Les Cahiers du GERAD ISSN: 0711–2440

Nested general variable neighborhood search for the periodic maintenance problem

R. Todosijević, R. Benmansour,
S. Hanafi, N. Mladenović, A. Artiba

G–2015–40

April 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Nested general variable neighborhood search for the periodic maintenance problem

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

[a] *LAMIH – Université de Valenciennes et du Hainaut-Cambrésis, 59313 Valenciennes Cedex 9, France*

[b] *GERAD, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

racatodosijevic@gmail.com
rachid.benmansour@univ-valenciennes.fr
said.hanafi@univ-valenciennes.fr
nenad.mladenovic@gerad.ca
abdelhakim.artiba@univ-valenciennes.fr

April 2015

Les Cahiers du GERAD
G–2015–40

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.

Key Words: Scheduling, preventive maintenance, mixed-integer linear programming, variable neighborhood search, nested general VNS.

Acknowledgments: This research has been supported by International Chair grant devoted to Université de Valenciennes, France.

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 [1]. 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 [3]. Hence, there are conflicts between maintenance planning, and production scheduling and consequently there is good reasons to try minimizing the cost of the two functions [14].

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 k machines. Obviously the parameter k depends on the capacity of the maintenance service. In our problem $k = 1$. 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 repair made within a scheduled or preventive mode [13]. Scheduling the repair minimizes the repair time and associated labor costs. It also reduces the negative impacts of expedited shipments and lost production.

In this paper we consider the *periodic maintenance problem* (PMP). 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 lenght 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, 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 $f(\pi^*) = 176$. Figure 1, also presents it.

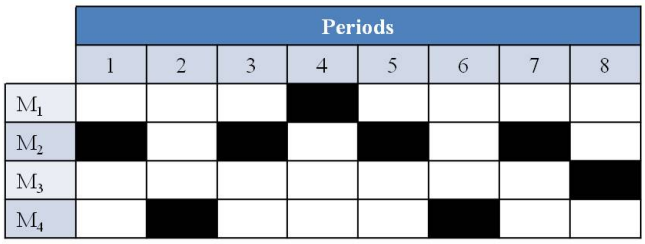

Figure 1: Example with $m = 4$ and $T = 8$, with optimal solution $\pi^* = (2, 4, 2, 1, 2, 4, 2, 3)$ and $f(\pi^*) = 176$.

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 et al. [4] 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:

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

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

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

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

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

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

The authors in [4] 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:

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

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

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

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

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

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

The objective function (6) minimizes the sum of operating costs and servicing costs. Equations (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:

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

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

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

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

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

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

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.

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

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

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

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

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 [4]). Furthermore, the LP relaxation of SP is stronger than the LP relaxation of FF (see [4]).

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:

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

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

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

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

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

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

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

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

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

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 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.

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

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.

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

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

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

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

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

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

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

$$\tag{39}$$

The following Table 2 gives a brief comparison of four formulations of the studied PMP problem. It appears that models 1 and 4 contains the less number of variables than other two.

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

Formulation	# of constraints	# of integer variables	# of binary variables
Linearisation 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

3 Nested general variable neighborhood search for the PMP

Variable Neighborhood Search (VNS) [11, 8, 7, 6] 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. [8, 7, 2, 10, 5] 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 $T \geq m$), 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 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 [7, 9]. 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.

Algorithm 1: Procedure for solving PMP

```
   Function NGVNS(π);
1  Improve ← True;
   while Improve do
2      for each π′ ∈ Replace(π) do
3          Improve ← False;
4          π″ ← GVNS(k_max, π′);
5          if π″ is better than π then
6              π ← π″;
7              Improve ← True;
8              break;
           end
       end
   end
```

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 fashion [9].

3.3 General variable neighborhood search used within NGVNS

General variable neighborhood search (GVNS) [7, 8] is a variant of VNS [11] 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 [9].

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 $Reverse\_part(\pi)$.

Algorithm 2: GVNS for solving PMP.

Function GVNS(k_{max}, π)
$k \leftarrow 1$;
while $k \leq k_{max}$ **do**
  $\pi' \leftarrow Shake(\pi, k)$;
  $\pi'' \leftarrow SeqVND(\pi')$;
  $k \leftarrow k + 1$;
  if π''*is better than* π **then**
    $\pi \leftarrow \pi''$; $k \leftarrow 1$;
  end
end
Return π

Neighborhood structures.

- *Reverse_two_consecutive*(π) (1-opt) - the neighborhood structure consists of all solutions obtained from the solution π swapping two consecutive elements of π (see e.g. [12]). 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*(π) (2-opt) - the neighborhood structure consisted of all solutions obtained from the solution π reversing a sub-sequence 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*(π) 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.

Algorithm 3: SeqVND

Function SeqVND(π);
while *there is an improvement* **do**
  $\pi' \leftarrow Reverse\_two\_consecutive(\pi)$;
  if(π' better than π) **then** {$\pi \leftarrow \pi'$;**continue;**}
  $\pi' \leftarrow Shift\_backward(\pi)$;
  if(π' better than π) **then** {$\pi \leftarrow \pi'$;**continue;**}
  $\pi' \leftarrow Shift\_forward(\pi)$;
  if(π' better than π) **then** {$\pi \leftarrow \pi'$;**continue;**}
  $\pi' \leftarrow Reverse\_part(\pi)$;
  if(π' better than π) **then** {$\pi \leftarrow \pi'$;**continue;**}
end

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 inserting an element in π at random either backward or forward ($Shift\_backward$ and $Shift\_forward$).

Algorithm 4: Shaking procedure

Function `Shake(`π, π'`,k)`;
1 **for** $i = 1$ **to** k **do**
2 | select $\pi' \in$ `Shift_backward`$(\pi) \cup$ `Shift_forward`(π) at random;
3 | $\pi \leftarrow \pi'$;
end
4 **return** π'

4 Computational results

In this section we compare five methods for solving the PMP. Four among them are exact and differ in mathematical programming formulation: our new formulation (see Section 2.5 of this paper) and another three taken from [4]: MIP model (see Section 2.2), flow formulation (FF) model (see Section 2.3) and set partitioning formulation (SP) (see Section 2.4). The last method included in comparison is NGVNS based heuristic presented in Section 3. After some preliminary testing, the GVNS parameter k_{max} has been set to 10. The numerical experiments were carried on a personal computer with 2.53GHz CPU and 3GB RAM memory. 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 300 seconds.

For testing purposes we consider the same test instances proposed in Grigoriev et al.[4]. Comparative results are reported in Tables 3–7. All results reported in Tables 3–7 are obtained on the same computer except those obtained by SP based model. We simply copied them from [4]. In the tables the following abbreviations are used:

- `OPT` – optimal solution value
- `MIP value` – value of solution obtained solving MIP formulation proposed in Grigoriev et al.[4]
- `MIP time` – CPU time, in seconds, consumed by MIP solver to solve MIP formulation proposed in Grigoriev et al.[4].
- `SP` – consumed CPU time, in seconds, for solving 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.
- `NGVNS` – CPU time (in seconds)consumed by NGVNS based heuristic to solve an instance of PMP.

Note that Table 6 does not provide results obtained by SP model since not all solution values had been provided in [4].

The Table 8 provides the average CPU times consumed by each solution approach on a considered data set.

From Tables 3–8 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 5 which appears to be the hardest.

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

T	a	OPT	MIP (s)	SP (s)	new-MIP (s)	FF (s)	NGVNS (s)
3	1 1 1	3	0.11	1	0.01	0.06	0.00
3	2 1 1	4	0.02	1	0.02	0.03	0.00
3	2 2 1	5	0.02	1	0.02	0.03	0.00
4	5 1 1	5.5	0.05	1	0.03	0.06	0.00
4	5 2 1	7	0.03	1	0.02	0.03	0.00
5	5 5 1	10	0.04	1	0.19	0.02	0.00
4	10 1 1	8	0.02	1	0.02	0.02	0.00
4	10 2 1	9.5	0.03	1	0.02	0.00	0.00
6	10 5 1	13.3333	0.04	1	0.05	0.05	0.01
16	10 10 1	17.25	3.04	114	0.16	0.61	0.00
8	30 1 1	14.5	0.10	1	0.08	0.03	0.00
17	30 2 1	17.2941	3.32	89	0.55	0.69	0.10
8	30 5 1	22.25	0.07	1	0.11	0.03	0.02
9	30 10 1	28.4444	0.05	1	0.09	0.11	0.02
13	30 30 1	42.9231	0.16	9	0.11	0.03	0.09
10	50 1 1	19	0.09	1	0.08	0.03	0.02
21	50 2 1	22.6667	11.74	604	1.64	0.98	0.12
10	50 5 1	29.5	0.15	2	0.08	0.05	0.03
10	50 10 1	36.5	0.14	1	0.09	0.03	0.03
15	50 30 1	55	0.37	27	0.17	0.73	0.10
17	50 50 1	66.8235	0.57	114	0.16	0.62	0.05
	Average time		0.960	46.333	0.176	0.202	0.028

Table 4: Instances with three machines ($m = 3, b_i = 0, i \in M$)

T	a	OPT	MIP value	MIP time(s)	SP (s)	new-MIP (s)	FF (s)	NGVNS (s)
4	1 1 1 1	6	6	0.04	1	0.05	0.00	0.00
9	2 1 1 1	7.3333	7.3333	0.35	1	0.16	0.05	0.02
10	2 2 1 1	8.8	8.8	0.68	1	0.14	0.05	0.05
15	2 2 2 1	10.4	10.4	8.84	1	0.34	0.72	0.04
6	5 1 1 1	10	10	0.06	1	0.11	0.02	0.00
16	5 2 1 1	11.75	11.75	17.84	1	0.33	0.78	0.05
22	5 2 2 1	13.7273	13.7273	300.01	3	1.12	2.32	0.21
6	5 5 1 1	15	15	0.07	1	0.09	0.03	0.00
6	5 5 2 1	17.5	17.5	0.06	1	0.09	0.02	0.00
24	5 5 5 1	22.25	22.25	300.16	3	2.45	2.25	0.02
6	10 1 1 1	12.5	12.5	0.08	1	0.05	0.01	0.00
6	10 2 1 1	15	15	0.07	1	0.09	0.01	0.00
6	10 2 2 1	17.5	17.5	0.09	1	0.09	0.02	0.00
8	10 5 1 1	19.5	19.5	0.27	1	0.12	0.09	0.01
6	10 5 2 1	22.5	22.5	0.08	1	0.06	0.03	0.00
8	10 5 5 1	27.875	27.875	0.16	1	0.16	0.22	0.00
8	10 10 1 1	24.5	24.5	0.13	1	0.12	0.02	0.01
6	10 10 2 1	27.5	27.5	0.10	1	0.11	0.01	0.00
9	10 10 5 1	34	34	0.26	1	0.14	0.08	0.00
33	10 10 10 1	40.4545	**41.0606**	300.02	17	5.04	3.62	0.89
8	30 1 1 1	21.75	21.75	0.18	1	0.11	0.03	0.00
8	30 5 1 1	29.5	29.5	0.15	1	0.09	0.06	0.01
10	30 5 5 1	40.5	40.5	0.42	1	0.11	0.10	0.02
8	30 10 1 1	37	37	0.13	1	0.08	0.05	0.03
12	30 10 5 1	49.6667	49.6667	2.09	1	0.17	0.18	0.04
30	30 10 10 1	58.3333	58.3333	300.18	19	6.75	4.52	0.66
26	30 30 1 1	55.8462	55.8462	300.01	3	13.81	2.36	0.42
24	30 30 5 1	70.5	70.5	300.01	4	3.18	3.20	0.35
14	30 30 10 1	81.5	81.5	3.02	1	0.36	1.25	0.05
19	30 30 30 1	108.4737	108.4737	21.09	1	0.69	1.17	0.22
		Average time		61.890	2.433	1.207	0.776	0.103

Table 5: Instances with three machines ($m = 3, a_i = 1, b_i = 0, i \in M$)

T	OPT	MIP value	MIP time(s)	SP (s)	new-MIP (s)	FF (s)	NGVNS (s)
50	3.04	3.04	300.02	51	3.81	4.52	0.75
51	3	3	25.96	21	0.47	3.54	1.84
52	3.0385	3.0385	300.01	154	4.77	5.40	0.38
53	3.0377	3.0377	107.24	247	3.20	4.98	0.38
54	3	3	219.56	32	0.76	4.76	1.66
55	3.0364	3.0364	53.13	117	4.26	5.63	0.68
56	3.0357	3.0357	300.12	590	4.73	5.13	0.46
57	3	3	16.37	46	0.42	3.95	1.43
58	3.0345	3.0345	300.01	366	5.63	4.99	2.42
59	3.0339	3.0339	164.97	407	5.12	4.68	1.41
60	3	**3.0667**	300.06	170	1.78	6.50	1.51
61	3.0328	3.0328	300.01	715	5.97	6.27	2.47
62	3.0323	**3.0882**	300.08	1437	4.79	7.67	1.47
63	3	3	233.27	195	1.04	6.04	2.93
64	3.0313	**3.0571**	300.08	1431	7.77	6.85	3.28
65	3.0308	**3.05**	207.69	1098	6.15	7.93	1.70
66	3	**3.0444**	46.11	470	2.62	5.57	2.81
67	3.0299	**3.16**	10.51	546	8.52	6.26	1.18
68	3.0294	**3.0882**	300.09	783	5.76	7.69	3.34
69	3	3	229.87	601	3.46	7.91	0.88
70	3.0286	**3.0571**	300.06	5150	6.52	7.53	1.87
80	3.025	3.025	300.17	1167	10.84	16.07	13.76
90	3	**3.0444**	300.06	1093	3.87	17.82	7.16
100	3.02	3.02	300.11	2618	16.30	24.85	16.11
	Average time		217.315	812.708	4.940	7.606	2.994

Table 6: Instances with positive maintenance costs ($m = 5, T = 24$)

a	b	OPT	MIP value	MIP time(s)	new-MIP (s)	FF (s)	NGVNS (s)
5 1 1 1 1	0 0 0 0 0	15	15	300.01	2.48	2.08	0.22
5 1 1 1 1	5 1 1 1 1	17.3333	17.3333	300.01	3.00	2.26	0.36
5 1 1 1 1	30 10 5 2 1	27.0417	**27.125**	300.00	6.49	8.94	0.58
5 5 1 1 1	0 0 0 0 0	21.9583	21.9583	300.01	31.84	2.51	0.32
5 5 1 1 1	5 5 1 1 1	25.4167	25.4167	300.00	57.49	25.96	0.04
5 5 1 1 1	30 10 5 2 1	33.8333	**34.125**	300.00	9.42	28.28	0.61
5 5 5 1 1	0 0 0 0 0	29.5	29.5	300.15	5.21	2.78	0.13
5 5 5 1 1	5 5 5 1 1	33.5	**33.9167**	300.01	4.63	5.46	0.06
5 5 5 1 1	30 10 5 2 1	41.125	41.125	300.00	6.94	39.61	0.33
5 5 5 5 1	0 0 0 0 0	40.375	40.375	300.14	229.15	8.72	0.66
5 5 5 5 1	5 5 5 5 1	44.875	44.875	300.02	111.71	6.16	0.04
5 5 5 5 1	30 10 5 2 1	50.375	50.375	300.17	86.30	**300.00**	0.33
10 5 1 1 1	0 0 0 0 0	26.75	26.75	300.07	9.89	2.95	0.48
10 5 1 1 1	10 5 1 1 1	32.125	**32.25**	300.12	6.71	59.33	0.35
10 5 1 1 1	30 10 5 2 1	41	41	300.17	27.88	38.39	0.3
10 10 5 1 1	0 0 0 0 0	43.5	**43.9167**	300.03	48.75	4.65	0.6
10 10 5 1 1	10 10 5 1 1	50.9583	50.9583	300.01	49.17	6.86	0.3
10 10 5 1 1	30 10 5 2 1	56.125	**56.625**	300.10	21.19	**300.00**	0.87
30 10 5 1 1	0 0 0 0 0	61.4167	61.4167	300.11	30.5	4.18	0.28
30 10 5 1 1	30 10 5 1 1	77.4167	77.4167	300.10	38.14	13.53	0.62
30 10 5 1 1	30 10 5 2 1	77.5	**77.7083**	300.10	43.34	208.28	0.52
30 30 1 1 1	0 0 0 0 0	69	69	300.09	55.72	3.26	0.46
30 30 1 1 1	30 30 1 1 1	91.75	91.75	300.06	26.40	3.49	0.22
30 30 1 1 1	30 10 5 2 1	84.6667	84.6667	300.08	79.14	261.57	0.55
30 30 30 1 1	0 0 0 0 0	129.5	129.5	300.01	202.88	4.26	0.51
30 30 30 1 1	30 30 30 1 1	155.875	155.875	300.01	79.80	9.53	0.59
30 30 30 1 1	30 10 5 2 1	142.7917	142.7917	300.13	103.88	**300.00**	0.41
30 30 30 30 1	0 0 0 0 0	207.75	207.75	300.02	47.25	3.09	0.16
30 30 30 30 1	30 30 30 30 1	236.5417	236.5417	300.03	37.03	3.76	0.35
30 30 30 30 1	30 10 5 2 1	218.2917	218.2917	300.16	37.99	74.46	0.21
			Average time	300.064	50.011	57.812	0.382

Table 7: Instances with many machines ($m = 10, T = 18, b_i = 0, i \in M$)

a	OPT	MIP value	MIP time(s)	SP (s)	new-MIP (s)	FF (s)	NGVNS
1 1 1 1 1 1 1 1 1 1	49	49	300.12	1	**300.00**	**300.00**	0.32
10 9 8 7 6 5 4 3 2 1	232	**232.3333**	300.12	29	**300.00**	**300.00**	0.38
10 10 10 10 10 10 10 10 10 1	413.5	413.5	300.25	2	**300.00**	**300.00**	0.22
100 1 1 1 1 1 1 1 1 1	126.5	126.5	300.08	1	**300.00**	11.71	0.03
1000 1 1 1 1 1 1 1 1 1	576.5	576.5	279.32	7	150.05	0.76	0.05
		Average time	295.798	8.000	270.010	182.494	0.199

Table 8: Average CPU times

Instances from	MIP (s)	SP (s)	new-MIP (s)	FF (s)	NGVNS (s)
Table 3	0.960	46.333	0.176	0.202	0.028
Table 4	61.890	2.433	1.207	0.776	0.103
Table 5	217.315	812.708	4.940	7.606	2.994
Table 7	295.798	8.000	270.010	182.494	0.199
Average time	143.991	217.369	69.083	47.770	0.831

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

(i) The best performance on average are reported by FF and our new MIP formulation. Further, the optimality of found solutions for 4 test instances (Table 7) were not proven solving new MIP formulation, while solving FF formulation the optimality were not proven for 6 test instances (3 instances in Table 6 and 3 instances in Table 7). For all these instances reported times are boldfaced.

(ii) The advantage of FF over new MIP formulation comes from test instances four in Table 7. There, the optimal solution was reached by new MIP formulation but not proven in 300 seconds.

(iii) The behaviour of SP formulation is interesting. It is the worst exact method for small instances in Table 3 but the best one for the largest instances presented in Table 7.

(iv) The old MIP model is least reliable. For example in Table 6 and Table 7, for only one instance (out of 35) the optimal solution has been proven within 300 seconds. However, the optimal solution has not been reached on eight instances (boldfaced values in those tables).

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. For all 110 test instances the optimal solutions were found. Moreover, NGVNS heuristic needed only 0.831 seconds on average to solve them.

It will be very convenient in the future research to generalize the PMP model taking into account more practical issues. Future research may also include proposing extensions of PMP model as well as development of NGVNS based heuristics to solve resulting problems.

References

[1] R. Benmansour, H. Allaoui, A. Artiba, S. Iassinovski, and R. Pellerin. 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, 2011.

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

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

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

[5] S. Hanafi, J. Lazic, N. Mladenovic, C. Wilbaut, and I. Crevits. New VNS based 0-1 MIP heuristics. Yugoslav Journal of Operations Research, doi:10.2298/YJOR140219014H, 2014.

[6] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. European Journal of Operational Research, 130(3):449–467, 2001.

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

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

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

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

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

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

[13] R. Mobley. An introduction to predictive maintenance. Butterworth-Heinemann, 2002.

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