Yugoslav Journal of Operations Research
27 (2017), Number 3, 301–322
DOI: 10.2298/YJOR160417017E

# VARIABLE NEIGHBORHOOD FORMULATION SEARCH APPROACH FOR THE MULTI-ITEM CAPACITATED LOT-SIZING PROBLEM WITH TIME WINDOWS AND SETUP TIMES

Ridha ERROMDHANI
*Université de Sfax, FSEGS, Tunisie*
*erromdhaniridha@yahoo.fr*

Bassem JARBOUI
*Emirates College of Technology, Abu Dhabi, United Arab Emirates*
*bassem_jarboui@yahoo.fr*

Mansour EDDALY
*Université de Sfax, FSEGS, Tunisie*
*eddaly.mansour@gmail.com*

Abdelwaheb REBAI
*Université de Sfax, FSEGS, Tunisie*
*abdelwaheb.rebai@fsegs.rnu.tn*

Nenad MLADENOVIĆ
*University of Belgrade, FOS, Belgrade, Serbia*
*Nenad.Mladenovic@brunel.ac.uk*

Received: April 2016 / Accepted: September 2016

**Abstract:** In this paper we suggest a new variant of Variable neighborhood search designed for solving Mixed integer programming problems. We call it Variable neighborhood formulation search (VNFS), since both neighborhoods and formulations are changed during the search. VNS deals with integer variables, while an available (commercial) solver is responsible for continues variables and the objective function value. We address the multi-item capacitated lotsizing problem with production time windows and setup times, under the non-customer specific case. This problem is known to be NP-hard and can be formulated as a mixed 0-1 program. Neighborhoods are induced from the Hamming distance in 0-1 variables, while the objective function values in the corresponding neighborhoods are evaluated using different mathematical programming formulations of the problem. The computational experiments show that our approach is more effective and efficient when compared with the existing methods from the literature.

**Keywords:** Production Planning, Multi-item lot-sizing, Variable Neighborhood Search, Formulation Space Search, Matheuristic.

**MSC:** 90B06, 90C05, 90C08.

## 1. INTRODUCTION

**Lot sizing problem.** The production planning process aims to establish an overall level of manufacturing output in order to satisfy the customer demands throughout a given time horizon $T$ divided into equal discrete periods ($t = 1, ..., T$). Several models of production planning were presented in the literature (see Gelders and Van Wassenhove [7] for an early review). A lot sizing problem (LSP) consists of determining the quantity of each final product (or item) that should be produced in each period $t$, while minimizing the total manufacturing cost (production cost, inventory cost, setup cost, etc.). The Lot sizing models could be classified in different ways:(i) following the number of items, models are single or multiple; (ii) following the structure of items, they are single level or multi-level and (iii) following the imposed constraints, models are capacitated, and/or with time windows, etc. Clearly, one can consider a combination of (i), (ii) and (iii). For example, LSP problem can be capacitated multi-item multi-level. In this paper, we address the capacitated multi-item LSP with production time windows constraints. The capacity constraint signifies that each period $t$ ($t = 1, ..., T$) is characterized by finite capacity of production. This constraint is retained because the available resources are limited [22]. The time window constraint signifies that there is a time interval where the customer demand can be satisfied without inventory and backlogging costs. Under this constraint, two cases may be presented. The specific customer case means that each specific demand should be satisfied within a given time interval. Therefore, the production between two periods, $t_1$ and $t_2$ can only satisfy the specific demand available before $t_2$. On the other hand, the non-specific customer case means that time windows are not inclusive, i.e., for any pair of time windows $(s_1, e_1)$ and $(s_2, e_2)$, it holds that $s_1 \leq s_2$ and $e_1 \leq e_2$.

**Literature review.** Trigeiro et al. [23] were the first who considered the problem of multi-item CLSP with setup time. They developed a heuristic production smoothing procedure based on lagrangian relaxation of the capacity constraints to generate feasible solutions for the problem. Hindi [9] has proposed a column generation method and a tabu search procedure to solve multi-item CLSP. First, a linear programming relaxation is solved by column generation algorithm. Then, the obtained solution is improved by introducing it as an initial point to tabu search algorithm. Özdamar and Bozyel [20] have developed new heuristic approaches including Hierarchical Production Planning (HPP), Simulated Annealing (SA) and Genetic Algorithm (GA) to solve dynamic CLSP with setup times. The results have shown that the SA provides the best solutions in low computational times according to the compared algorithms. In fact, the use of the local search based algorithm appears more efficient than the population based algorithm in this kind of problem. Karimi et al. [11] have presented a review of the multi-item CLSP and its extensions. The authors mainly focused on discussing the single level variant in both uncapacitated and capacitated cases. They concluded that less attention was paid in the litterature to the CLSP with setup times, and using the metaheuristics based approaches was a fruitful area of research. Miller et al. [19] have exploited the interaction of demand and capacity constraints over a finite time horizon to propose a single period relaxation model where these interactions are considered for each time period. This relaxation is called Preceding Inventory (PI). The polyhedral structure of the convex hull is analyzed and valid inequalities for the original problem are obtained. Jans and Degraeve [10] have presented a review of various metaheuristics, including TS, SA, and GA, developed for solving lotsizing problems (Single level CLSP, Multi level lot sizing without capacities, Capacitated multi level lot sizing, Multi level PLSP, CSLP and DLSP with batch availability). The authors have concluded that the performance of each solutions techniques depends on the way of its implementation. Also, they noted an absence of research on meta-heuristics for the regular DLSP. Absi and Kedad-Sidhoum [1] have considered the CLSP with setup times and shortage costs, using single period relaxation of Miller et al. [19]. The resulted valid inequalities are applied in a branch and cut algorithm.

The work of Dauzère-Pérès et al. [5] is the pioneer work presenting the LSP with the production time windows constraint. The authors have considered uncapacitated single item LSP with both specific and non specific customer demands. They proposed an extension of Wagner-Within algorithm (WW) [25] for solving the non-specific customer case and a dynamic programming (DP) for the case of customer specific. The authors have shown that the DP used for the customer specific case runs in polynomial time with a complexity of $O(T^4)$ for the non customer specific problem. Lee et al. [12] have considered the time window with two cases with and without backlogging. For the no-backlogging problem, an $O(T^2)$ algorithm is proposed. When backlogging is allowed, the problem is solved in $O(T^3)$.

In Wolsey [26] the LSP with time windows under non-specific customer case is considered. It is shown that this problem is equivalent to the single item problem with stock upper bounds. Heuvel and Wagelmans [24] have shown that the same problem is equivalent to the lot-sizing model with a remanufacturing option and also with the lot-sizing model with cumulative capacities.

For solving the capacitated LSP with non customer specific a Lagrangian relaxation based heuristic is proposed in Brahimi et al. [3], where the capacity constraints are relaxed and the initial problem transformed into the uncapacitated single item LSP with time windows. Then a dynamic programming approach is used to solve the problem. Moreover, two new mix-integer 0-1 formulations of the problem are proposed and solved by using commercial software. The binary variables indicate the setups of each item for each period, while the continuous variables indicate the production and inventory quantities.

In terms of complexity, in the case of the single item, the lot sizing problems with concave cost functions and no capacity limits or constant capacity and those with convex cost functions and no setup costs are polynomially solvable [10]. Florian et al. [6] have shown that the general case of the single item CLSP is NP Hard. Moreover, Trigeiro et al. [23] have stated that when set-up times are introduced in the multi-item CLSP, even the feasibility problem becomes NP Complete. Since the considered problem is NP-hard, an exact method suffers from usage of an excessive computational time. To the best of our knowledge, a meta-heuristic approach has never been used.

**Contribution and outline.** In this paper we propose Variable neighborhood search (VNS) based heuristics [15, 8, 4, 18]for solving the multi-item CLSP with setup times and time windows constraints under non customer specific case (CLSP-TW-ST). Within the method, different mathematical programming formulations of subproblems are proposed and solved with exact solver. Therefore, designed VNS based heuristic explores an idea of formulation space search (FSS) [16, 17] as well. We call our variant of VNS as Variable neighborhood formulation search (VNFS), which can be used for solving any Mixed integer nonlinear optimization problem: apply VNS heuristic on the integer part of the problem by fixing continuous variables; use available (commercial) solver to get values of continuous variables, as well as the objective function value, for a given integer variables. Note that another VNS version for solving Mixed nonlinear problems, that is more dependent on available solvers, is proposed in Liberti et al. [13]. VNFS idea has recently been explored also for solving Min-max integer programs in Pardo et al. [21].

Our VNFS based heuristic starts with finding an initial solution for the set of binary variables (the production days for each item during the time horizon) heuristically, using a mathematical programming formulation. In each VNS based method developed in this paper, after fixing 0-1 variables, the continues subproblem is solved by using an exact method through the application of mathematical programming. In that way, the production and inventory quantities for each item for all periods are found.

In the first proposed VNS heuristic (VNFS-1), the continuous subproblems are solved in each neighboring point of 0-1 variables. The number of variables and constraints can be large, and therefore the time spent to solve LP sub-problems can be too long. In order to reduce the running time, we develop a second algorithm VNFS-2. It is based on transformation of the multi-item CLSP into a sequence of single-item CLSP. In VNFS-2, we solve the LP subproblem item by item. While optimizing the production days of an item, we assume that the production days and the produced quantities of all the remaining items are fixed to the best values found so far. However, this algorithm appears unable to perform a deeper exploration of the search space because of the hardness of constraints presented in some instances. This fact has motivated the development of the third VNS based heuristic (VNFS), that combines the main features of the two previous heuristics. It takes the deeper exploration from the first algorithm, and the efficiency of the resolution of the mathematical programming from the second one.

The paper is organized as follows. In section 2 we give the steps of our VNFS method. Section 3 contains a description of the problem addressed in this paper. The initialization phase of the VNFS algoritm is presented in section 4. Our methods, VNFS-1, VNFS-2 and VNFS are described in sections 5, 6 and 7, respectively. Computational results are reported in Section 8. The conclusions are given in Section 9.

## 2. VARIABLE NEIGHBORHOOD FORMULATION SEARCH

Variable neighborhood formulation search (VNFS) is a VNS based method designed for solving Mixed integer programs (MIPs). Therefore, in order to use it, the mathematical programming formulation of the problem is necessary

$$\min_{X,Y}\{f(X,Y) \mid g_i(X,Y) \le 0, i = 1,..,m,\ X \in R^p,\ Y \in Z^q\}. \tag{1}$$

Variables $X = (x_1,\dots,x_p)^T$ are real and variables $Y = (y_1,\dots,y_q)^T$ are integer. Functions $f$ and $g_i$ map the corresponding mixed spaces into a set of real numbers $R$. All $m + 1$ functions need to be at least defined for each $(X, Y)$. More specific conditions on those functions, which are problem dependent, may be added.

VNFS follows a scheme of General VNS (GVNS) for solving discrete (combinatorial) problem in integer variables $Y$. Continuous variables, as well as the objective function values are found within local search step. GVNS algorithm is given in Algorithm 8.

**Algorithm 1:** Steps of the general VNS

```
  Function GVNS ($Y, \ell_{max}, k_{max}, t_{max}$);
1 $Y \leftarrow$ Initial // Get Initial solution;
2 repeat
3     $k \leftarrow 1$;
4     repeat
5         $Y' \leftarrow$ Shake($Y, k$);
6         $Y'' \leftarrow$ VND($Y', \ell_{max}$) ;
7         NeighborhoodChange($Y, Y'', k$);
      until $k = k_{max}$;
8     $t \leftarrow$ CpuTime()
  until $t > t_{max}$;
```

As it is well known [8, 18] the input parameters of GVNS are $k_{max}$, $\ell_{max}$ and $t_{max}$, representing the number of neighborhood structures used in `Shaking`, the number of neighborhood structures used in `VND`, and the total CPU time that will be used in the search, respectively. VND is a deterministic version of VNS, where $\ell_{max}$ neighborhoods are explored one after another, unless the final solution becomes a local minimum with respect to all neighborhoods. `Shake`, or perturbation, procedure generates random point from the predefined neighborhood $N_k$. Its aim is a diversification of the search, while `VND` is used for intensification. In `NeighborhoodChange` the values of $f(Y)$ and $f(Y'')$ are compared. If $Y''$ is better, the search is re-centered around $Y''$ ($Y \leftarrow Y'', k \leftarrow 1$). Otherwise the neighborhood counter is increased by 1 for the next shaking step.

For solving MIPs, the VNFS explores discrete neighborhood structures. When point in $k^{th}$ neighborhood of $(Y)$ is chosen, i.e., when all discrete values are fixed, then much simplified mathematical program using continuous variables $X$ is obtained. Moreover, VNFS may use different formulations of the problem (1). Initial model may be reformulated by introducing penalties, Lagrangian multipliers, could be decomposed etc. All those reformulation possibilities may give some advantages in the search. Change of formulation is in fact included in each local search step. Since VND may contain several ($\ell_{max}$) local searches, this implies the use of different mathematical programming models within local search step of VND. Thus, in each neighborhood point considered, the corresponding mathematical program is solved. General steps of VNFS are given below in Algorithm 8. The details for VND step within VNFS are problem specific and will not be given here.

**Algorithm 2:** Steps of the general VNFS

```
  Function VNFS ($Y, X, \ell_{max}, k_{max}, t_{max}$);
1 $X, Y \leftarrow$ Initial // Get Initial solution;
2 repeat
3     $k \leftarrow 1$;
4     repeat
5         $Y' \leftarrow$ Shake($Y, k$);
6         $X', Y'' \leftarrow$ VND($Y', X, \ell_{max}$) ;
7         NeighborhoodChange($Y, Y'', X, X', k$);
      until $k = k_{max}$;
8     $t \leftarrow$ CpuTime()
  until $t > t_{max}$;
```

The differences between GVNS and VNFS are in steps 6 and 7. Steps 6 and 7 of VNFS contain continuous variables $X$, as well. In Step 6 different formulations are used in finding the best values $(X, Y)$. In step 7, a move to new mixed integer solution is made, or neighborhood counter increased. Obviously, the details of step 6 are problem dependent and we will give its details when solving a variant of lot-sizing problem.

## 3. DESCRIPTION OF CLSP-TW-ST PROBLEM

We consider the general approaches for lot-sizing problem with production time windows and setup times, where the objective is to minimize the total cost. In order to achieve the optimal plan, one needs to find optimal production and inventory quantities, as well as a setup decisions. While doing so, capacities should not be exceeded and demand should be satisfied. The notation and meaning of terms used throughout the paper are:

| | |
|---|---|
| $N$ | Number of items ($i = 1, \dots, N$); |
| $T$ | Number of time periods ($t = 1, \dots, T$); |
| $p_{it}$ | Production cost for item $i$ at the end of period $t$; |
| $S_{it}$ | Setup cost of item $i$ in period $t$; |
| $d_{ikt}$ | External demand for item $i$ available at period $k$ that should be produced before period $t$. In the non-customer specific demand case, it is assumed that the demand of item $i$ in period $t$ can be preprocessed (available and produced) at another availability period $k$; |
| $D_{it}$ | The aggregate demand in period $t$ of item $i$: $D_{it}=\sum_{k=1}^{t} d_{ikt}$; |
| $I_{i0}$ | Initial inventory of item $i$; |
| $C_t$ | The total available capacity at period $t$; |
| $h_{it}$ | Non-negative holding costs for item $i$ in period $t$; |
| $st_i$ | Setup time of product $i$; |
| $r_i$ | Unit resource consumption for item $i$; |

Decision variables are:

| | |
|---|---|
| $I_{it}$ | Inventory for item $i$ at the end of period $t$. |
| $x_{it}$ | The quantity of item $i$ produced at period $t$. |
| $y_{it}$ | Binary variable which indicates whether a setup for item $i$ occurs in period $t$. |

Referring to Brahimi et al. [3], this problem can be formulated as follows.

$$f(Y, X, I) = \min_{Y,X,I} \sum_{i=1}^{N} \sum_{t=1}^{T} (S_{it} Y_{it} + p_{it} X_{it} + h_{it} I_{it}) \tag{2}$$

Subject to:

$$I_{i,t-1} + x_{it} - I_{it} = D_{it}, \ (\forall i, t); \tag{3}$$

$$\sum_{k=1}^{t} x_{ik} \leq \sum_{k=1}^{t} \sum_{l=k}^{T} d_{ikl} \ (\forall i, t); \tag{4}$$

$$\sum_{i=1}^{N} (r_i x_{it} + st_i y_{it}) \leq C_t \ (\forall\ t); \tag{5}$$

$$x_{it} \leq (\sum_{l=t}^{T} D_{il}) y_{it} \ (\forall\ i, t); \tag{6}$$

$$y_{it} \in \{0, 1\} \ (\forall i, t); \tag{7}$$

$$I_{it}, x_{it} \geq 0 \ (\forall i, t). \tag{8}$$

The objective function (2) gives the total production, inventory and setup costs in $T$ periods that should be minimized. Constraints (3) represent the inventory balance equations. Constraints (4) state that cumulative production in the first $t$ periods does not exceed the cumulative quantity available from periods 1 to $t$. Constraints (5) indicate that the amount of capacity used for production is limited. Constraints (6) relate the binary setup variables $y_{it}$ to the continuous variables $x_{it}$. Constraints (7) and (8) characterize the variable types: $y_{it}$ are binary and $x_{it}$ and $I_{it}$ are non-negative real variables, for $i = 1, \ldots, N$ and $t = 1, \ldots, T$.

## 4. INITIALIZATION OF VNFS FOR SOLVING THE CLSP-TW-ST

In the first step of VNFS heuristic we need to find an initial solution. As an input of the problem, we consider the setup sequence presented by the binary matrix $Y = \{y_{it}\}_{N \times T}$. Besides the usual objective, we introduce a new one that minimizes infeasibility of the current solution, as well. A new formulation of the CLSP-TW-ST may be presented by a linear mixed integer program as follows:

Let $E_i$ denote the set of periods where $y_{it}$ takes the value of 1, $(\forall\ i)$, and let $F_t$ denote the set of items produced in period $t$ $(\forall\ t)$.

$$f(Y) = \min_{X,I,Z} \left[ \sum_{i=1}^{N} \sum_{t \in E_i} p_{it} x_{it} + \sum_{i=1}^{N} \sum_{t=1}^{T} (h_{it} I_{it} + \gamma_{it} z_{it}) \right] \tag{9}$$

subject to

$$I_{i,t-1} + x_{it} + z_{it} - I_{it} = D_{it} \ (\forall\ i), \ t \in E_i \tag{10}$$

$$I_{i,t-1} + z_{it} - I_{it} = D_{it} \ (\forall\ i), \ t \in \{1, \ldots, T\} \backslash \{E_i\} \tag{11}$$

$$\sum_{k=1}^{t} x_{ik} \leq \sum_{k=1}^{t} \sum_{l=k}^{T} d_{ikl} \ (\forall\ i), \ t \in E_i \tag{12}$$

$$\sum_{i \in F_t} r_i x_{it} \leq C_t - \sum_{i \in F_t} st_i \ (\forall\ \ t) \tag{13}$$

$$I_{it}, x_{it}, z_{it} \geq 0 \ (\forall\ i), (\forall\ t) \tag{14}$$

In this formulation, new variables $z_{it}$ are introduced. They represent the shortage of the demand for item $i$ at period $t$. In other words, variables are used for creating a feasible solution by minimizing the total infeasibility. Indeed, the measure of infeasibility may be presented as

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\gamma_{it}z_{it},$$

where $\gamma_{it}$ denotes the associated penalty. In addition, the acceptance of unfeasible solutions (i.e. the shortage of the demand) is associated with high cost $\gamma_{it}$.

At first, we set $y_{it} = 1$ for all items and for all periods, and then solve the corresponding program. In our experimentation we have considered the instances of Trigeiro et al. [23]. The parameter capacity is generated by setting a target average utilization capacity. These instances are used under the assumption that the presence of setup time allows feasible solutions for the problems that initially use more than 100% of capacity. For all instances, at each period, the sum of setup times of all items is always lower than the capacity. Therefore, our initialization is operational.

As the result, the production quantities $x_{it}$, $\forall i \in \{1,\ldots,N\}$ and $\forall t \in \{1,\ldots,T\}$ are found. In such a solution, some $x_{it}$ may be zero, whereas all $y_{it}$ are initially set to 1. In order to eliminate such cases, we set $y_{it} = 0$ for all $i$ and $t$, where $x_{it} = 0$ . Based on this modification of $Y$, the problem is reformulated and resolved. These steps are repeated until there is no $x_{it} = 0$ with the corresponding $y_{it} \neq 0$.

The initialization procedure obviously requires $N \times T$ iterations, at most. At each iteration, at least one variable is eliminated. In that way, the capacity constraint becomes 'more relaxed' and the value of the objective function decreases.

## 5. VNFS-1 FOR THE CLSP-TW-ST

In this section we describe our first algorithm VNFS-1. The initial solution is obtained by the algorithm described in the previous section. Then, we give details of Shake (step 5) and VND (step 6) developed for solving CLSP-TW-ST.

### 5.1. *Shaking.*

An improvement phase starts with moves in the setup sequence $Y = \{y_{it}\}_{N\times T}$. Since $Y$ is a binary matrix, the Hamming distance is a natural choice in defining the neighborhood of a solution: a new solution $Y'$ from the neighborhood $k$ of $Y$ ($Y' \in \mathcal{N}_k(Y)$) is obtained by flipping $k$ components simultaneously. Let us denote, $Y'$ the neighboring solution of $Y$ ($Y' \in \mathcal{N}_1$) and $y'_{it}$ the value of $y_{it}$ after being flipped. We denote by $Y_{best}$ the incumbent solution (the best solution obtained so far).

*5.2. The first local search procedure LS-1.*

The first neighborhood structure is based on one flip move, i.e. one element of $Y$ is flipped at each iteration: $y'_{it} = 1 - y_{it}$, $(\forall i, t)$. Therefore, if the setup of an item $i$ occurs over one period $t$ in the current solution ($y_{it} = 1$), then it will be cancelled in the new solution ($y_{it} = 0$) and vice-versa. After each such move, the value of the objective function is computed by solving the mathematical program (9)-(14). If $f(Y') < f(Y_{best})$ then $Y_{best} = Y'$. It should be noted that the first improvement strategy is used. Therefore, an improved neighboring solution is immediately selected to replace the current one. This procedure is repeated until no improvement is found (Algorithm 3). If a local optimum is found, the second neighborhood structure will be explored.

**Algorithm 3:** Local search LS-1

```
1  $f_{best} = f(Y)$;
2  repeat
3      improve ← false;
4      for $i = 1, \ldots, N$ do
5          for $t = 1, \ldots, T$ do
6              $y_{it} = 1 - y_{it}$; // Flipping move
7              Find $Y$ by solving the mathematical program (9)-(14);
8              if $f(Y) < f_{best}$ then
9                  $f_{best} = f(Y)$, improve ← true
               else
10                 $y_{it} = 1 - y_{it}$
   until no improve;
11 output: $Y$
```

*5.3. The Second local search procedure LS-2*

In this algorithm, the neighborhood is defined by flipping two 0-1 variables, i.e., neighboring solutions are chosen among those whose Hamming distance from $Y$ is equal to two. For each item $i$ and for each period $t$, all possible 2-flip moves are checked out. An exchange move consists of exchanging the value of two distinct periods for one item $i$ in $Y$. So, if $y_{it} \neq y_{i\acute{t}}$ with $t < \acute{t}$ then, $\acute{Y}_{it} = y_{i\acute{t}}$ and $\acute{Y}_{i\acute{t}} = y_{it}$. If $f(\acute{Y}) < f(Y_{best})$ then, $Y_{best} = \acute{Y}$ and $f(Y_{best}) = f(\acute{Y})$. This step is repeated until no improvement is found (Algorithm 4). The choice of changing the structure of one item and not of two distinct items is based on an extensive experimentation. We have found that 90% of improvements are obtained when we perform the moves in the structure of the same item between two distinct periods. Thus, the number of possible flip moves (and therefore the running time of the algorithm) is reduced, for the price of loosing 10% of improving moves.

**Algorithm 4:** Local search LS-2

```
1  $f_{best} = f(Y)$;
2  repeat
3      improve ← false;
4      for $i = 1, \dots, N$ do
5          for $t = 1, \dots, T-1$ do
6              for $t' = t+1, t+2, \dots, T$ do
7                  if $y_{it} \neq y_{it'}$ then
8                      Flip $y_{it}$ and $y_{it'}$
9                      Solve mathematical program (9)-(14) to find $Y$;
10                     if $f(Y) < f_{best}$ then
11                         $f_{best} = f(Y)$; improve ← true
                       else
12                         Flip back $(y_{it}, y_{it'})$
   until no improve;
13 Output$Y$
```

The local optimum of the second local search procedure LS-2 will be than an input for the first procedure LS-1. The algorithm will stop when the local optima of the second and the first procedures are the same (see Algorithm 5).

**Algorithm 5:** VND local search

```
1  improvement = true
2  repeat
3      $Y' \leftarrow$ LS-1($Y$)
4      $Y' \leftarrow$ LS-2($Y'$)
5      if $f(Y') \geq f(Y)$ then
6          not improvement
7      $Y \leftarrow Y'$
   until improvement = false;
8  output: $Y$
```

Each move is evaluated according to the resolution of the mathematical program with $(\sum_{i=1}^{N} |E_i| + 2 \times N \times T)$ variables and $(\sum_{i=1}^{N} |E_i| + (N \times T) + T)$ constraints. Thus the complexity increases according to the number of items. Moreover, the size of the neighborhood is very large and large computational time is needed to complete the search.

The experimental experience has shown that the efficiency of this algorithm depends a lot on the size of the problem. Moreover, the final solution quality is very sensitive to the choice of the starting point.

## 6. DECOMPOSITION APPROACH FOR SOLVING CLSP-TW-ST (VNFS-2)

It is shown that when using VNFS-1, the mathematical programming part is relatively slow. Here, we propose a decomposition scheme of the problem to reduce the number of constraints and variables in mathematical programming phase of the method.

The move is evaluated for the single item LSP by considering the remaining items as fixed. So, the multi-item LSP is transformed into the single-item LSP by performing the moves to the setup sequence of an item $\bar{i}$ and assuming that the setup sequence for all remaining items are known. Therefore, in each iteration, a new mathematical formulation of the problem is developed as follows:

$$f(Y) = \min_{X,I,Z} \left[ \sum_{t \in E_{\bar{i}}} (p_{\bar{i}t} x_{\bar{i}t}) + \sum_{t=1}^{T} (h_{\bar{i}t} I_{\bar{i}t} + \gamma_{\bar{i}t} z_{\bar{i}t}) \right] \tag{15}$$

Subject to:

$$I_{\bar{i},t-1} + x_{\bar{i}t} + z_{\bar{i}t} - I_{\bar{i}t} = D_{\bar{i}t} \ \ \forall\ t \in E_{\bar{i}} \tag{16}$$

$$I_{\bar{i},t-1} + z_{\bar{i}t} - I_{\bar{i}t} = D_{\bar{i}t} \ \ \forall\ \ t \in \{1,\ldots,T\} \setminus \{E_{\bar{i}}\} \tag{17}$$

$$\sum_{k=1}^{t} x_{\bar{i}k} \leq \sum_{k=1}^{t} \sum_{l=k}^{T} d_{\bar{i}kl} \ \ \forall\ t \in E_{\bar{i}} \tag{18}$$

$$\delta_{i*} x_{\bar{i}t} \leq C_t - \sum_{\bar{i}=1}^{N} \tau_{\bar{i}} y_{\bar{i}t} - \sum_{\substack{\acute{i}=1 \\ \acute{i} \neq \bar{i}}}^{N} \delta_{\acute{i}} x_{\acute{i}t} \ \ \forall\ \ t = 1,\ldots,T \tag{19}$$

$$x_{\bar{i}t} \leq \sum_{l=t}^{T} D_{\bar{i}l} \ \ \forall\ t \in E_{\bar{i}} \tag{20}$$

$$I_{\bar{i}t}, x_{\bar{i}t}, z_{\bar{i}t} \geq 0 \ \ \forall\ \ t = 1,\ldots,T \tag{21}$$

where $E_{\bar{i}}$, the set of periods whose $y_{\bar{i}t} = 1 \ \forall\ \bar{i} = 1,\ldots,N$.

The objective function (15) minimizes the total cost induced by the production plan, like production, inventory, and penalty of the demand shortage costs for the item $\bar{i}$. The capacity constraints (19) is modified while taking into account the known production of the items other than $\bar{i}$.

The evaluation of each move is computed according to the resolution of the mathematical programming presented above. This reformulation includes ($\left|E_{\bar{i}}\right|$ + $2 \times T$) variables and ($3 \times \left|E_{\bar{i}}\right| + (T - \left|E_{\bar{i}}\right|) + T$) constraints. We observe that the evaluation of moves in VNFS-2 is faster than the evaluation of VNFS-1 due to the obtained reduction of the size of the problem in the mathematical programming model.

The initialization of this algorithm has the same steps as VNFS-1. Our idea is to propose a new improvement procedure by using the Variable neighborhood search algorithm (VNS) [15, 17, 4, 18, 21]. This main loop of the algorithm consists of three steps, after defining the neighborhood structures: (i) the shaking phase; (ii) the local search phase; (iii) the neighborhood change. The first step leads to performing moves onto the current solution. The local optima found will be subject to the shaking by applying random moves. These three steps will be repeated until a given stopping criterion is reached.

The VNS algorithm advantage is to explore regions far from the current region of the searching space. So, once the best solution is found in a large region, it is necessary to go quite far to get better solutions. Therefore, this algorithm employs the concept of shaking to move from one region to another which can be more profitable. In the literature, the VNS algorithm has not been much explored for solving the LSP. We can find the work of Xiao et al. [27] where this algorithm is used for solving the uncapacitated multi-level LSP.

In our application, the neighborhood structures are the same as those used in the first algorithm. However, within the local search, the way solutions are evaluated in each move is based on the decomposition scheme described above (Algorithms 6, 7 and 9).

**Algorithm 6:** VNFS-2, i.e., LS-1 local search for decomposed problem

```
   begin
1      $LS1'(Y)$ //first local search in VNFS-2
2      $f_{best} = f(Y)$;
3      repeat
4          for $i^* = 1, \dots, N$ do
5              for $t = 1, \dots, T$ do
6                  $y_{i^*t} = 1 - y_{i^*t}$; //perform move
7                  solve the mathematical program (15)-(21) according to $Y$;
8                  if $f(Y) < best$ then
9                      $best = f(Y)$
                   else
10                     $y_{i^*t} = 1 - y_{i^*t}$
       until no improvement;
       output: $Y$
```

The step of perturbation consists in performing $h$ consecutive random moves into the binary vector. Two items and two periods are selected at random and their values are exchanged (Algorithm 10).

The reduction of the computational time of the evaluation is accompanied by reduction of the size of the neighborhood. Therefore, the exploration of the space search remains partial, and some promising regions may not be visited by this algorithm.

**Algorithm 7:** Second local search procedure in VNFS-2

```
begin
1   $LS'2(Y)$ //second local search in VNFS-2
2   $best = f(Y)$;
3   repeat
4       for $i^* = 1, \ldots, N$ do
5           for $t = 1, \ldots, T-1$ do
6               for $t = t+1, t+2, \ldots, T$ do
7                   if $y_{i^*t} \neq y_{i^*t'}$ then
8                       exchange $(y_{i^*t}, y_{i^*t'})$
9                       Find $Y$ by solving the mathematical program (14)-(20);
10                      if $f(Y) < f_{best}$ then
11                          $f_{best} = f(Y)$
                        else
12                          exchange $(y_{i^*t}, y_{i^*t'})$
    until no improvement;
    output: $Y$
```

**Algorithm 8:** VND local search in VNFS-2

```
begin
1   VND-1
2   repeat
3       $Y \leftarrow RD - 1(Y)$
4       $Y \leftarrow RD - 2(Y)$
    until no improvement;
    output: $Y$
```

**Algorithm 9:** VND local search in VNFS-2

```
begin
1   VND-1
2   repeat
3       $Y \leftarrow RD - 1(Y)$
4       $Y \leftarrow RD - 2(Y)$
    until no improvement;
    output: $Y$
```

**Algorithm 10:** Variable neighborhood formulation search method

```
begin
1    $VNFS - 1(Y)$
2    $Y_{best} = Y$
3    repeat
4        $k = 1$;
5        while $k < k_{max}$ do
6            Select random solution $Y \in N_k(Y_{best})$;
7            $Y' \leftarrow$ VND-1($Y$);
8            if $f(Y') < f(Y_{best})$ then
9                $Y_{best} \leftarrow Y'$; $k = 1$;
             else
10               $k = k + 1$
     until until stopping criterion;
     output: $Y$
```

## 7. GENERAL VNFS FOR SOLVING CLSP-TW-ST

The third proposed approach is a combination of the two previous approaches. Note that the VNFS-1 offers a deep exploration of the space search, and that the VNFS-2 offers the convergence speed. In VNFS we used both of them in sequential way as in the Variable neighborhood descent VND algorithm.

VND is a deterministic version of VNS, where the best neighbor of the current solution is considered instead of a random one. Also, no local descent is performed with this neighbor. Rather, it automatically becomes the new current solution if an improvement is obtained, and the search is then restarted from the first neighborhood. The VND algorithm presents a particular feature where the different neighborhoods are explored in an increasing order of their sizes and evaluations. That is to say, it starts by exploring the smallest and thus the fastest neighborhoods, finishing with the slowest (the largest) neighborhoods. While investigating this idea in our VNFS algorithm, two local search procedures are used within VND local search. They have the same structure as the two local searches given in Algorithms 3, 4 and, 5. Therefore, we will not repeat their steps here.

In the first phase of our final VNFS, the decomposition algorithm VNFS-2 is applied. As presented above, the size of neighborhoods is small and less computation time is used than for VNFS-1. After finding the local optima, the algorithm VNFS-1 is run in the second phase.

The algorithm leaves this second structure after performing a given number of successive improvements and returns to the first structure. If the local optima of the first and the second structures are the same, then we perform some random moves (shake) into the current solution and return to the first structure. These steps will be repeated until the given stopping criteria are met

## 8. COMPUTATIONAL RESULTS

This section reports the computational results that evaluate the effectiveness of our algorithms against the proposed algorithms in the literature of the CLSP with time windows and setup times. The algorithms were implemented in C++ and run in Windows XP on desktop PC with Intel Pentium IV, running at 3.2 GHz processor with 512 MB of RAM memory.

The data sets used in our experiments are generated by Brahimi et al. [3]. These instance problems are based on the data sets of Trigeiro et al. [23] and extended by adding the time windows. More detailed information about these test problems is available in Brahimi et al. [3]. The comparison was conducted after classifying the instances into 13 classes. The first three classes depend on the number of items $N$ =10, 20 and 30. The utilization rate of total capacity is 75%, 85%, and 95%. The time between two successive orders (TBO) is 1, 2, and 4. The variation coefficient of the demand is set to 0.35, and 0.59. The average value of setup times over all products is 11, and 43 capacity unit. In total, there are 540 instances to be tested.

The competing approaches used in our comparative study include the approaches developed by Brahimi et al. [3]. The first approach denoted by (LR) is a Lagrangian relaxation based heuristic. The second is an exact mixed integer programming approach, based on a new reformulation of the problem and solved with the commercial software (MIPX). The last one is a mixed integer approach, based on the original formulation of the problem (MIPB).

In order to evaluate our results according to the competing approaches, we use the following measure of performances as in [14, 2, 3]:

$$Gap(UB_A, LB) = 200 \times \frac{UB_A - LB}{UB_A + LB} \tag{22}$$

where $Gap(UB_A, LB)$ denotes the gap between upper bound values of the algorithm A and the best lower bound values, for each instance, among the different lower bounds provided by [3].

$$Gap(UB_A, UB_B) = 200 \times \frac{UB_A - UB_B}{UB_A + UB_B} \tag{23}$$

where $Gap(UB_A, UB_B)$ is the gap between the upper values of the algorithm A,$UB_A$, and those of the algorithm B,$UB_B$, such that $UB_A \geq UB_A$.

Brahimi et al. [3] have noted that the use of the relative percentage deviation as a performance measure computed by $\frac{UB_A - LB}{LB}$, according to the obtained solution by an algorithm A, and the lower bound value may underestimate or overestimate the gap. Also, the same stands for comparing to upper bound values. Extensive experiments were conducted in order to set the values of parameters of our proposed algorithms as follows:

VNFS-2: the number of successive random haves is $h = 1 + random$ number in $[1, 10]$. The stopping criterion is the number of iterations which is fixed to 100. For the algorithm (VNFS), the number of iterations is used as a stopping criterion and is set to be the number of items $\times$ 50.

Table 1: Computational experiments of the proposed approaches

| | | GAP UB (VNFS-1,VNFS-2) | | GAP UB (VNFS-2,VNFS) | |
|---|---|---|---|---|---|
| | | VNFS-1 is better | VNFS-2 is better | VNFS-2 is better | VNFS is better |
| parameter | Value | gap | gap | gap | gap |
| | 10 | 0,4 | 0,93 | 0,08 | 0,62 |
| N | 20 | 0,25 | 1,41 | 0,1 | 0,71 |
| | 30 | 0,07 | 1,74 | 0,47 | 0,1 |
| | 75% | 0,21 | 0,68 | 0,16 | 0,26 |
| Capacity | 85% | 0,18 | 0,86 | 0,24 | 0,38 |
| | 95% | 0,63 | 2,51 | 0,65 | 1,09 |
| | 1 | 0,28 | 1,26 | 0,55 | 0,62 |
| TBO | 2 | 0,62 | 1,16 | 0,26 | 0,65 |
| | 4 | 0,24 | 1,73 | 0,3 | 0,42 |
| CV | 0.35 | 0,27 | 1,8 | 0,32 | 0,53 |
| | 0.59 | 0,2 | 1,37 | 0,47 | 0,58 |
| STC | 11 | 0,26 | 1,7 | 0,4 | 0,57 |
| | 43 | 0,16 | 1,44 | 0,38 | 0,54 |
| Global average | | **0,29** | **1,43** | **0,34** | **0,54** |
| Number of better UBs | | 41 (0,04%) | 374 (79%) | 64 (12%) | 171 (31%) |

Table 1 shows the comparative study between the three proposed approaches. In the third and fourth columns, we compare the gap between the upper bound values provided by VNFS-1 and VNFS-2 according to Eq. (23). It can be seen that VNFS-2 is better than VNFS-1 both in terms of average percentage deviations (0.29% in favor of VNFS-1 and 1.43% in favor of VNFS-2) and the number of better values of upper bounds.

When VNFS-2 is better than VNFS-1, we find that the number of better upper bounds is larger than in the opposite case (41 upper bounds in favor of VNFS-1 and 374 in favor of VNFS-2). This is provided by the decomposition scheme, introduced in VNFS-2 for improving the results. However, in some cases, VNFS-1 is better than VNFS-2 because VNFS-1 explores more deeply the space searched.

In the fifth and sixth columns, we have compared the gap between the upper bound values of VNFS-2 and VNFS. We see that VNFS outperforms VNFS-2 both in terms of average percentage deviations (0.34% in favor of VNFS-2 and 0.54% in favor of VNFS) and the number of better upper bounds (64 upper bounds in favor of VNFS-2 and 171 in favor of VNFS).

This result is expected since the use of the VND within VNFS, coordinates the main advantages of the two algorithms VNFS-1 and VNFS-2. Therefore, in the first neighborhood structure we have used the local search procedure based on the decomposition scheme of the VNFS-2 in order to accelerate the exploration of the space search. Then we have used the second neighborhood structure as employed in VNFS-1 in order to perform a deeper exploration of the search space.

Table 2 presents the number of feasible solutions provided by each approach of resolution. It can be seen that VNFS-2 and VNFS are able to find 535 feasible solutions, i.e. five instances remain unsolved as in LR algorithm.

Table 2: Number and percentage of problems for which feasible solutions

| | MIPB | MIPX | LR | VNFS-1 | VNFS-2 | VNFS |
|---|---|---|---|---|---|---|
| Number of feasible solutions | 466 | 529 | 535 | 535 | 535 | 535 |
| Percentage (%) | 86.3 | 98 | 99.1 | 99.1 | 99.1 | 99.1 |

Table 3: Computational experiments of the proposed approaches

| | | GAP (UB,LB) | | | | | |
|---|---|---|---|---|---|---|---|
| parameter | Value | MIPB | MIPX | LR | VNFS-1 | VNFS-2 | VNFS |
| | 10 | 3,78 | 2,92 | 2,54 | 2,48 | 2,01 | 1,79 |
| N | 20 | 18,67 | 1,32 | 0,96 | 2,35 | 1,27 | 0,94 |
| | 30 | 19,40 | 1,26 | 0,55 | 1,92 | 0,50 | 0,60 |
| | 75% | 15,19 | 0,332 | 0,29 | 0,66 | 0,26 | 0,19 |
| Capacity | 85% | 17,77 | 1,20 | 0,94 | 1,35 | 0,72 | 0,62 |
| | 95% | 19,54 | 4,11 | 2,97 | 4,91 | 2,94 | 2,63 |
| | 1 | 2,10 | 0,60 | 0,54 | 1,37 | 0,77 | 0,66 |
| TBO | 2 | 16,86 | 1,01 | 0,86 | 2,48 | 1,15 | 0,87 |
| | 4 | 40,58 | 4,05 | 2,79 | 3,40 | 1,93 | 1,88 |
| CV | 0.35 | 23,64 | 1,61 | 1,43 | 2,22 | 0,87 | 0,77 |
| | 0.59 | 14,08 | 0,97 | 1,31 | 2,05 | 0,91 | 0,76 |
| STC | 11 | 20,95 | 1,31 | 1,42 | 2,41 | 1,27 | 1,12 |
| | 43 | 12,42 | 1,26 | 1,31 | 2,09 | 1,26 | 1,11 |
| Global average | | **17,31** | **1,7** | **1,38** | **2,25** | **1,22** | **1,07** |

Table 3 displays the computational results based on the gap between the upper bounds, got by our proposed approaches and the approaches of [3], and the lower bound according to Eq. (22). Note that these gaps are computed according to the feasible solutions obtained by each approach. It can be seen that VNFS has got the best gaps in average for 10 classes of instances among 13, whereas the LR algorithm has provided 3 best gaps.

The average gaps obtained by VNFS are lower than those obtained by LR, except for three classes of $N = 30$, TBO=1 and 2. In terms of global average, for all instances, VNFS outperforms all the other approaches with 1, 07%. Moreover, VNFS-2 outperforms LR algorithm when $N = 30$. Concerning our proposed approaches, we notice that VNFS is the best algorithm, and VNFS-2 is better than VNFS-1.

Table 4 provides a comparison between the upper bounds of VNFS-2, MIPX and LR. It is shown that the proposed algorithm outperforms the two other approaches. In average, for all classes, the percentage gap between VNFS-2 and MIPX is equal 1.91%, where VNFS-2 is better than MIPX and is equal to 0.54% in the opposite case. The number of better upper bounds in favor of VNFS-2 according to MIPX algorithm is 206. Regarding the LR algorithm, we see that VNFS-2 is better in terms of average percentage deviations, whereas the former is better in terms of the number of improved upper bounds.

Table 5 displays the gaps between UB (VNFS), UB (MIPX) and UB (LR). It is shown that, in average, when VNFS is better, the average relative deviations are larger than MIPX and LR. Moreover, according to MIPX algorithm, the VNFS has provided 33% of better upper bounds, and MIPX has provided 21%, the

Table 4: Gaps between UB(VNFS-2),UB(MIPX) and UB(LR)

| | | GAP UB (VNFS-2,MIPX) | | GAP UB (VNFS-2,LR) | |
|---|---|---|---|---|---|
| | | VNFS-2 is better | MIPX is better | VNFS-2 is better | LR is better |
| parameter | Value | gap | gap | gap | gap |
| | 10 | 2,32 | 0,49 | 1,54 | 0,82 |
| N | 20 | 1,81 | 0,77 | 0,78 | 0,76 |
| | 30 | 2,13 | 0,15 | 0,26 | 0,22 |
| | 75% | 0,69 | 0,29 | 0,51 | 0,33 |
| Capacity | 85% | 2,08 | 0,41 | 1,10 | 0,45 |
| | 95% | 2,88 | 1,13 | 1,63 | 1,23 |
| | 1 | 0,22 | 0,70 | 0,25 | 0,85 |
| TBO | 2 | 0,92 | 0,57 | 0,53 | 0,76 |
| | 4 | 3,32 | 0,40 | 1,52 | 0,39 |
| CV | 0.35 | 2,57 | 0,52 | 0,63 | 0,74 |
| | 0.59 | 1,69 | 0,57 | 0,64 | 0,60 |
| STC | 11 | 1,86 | 0,55 | 0,56 | 0,61 |
| | 43 | 2,39 | 0,55 | 0,77 | 0,73 |
| Global average | | **1,91** | **0,54** | **0,82** | **0,65** |
| Number of better UBs | | 206 (38%) | 178 (33%) | 165 (31%) | 187 (35%) |

Table 5: Gaps between UB(VNFS),UB(MIPX) and UB(LR)

| | | GAP UB (VNFS,MIPX) | | GAP UB (VNFS,LR) | |
|---|---|---|---|---|---|
| | | VNFS is better | MIPX is better | VNFS is better | LR is better |
| parameter | Value | gap | gap | gap | gap |
| | 10 | 2,30 | 0,33 | 1,40 | 0,53 |
| N | 20 | 1,77 | 0,52 | 0,46 | 0,50 |
| | 30 | 2,86 | 0,42 | 0,19 | 0,45 |
| | 75% | 0,78 | 0,22 | 0,39 | 0,26 |
| Capacity | 85% | 2,10 | 0,38 | 0,78 | 0,35 |
| | 95% | 3,07 | 0,82 | 1,33 | 0,70 |
| | 1 | 0,34 | 0,66 | 0,21 | 0,68 |
| TBO | 2 | 1,20 | 0,42 | 0,36 | 0,43 |
| | 4 | 3,46 | 0,32 | 1,41 | 0,33 |
| CV | 0.35 | 2,93 | 0,42 | 0,43 | 0,43 |
| | 0.59 | 1,93 | 0,49 | 0,42 | 0,53 |
| STC | 11 | 2,13 | 0,54 | 0,39 | 0,43 |
| | 43 | 2,80 | 0,39 | 0,46 | 0,51 |
| Global average | | **2,13** | **0,46** | **0,63** | **0,47** |
| Number of better UBs | | 177 (33%) | 116 (21%) | 232 (43%) | 140 (26%) |

average deviations of VNFS are larger than those of MIPX. On the other hand, VNFS outperforms LR algorithm both in terms of average gaps and the number of better upper bounds.

Table 6 displays the average CPU times of the approaches for each class of instances. We can see that VNFS-2 has the smallest CPU time (5,86) according to VNFS-1 and VNFS. This is due to the decomposion scheme used in this algorithm, where, at each iteration, a single item problem is solved. On the other hand, the VNFS-1 algorithm appears to be the slowest algorithm. Besides, the CPU time and the gaps of the proposed algorithms increase when the number of items and TBO increase, and when the capacity is tightly constrained. In comparison with Brahimi et al. [2] approaches, the average CPU time values of our algorithms are relatively higher.

Table 6: Average CPU times of the six solution approaches proposed

| | | CPU (s) | | | | | |
|---|---|---|---|---|---|---|---|
| parameter | Value | MIPB | MIPX | LR | VNFS-1 | VNFS-2 | VNFS |
| | 10 | 3,32 | 3,33 | 2,09 | 5,3 | 5,28 | 5,47 |
| N | 20 | 5,79 | 5,81 | 4,14 | 6,96 | 6,39 | 6,63 |
| | 30 | 7,84 | 7,95 | 5,92 | 8,33 | 6,67 | 7,62 |
| | 75% | 4,73 | 4,05 | 3,09 | 5,13 | 4,95 | 5,86 |
| Capacity | 85% | 5,48 | 5,38 | 3,94 | 6,24 | 5,18 | 6,79 |
| | 95% | 6,73 | 7,66 | 5,12 | 9,01 | 5,32 | 6,54 |
| | 1 | 4,16 | 3,36 | 2,69 | 5,16 | 5,12 | 5,11 |
| TBO | 2 | 6,25 | 6,09 | 4,55 | 6,13 | 6,92 | 6,32 |
| | 4 | 6,54 | 7,64 | 4,90 | 8,18 | 7,28 | 7,43 |
| CV | 0.35 | 5,67 | 5,81 | 3,94 | 7,39 | 6,06 | 7,92 |
| | 0.59 | 5,63 | 5,59 | 4,16 | 9,04 | 5,73 | 6,64 |
| STC | 11 | 5,64 | 5,66 | 4,05 | 6,42 | 5,66 | 5,79 |
| | 43 | 5,66 | 5,73 | 4,05 | 6,53 | 5,62 | 5,87 |
| Global average | | **5,70** | **5,65** | **4,05** | **6,90** | **5,86** | **6,46** |

## 9. CONCLUDING REMARKS

In this paper we introduce a new variant of the Variable Neighborhood Search (VNS). It is designed for solving Mixed integer nonlinear programs, and we call it Variable neighborhood formulation search (VNFS). The integer part of the problem is treated as in a general VNS, while the continuous part is fixed. Then, for the fixed neighboring solutions, if variables are discrete, the optimal continues values are found by using some available commercial solver. The term 'formulation' comes from the fact that several mathematical programming formulations of the same problem may be used during the search.

Three variants of VNFS based heuristics are proposed for solving the multi-item capacitated lot-sizing problem with time windows and setup times, where the non-customer specific demand is considered. In the literature, this problem was never solved by a meta-heuristic framework. Therefore, we have taken the challenge to prove the efficiency of this class of methods.

The problem is formulated as a nonlinear mixed integer program. In each VNFS variant the problem is divided into two sub-problems: (i) search through the space of binary variables; (ii) for the fixed binary variables, find the continuous variables and the objective function values by using LP solver. Search through the 0-1 space following the rules of general VNS: the neighborhoods for shaking phase are induced from the Hamming distance; two neighborhoods are used within the local search. The VNS variants differ in mathematical programming formulation that is used to get the continuous values. We call our final heuristic Variable Neighborhood Formulation Search (VNFS) since it changes both the solution space and the formulation during the search for the better solution. The computational results have shown the efficiency of the proposed approaches. Our results compare favorably with recent state-of-the-art heuristics. Our work contributes to adding new value to the literature of lot sizing problems when the setup times and time windows constraints are considered. In the future, the proposed approach can be extended to solve other CLSP, inter alia, the multi-level problems. Also, forthcoming work may include application of our VNFS to other optimization problems which can be formulated as mixed integer programs.

## REFERENCES

[1] Absi, N., and Kedad-Sidhoum, S.,"The multi-item capacitated lot-sizing problem with setup times and shortage costs", *European Journal of Operational Research*, 185 (3) (2008) 1351-1374.

[2] Brahimi, N., Dauzère-Pérès, S., and Najid, N.M., "Capacitated multi-item lot-sizing problems with time windows", *Operations Research*, 54 (5)(2006) 951-967.

[3] Brahimi, N., Dauzère-Pérès, S., and Wolsey, L.A., "Polyhedral and Lagrangian approaches for lot sizing with production time windows and setup times", *Computers and Operations Research*, 37 (1) (2010) 182-188

[4] Brimberg, J., Hansen, P. ,and Mladenovic, N., "Attraction probabilities in variable neighborhood search", 4OR, 8(2) (2010) 181-194.

[5] Dauzère-Pérès, S., Brahimi, N., Najid, N., and Nordli, A., "The single-item lot sizing problem with time windows", Tech. Rep. 02/4/AUTO, Ecole des Mines de Nantes, France, 2002.

[6] Florian, M., Lenstra, J.K., and Rinnooy, Kan. A.H.G., "Deterministic production planning: Algorithms and complexity", *Management Science*, 26 (7) (1980) 669-679.

[7] Gelders, L.F., and Van Wassenhove, L.N., "Production planning: a review" *European Journal of Operational Research*, 7(2)(1981) 101-110.

[8] Hansen, P., and Mladenovic, N., "Developments in Variable Neighbourhood Search", in: *Esseys and Surveys in Metaheuristics*, C. Ribeiro, and P. Hanasen (eds.), Kluwer Academic Publishers, Dordrecht, 2002, 415–439.

[9] Hindi, K.S., "Solving the CLSP by a Tabu Search Heuristic", *The Journal of the Operational Research Society*, 47(1) (1996) 151-161.

[10] Jans, R., and Degraeve, Z., "Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches", *European Journal of Operational Research*, 177 (3) (2007) 1855-1875.

[11] Karimi, B., Ghomi, S.F., and Wilson, J., "The capacitated lot sizing problem: a review of models and algorithms", *Omega*, 31 (5) (2003) 365-378.

[12] Lee, C.Y., Çetinkaya, S., and Wagelmans, A.P.M., "A dynamic lot-sizing model with demand time windows", *Management Science*, 47 (10) (2001) 1384-1395.

[13] Liberti, L., Mladenović, N., and Nannicini, G., "A recipe for finding good solutions to MINLPs", *Mathematical Programming Computation*, 3 (4) (2011) 349-390.

[14] Millar, H.H., and Yang, M., "Lagrangian heuristics for the capacitated multi-item lot-sizing problem with backordering", *International Journal of Production Economics*, 34(1)(1994) 1-15.

[15] Mladenović, N., and Hansen, P., "Variable neighborhood search", *Computers and Operations Research*, 24(11)(1997), 1097-1100.

[16] Mladenović, N., Plastria, F., and Urosevic, D., "Reformulation descent applied to circle packing problems" ,*Computers and Operations Research*, 32 (5) (2005) 2419-2434.

[17] Mladenović, N., Plastria, F., and Urosevic, D., "Formulation space search for circle packing problems", *Lecture Notes in Computer Science*, 4638 (2007) 212-216.

[18] Mladenović, N., Todosijevic, R., and Urosevic, D., "An efficient general variable neighborhood search for large TSP problem with time windows " , *Yugoslav Journal of Operations Research*, 23 (1) (2013) 19-31.

[19] Miller, A.J., Nemhauser, G.L., and Savelsberg, M.W.P.," On the polyhedral structure of a multi-item production planning model with setup times" , *Mathematical Programming*, 94 (2003) 375-405.

[20] Özdamar, L., and Bozyel, M.A.," The capacitated lot sizing problem with overtime decisions and setup times", *IIE Transactions*, 32 (11) (2000) 1043-1057.

[21] Pardo, E., Mladenović, N., Pantrigo, J., and Duarte, A.," Variable formulation search for the cutwidth minimization problem", *Applied Soft Computing*, 13 (5) (2013) 2242-2252.

[22] Shim, I.S., Kim, H., Doh, H.H., and Lee, D.H., "A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs" *Computers and Industrial Engineering*, 61 (4) (2011) 920-929.

[23] Trigeiro, W.W., Thomas, L.J., and McClain, J.O., "Capacitated lot sizing with setup times", *Management Science*, 35 (3) (1989) 353-366.

[24] Van Den Heuvel,W., and Wagelmans,A.P.M., “Four equivalent lot-sizing models". *Operations Research Letters*, 36 (4) (2008) 465-470.

[25] Wagner,H.M., and Whitin, T.M., “Dynamic version of the economic lot size model". *Management Science*, 5 (1) (1958) 89-96.

[26] Wolsey, L.A., “Lot-sizing with production and delivery time windows", *Mathematical Programming*, 107 (3) (2006) 471-489.

[27] Xiao, Y., Kaki, I., Zhao, Q., and Zhang, R., “A variable neighborhood search based approach for uncapacitated multilevel lot-sizing problems" *Computers and Industrial Engineering*, 60 (2) (2011) 218-227.