A Variable Neighborhood Search Based Heuristic for the Lot Sizing Problem with an Emission Constraint

Norodin Ty

Erasmus University Rotterdam

Bachelor Thesis

2011

Supervisor Prof. dr. Albert P.M. Wagelmans

Abstract

In this paper we developed a Variable Neighborhood Search based heuristic for the lot sizing problem with an emission constraint. This is an extended version of the standard lot sizing problem, where the emission constraint is a capacity of the production quantities and inventory. The purpose of the heuristic is to rapidly find a good solution. The results show that this heuristic is highly efficient for production planning with a short time horizon.

Contents

1 Introduction

For every industrial company production planning is an important factor. They have to determine when and how much they will produce in order to satisfy all their clients by meeting their demands, while they want to keep the costs as low as possible. This form of production planning is called the *lot sizing problem*, where the total costs which exist of setup, inventory holding and production costs, have to be minimized over a certain time horizon.

Nowadays companies are bound to much stricter rules. Because of the climate change everyone becomes more aware of environmental issues like the CO_2 emissions. In order to reduce these CO_2 emissions and the pollute of plants, governments set environmental restrictions. In terms of the lot sizing problem this means that companies are limited in the quantity of products they produce and store in a certain period.

The original lot sizing problem is an LP-problem which is relatively easy to solve (see Wagner and Whitin (1958)). The lot sizing problem with an emission constraint however has proven to be NP-hard and is not as easy to solve. This thesis is a part of the research of M.J. Retel Helmrich on solving lot sizing problem with an emission constraint. The purpose of this thesis is to find a heuristic that provides a good solution for this problem without having a large computation time. The heuristic that we developed is based on a Variable Neighborhood Search. This method, developed by Hansen and Mladenovic (1999), has proven to be an efficient method for different kinds of problems. It uses a local search in different neighborhoods to find the global optimum. In this thesis we have datasets available of different lot sizing problems with an emission constraint. For every problem the optimal solution is also included in the datasets. In the end we will the compare the costs that are found by our heuristic with the optimal solution, and report the computation time.

2 Problem Description

In this section we will provide an exact explanation of our problem. The standard lot sizing problem consist of an objective function and a few restrictions, parameters and decision variables. The lot sizing problem with emission constraint is extended with an extra restriction and four parameters. In his paper we consider a finite horizon of T periods where $t = 1, .., T$ and a plant with an infinite capacity, meaning that there is no limit on how many products can be produced or stored in a period. In the problem we consider the plant only produces one kind of product.

2.1 Parameters

The first four parameters are the parameters for the standard lot sizing model. The last three ($\hat{K}_t$, $\hat{p}_t$, $\hat{h}_t$ and $\hat{C}$) are the extra parameters needed for the emission constraint. All the costs and demand are known beforehand, but can vary over time.

$K_t =$ set up costs in period t $\quad t = 1, \ldots, T$

$p_t =$ production costs per unit for period t $\quad t = 1, \ldots, T$

$h_t =$ inventory holding costs per unit remaining at end of period t $\quad t = 1, \ldots, T$

$D_t =$ demand in period t $\quad t = 1, \ldots, T$

$\hat{K}_t =$ set up emission costs in period t $\quad t = 1, \ldots, T$

$\hat{p}_t =$ production emission costs per unit for period t $\quad t = 1, \ldots, T$

$\hat{h}_t =$ inventory holding emission costs per unit remaining at end of period t $\quad t = 1, \ldots, T$

$\hat{C} =$ the emission capacity

2.2 Decision variables

For every period t, we have to decide if we will produce (y_t=0/1) and if this is the case, the quantity (x_t). The inventory (I_t) depends on the production periods.

$y_t = 1$ if there will be produced in period t, 0 otherwise $\quad t = 1, \ldots, T$

$x_t =$ Production quantity in period t $\quad t = 1, \ldots, T$

$I_t =$ Inventory remaining at the end of period t $\quad t = 1, \ldots, T$

2.3 Objective function and restrictions

$$min \sum_{t=1}^{T} (K_t y_t + p_t x_t + h_t I_t) \tag{1}$$

$$I_t = I_{t-1} + x_t - d_t \quad t = 1, \ldots, T \tag{2}$$

$$s.t. \qquad x_t \leq D_{t,T} y_t \qquad t = 1, \ldots, T \tag{3}$$

$$x_t, I_t \geq 0 \qquad t = 1, \ldots, T \tag{4}$$

$$y_t \in \{0, 1\} \qquad t = 1, \ldots, T \tag{5}$$

$$I_0 = 0 \tag{6}$$

$$\sum_{t=1}^{T} (\hat{K}_t \hat{y}_t + \hat{p}_t \hat{x}_t + \hat{h}_t \hat{I}_t) \leq \hat{C} \tag{7}$$

In (1) the objective function is given, it wants to minimize to total costs. These costs consists out of the set up costs ($K_t y_t$), the total production costs ($p_t x_t$) and the inventory cost ($h_t I_t$) per period t. Note that in some periods there will not be produced so there will not be any set up and production costs.

The objective function is subject to several restrictions. The restriction in (2) makes sure that the demand in every period satisfied. The amount of the production in stock at the beginning minus the amount of production in stock at the end of a period plus the amount produced must be equal to the demand of that period. The constraint in (3) has the function that if there will not be produced in a period, meaning that $D_{t,T} y_t$ is equal to zero, x_t will also be equal to zero. The third constraint (4) says that the production quantity and inventory cannot be negative. In (5) it says that y_t can only be one and zero. A one when there will be produced in period t, and a zero when this is not the case. The restriction in (6) says that we start with no products in stock. The last constraint in (7) is the emission constraint. Just like the objective function the sum of all the costs per periods are taken except they are now calculated with the emission parameters. These total costs, (actually it is more a penalty because these are not actual costs), has to be less than or equal to $\hat{C}$, the emission capacity.

2.4 Data sets

For his research on the lot sizing problem with an emission constraint, M.J. Retel Helmrich created 600 test problems. In this paper we use these test problems to test the heuristics we created. The problems can be divided in different categories. In defining the these categories the zero inventory property (ZIP) plays an important role. This property makes sure that in every period t where there will be produced, there is no inventory left from the previous periods. Meaning that for all production periods only the exact quantity of products that is needed till the next production period will be produced. In the lot sizing problem with emission constraint this property does not necessarily hold. However it is of importance because the property holds for all the solutions of the algorithms we create in this paper. Below we provide a short description of the different test problems:

A) 1. Optimal solution meets the zero inventory property.

   2. Optimal solution does not necessarily meet the zero inventory property.

   3. These problems have the feature that all periods are alternately cheap and clean for the even periods and expensive and polluting for the odd periods.

B) The set-up costs and the set-up emission costs are generated from three different uniform distributions with the mean 1000,5000,10000.

C) The length of the periods T are 25, 50 or 100.

D) For every category there are 10 unique problems generated.

E) For every problem three different emission capacities $\hat{C}$ are given, the demand D per period and all the costs K,p,h,$\hat{K}$,$\hat{p}$,$\hat{h}$ for every period.

Besides all these parameters we need for the heuristic, the optimal solution is also known for every test problem.

3 Methods

The heuristic we present in this paper consists of three parts: the Initialization, the Variable Neighborhood Search (VNS) and a disruption. In this section we will describe the three parts individually. In figure 1 the pseudo code is given. The heuristic starts with finding an initial solution and is followed by an iterative process where the VNS and the disruption part are repeated until the search is terminated. For every iteration where a better solution is found, the solution with corresponding costs will be saved. In section 4 we will discuss how many iterations are preferred before we terminate the search.

```
Create initial solution
REPEAT
        Variable Neighborhood Search
        IF current solution is feasible
                IF current costs < minimum costs
                        minimum costs = current costs
                        best solution = current solution
                ENDIF
        ENDIF
        current solution = best solution
        Disruption
UNTIL search terminated
```

Figure 1: *Pseudo code for the algorithm*

3.1 Initialization

The first step of the heuristic is to find an initial solution. The work of Wagner and Whitin (1958) describes how to find the optimal solution for the dynamic lot sizing problem given that the solution has the zero inventory property. If we apply this Wagner-Whitin algorithm for our problem, the solution probably will not be feasible because of the emission constraint. However if we solve the algorithm using the costs of these emission constraints we can guarantee that the solution is feasible. That this solution could be far from optimal does not really matter, as it is just an initial solution. In Evans (1985) an efficient way to implement the Wagner-Whitin algorithm is described. This implementation guarantees a fast computation time.

3.2 Variable Neighborhood Search

When the initial solution has been created the Variable Neighborhood Search is executed. The basic VNS described in Hansen and Mladenovic (1999) consists of an initial solution x_0 and a set of neighborhood structures N_k with k different neighborhoods. The incumbent solution will be set to $x_0(x \leftarrow x_0)$ and k to the first neighborhood ($k \leftarrow 1$). A local search will be applied to find a better solution. If a better solution cannot be found k will be set to $k+1$. If there is a better solution, this solution will be set as the new incumbent solution and k will be set to 1. This will be repeated until a certain stopping condition is met, this could be the maximum CPU time allowed or a maximum number of iterations. In figure 2 the steps of this basic VNS are given.

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:
(1) Set $k \leftarrow 1$; (2) Until $k = k_{max}$, repeat the following steps:
(a) *Shaking.* Generate a point x' at random from the k^{th} neighborhood of x ($x' \in \mathcal{N}_k(x)$);
(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Figure 2: *Pseudo code for the basic VNS as described in Hansen and Mladenovic (1999)*

3.2.1 Neighborhood structure

All the solutions in our algorithms meet the zero inventory property, this way the solution for the lot sizing problem only depends on the production periods y. For every possible y, there can be only one possible combination of x and I because there is no inventory left in the next production period. In our heuristic there are two different neighborhood structures. We define the neighborhood as the difference between any two solutions:

$P(y/y^{'})$=$|y/y^{'}|$, where $|./.|$ denotes the number of different points between the two solutions $\sum_{t=1}^{T} |y_t - y_t^{'}|$.

1. The first neighborhood N_k has all the set of all solutions where k=$P(y/y^{'})$=1.

$$\text{y}= \begin{bmatrix} 1 \\ 0 \\ [1] \\ 1 \\ 0 \end{bmatrix} \qquad \text{y}^{'} = \begin{bmatrix} 1 \\ 0 \\ [0] \\ 1 \\ 0 \end{bmatrix}$$

In this neighborhood we change one production period y_t, from a one to a zero or vice versa. As we can see in the notations shown above only one production period changes. If the production horizon is T we can calculate that this neighborhood consists of T possible solutions.

2. The second neighborhood N_k has all the set of all solutions where k=$P(y/y^{'})$=2.

$$\text{y}= \begin{bmatrix} 1 \\ [0] \\ [1] \\ 1 \\ 0 \end{bmatrix} \qquad \text{y}^{''} = \begin{bmatrix} 1 \\ [1] \\ [1] \\ 1 \\ 0 \end{bmatrix}$$

The notations shown above indicates we change two production periods in this neighborhood. However this neighborhood does not consist all the $P(y/y^{'})$=2 solutions. We limited this neighborhood to all the production periods where it is possible to produce in one period earlier or in a period later. This means that $y_t = 1 \rightarrow 0$ and $y_{t-1} = 0 \rightarrow 1$ (producing in period $t-1$ instead of t) or $y_t = 1 \rightarrow 0$ and $y_{t+1} = 0 \rightarrow 1$ (producing in period $t+1$ instead of t). This is only possible if y_{t-1} or y_{t+1} is zero in the first place. On forehand we cannot calculate the size of this neighborhood because it depends on y and how many times this situation occurs.

3.2.2 Best improvement and first improvement

In Hansen and Mladenovic (1999) there are two different types of VNS discussed. The first one is the VNS in a steepest descent manner. Meaning that each possible solution in a neighborhood is examined and the solution that would bring the most improvement will be set as the incumbent solution. The disadvantage of this Best improvement implementation is that it takes time to examine every solution and calculate the corresponding costs. The other possibility is the quickest descent manner or First improvement. Instead of examining the whole neighborhood, every time a solution that brings improvement is found this solution will be set as the incumbent solution immediately. In this paper we will create two algorithms, in the first Best improvement algorithm the VNS is executed with a steepest descent manner, in the second First improvement algorithm the VNS is executed with the quickest descent manner.

3.2.3 Best Improvement

In figure 3 the pseudocode for the VNS where 'Best improvement 'is implented is given.

```
1 Start with initial solution y0, and calculate costs.
2 Lowest Costs (LC) = costs y0.
3
4 Neighborhood 1:
5       Check all possible solutions y' in Neighborhood 1 and calculate emission costs
6               IF y' is feasible
7                       Calculate costs of y' and store in vector N1C
8               ELSE N1C is 10.0E99
9               ENDIF
10      IF minimum N1C < LC
11      Set y <- y' belonging to min(N1C), LC=min(N1C) and go back to step 5.
12      ELSE Keep y, clear N1C and go to Neighborhood 2.
13      ENDIF
14 Neighborhood 2:
15      Check all possible solutions y'' in Neighborhood 2 and calculate emission costs
16              IF y'' is feasible
17                      Calculate costs of y' and store in vector N2C
18              ELSE N2C is 10.0E99
19              ENDIF
20      IF minimum N2C < LC
21      Set y <- y' belonging to min(N2C), LC=min(N2C) and go back to step 5.
22      ENDIF
23 End of VNS
```

Figure 3: *Pseudo code for the VNS with the 'Best improvement' implemented*

In the first step the incumbent solution y is the initial solution that was found in section 3.1 using the Wagner Whitin algorithm. Every time when a neighborhood is entered all the solutions, $y^{'}$ (or $y^{''}$) from the neighborhood of the incumbent solution y are examined. For every $y^{'}$ (or $y^{''}$) the emission costs are calculated. If these costs do not exceed the

emission capacity the total costs will be calculated and will be stored. If the emission costs do exceed the emission capacity, a penalty will be given and instead that the total costs will be calculated a very large number will stored. When all the possible solutions in the neighborhood are examined, y will be set to the solution of $y^{'}$ that gives the lowest total costs. This new solution will enter N_1 and the whole process will be repeated. Note that it does not matter if we are in N_1 or N_2 if a better solution is found it will go back to the first neighborhood. It is also possible that after all the solutions in a neighborhood are examined, there is no better solution found than the incumbent. When this happens in N_1, we move on to the second neighborhood. If this happens in N_2, the VNS is finished and the output is y.

3.2.4 First Improvement

In figure 4 the pseudocode for the VNS where 'First improvement 'is implented is given.

```
1 Start with initial solution y0, and calculate costs.
2 Lowest Costs (LC)=costs y0
3
4 Neighborhood 1:
5       Search for first solution y' in Neighborhood 1 and calculate emission costs
6               IF y' is feasible
7                       Calculate costs of y'
8                                       IF costs y' < LC
9                                       Set y <- y', LC=costs y' and go back to step 5.
10                                      ENDIF
11              ELSE IF all possible solution of y' are examined and none are feasible
12                      go to Neighborhood 2
13              ELSE check next solution y' in Neighborhood 1, calculate emission costs and
14                      go back to step 6
15              ENDIF
16 Neighborhood 2:
17      Search for first solution y'' in Neighborhood 1 and calculate emission costs
18              IF y'' is feasible
19                      Calculate costs of y''
20                                      IF costs y'' < LC
21                                      Set y <- y'', LC=costs y'' and go back to step 5.
22                                      ENDIF
23              ELSE IF all possible solution of y'' are examined and none are feasible
24                      go to step 25
25              ELSE check next solution y'' in Neighborhood 2, calculate emission costs and
24                      go back to step 18
25              ENDIF
25 End of VNS
```

Figure 4: *Pseudo code for the VNS with the 'First improvement' implemented*

Just like in the VNS where Best improvement is implemented, the VNS with First Improvement starts with setting the incumbent solution y as the initial solution found by the Wagner Whitin algorithm. The difference from Best improvement is that if we enter a neighborhood, instead of checking all possible solutions in the neighborhood, only one

solution will be examined. The emission costs will be calculated to check if this solution is feasible. If this is the case the total costs will be calculated and compared with the costs of the incumbent solution. When these total costs are lower, $y^{'}$ (or $y^{''}$) will be set to the incumbent solution immediately. Independent of what neighborhood we are currently in, we enter N_1 again with the new solution. If $y^{'}$ (or $y^{''}$) is not feasible or if the costs are not improving, we search for a new solution $y^{'}$ (or $y^{''}$) in the same neighborhood. If there are no improving solutions are found in a neighborhood we exit the neighborhood. If we are in N_1, we move on to N_2. If this happens in the second neighborhood the VNS is finished and have y as output.

3.3 Disruption

The variable neighborhood search is followed by a disruption. The VNS part of the heuristic is able to find a local optimum. However it could be possible that the global optimum lie far away from the local optimum we just found. By running the VNS again but starting with a different initial solution y we may find this global optimum. It has no use to run the initialization algorithm of section 3.1 again because the Wagner Whitin algorithm always gives the same outcome for a given problem. That is why we disrupt the current best solution y. The disrupted y will be the new 'initial' solution for the VNS. The disruption is done by randomly change a few production periods, from a one to a zero or vice versa. Only the first period cannot be changed because we always produce in the first period. The number of periods that will be disrupted is a percentage of the number of periods T. This percentage is a parameter we can adjust. Later we will determine what percentage is preferred for the heuristic.

4 Results

In this section the performance of the heuristics will be tested. The test problems, described in section 2.4, are divided into 9 categories: the three different 'A' problems and for each 'A' problem the three different planning periods T are separated. For every problem we calculated the relative difference with the optimal solution percentage wise ((solution found - optimal solution) / optimal solution*100 %) and the computation time in seconds. For each category the average relative difference with the optimal solution, the average computation time and the percentage that the optimal solution is found are reported. Because for every problem there were three different emission capacities $\hat{C}$ we find for every category three values, where the first one stands for the lowest (strictest) capacity and the third for the highest capacity. We start with comparing the Best improvement with the First improvement heuristic and deciding the two input parameters.

4.1 Best heuristic

The heuristics we described earlier are quite similar but it is expected that the Best improvement heuristic gives a better solution while the First improvement heuristic is expected to have lower computation time. Because of the large number of test problems we first want to test what heuristic is more efficient before we run all the problems. That is why we compare the two heuristics with only a part of the test cases. We also used these test cases to decide the other input parameters: the amount of iterations and the percentage of the periods that is disrupted, the disruption percentage. After some trial and error we discovered that if the disruption percentage is too high, then it is almost impossible to find a feasible solution. Intuitively this seems logic: the chances to find a feasible solution after a random disruption slink when the new solution differs a lot. We found that a disruption percentage around 5 or 10 percent would be the most efficient. For the amount of iterations we tested for several problems how much iteration it took before they found their minimum costs. We found out that most problems found their best solutions before the fiftieth iteration, however there were also some problems who found there optimum in a later stadium. Figure 5 gives an example of the progress for one test case.

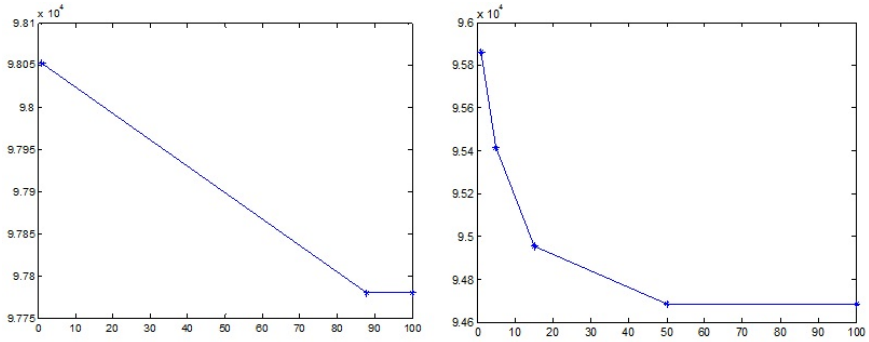

Figure 5: *This is an example of the progress of the best solution found so far(on the y-axes) after a certain amount of iterations (x-axes). Both graphs are from the same test problem (A=0, 50 periods) but the graph on the left has an stricter emission constraint.*

The two heuristics can be compared by their computation time, their relative difference with the optimal solution and how many times the optimal solution is found. The comparison is made using 90 test problems and we have tried three different inputs:

- Input 1: 100 iterations and a disruption percentage of 5
- Input 2: 50 iterations and a disruption percentage of 10
- Input 3: 100 iterations and a disruption percentage of 10

In the tables below the results of the two heuristics given these inputs are shown:

Table 1: *Input 1*

	Best improvement			First improvement		
	25	50	100	25	50	100
Computation time in seconds	1.24	3.95	13.15	1.07	3.36	11.02
	1.41	5.32	18.52	1.21	4.09	13.47
	1.59	6.72	25.23	1.37	4.78	16.83
Relative difference optimum in %	**0.06**	0.28	**0.38**	0.08	0.60	1.74
	0.01	**0.03**	**0.17**	0.14	0.25	0.41
	0.05	0.06	**0.05**	0.10	0.13	0.16
Optimum found in %	**96.67**	43.33	**13.33**	90.00	16.67	0
	96.67	80.00	**36.67**	86.67	50.00	13.33
	96.67	70.00	**46.67**	80.00	60.00	26.67

Table 2: *Input 2*

	Best improvement			First improvement		
	25	50	100	25	50	100
Computation time in seconds	0.54	1.71	6.20	0.45	1.41	5.04
	0.68	2.38	7.81	0.54	1.74	5.49
	0.83	3.35	10.32	0.65	2.05	6.18
Relative difference optimum in %	0.06	0.58	0.72	0.90	1.90	4.01
	0.02	0.09	0.40	0.14	0.56	1.76
	0.06	0.09	0.23	0.13	0.22	0.42
Optimum found in %	93.33	30.00	06.67	73.33	03.33	0
	96.67	66.67	13.33	83.33	26.67	03.33
	90.00	50.00	26.67	80.00	46.67	13.33

Table 3: *Input 3*

	Best improvement			First improvement		
	25	50	100	25	50	100
Computation time in seconds	1.03	3.24	11.19	0.90	2.83	10.05
	1.34	4.48	13.48	1.10	3.37	10.77
	1.64	6.21	18.38	1.30	4.10	12.14
Relative difference optimum in %	**0.06**	**0.25**	0.66	0.13	1.61	3.28
	0.00	**0.03**	0.41	0.09	0.41	1.36
	0.05	**0.02**	0.18	0.15	0.15	0.23
Optimum found in %	**96.67**	**43.33**	06.67	83.33	0	0
	96.67	76.67	13.33	86.67	40.00	0
	90.00	**76.67**	33.33	80.00	60.00	10.00

If we compare the two heuristics it can be seen that Best improvement gives, for all categories and inputs, a solution that is closer to the optimum, while First improvement results in lower computation times. It seems that the difference in computation times seems smaller than the difference of the best solution between the two algorithms, indicating that the Best improvement heuristic gives a better algorithm. However it has to be taken into account that there is a possibility that the First improvement heuristic could give a better solution if the running time was just as long. We can compare the results of Best improvement input 2 with First improvement input 3. Although First improvement has two times as many iterations, what results in a longer computation time, the solutions still cannot match the solutions of Best improvement. We assume that Best improvement gives us the best model, but we still have to compare the three inputs. In three tables the best values are expressed in boldface. For T= 25 and T=50, input 1 and 3 gives almost similar results, however the computation times of input 3 are slightly faster, so input 3 is preferred. For 100-periods problems input 1 gives clearly better results than the other two. That input 1 and 3 are better models is because it has two times as many iterations. The fact that a disruption percentage of 5 percent is preferred instead of the 10 percent for T=100 also seems logical. When 10 instead of 5 periods are disrupted the chances to find a feasible

solution slink. For 25 and 50 periods, 10 percent disruption only means that 3 and 5 periods are disrupted.

4.2 Final results

In the last section we concluded that the most efficient heuristic is the Best improvement heuristic with 100 iterations, a disruption percentage of 10% for 25 and 50 periods and a disruption percentage of 5% for 100 periods. We used this model for all the 600 test problems. The results are given separately for the three categories of A and each for the three different lengths of periods. Every category is also divided with the three different emission capacities. Where 1 stands for the lowest emission costs allowed and 3 for the highest. All the results are an average of the corresponding test problem per category. In table 4 these results are given.

Table 4: *Final results*

	A:	1			2			3		
	T:	25	50	100	25	50	100	26	50	100
Relative difference	1	0.48	2.09	2.27	0.48	2.59	2.47	1.63	5.28	7.06
optimum in %	2	0.44	1.29	1.70	0.43	1.33	1.45	1.49	5.19	5.56
	3	0.19	0.62	0.86	0.36	0.64	0.67	1.15	1.70	2.86
Optimum found in %	1	77.8	20.0	4.4	58.9	6.7	1.1	20.0	0	0
	2	82.2	33.3	7.8	64.4	27.8	6.7	30.0	0	0
	3	84.4	47.8	13.3	76.7	47.8	11.1	65.0	45.0	20.0
Computation time in seconds	1	0.84	2.53	9.79	0.99	3.06	11.64	1.33	3.51	13.72
	2	1.03	3.14	12.54	1.32	4.33	17.71	1.68	5.75	22.76
	3	1.23	4.11	16.79	1.63	5.91	24.30	2.06	8.49	32.48

The results show some logical patterns. It can also been seen that the computation times take longer when the emission cost become larger. When the emission capacity become larger, the set of feasible solution become bigger what could result in a larger computation time. There are also differences in the results between the three periods T, as well the solutions as the computation times are better if the time period is shorter. That is why we will take a look at the results for each period separately.

- T=25: The category A1 and A2 does not differ very much from each other. It was expected that the A1 problems would perform better because of the zero-inventory-optimum. This is the case, but except for the least strict capacity, the difference is not very large and the computation times are quite similar. The results for the A3 category stays behind on the other two, besides the longer computation time the difference with the optimum is almost twice as large compared to the other categories. Overall the heuristic seems to do an excellent job for T=25, in the best case the relative difference is on average only 0.19% and the worst case 1.63% on average. While the computation times are around one or two seconds.

- T=50: The same pattern can be seen as from the 25-periods problem. The A1 and A2 results are quite similar. The A1 problems are slightly better but the difference is not that much, only for the strictest problem the difference is a little bit larger. The results of the A3 category on the other hand stay behind. Longer computation times and the difference percentage wise with the optimum is in for two capacities more than 5% on average. The results for the other two categories seem very decent.

- T=100: For the 100 period problems the results for A1 and A2 are very similar. For two capacities the relative difference with the optimum is even better in the A2. Again we see that the results of the A3 category stay behind. With average relative difference between 0.67 and 2.47% we can conclude that the model gives a decent solution. The computation time on the other hand are almost three times as much as in the T=50 period. We can see that the model is less effective for 100 period problems.

5 Conclusion

In this paper a VNS based heuristic is used to solve the lot-sizing problem with an emission constraint. Because this problem is a NP-hard problem it is not easy to find an exact solution. The heuristic introduced in this paper shows that the results are very decent. They are close to the optimum and in many cases the optimum was found. Especially for short horizon of 25 periods the heuristic performed well: the computation time took on average about one second and in 80% of the problems the optimum was found. For the problems where the costs come in pairs (A3 problems) the results stay behind. The relative difference from the optimum was larger and it also took more time to compute. Even for the problems where the optimum does not necessarily meet the zero inventory property, very good results are reported, with only a slightly longer computation time than the problems where the optimum meet the ZIP. We can conclude that this heuristic is efficient for production planning with a short time horizon.

References

James R. Evans. An efficient implementation of the wagner-whitin algorithm for dynamic lot-sizing. *Journal of Operations Management*, 1985.

Pierre Hansen and Nenad Mladenovic. Variable neighborhood search: Principles and applications. *Elsevier*, 1999.

Harvey M. Wagner and Thompson M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 1958.