadratic model is linearized by introducing the binary variable $y_{i,t}$ that takes value 1 if the machine i is serviced in period t and 0 otherwise. The formulation of linearized model is given bellow:

$$\min_x \sum_{i \in M} \sum_{t \in U} (a_i x_{i,t} + b_i y_{i,t}) \tag{6}$$

$$s.t.\ x_{i,t+1} \geq x_{i,t} + 1 - T y_{i,t+1}, i \in M, t \in U, \tag{7}$$

$$x_{i,1} \geq x_{i,T} + 1 - Ty_{i,1}, i \in M, \tag{8}$$

$$\sum_{i \in M} y_{i,t} \leq 1, t \in U, \tag{9}$$

$$x_{i,t} \in \mathbb{Z}^+, i \in M, t \in U, \tag{10}$$

$$y_{i,t} \in \{0,1\}, i \in M, t \in U. \tag{11}$$

The objective function (6) minimizes the sum of operating costs and servicing costs. Eqs. (7) and (8) enforce the variables $x_{i,t}$ to behave in the same way as in the previous model. Inequality (9) assure that we cannot service more than one machine in a single period. Restrictions (10) and (11) are usual integrality constraints.

2.3. A flow formulation of PMP

The PMP may be modeled using the binary variable $x_i^{s,t}$ which equals 1 if machine $i \in M$ is serviced in period $s \in U$, and serviced next (cyclically) in period $t+1 \in U$, and 0 otherwise. In this model cost $c(s, t)$ is defined as:

$$c(s,t) = \begin{cases} \frac{(t-s)(t-s+1)}{2} & \text{if } s \leq t, \\ \frac{(T-s+t)(T-s+t+1)}{2} & \text{if } s > t, \end{cases}$$

Hence, the flow formulation model is stated as:

$$\min \sum_{i \in M} \sum_{s \in U} \sum_{t \in U} \left(a_i c(s,t) x_i^{s,t} + b_i x_i^{s,t} \right) \tag{12}$$

$$s.t. \sum_{i \in M} \sum_{s \in U} x_i^{s,t} \leq 1, t \in U \tag{13}$$

$$\sum_{s \in U} x_i^{s,t} = \sum_{s \in U} x_i^{t+1,s}, i \in M, t \in U, \tag{14}$$

$$\sum_{s \in U} x_i^{s,T} = \sum_{s \in U} x_i^{1,s}, i \in M, \tag{15}$$

$$\sum_{s \in U} \sum_{t \in U} x_i^{s,t} \geq 1, i \in M \tag{16}$$

$$x_i^{s,t} \in \{0,1\}, i \in M, s \in U, t \in U. \tag{17}$$

The objective function (12) minimizes the total operating costs and servicing costs. Inequalities (13) assure that, at each period, at most one machine can be serviced. Equality constraints (14) and (15) imply that there is a next period in which a machine will be serviced. Constraint (16) assures that each machine is serviced at least once. Finally, restrictions (17) represent the integrality conditions.

2.4. A set partitioning formulation

Let S be the set of all non-empty subsets of U. Clearly, every $s \in S$ is a possible set of periods for servicing a machine $i \in M$. Let us call $s \in S$ a service strategy or simply strategy. For every pair consisting of a machine $i \in M$ and a strategy $s \in S$, we can compute the cost $c_{i,s}$ incurred when servicing machine i in the periods contained in s as follows: let p_s be the cardinality of s and let q_j, $j \in \{1, 2, \ldots, p_s\}$, be the distance between neighboring services in s. The total service and operating cost associated with machine $i \in M$ and strategy $s \in S$ is

$$c(i,s) = b_i p_s + a_i \sum_{j=1}^{p_s} (q_j - 1) q_j / 2.$$

The set-partitioning formulation (SP) is as follows: The binary variable $x_{i,s}$ is equal to 1 if the machine $i \in M$ is serviced in the periods contained in strategy $s \in S$, and 0 otherwise.

$$\min \sum_{i \in M} \sum_{s \in S} c_{i,s} x_{i,s} \tag{18}$$

$$s.t. \sum_{s \in S} x_{i,s} = 1, i \in M \tag{19}$$

$$\sum_{i \in M} \sum_{s \in S: t \in s} x_{i,s} \leq 1, t \in T \tag{20}$$

$$x_{i,s} \in \{0,1\}, i \in M, s \in S. \tag{21}$$

The total servicing and operating cost is minimized by objective function (18). Constraints (19) imply that one service strategy has to be selected for each machine. Constraints (20) ensure that no two strategies make use of a same period, while constraints (21) represent usual integrality constraints.

Despite the fact that the set-partitioning formulation (SP) has an exponential number of variables, its linear relaxation, which is quite strong, is solvable in a polynomial time in m and T (see Grigoriev et al., 2006). Furthermore, the LP relaxation of SP is stronger than the LP relaxation of FF (see Grigoriev et al., 2006).

2.5. A new MIP formulation of PMP

The PMP may be modeled in terms of the following variables. Let $x_{i,t}$ be a binary variable that takes the value of 1 if machine i is serviced in the time period t and 0 otherwise. Further, let $z_{i,t}$ be a variable such that the difference $(t - z_{i,t})$ is equal to 0 if the machine i is serviced in the time period t, while otherwise, it is equal to the number of time periods elapsed since the last service of machine i. Mathematically, the variable $z_{i,t}$ may be stated as:

$$z_{i,t} = \max\{\tau \in \{h | x_{i,h} = 1,\ 1 \leq h \leq t\} \cup \{h - T | x_{i,h} = 1,\ t+1 \leq h \leq T\}\} \tag{22}$$

Note that we also introduce the variable $z_{i,t}$ for $t = 0$ that corresponds to the last period before starting new cycle. Then the PMP may be formulated as follows:

$$\min \sum_{i \in M} \sum_{t \in U} b_i x_{i,t} + a_i (t - z_{i,t}) \tag{23}$$

$$s.t. \sum_{i \in M} x_{i,t} \leq 1, \quad t \in U \tag{24}$$

$$\sum_{t \in U} x_{i,t} \geq 1, \quad i \in M \tag{25}$$

$$z_{i,t} \geq x_{i,t}(t+T) - T, \quad t \in U, \quad i \in M \tag{26}$$

$$z_{i,t} \geq z_{i,t-1}, \quad t \in U, \quad i \in M \tag{27}$$

$$z_{i,t-1} + x_{i,t}(t+T) - z_{i,t} \geq 0, \quad t \in U, \quad i \in M \tag{28}$$

$$z_{i,0} = z_{i,T} - T \quad i \in M \tag{29}$$

$$-T \leq z_{i,t} \leq t, \quad t \in U, \quad i \in M \tag{30}$$

$$x_{i,t} \in \{0,1\}, \quad i \in M, \quad t \in U. \tag{31}$$

The objective (23) minimizes the sum of operating costs and servicing costs. The meaning of the constraints are as follows: constraint (24) guarantees that in each time period at least one

Table 1
Optimal solution of the MIP formulation.

	Time period t								
	0	1	2	3	4	5	6	7	8
Serviced machine		2	4	2	1	2	4	2	3
$x_{1,t}$		0	0	0	1	0	0	0	0
$x_{2,t}$		1	0	1	0	1	0	1	0
$x_{3,t}$		0	0	0	0	0	0	0	1
$x_{4,t}$		0	1	0	0	0	1	0	0
$z_{1,t}$	−4	−4	−4	−4	4	4	4	4	4
$z_{2,t}$	−1	1	1	3	3	5	5	7	7
$z_{3,t}$	0	0	0	0	0	0	0	0	8
$z_{4,t}$	−2	−2	2	2	2	2	6	6	6

Table 2
Number of constraints and number of variables for each model.

Formulation	# of constraints	# of integer variables	# of binary variables
Linearization of the quadratic programming	$mT+m+T$	mT	mT
Flow model	$mT+2m+T$	0	mT^2
A set partitioning	$m+T$	0	$m(2^T-1)$
New MIP	$T+2m+4mT$	0	mT

machine will be serviced, while the constraint (25) allows the servicing of each machine at least once. From the definition of variables $z_{i,t}$ (see (22)) follows that $z_{i,t}$ equals to t if $x_{i,t}=1$ and $z_{i,t-1}$ otherwise. This simple observation is used for modeling the constraints (26–30) keeping in mind that the whole process is cyclic (constraint (29)).

The model is illustrated using the example given in the Section 1. The optimal solution π^* of the formulation is presented in Table 1.

In what follows, we present a refinement of the model. Since we consider the minimization, the all constraints which bound variables $z_{i,t}$ from below are redundant. So, constraint (26) and (27) can be excluded from the model in order to obtain a model with smaller number of constraints. Such model is given below.

$$\min \sum_{i\in M}\sum_{t\in U} b_i x_{i,t} + a_i(t - z_{i,t}) \tag{32}$$

$$s.t. \sum_{i\in M} x_{i,t} \le 1, \quad t \in U \tag{33}$$

$$\sum_{t\in U} x_{i,t} \ge 1, \quad i \in M \tag{34}$$

$$z_{i,t-1} + x_{i,t}(t+T) - z_{i,t} \ge 0, \quad t \in U, \quad i \in M \tag{35}$$

$$z_{i,0} = z_{i,T} - T \quad i \in M \tag{36}$$

$$-T \le z_{i,t} \le t, \quad t \in U, \quad i \in M \tag{37}$$

$$x_{i,t} \in \{0,1\}, \quad i \in M, \quad t \in U. \tag{38}$$

The following Table (2) gives a brief comparison of four formulations of the studied PMP problem. In this table, the # symbol stands for “number”.

It appears that models 1 and 4 contains the less number of variables than other two.

3. Nested general variable neighborhood search for the PMP

Variable Neighborhood Search (VNS) (Hansen & Mladenović, 2001; Hansen, Mladenović, & Pérez, 2008; 2010; Mladenović & Hansen, 1997) is a flexible framework for building heuristics. VNS changes systematically the neighborhood structures during the search for an optimal (or near-optimal) solution. The changing of neighborhood structures is based on the following observations: (i) A local optimum relatively to one neighborhood structure is not necessarily a local optimal for another neighborhood structure; (ii) A global optimum is a local optimum with respect to all neighborhood structures; (iii) Empirical evidence shows that for many problems all local optima are relatively close to each other. The first property is exploited by increasingly using complex moves in order to find local optima with respect to all neighborhood structures used. The second property suggests using several neighborhoods, if local optima found are of poor quality. Finally, the third property suggests exploitation of the vicinity of the current incumbent solution. The VNS based heuristics have been successfully applied for solving many optimization problems (see e.g. Carrizosa, Mladenović, and Todosijević, 2013; Guo, Chen, and Wang, 2014; Hanafi, Lazic, Mladenovic, Wilbaut, and Crevits, 2015; Hansen et al., 2008; 2010; Lazić, Todosijević, Hanafi, and Mladenović, 2014; Mladenović, Urošević, Hanafi, and Ilić, 2012 for recent applications).

In this section we give details of our Nested general variable neighborhood search (NGVNS) based heuristic. Before giving its pseudo-code and explaining in details each its step, we first discuss important question in applying each heuristic: how to represent the solution of PMP in the computer.

3.1. Solution presentation and solution space

The following necessary condition allow us to efficiently define the solution space of the PMP.

Property 3.1. If there is a machine j such that $a_j > b_j$ and if solution of PMP is optimal then, exactly one machine is serviced in each time period.

Proof. Let us assume that opposite is true, i.e. that there is an optimal solution such that in the time period t' none of machines is serviced. In that case, in the time period t' operating cost $op\_cost(i,t') = n_i(t') \times a_i$ occurs for each machine i. Furthermore, the operating cost of machine j, $op\_cost(j,t')$ is greater or equal to a_j. Therefore, in the case when $a_j > b_j$, if we service the machine j in the time period t', we would obtain better solution than the optimal one. This is obviously a contradiction. □

In addition, we may draw the following property.

Property 3.2. Let us assume that there is a machine j such that $a_j = b_j$ and that in a given solution π of PMP in time period t' none of machines is serviced, then servicing machine j in the time period t' will yield the solution with objective value better than or equal to that one of a given solution.

Proof. Let π be a solution such that in time period t' none of machines is serviced and $f(\pi)$ its corresponding objective value. Therefore, the operating cost of machine j incurred in time period t', i.e. $op\_cost(j,t')$ is greater or equal to a_j. However, if we service machine j in the time period t', $op\_cost(j,t')$ will be zero, while servicing cost induced by machine j in time period t' will be b_j. So, the value of such obtained solution π' will be $f(\pi') = f(\pi) - op\_cost(j,t') + b_j$. Hence, objective value of resulting solution π' can not be greater than the solution value of a given solution π. □

Solution space. Based on the problem definition (including $T \ge m$), the features of test instances (see Section 4) and Properties 3.1 and 3.2, we may conclude that the solution space of PMP consists of all vectors $\pi = (\pi_1, \pi_2, \ldots, \pi_T)$, with $\pi_t \in M$ for $t \in U$ such that $M \subset \pi$. In such representation, π_t corresponds to the index of a machine serviced in the t^{th} time period. In what follows the solution space of PMP will be denoted by P. For example

Algorithm 1: Procedure for solving PMP.

```
Function NGVNS(π);
Improve ← True;
while Improve do
    for each π' ∈ Replace(π) do
        Improve ← False;
        π'' ← GVNS(k_max, π');
        if π''is better than π then
            π ← π'';
            Improve ← True;
            break;
        end
    end
end
```

if $M = \{1, 2, 3\}$ and $T = 6$ then solution π may be represented as $\pi = \{1, 1, 2, 2, 3, 1\}$

3.2. Pseudo-code of NGVNS

In order to explore the solution space we propose new variant of VNS that we call Nested GVNS. It may be seen as an extension of the nested variable neighborhood descent (NVND) already proposed in Hansen et al. (2008); Ilić, Urošević, Brimberg, and Mladenović (2010). It applies general variable neighborhood search (GVNS) on each element of the preselected neighborhood structure, unlike NVND that applies sequential Variable neighborhood descent (VND) instead. The steps of our NGVNS are depicted at Algorithm 1.

The proposed NGVNS applies GVNS starting from each element of the neighborhood structure *Replace* of the current solution π. The neighborhood $Replace(\pi)$ contains all sets $\pi' \in P$ that may be derived from the set π by replacing one element of π (say π_j) with one from the set M, e.g $\pi_k \in M, \pi_k \neq \pi_j$. Therefore the following property holds.

Property 3.3. The cardinality of the neighborhood $Replace(\pi)$ is $O(m \cdot T)$.

If an improvement is detected, it is accepted as a new incumbent solution π and whole process is repeated starting from that solution. NGVNS finishes its work if there is no improvement. An initial solution for the proposed NGVNS is built as follows. In the first m periods all m machines are chosen to be serviced. In the remaining $T - m$ periods, machines to be serviced are chosen at random. In that way the feasibility of the initial solution is achieved. The reason why we decide to use NGVNS instead of NVND is that solution obtained by GVNS can not be worse than one obtained by VND, used within GVNS. Therefore the solution quality found by NGVNS is at least as good as one found by NVND.

Note that in Algorithm 1 the GVNS based heuristic is applied in each point of only one, i.e., *Replace neighborhood*. Clearly, as in building Nested VND, more than one neighborhood structures could be nested before GVNS is applied. That would obviously increase the procedure complexity, but enable much deeper exploration of the solution space. In solving PMP we got very good results with only one initial or higher level neighborhood structure. That is why we did not include more structures in NGVNS.

In NGVNS all neighborhoods used may be divided in 2 groups: *higher level* neighborhoods that are nested and *lower level* neighborhood structures that are used in GVNS. Of course, within GVNS, lower level neighborhoods can be used in sequential, nested and mixed nested manner (Ilić et al., 2010).

Algorithm 2: GVNS for solving PMP.

```
Function GVNS(k_max, π)
k ← 1;
while k ≤ k_max do
    π' ← Shake(π, k) ;
    π'' ← SeqVND(π') ;
    k ← k + 1;
    if π''is better than π then
        π ← π''; k ← 1;
    end
end
Return π
```

3.3. General variable neighborhood search used within NGVNS

General variable neighborhood search (GVNS) (Hansen et al., 2008; 2010) is a variant of VNS (Mladenović & Hansen, 1997) which uses Variable neighborhood descent (VND) as a local search. VND may be seen as a generalization of a local search since it explores several neighborhood structures at once instead of exploring just one. The different neighborhood structures may be explored in sequential, nested or mixed nested fashion (Ilić et al., 2010).

The proposed GVNS (Algorithm 2) includes a shaking phase used in order to escape from the local minima traps and an intensification phase in which sequential VND (seqVND) is applied. Within seqVND the following neighborhood structures of a solution π are explored: $Reverse\_two\_consecutive(\pi)$, $Shift\_backward(\pi)$, $Shift\_forward(\pi)$ and $Reverse\_part(\pi)$.

Neighborhood structures.

- $Reverse\_two\_consecutive(\pi)$ (1-opt) – the neighborhood structure consists of all solutions obtained from the solution π swapping two consecutive elements of π (see e.g. Mladenovic, Todosijevic, & Urosevic, 2012). The complexity of this neighborhood structure is $O(T)$.
- $Shift\_backward(\pi)$ (Or-opt) – the neighborhood structure consists of all solutions obtained from the solution π moving some element π_t backward immediately after some element π_s for all $s > t$. The complexity of this neighborhood structure is $O(T^2)$.
- $Shift\_forward(\pi)$ (Or-opt)– the neighborhood structure consists of all solutions obtained from the solution π moving some element π_t immediately after some element π_s for all $s > t$. The complexity of this neighborhood structure is $O(T^2)$.
- $Reverse\_part(\pi)$ (2-opt) – the neighborhood structure consisted of all solutions obtained from the solution π reversing a subsequence of π. Each solution in this neighborhood structure is deduced from the solution π reversing the part starting at π_t and ending at π_s $(t < s)$ and therefore the complexity of this neighborhood structure is $O(T^2)$. In other words from a solution $\pi = \{\pi_1, \ldots, \pi_t, \pi_{t+1}, \ldots, \pi_s, \ldots, \pi_T\}$ we will obtain $\pi' = \{\pi_1, \ldots, \pi_s, \ldots, \pi_{t+1}, \pi_t, \ldots, \pi_T\}$. In fact, this neighborhood structure is a generalization of the $Reverse\_two\_consecutive(\pi)$ since it permits reversing the part of solution π consisted of more than two consecutive elements.

The reason why we embedded these neighborhood structures within seqVND scheme is that all of them are based on changing order of servicing machines. In Algorithm 3 we give pseudo-code for our seqVND. Note that neighborhoods are changed according to "first improvement" strategy.

Shaking. The Shaking phase of GVNS, is presented at Algorithm 4. It takes as input the solution π and the parameter k. At the output it returns the solution obtained after performing k-times random shift move on π. Each random shift consists of

Algorithm 3: SeqVND .

```
Function SeqVND(π);
while there is an improvement do
    π' ← Reverse_two_consecutive(π);
    if(π' better than π) then {π ← π';continue;}
    π' ← Shift_backward(π);
    if(π' better than π) then {π ← π';continue;}
    π' ← Shift_forward(π);
    if(π' better than π) then {π ← π';continue;}
    π' ← Reverse_part(π);
    if(π' better than π) then {π ← π';continue;}
end
```

Algorithm 4: Shaking procedure.

```
Function Shake(π, π',k);
for i = 1 to k do
    select π' ∈ Shift_backward(π) ∪ Shift_forward(π) at random;
    π ← π';
end
return π'
```

inserting an element in π at random either backward or forward (*Shift_backward* and *Shift_forward*).

4. Computational results

All experiments described in this section have been carried out on a personal computer with Intel i3 2.53 gigahertz CPU and 3 gigabyte RAM memory. For testing purposes we consider the test instances proposed in Grigoriev et al. (2006) as well as a set of large test instances generated by us. The instances proposed in Grigoriev et al. (2006) have no more than 10 machines. Thus, we propose new set of 30 instances generated setting the parameter T to 52 (what corresponds to the number of weeks of a year) and setting the number of machines m to 15, 20 and 30. For each choice of T and m values 10 instances were generated choosing a_i and b_i values as random integer numbers from intervals [10,50] and [1,20], respectively. These instances as well as the corresponding best known solutions are publicly available at https://sites.google.com/site/dataforpmp/data.

Note that in tables presented in this section, for each instance we report the average maintenance and operating cost of a solution (i.e., the objective value divided by T) as it was done in Grigoriev et al. (2006).

4.1. Parameter calibration of NGVNS heuristic

In this section, we examine the influence of the parameter k_{max} on NGVNS heuristic. We tested different values for k_{max} for instances of all sizes. However, to save the space, here, we present results on 10 test instances generated by us with $m = 20$ and $T = 52$. These instances are of large size and reveal the typical performance of NGVNS on most of instances. The testing is performed by varying the value of k_{max} from 5 to 30 with step 5. The obtained results are presented in Table 3. For each choice of k_{max} value, we report the average solution value found, as well as the average CPU time consumed upon reaching these solutions.

From the results presented in Table 3, it follows that that NGVNS is not very sensitive on value of k_{max} parameter. Namely the average solution values as well as the average CPU times consumed for different values of k_{max} are very close. However, it turns out that NGVNS offers the best solution values when the parameter k_{max} is set to 10 consuming the least amount of CPU time. Therefore, for the rest of testing, we set k_{max} to 10.

Table 3
Comparison of NGVNS with different values of k_{max} parameter.

k_{max}	Av.value	Av.time
5	971.721	117.591
10	971.652	110.811
15	971.659	123.081
20	971.758	111.581
25	971.675	114.396
30	971.679	112.435

4.2. Testing local search procedures

In this section, we compare local searches in the five defined neighborhood structures (i.e., *Reverse_two_consecutive*(π), *Shift_backward*(π), *Shift_forward*(π), *Reverse_part*(π), and *Replace*(π)). These local searches are tested on the instance with $m = 20$ and $T = 52$. Each local search is executed 1000 times, starting each time from a different random solution. The summarized results are reported in Table 4, where columns 2, 3, and 4 give respectively the minimum, the average, and the maximum deviation percent from the best known solution over 1000 runs. Columns 5, 6, and 7 report the minimum, average and maximum distance between the generated local optima over 1000 runs and the best known solution. The distance between solutions π^1 and π^2 is defined as follows:

$$d(\pi^1, \pi^2) = |\{t : \pi_t^1 \neq \pi_t^2\}|.$$

The last column reports the average computing time spent to reach a local minimum (in seconds). Fig. 2 (also known as a distance-to-target diagram) shows the distribution of local minima, where each point (x, y) plots the distance x and percentage deviation y of the local minimum from the best-known solution.

Comparing the results in Table 4 we observe that:

- It turns out that local searches in *Reverse_two_consecutive*, *Shift_backward*, *Shift_forward* and *Reverse_part* neighborhoods return solutions of very similar quality.
- The local search in the *Reverse_two_consecutive* neighborhood is faster than the others, but it provides the worst quality (regarding the average deviation objective value).
- The local search with respect to *Replace* neighborhood significantly outperforms all the others regarding quality of the obtained local optima (i.e., percentage deviation of objective value). In addition this local search is the second fastest.

These observations justify the chosen order of neighborhoods within seqVND. Indeed, it is usual to explore neighborhoods in the increasing order with respect to their power. Also it should be emphasized that in papers (Ilić et al., 2010; Todosijević, Urošević, Mladenović, & Hanafi, 2015), the most powerful neighborhood is chosen as *higher level* neighborhood that is nested as it is done here nesting *Replace* neighborhood.

4.3. NGVNS versus GVNS

In this section we compare NGVNS proposed in Section 3 and GVNS heuristic that follows rules described in Section 3.3. The compared GVNS heuristic, denoted GVNS_PMP, uses as a shaking procedure one that has been presented in Algorithm 4 while as a local search uses a seqVND that explores *Reverse_two_consecutive*, *Shift_backward*, *Shift_forward*, *Reverse_part* and *Replace* neighborhoods in that order. Note that steps of such one seqVND procedure may de deduced directly from the Algorithm 3. As a stoping

Table 4
Comparison of different local search procedures.

Local search	Deviation (percent)			Distance			
	Min	Avg.	Max	Min	Avg.	Max	Time
Reverse_two_consecutive	7.265	39.205	75.208	44	49.464	51	0.009
Shift_backward	7.265	39.117	75.296	45	49.140	50	0.116
Shift_forward	7.285	39.054	75.167	44	49.640	50	0.203
Reverse_part	7.199	39.049	75.245	41	48.764	51	0.114
Replace	2.741	5.480	8.873	42	46.276	52	0.088

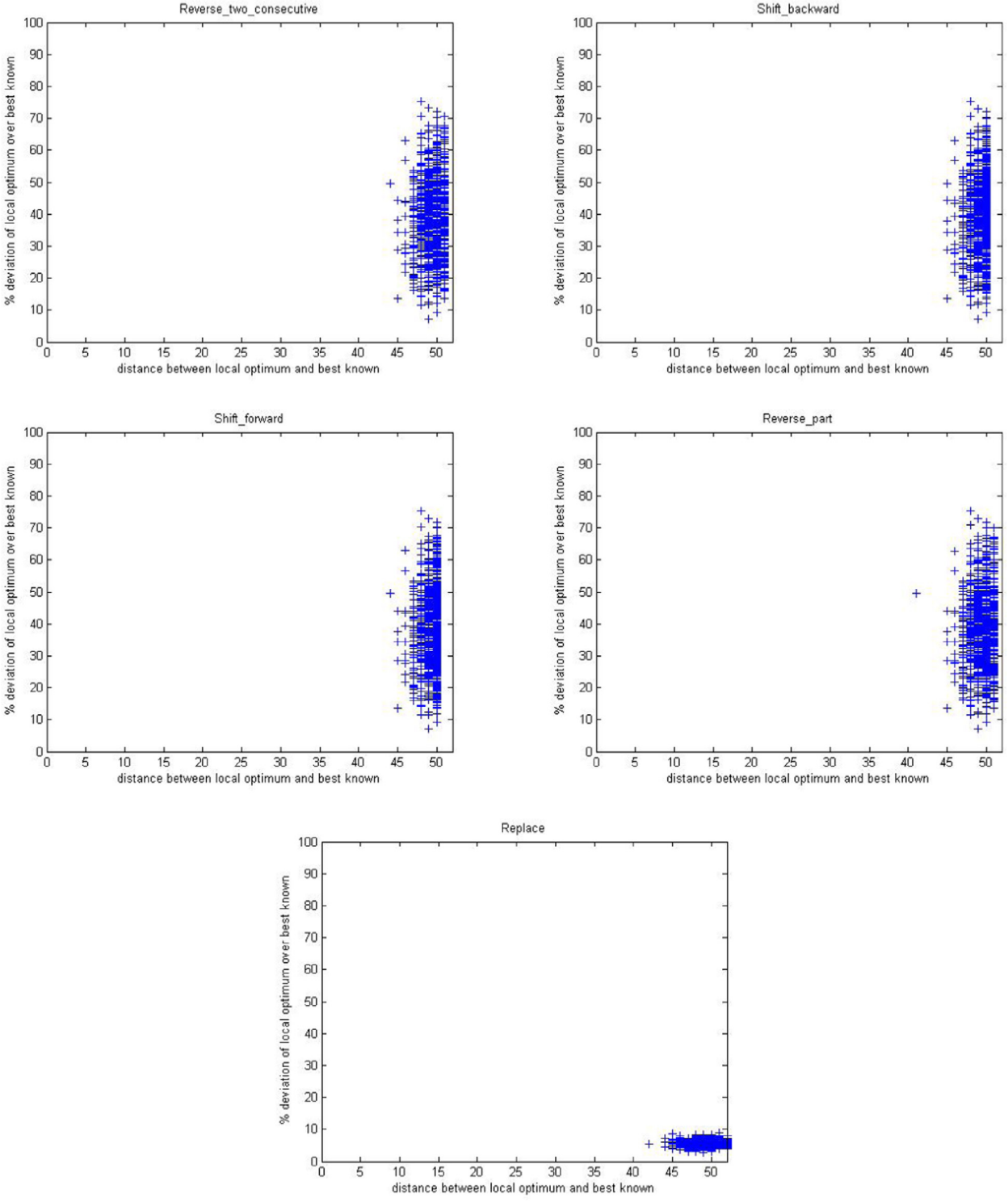

Fig. 2. Distribution of 1000 local minima on distance-to-target diagram for different local search algorithms.

criterion, GVNS_PMP uses the maximum CPU time allowed which is set to 300 seconds.

Here we present comparison of GVNS_PMP and NGVNS on the largest instances generated by us since on small instances from Grigoriev et al. (2006) there is no significant difference between GVNS_PMP and NGVNS. The comparison is presented in Table 5. For each heuristic we report the value found on each test instance (Columns 'value') and time consumed upon reaching this value (Columns 'time'). Finally, in columns 'Dev. (percent)' we report the percentage deviations of values returned by GVNS_PMP from corresponding values offered by NGVNS.

From the presented results follows that NGVNS significantly outperform GVNS_PMP regarding the solution values returned. Regarding average CPU time consumed upon reaching the reported value it follows that GVNS_PMP is faster than NGVNS on instances with $m = 15$ and $m = 30$, while on instances with $m = 20$ NGVNS is faster. Such behavior of NGVNS may be explained by the fact that NGVNS performs more thorough exploration of the solution space (based on nesting neighborhoods) than GVNS_PMP. Thus, within the imposed time limit, NGVNS is able reach high quality solutions and efficiently avoid possible local optima traps.

4.4. Computational results on instances from the literature

Keeping in mind that NGVNS performs much better than GVNS_PMP, in this section, we compare it with four exact methods for solving the PMP. These exact methods differ in mathematical programming formulation: our new formulation (see Section 2.5 of this paper) and another three taken from Grigoriev et al. (2006): MIP model (see Section 2.2), flow formulation (FF) model (see Section 2.3) and set partitioning formulation (SP) (see Section 2.4). All mathematical models, except SP model, were solved using the MIP solver IBM ILOG CPLEX 12.4. The time limit for MIP solver were set to 3600 seconds.

For testing purposes we consider the same test instances proposed in Grigoriev et al. (2006). Comparative results are reported in Tables 6–10. The results reported for NGVNS heuristic corresponds to the results of a single run. All results reported in Tables 6–10 are obtained on the same computer except those obtained by SP based model. Due to different computing facilities (SP based model were run on a computer with AMD Athlon 2400 XP+ CPU, while all the others were run on an Intel i3 machine), we have normalized the computational times of SP based model using the approach described in Dongarra (2014) and data from http://www.cpubenchmark.net/. All comparisons were made according to the Passmark CPU Score (PCPUS). The running times were normalized by using our machine as the reference point, i.e., Norm.Time(SP)= PCPUS(AMD Athlon 2400 XP+)Time(SP)/PCPUS(Intel i3).

In Tables 6–10 the following abbreviations are used:

- OPT – the average maintenance and operating cost of the optimal solution (i.e., the objective value divided by T)
- MIP value – the average maintenance and operating cost of solution obtained solving MIP formulation proposed in Grigoriev et al. (2006)
- MIP time – CPU time, in seconds, consumed by MIP solver to solve MIP formulation proposed in Grigoriev et al. (2006).
- SP – normalized CPU time, in seconds, consumed to solve an instance using set-partitioning (SP) formulation.
- FF – CPU time, in seconds, needed to solve an instance using flow formulation (FF) formulation.
- new-MIP – CPU time, in seconds, spent by MIP solver to solve MIP formulation proposed in this paper.

Table 5
NGVNS versus GVNS.

Instance	Instances with $m = 15$ and $T = 52$					Instances with $m = 20$ and $T = 52$					Instances with $m = 30$ and $T = 52$				
	NGVNS		GVNS_PMP			NGVNS		GVNS_PMP			NGVNS		GVNS_PMP		
	Value	Time	Value	Time	Dev. (percent)	Value	Time	Value	Time	Dev. (percent)	Value	Time	Value	Time	Dev. (percent)
1	3026.38	27.42	3211.27	54.94	6.11	1030.81	88.73	1067.50	94.77	3.56	12073.31	149.68	12444.94	37.06	3.08
2	2964.33	42.07	3137.23	11.01	5.83	908.69	104.43	954.77	175.76	5.07	14029.96	194.74	14317.38	28.03	2.05
3	2512.25	29.53	2692.54	9.73	7.18	974.88	94.92	1006.35	103.97	3.23	12464.63	175.07	12582.23	41.40	0.94
4	3021.98	40.11	3070.67	20.75	1.61	1011.15	78.07	1046.81	295.74	3.53	11280.12	269.82	11323.79	186.83	0.39
5	2649.75	37.59	2733.98	30.43	3.18	1042.00	85.10	1153.37	58.56	10.69	12285.62	189.04	12354.02	48.78	0.56
6	2573.37	46.42	2886.54	3.30	12.17	1143.85	134.41	1188.54	24.14	3.91	13576.79	232.30	13657.38	42.11	0.59
7	3225.81	22.50	3348.02	59.59	3.79	893.08	108.75	949.62	150.72	6.33	12414.67	148.78	12510.29	11.02	0.77
8	3181.88	42.97	3306.83	8.36	3.93	951.73	117.78	985.73	213.24	3.57	13156.63	136.65	13313.35	290.28	1.19
9	3343.50	43.49	3504.04	14.65	4.80	962.35	118.57	987.87	180.31	2.65	11230.77	215.30	11421.27	6.95	1.70
10	3193.98	52.57	3334.50	23.94	4.40	797.98	177.36	838.27	199.78	5.05	11263.08	129.20	11368.77	25.74	0.94
Average:	2969.32	38.47	3122.56	23.67	5.30	971.65	110.81	1017.88	149.70	4.76	12377.56	184.06	12529.34	71.82	1.22

Table 6
Instances with three machines ($m = 3, b_i = 0, i \in M$).

T	a	OPT	MIP (second)	SP (second)	New-MIP (second)	FF (second)	NGVNS (second)
3	1 1 1	3	0.11	0.22	0.01	0.06	0.00
3	2 1 1	4	0.02	0.22	0.02	0.03	0.00
3	2 2 1	5	0.02	0.22	0.02	0.03	0.00
4	5 1 1	5.5	0.05	0.22	0.03	0.06	0.00
4	5 2 1	7	0.03	0.22	0.02	0.03	0.00
5	5 5 1	10	0.04	0.22	0.19	0.02	0.00
4	10 1 1	8	0.02	0.22	0.02	0.02	0.00
4	10 2 1	9.5	0.03	0.22	0.02	0.00	0.00
6	10 5 1	13.3333	0.04	0.22	0.05	0.05	0.01
16	10 10 1	17.25	3.04	24.88	0.16	0.61	0.00
8	30 1 1	14.5	0.10	0.22	0.08	0.03	0.00
17	30 2 1	17.2941	3.32	19.42	0.55	0.69	0.10
8	30 5 1	22.25	0.07	0.22	0.11	0.03	0.02
9	30 10 1	28.4444	0.05	0.22	0.09	0.11	0.02
13	30 30 1	42.9231	0.16	1.96	0.11	0.03	0.09
10	50 1 1	19	0.09	0.22	0.08	0.03	0.02
21	50 2 1	22.6667	11.74	131.81	1.64	0.98	0.12
10	50 5 1	29.5	0.15	0.44	0.08	0.05	0.03
10	50 10 1	36.5	0.14	0.22	0.09	0.03	0.03
15	50 30 1	55	0.37	5.89	0.17	0.73	0.10
17	50 50 1	66.8235	0.57	24.88	0.16	0.62	0.05
		Average time	0.960	10.111	0.176	0.202	0.028

Table 7
Instances with three machines ($m = 3, b_i = 0, i \in M$).

T	a	OPT	MIP value	MIP time (second)	SP (second)	new-MIP (second)	FF (second)	NGVNS (second)
4	1 1 1 1	6	6	0.04	0.22	0.05	0.00	0.00
9	2 1 1 1	7.3333	7.3333	0.35	0.22	0.16	0.05	0.02
10	2 2 1 1	8.8	8.8	0.68	0.22	0.14	0.05	0.05
15	2 2 2 1	10.4	10.4	8.84	0.22	0.34	0.72	0.04
6	5 1 1 1	10	10	0.06	0.22	0.11	0.02	0.00
16	5 2 1 1	11.75	11.75	17.84	0.22	0.33	0.78	0.05
22	5 2 2 1	13.7273	13.7273	2254.24	0.65	1.12	2.32	0.21
6	5 5 1 1	15	15	0.07	0.22	0.09	0.03	0.00
6	5 5 2 1	17.5	17.5	0.06	0.22	0.09	0.02	0.00
24	5 5 5 1	22.25	22.25	3600.16	0.65	2.45	2.25	0.02
6	10 1 1 1	12.5	12.5	0.08	0.22	0.05	0.01	0.00
6	10 2 1 1	15	15	0.07	0.22	0.09	0.01	0.00
6	10 2 2 1	17.5	17.5	0.09	0.22	0.09	0.02	0.00
8	10 5 1 1	19.5	19.5	0.27	0.22	0.12	0.09	0.01
6	10 5 2 1	22.5	22.5	0.08	0.22	0.06	0.03	0.00
8	10 5 5 1	27.875	27.875	0.16	0.22	0.16	0.22	0.00
8	10 10 1 1	24.5	24.5	0.13	0.22	0.12	0.02	0.01
6	10 10 2 1	27.5	27.5	0.10	0.22	0.11	0.01	0.00
9	10 10 5 1	34	34	0.26	0.22	0.14	0.08	0.00
33	10 10 10 1	40.4545	40.4545	3600.02	3.71	5.04	3.62	0.89
8	30 1 1 1	21.75	21.75	0.18	0.22	0.11	0.03	0.00
8	30 5 1 1	29.5	29.5	0.15	0.22	0.09	0.06	0.01
10	30 5 5 1	40.5	40.5	0.42	0.22	0.11	0.10	0.02
8	30 10 1 1	37	37	0.13	0.22	0.08	0.05	0.03
12	30 10 5 1	49.6667	49.6667	2.09	0.22	0.17	0.18	0.04
30	30 10 10 1	58.3333	58.3333	3600.05	4.15	6.75	4.52	0.66
26	30 30 1 1	55.8462	55.8462	2063.60	0.65	13.81	2.36	0.42
24	30 30 5 1	70.5	70.5	3094.73	0.87	3.18	3.20	0.35
14	30 30 10 1	81.5	81.5	3.02	0.22	0.36	1.25	0.05
19	30 30 30 1	108.4737	108.4737	21.09	0.22	0.69	1.17	0.22
			Average time	608.969	0.531	1.207	0.776	0.103

- `NGVNS` – CPU time (in seconds) consumed by NGVNS based heuristic to solve an instance of PMP.

Note that Table 9 does not provide results obtained by SP model since not all solution values had been provided in Grigoriev et al. (2006).

The Table 11 provides the average CPU times consumed by each solution approach on a considered data set.

From Tables 6–11 the following conclusions may be drawn:

(i) Overall NGVNS approach appears to be most reliable. It solved all test instances to the optimality in the shortest CPU time (i.e., 0.831 seconds on average for all test instances).
(ii) NGVNS needed, in most of instances, less than a second to get an optimal solution, except on instances in Table 10 which appear to be the hardest (since these instances consider the largest number of machines m (the number

Table 8
Instances with three machines ($m = 3, a_i = 1, b_i = 0, i \in M$).

T	OPT	MIP value	MIP time (second)	SP (second)	New-MIP (second)	FF (second)	NGVNS (second)
50	3.04	3.04	3577.89	11.13	3.81	4.52	0.75
51	3	3	25.96	4.58	0.47	3.54	1.84
52	3.0385	3.0385	3600.01	33.61	4.77	5.40	0.38
53	3.0377	3.0377	107.24	53.90	3.20	4.98	0.38
54	3	3	219.56	6.98	0.76	4.76	1.66
55	3.0364	3.0364	53.13	25.53	4.26	5.63	0.68
56	3.0357	3.0357	1731.56	128.76	4.73	5.13	0.46
57	3	3	16.37	10.04	0.42	3.95	1.43
58	3.0345	3.0345	1159.67	79.87	5.63	4.99	2.42
59	3.0339	3.0339	164.97	88.82	5.12	4.68	1.41
60	3	**3.0667**	3600.00	37.10	1.78	6.50	1.51
61	3.0328	3.0328	661.13	156.04	5.97	6.27	2.47
62	3.0323	3.032258	3600.52	313.60	4.79	7.67	1.47
63	3	3	233.27	42.56	1.04	6.04	2.93
64	3.0313	3.03125	3600.08	312.29	7.77	6.85	3.28
65	3.0308	3.030769	2442.31	239.62	6.15	7.93	1.70
66	3	**3.0444**	3600.11	102.57	2.62	5.57	2.81
67	3.0299	3.0299	2899.25	119.16	8.52	6.26	1.18
68	3.0294	3.0294	2610.39	170.88	5.76	7.69	3.34
69	3	3	229.87	131.16	3.46	7.91	0.88
70	3.0286	3.0286	3600.06	1123.90	6.52	7.53	1.87
80	3.025	3.025	3600.17	254.68	10.84	16.07	13.76
90	3	3.022222	3600.06	238.53	3.87	17.82	7.16
100	3.02	3.02	3600.11	571.33	16.30	24.85	16.11
		Average time	2022.237	177.360	4.940	7.606	2.994

Table 9
Instances with positive maintenance costs ($m = 5, T = 24$).

a	b	OPT	MIP value	MIP time (second)	New-MIP (second)	FF (second)	NGVNS (second)
5 1 1 1 1	0 0 0 0 0	15	15	3600.02	2.48	2.08	0.22
5 1 1 1 1	5 1 1 1 1	17.3333	17.3333	3600.01	3.00	2.26	0.36
5 1 1 1 1	30 10 5 2 1	27.0417	27.0417	3600.01	6.49	8.94	0.58
5 5 1 1 1	0 0 0 0 0	21.9583	21.9583	3600.01	31.84	2.51	0.32
5 5 1 1 1	5 5 1 1 1	25.4167	25.4167	3600.00	57.49	25.96	0.04
5 5 1 1 1	30 10 5 2 1	33.8333	**33.9583**	3600.00	9.42	28.28	0.61
5 5 5 1 1	0 0 0 0 0	29.5	29.5	3600.00	5.21	2.78	0.13
5 5 5 1 1	5 5 5 1 1	33.5	33.5	3600.07	4.63	5.46	0.06
5 5 5 1 1	30 10 5 2 1	41.125	41.125	3600.01	6.94	39.61	0.33
5 5 5 5 1	0 0 0 0 0	40.375	40.375	2160.30	229.15	8.72	0.66
5 5 5 5 1	5 5 5 5 1	44.875	44.875	3600.00	111.71	6.16	0.04
5 5 5 5 1	30 10 5 2 1	50.375	50.375	3599.99	86.30	**3600.00**	0.33
10 5 1 1 1	0 0 0 0 0	26.75	26.75	3599.99	9.89	2.95	0.48
10 5 1 1 1	10 5 1 1 1	32.125	32.125	3600.00	6.71	59.33	0.35
10 5 1 1 1	30 10 5 2 1	41	41	3600.01	27.88	38.39	0.3
10 10 5 1 1	0 0 0 0 0	43.5	**43.9167**	3600.01	48.75	4.65	0.6
10 10 5 1 1	10 10 5 1 1	50.9583	50.9583	3600.32	49.17	6.86	0.3
10 10 5 1 1	30 10 5 2 1	56.125	**56.4167**	3600.00	21.19	650.25	0.87
30 10 5 1 1	0 0 0 0 0	61.4167	61.4167	3600.01	30.50	4.18	0.28
30 10 5 1 1	30 10 5 1 1	77.4167	77.4167	3600.00	38.14	13.53	0.62
30 10 5 1 1	30 10 5 2 1	77.5	77.5	3600.01	43.34	208.28	0.52
30 30 1 1 1	0 0 0 0 0	69	69	3600.00	55.72	3.26	0.46
30 30 1 1 1	30 30 1 1 1	91.75	91.75	3599.99	26.40	3.49	0.22
30 30 1 1 1	30 10 5 2 1	84.6667	84.6667	3599.99	79.14	261.57	0.55
30 30 30 1 1	0 0 0 0 0	129.5	129.5	3600.01	202.88	4.26	0.51
30 30 30 1 1	30 30 30 1 1	155.875	155.875	3599.99	79.80	9.53	0.59
30 30 30 1 1	30 10 5 2 1	142.7917	142.7917	3600.00	103.88	**3600.00**	0.41
30 30 30 30 1	0 0 0 0 0	207.75	207.75	3600.01	47.25	3.09	0.16
30 30 30 30 1	30 30 30 30 1	236.5417	236.5417	3600.01	37.03	3.76	0.35
30 30 30 30 1	30 10 5 2 1	218.2917	218.2917	2357.60	37.99	74.46	0.21
			Average time	3510.612	50.011	289.487	0.382

of constraints and variables in MIP formulations depend on m)).

Regarding exact solution methods, several interesting observations may be derived:

(i) The behavior of SP formulation is interesting. It is the worst exact method for small instances in Table 6 but the best one for the largest instances presented in Table 10. This may be explained by the fact that increasing the number of machines by one, SP formulation gets just one additional constraint, while the other formulations get $O(T)$ additional constraints. In addition, as shown in Grigoriev et al. (2006) the LP relaxation of SP formulation turns out to be very strong.

(ii) The average computational times of FF and our new MIP formulation are very similar. Further, the optimality of the solution found for one test instance (Table 10) were not proven

Table 10
Instances with many machines ($m = 10, T = 18, b_i = 0, i \in M$).

a	OPT	MIP value	MIP time (second)	SP (second)	new-MIP (second)	FF (second)	NGVNS
1 1 1 1 1 1 1 1 1 1	49	49	**3600.02**	0.22	2318.77	**3600.00**	0.32
10 9 8 7 6 5 4 3 2 1	232	232	**3600.09**	6.33	2079.20	3157.77	0.38
10 10 10 10 10 10 10 10 10 1	413.5	413.5	**3599.25**	0.44	**3600.94**	**3600.80**	0.22
100 1 1 1 1 1 1 1 1 1	126.5	126.5	**3600.18**	0.22	2485.54	11.71	0.03
1000 1 1 1 1 1 1 1 1 1	576.5	576.5	279.32	1.53	150.05	0.76	0.05
	Average time		2935.772	1.746	2126.900	2074.208	0.200

Table 11
Average CPU times.

Instances from	MIP (second)	SP (second)	New-MIP (second)	FF (second)	NGVNS (second)
Table 6	0.960	10.111	0.176	0.202	0.028
Table 7	608.969	0.531	1.207	0.776	0.103
Table 8	2022.237	177.360	4.940	7.606	2.994
Table 10	2935.772	1.746	2126.900	2074.208	0.200
Average time	1391.985	47.437	533.306	520.698	0.831

solving new MIP formulation, while solving FF formulation the optimality were not proven for 4 test instances (2 instances in Tables 9 and 2 instances in Table 10). For all these instances reported times are boldfaced.

(iii) The advantage of FF over new MIP formulation comes from the fourth test instance in Table 10. There, new MIP formulation needs about 2500 seconds to provide an optimal solution, while FF formulation does so within 12 seconds.

(iv) The old MIP model is the least reliable. For example in Tables 9 and 10, for only two instance (out of 35) the optimal solutions have been proven within 3600 seconds. However, optimal solutions have not been reached on three instances (boldfaced values in these tables).

4.5. Computational results on large test instances

In this section we present results obtained testing NGVNS (NGVNS is again run only once on each test instance), our new formulation (see Section 2.5 of this paper), MIP formulation (see Section 2.2), and flow formulation (FF) (see Section 2.3) on large test instances generated by us. Again, all mathematical models were solved using the MIP solver IBM ILOG CPLEX 12.4 setting time limit to 3600 seconds.

The results are provided in Tables 12–14. In each table we report the average maintenance and operating costs of obtained solutions (Columns '`value`') as well as CPU times in seconds consumed by certain method to solve certain test instance (Columns '`time`'). Finally, in Columns '`Dev. (percent)`' the percentage deviations of values found by exact methods from corresponding values offered by NGVNS are reported.

From Tables 12–14 the following conclusions may be drawn:

(i) NGVNS significantly outperforms the exact methods regarding both solution quality and average computation time.

Table 12
Instances with $m = 15$ and $T = 52$.

Instance	NGVNS		MIP			New-MIP			FF		
	Value	Time	Value	Time	Dev. (percent)	Value	Time	Dev. (percent)	Value	Time	Dev. (percent)
1	3026.38	27.42	3177.90	3600.00	5.01	3042.87	3600.00	0.54	3433.83	3600.02	13.46
2	2964.33	42.07	3038.81	3600.00	2.51	2977.23	3600.01	0.44	3263.13	3600.03	10.08
3	2512.25	29.53	2662.08	3600.26	5.96	2518.52	3600.01	0.25	3023.92	3600.03	20.37
4	3021.98	40.11	3207.38	3600.00	6.14	3030.85	3600.00	0.29	3286.60	3600.02	8.76
5	2649.75	37.59	2755.94	3600.01	4.01	2655.83	3600.00	0.23	3659.83	3600.02	38.12
6	2573.37	46.42	2654.65	3600.02	3.16	2578.37	3600.00	0.19	3267.42	3600.03	26.97
7	3225.81	22.50	3347.60	3600.01	3.78	3231.60	3600.18	0.18	3642.73	3600.03	12.92
8	3181.88	42.97	3295.83	3600.00	3.58	3185.04	3600.00	0.10	4310.52	3600.03	35.47
9	3343.50	43.49	3445.04	3600.01	3.04	3365.67	3600.00	0.66	4642.42	3600.02	38.85
10	3193.98	52.57	3321.73	3600.00	4.00	3206.38	3600.00	0.39	4163.21	3600.03	30.35
Average	2969.32	38.47	3090.70	3600.03	4.12	2979.23	3600.02	0.33	3669.36	3600.03	23.53

Table 13
Instances with $m = 20$ and $T = 52$.

Instance	NGVNS		MIP			New-MIP			FF		
	Value	Time	Value	Time	Dev. (percent)	Value	Time	Dev. (percent)	Value	Time	Dev. (percent)
1	1030.81	88.73	1068.12	3600.01	3.62	1033.98	3600.02	0.31	2459.17	3600.07	138.57
2	908.69	104.43	931.92	3600.00	2.56	913.96	3600.00	0.58	1619.50	3600.08	78.22
3	974.88	94.92	1015.58	3600.00	4.17	980.65	3600.02	0.59	1344.52	3600.03	37.92
4	1011.15	78.07	1049.90	3600.01	3.83	1016.46	3600.00	0.52	2080.87	3600.03	105.79
5	1042.00	85.10	1093.96	3600.01	4.99	1043.98	3600.00	0.19	1399.75	3600.03	34.33
6	1143.85	134.41	1183.42	3600.01	3.46	1149.69	3600.00	0.51	2812.77	3600.05	145.90
7	893.08	108.75	930.10	3600.00	4.15	896.40	3600.00	0.37	1378.08	3600.01	54.31
8	951.73	117.78	978.38	3600.00	2.80	954.98	3600.02	0.34	1862.40	3600.03	95.69
9	962.35	118.57	996.50	3600.01	3.55	965.46	3600.00	0.32	2417.42	3600.07	151.20
10	797.98	177.36	826.87	3600.00	3.62	803.96	3600.01	0.75	1333.48	3600.03	67.11
Average	971.65	110.81	1007.48	3600.01	3.67	975.95	3600.01	0.45	1870.80	3600.04	90.90

Table 14
Instances with $m = 30$ and $T = 52$.

Instance	NGVNS		MIP			New-MIP			FF		
	Value	Time	Value	Time	Dev. (percent)	Value	Time	Dev. (percent)	Value	Time	Dev. (percent)
1	12073.31	149.68	12399.19	3600.01	2.70	12101.44	3600.01	0.23	17299.92	3600.06	43.29
2	14029.96	194.74	14567.96	3600.01	3.83	14080.08	3600.01	0.36	18353.67	3600.04	30.82
3	12464.63	175.07	13072.02	3600.00	4.87	12571.35	3600.01	0.86	16805.62	3600.03	34.83
4	11280.12	269.82	11416.73	3600.01	1.21	11312.10	3600.00	0.28	13806.85	3600.02	22.40
5	12285.62	189.04	12907.27	3600.02	5.06	12325.65	3600.02	0.33	16282.27	3600.04	32.53
6	13576.79	232.30	13961.17	3600.01	2.83	13674.13	3600.00	0.72	16816.40	3600.05	23.86
7	12414.67	148.78	13614.21	3600.01	9.66	12444.38	3600.02	0.24	16159.79	3600.04	30.17
8	13156.63	136.65	14073.12	3600.01	6.97	13218.85	3600.00	0.47	16885.98	3600.05	28.35
9	11230.77	215.30	11532.90	3600.02	2.69	11260.42	3600.00	0.26	16436.19	3600.11	46.35
10	11263.08	129.20	11691.90	3600.03	3.81	11303.52	3600.00	0.36	14478.12	3600.07	28.54
Average	12377.56	184.06	12923.65	3600.01	4.36	12429.19	3600.01	0.41	16332.48	3600.05	32.11

There is no test instance where NGVNS offered worse solution than any of exact methods. Furthermore, NGVNS consumes not more than 270 seconds to solve an instance while exact methods spend 3600 seconds on each test instance.

(ii) Regarding the solution quality the considered formulations may be ranked as follows. The best one turns out to be our new MIP (the percentage deviations of new MIP solutions from corresponding NGVNS solutions lies in range 0.10–0.75 percent), followed by old MIP formulation (the percentage deviations of new MIP solutions from corresponding NGVNS solutions lies in range 1.21–9.69 percent). Finally, the last place is taken by FF formulation which offers solution values 10 percent or more far from those obtained applying NGVNS.

5. Concluding remarks

In this paper, we study the periodic maintenance problem (PMP) that consists of finding the cyclic maintenance schedule of machines in a given time period. We present a new MIP formulation for PMP and compare its performance with the models from the literature. Due to the limitations of these models, we also propose a heuristic method based on the Nested General Variable Neighborhood Search (NGVNS) to tackle hard instances. Computational results show that the proposed NGVNS approach is very efficient. On all 110 test instances with known optimal solutions NGVNS succeeded to find these optimal solutions. Moreover, NGVNS heuristic needed only 0.831 seconds on average to solve them. In addition, on large test instances proposed in this paper NGVNS performs much better than any exact approach. These large test instances also reveal superiority of our new MIP formulation over two previous formulations.

In future work it will be interesting to compare different metaheuristic approaches for solving PMP (such as Simulated Annealing, Genetic Algorithm, Tabu Search, etc. or even hybrid approaches (e.g., matheuristics)). It will be very convenient in the future research to generalize the PMP model taking into account more practical issues. For example number of machines serviced in each period may be greater than one. Future research may also include proposing extensions of PMP model as well as development of NGVNS based heuristics to solve resulting problems.

Acknowledgments

This research has been supported by International Chair “OPTIFER” as well as by ELSAT program.

The authors would like to thank the anonymous referees for their helpful suggestions which have improved the presentation of this paper.

References

Benmansour, R., Allaoui, H., Artiba, A., Iassinovski, S., & Pellerin, R. (2011). Simulation-based approach to joint production and preventive maintenance scheduling on a failure-prone machine. *Journal of Quality in Maintenance Engineering, 17*(3), 254–267.

Carrizosa, E., Mladenović, N., & Todosijević, R. (2013). Variable neighborhood search for minimum sum-of-squares clustering on networks. *European Journal of Operational Research, 230*(2), 356–363.

Cassady, C., & Kutanoglu, E. (2005). Integrating preventive maintenance planning and production scheduling for a single machine. *IEEE Transactions on Reliability, 54*(2), 304–309.

Dongarra, J. J. (2014). *Performance of various computers using standard linear equations software*. University of Manchester. CS - 89 - 85.

Grigoriev, A., Van De Klundert, J., & Spieksma, F. (2006). Modeling and solving the periodic maintenance problem. *European Journal of Operational Research, 172*(3), 783–797.

Guo, P., Chen, W., & Wang, Y. (2014). A general variable neighborhood search for single-machine total tardiness scheduling problem with step-deteriorating jobs. *Journal of Industrial and Management Optimization, 10*(4), 1071–1090.

Hanafi, S., Lazić, J., Mladenović, N., Wilbaut, C., & Crevits, I. (2015). New variable neighbourhood search based 0-1 mip heuristics.. *Yugoslav Journal of Operations Research, 25*(3), 343–360.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research, 130*(3), 449–467.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2008). Variable neighbourhood search: methods and applications. *4OR, 6*(4), 319–360.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research, 175*(1), 367–407.

Ilić, A., Urošević, D., Brimberg, J., & Mladenović, N. (2010). A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. *European Journal of Operational Research, 206*(2), 289–300.

Lazić, J., Todosijević, R., Hanafi, S., & Mladenović, N. (2014). Variable and single neighbourhood diving for mip feasibility. *Yugoslav Journal of Operations Research*. doi:10.2298/YJOR140417027L.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100.

Mladenovic, N., Todosijevic, R., & Urosevic, D. (2012). An efficient general variable neighborhood search for large travelling salesman problem with time windows. *Yugoslav Journal of Operations Research, 23*(1), 19–30.

Mladenović, N., Urošević, D., Hanafi, S., & Ilić, A. (2012). A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research, 220*(1), 270–285.

Mobley, R. (2002). *An introduction to predictive maintenance*. Butterworth-Heinemann.

Todosijević, R., Urošević, D., Mladenović, N., & Hanafi, S. (2015). A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. *Optimization Letters*. doi:10.1007/s11590-015-0867-6.

Weinstein, L., & Chung, C. (1999). Integrating maintenance and production decisions in a hierarchical production planning environment. *Computers & Operations Research, 26*(10), 1059–1074.