Erasmus Universiteit Rotterdam
Erasmus School of Economics

Master Thesis

Econometrics & Management Science

---

# Parameter Determination in Relax-and-Fix Heuristics

---

*Author* | *Student number*

Kars Smidt | 400201

*Supervisor*

Wilco van den Heuvel

*Second Assessor*

Albert Wagelmans

**Abstract**

Relax-and-Fix (RF) heuristics are normally executed using parameters determined by trial-and-error. Using numerical methods, the parameters of the Relax-and-Fix heuristic can be optimized. In this thesis, the definition of a Relax-and-Fix parameter is reviewed. Then, the parameters are optimized using two different methods. At first, a method which seeks optimal subproblem orders for the RF heuristic is introduced. Secondly, the subproblem sizes of the RF heuristic are made dynamic, to check if the commonly used homogeneous subproblem size can be improved.

April 30, 2020

# Contents

# 1 Introduction

Through the years, the boundaries of computational power have been sought and stretched. Central processing units have exponentially grown in their processing power over the years. However, there are still many optimization problems which become too large too fast, such that they can not be solved efficiently by common computers using common commercial solvers. Therefore, we should strive to not only improve in computing power, but also to reduce optimization problems to easily-solvable problems for the solving software.

To reduce problem complexity, a technique called 'relax-and-fix' (RF) can be used. This technique is described in Wolsey (1998). The relax-and-fix heuristic can be applied to many problems, for which it can have differing results. In short, the relax-and-fix heuristic optimizes the mathematical problem sequentially, with different fixed, binary and relaxed variables in every iteration, until a final solution is reached. The RF heuristic generally gives good results for problems with a natural ordering of variables. In the case of a lot sizing problem, there is a natural ordering of the variable when looking at time. Production levels are determined periodically and items can be in hold for a period of time to suffice the demands of the subsequent periods. For problems without a natural ordering of the variables, such as a Vehicle Routing Problem (VRP), the RF heuristic is not as widely used in previous literature. This raises the question if it is possible to find some ordering of variables such that the RF heuristic gives satisfying results for problems without a natural variable ordering.

The RF heuristic does not have to result in a global optimum for the complete mathematical problem, because the composite solution of the subproblems is only guaranteed to be optimal on the subproblem level. Therefore it is of paramount importance to make an educated decision of the parameters that define the subproblems. The sizes of the subproblems and the total amount of the subproblems, together with the sequence in which they are solved, are the key parameter decisions to be made.

This leads to the question: How do you define the parameters of the relax-and-fix heuristic such that it performs the best it can? Furthermore, is it possible to construct a

heuristic which finds a natural ordering of variables for any mathematical problem, such that the RF heuristic gives satisfying results? And finally, does this heuristic perform equally well on multiple different mathematical programming problems?

In the coming section, a literature review is given to provide academical context to the reader. Afterwards, the methods to determine the relax-and-fix parameters are explained. Consequently, the results of these methods are compared to each other and to the general solution. Finally, a conclusion will be drawn from these results.

# 2 Literature Review

The RF heuristic has been used over a varying set of problems in previous research, but mostly for variants of the Lot Sizing Problem (LSP). The first ones to implement the idea of Relax-and-Fix on the Lot Sizing Problem are Dillenberger et al. (1994). Their RF heuristic has been implemented in the resource planning structure of the company for which they have done the research, indicating the practical relevance of the subject. The RF heuristic has also been implemented by Stadtler (2003). In this paper, the method is called an "Internally Rolling Schedules with Lot-Sizing Windows" approach, but the idea is the same as in Relax-and-Fix. Other authors to have successfully used RF on variants of the LSP more recently are Soler et al. (2019), Roshani et al. (2017) and Mohammadi et al. (2010)

A comparable method to the RF heuristic is the Fix-and-Optimize (FO) heuristic. This heuristic does not relax variables, but instead fixes them. The FO heuristic runs multiple times over the set of binary variables to be fixed, to converge to a solution. For the execution of the FO heuristic, an initial feasible solution is needed, as all non-binary variables should already by fixed in the starting situation. Toledo et al. (2015) combine Relax-and-Fix with a Fix-and-Optimize heuristic to produce quality results for the ML-CLSP.

Metaheuristics have already been used in previous literature as a means to improve the solution quality or the computation time of the RF Heuristic. For example, Pedroso and Kubo (2005) introduced a Tabu Search heuristic for the RF for lot sizing problems. The Tabu Search is used to destruct a part of an initial RF heuristic solution and reconstruct it using the RF heuristic again, this way the initial solution is improved.

Further more, the RF heuristic has been used on other problems than the LSP, although not as much. For example, De Oliveira et al. (2014) use the RF heuristic to determine the solution for their Traveling Umpire Problem, a variant of the Assignment Problem. Another example is given by Noor-E-Alam and Doucette (2012). They use the RF heuristic to compute solutions for their Grid-Based Location Problem.

A few authors have also experimented with improving the RF heuristic. An example is given by Gonźalez et al. (2014), who use the RF heuristic to solve the Fixed Charge Network Design Problem. They fix variables only if it is certain that it will lead to feasible solutions in the following iterations. Lastly, Ribeiro et al. (2019) solved a Multicast Routing Problem, formulated using a Multi-Commodity Flow Network, using the RF heuristic.

# 3 Methodology

To determine good parameters for the RF heuristic, we want to improve the RF heuristic using two different angles of approach. The first approach optimizes the ordering of the variables over which the RF heuristic is executed. The second approach optimizes the way the RF heuristic is executed over a given set of variables which are already ordered in a certain way. Before the methods are explained in greater detail, a more precise definition of the standard relax-and-fix heuristic is given.

## 3.1 Standard Relax-and-Fix Heuristic

A full definition of the relax-and-fix heuristic can be found in Wolsey (1998), but because of its importance for this report, the RF heuristic will be explained briefly by means of its pseudocode and a short example. For this, we will walk through the execution of an RF heuristic over a standard binary knapsack problem. The RF heuristic is written in pseudocode as follows:

**Algorithm 1** Relax-and-Fix heuristic

1: **procedure** RELAX-AND-FIX($window$,$increment$,$knapsack$)
2: $x_i^{binary} \in \{0,1\} \quad \forall i \in window$
3: $x_i^{relax} \in [0,1] \quad \forall i \notin window$
4: $x_i^{fix} \leftarrow \emptyset$
5: $Solution \leftarrow \text{SOLVE}(knapsack, x_i^{binary}, x_i^{relax}, x_i^{fix})$
6: **while** not all variables are fixed **do**
7: $newWindow \leftarrow move(window, increment)$
8: $x_i^{fix} \leftarrow x_i^{binary} \quad \forall i \in window \setminus newWindow$
9: $x_i^{binary} \in \{0,1\} \quad \forall i \in newWindow$
10: $Solution \leftarrow \text{SOLVE}(knapsack, x_i^{binary}, x_i^{relax}, x_i^{fix})$
11: $window \leftarrow newWindow$
12: **end while**
13: **return** $Solution$
14: **end procedure**

The mathematical programming formulation of a knapsack problem instance can be found in Equations (1)-(3).

$$\textbf{max } 10x_1 + 40x_2 + 30x_3 + 50x_4 \tag{1}$$

$$\textbf{s.t. } 5x_1 + 4x_2 + 6x_3 + 3x_4 \leq 10 \tag{2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in N \tag{3}$$

Where item $i$ is from set of items $N$ and $x_i$ is a binary variable where $x_i = 1$ if we take item $i$ with us. The objective is to maximize utility. The formulation where all binary variables are kept binary will be called the master problem.

To execute the RF heuristic, we need to determine the window parameter and the increment parameter. For this example, we set the window parameter to 2 and the increment to 1. In the first iteration the first two variables are set as binaries, the rest is relaxed. This gives the solution as in Table 1b. Such a problem formulation, where a part of the binary variables in the master problem are fixed or relaxed, will be called an RF subproblem, as multiple of these subproblems have to be solved to get a feasible solution to the master problem. In the following iteration, the window is incremented by 1, and the first binary variable of the previous iteration will be set as a fixed value. This process continues until all values are fixed or have a binary value.

Table 1: Solution values of the RF heuristic on the knapsack problem

(a) Coloring code

| | |
|---|---|
| Fixed | $x_i$ |
| Binary | $x_i$ |
| Relaxed | $x_i$ |

(b) Iteration 1

| Variable | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| Value | 0 | 1 | 0.5 | 1 |

(c) Iteration 2

| Variable | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| Value | 0 | 1 | 0 | 1 |

(d) Iteration 3

| Variable | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| Value | 0 | 1 | 0 | 1 |

(e) Iteration 4

| Variable | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| Value | 0 | 1 | 0 | 1 |

The global optimum of this knapsack problem is 90, with $x_2$ and $x_4$ being 1 and $x_1$,

$x_3$ are 0. As can be seen in Table 1e, the heuristic solution is the same as global optimum in this case, but many times it is not. For larger, more complicated problems, the RF heuristic does almost never return optimal values.

It is known that the Relax-and-Fix heuristic can give satisfactory solutions to the lot sizing problem, as described in Helber and Sahling (2010) and Toledo et al. (2015). But the parameter determination is always done by making an educated guess and possibly improving them by trial and error. We aim to automate the parameter definition in a general way, while striving for a solution with a very small optimality gap. The optimization of heuristic parameters is more commonly called 'meta-optimization', as the optimization itself is being optimized.

## 3.2 Feasibility Issues

Theoretically it is possible that the RF heuristic does not return a feasible solution. In any problem formulation, there is only one way in which this could be the case. Between RF iterations it could happen that the mathematical problem becomes infeasible, when the binary window moves such that some previously relaxed variables now have to return binary values. Intuitively this is correct, as the solution space decreases when a variable can be anything between zero and one beforehand, but afterwards can only become zero or one distinctly. It is more easily noticeable when discussed using a small example. We consider a small instance of a vehicle routing problem in Equations (4)-(10).

$$\min \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} \tag{4}$$

s.t.

$$\sum_{j=1,j\neq i}^{n+1} x_{ij} = 1, \qquad i = 1,\dots,n \tag{5}$$

$$\sum_{i=0,i\neq h}^{n} x_{ih} - \sum_{j=1,j\neq h}^{n+1} x_{hj} = 0 \qquad h = 1,\dots,n \tag{6}$$

$$\sum_{j=1}^{n} x_{0j} \leq K \tag{7}$$

$$y_j \geq y_i + d_j x_{ij} - Q(1 - x_{ij}) \qquad i,j = 0,\dots,n+1, i \neq j \tag{8}$$

$$d_i \leq y_i \leq Q, \qquad i = 0,\dots,n+1 \tag{9}$$

$$x_{ij} \in \{0,1\}, \qquad i,j = 0,\dots,n+1 \tag{10}$$

The binary decision to make is if a vehicle travels to node $j$ after node $i$. This is denoted using the binary variable $x_{ij}$. Here, $y_i$ is a variable which denotes the cumulative demand satisfied of the traversed route up to and containing node $j$. $c_{ij}$ is defined as the euclidean distance between node $i$ and $j$. Furthermore, node $i$ has demand $d_i$ and we have $K$ vehicles available with maximum capacity $Q$. Equations (5) ensure that every node is visited. Equations (6) guarantee that each vehicle entering a non-depot node also leaves the node. Then, Equation (7) caps the amount of departing vehicles from the depot to $K$. The Equations (9) and (10) make sure the vehicle does not exceed its capacity and prohibit the creation of subtours.

When relaxing the binary variable $x_{ij}$, it can for example take on the value 0.5. This would mean that a node is visited partially. Intuitively, one can understand this by stating that a vehicle can pull a total of 4 cargo-holding units, but it only takes 2, so half of the vehicle travels to a certain node.

If a node is visited partially, the capacity constraint will become less restricted. Normally, when a node is visited, the capacity constraint will be of the following form:

$$y_j \geq y_i + d_j x_{ij} \tag{11}$$

But because the demand of a node can be satisfied using partial visits, the capacity constraint of a feasible solution could look like equation (12) if $x_{ij} = 0.75$.

$$y_j \geq y_i + d_j 0.75 - 0.25Q \tag{12}$$

This means that if a node $j$ is visited partially from 2 different nodes 1 and 2, but their total $\sum_{i=1,2} x_{ij}$ equals 1, then the cumulative demand constraint does not hold as strictly because of the subtraction of a value $\alpha Q$ to the constraint, where $\alpha \in (0, 1)$.

This causes issues in the RF heuristic, where we want to deduce a solution of the CVRP formulation from a relaxed CVRP formulation. As an example of where this can cause problems, we let CPLEX solve an instance with 3 nodes which together have a demand above the capacity, the instance can be seen in Figure 1.

As we can see, the demand of node 1 and 3 equal 10, while the demand of node 2 equals 20. Let us assume the vehicle capacity is 20 and that we have 2 vehicles available. The instance of Figure 1 is a feasible and optimal solution for an iteration of the RF heuristic.

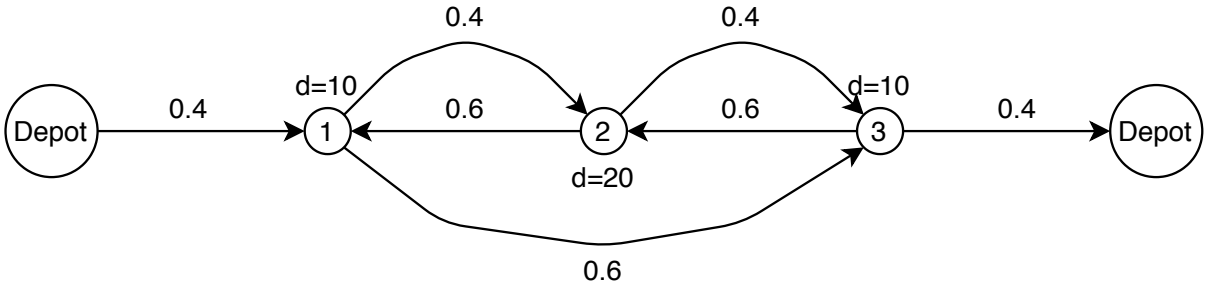


Figure 1: A possible solution to the relaxed capacitated vehicle routing problem

In this example, we have previously fixed the arcs $x_{depot,2}$ and $x_{2,depot}$ to 0. This is possible because the current solution, where node 2 is visited partially, gives us a better objective value. This means that for the rest of the iterations, it is impossible to make a route direct from the depot to node 2 or directly from node 2 to the depot. In one of the following iterations of the RF heuristic, we need to decide whether to set $x_{1,2}$ and $x_{3,2}$ to 0 or 1. We know that neither of them can be 1, because then we would have a route with a cumulative demand of 30, which exceeds the vehicle capacity. But it also can not be 0, as there would not be any way to visit node 3. Therefore, the problem is infeasible.

For an ideal execution of a RF heuristic, we want the RF subproblems to never be infeasible. Therefore, we adapt the RF heuristic for the CVRP such that it will always be

feasible. The simplest way of accomplishing this is to add two variables to each constraint which allows the solver to break constraints. If we let these variables add large penalties to the objective value if the constraints are not met, the RF heuristic might still find a feasible solution. Sadly, this increases the solution space substantially, and it does not give a guarantee that a feasible solution will be found. Therefore, we will also look at another method which changes the basis of the RF heuristic.

## 3.3 Infeasibility Fixes

As has been explained in Section 3.2, the CVRP can become infeasible if the RF heuristic is executed. As the RF heuristic is incorporated in both the VORF and the DSRF heuristic, we have to make sure that any RF subproblem is feasible. In the example given in Figure 1 there is a possibility that previously fixed variables cause infeasibilities. To counter the infeasibilities, we use two different methods. The first method is adding penalty variables to the CVRP formulation. The second method exploits the idea that only fixing variables to zero can cause infeasibilities.

### 3.3.1 Penalty Variables

To ensure that any created solution is feasible, we can change the CVRP formulation of Equations (4)-(10) to contain penalty variables. These variables allow certain infeasibilities in the CVRP formulations and captures them with the penalty variables. When these variables are nonzero they will add a penalty to the objective value. The CVRP penalty formulation is defined as:

$$\min \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} + M\left(\sum_{i=1}^{n} \sum_{k \in \{1,2,4\}} u_i^k + \sum_{i=1}^{n} v_i + u^3\right) \tag{13}$$

s.t.

$$\sum_{j=1,j\neq i}^{n+1} x_{ij} = 1 - u_i^1, \qquad i = 1,\dots,n \tag{14}$$

$$\sum_{i=0,i\neq h}^{n} x_{ih} + u_h^2 = \sum_{j=1,j\neq h}^{n+1} x_{hj} + v_h^2 \qquad h = 1,\dots,n \tag{15}$$

$$\sum_{j=1}^{n} x_{0j} \leq K + u^3 \tag{16}$$

$$u_{ij}^4 + y_j \geq y_i + d_j x_{ij} - Q(1 - x_{ij}) \qquad i,j = 0,\dots,n+1, i \neq j \tag{17}$$

$$d_i \leq y_i \leq Q, \qquad i = 0,\dots,n+1 \tag{18}$$

$$x_{ij} \in \{0,1\}, \qquad i,j = 0,\dots,n+1 \tag{19}$$

$$u_i^k, v_i \geq 0 \qquad i = 0,\dots,n+1, k \in \{1,2,4\} \tag{20}$$

Compared to the original formulation, we see that the penalty variables $u_i^k$ and $v_i$ are added to the formulation. Constraints (15) now allow vehicles routes to be cut off at a random point in the graph. This means that a route can end or start from wherever in the graph at the cost of a penalty $M$. Furthermore, we allow for nodes not to be visited at all in constraints (14), again for a cost of $M$. Extra vehicles are allowed by constraint (16). Lastly, the capacity constraint may be exceeded as by constraints (17).

### 3.3.2 Adjusted Relax-and-Fix

In the adjusted RF heuristic, we adjust the fixing step of the standard RF heuristic. In an RF heuristic subproblem we normally fix all the binary variables after solving them, in each subproblem. Now, we only fix the binary variables which return a value of 1. If the binary variable returns 0, we relax the binary variable again. In this way, no feasible routes can be cut off, and the problem should always remain feasible. When all variables have been binary once, we are left with a set of fixed and relaxed variables. If all relaxed variables equal 0, we have found a feasible solution and the RF heuristic terminates. If not, we reiterate the list of variables again, such that some binary variables are fixed to 1. If no variables are set to 1 in a complete execution of the RF heuristic, we force a variable to be fixed to one. This is done by setting all non-fixed variables as relaxed variables, while including an extra constraint and an auxiliary variable to the two-index VRP formulation.

The following constraints are added to the formulation of (4)-(10):

$$x_{ij} \geq r_{ij} \qquad \forall (i,j) \in R \tag{21}$$

$$\sum_{(i,j)\in R} r_{ij} \geq 1 \tag{22}$$

Here, $r_{ij}$ is a binary variable and $R$ is the set of relaxed variables in the current iteration of the RF heuristic. Because only a single of the $r_{ij}$ variables has to be 1, it does not add much to the complexity of the RF heuristic. Furthermore, these constraints are only appended to the formulation when no variables are fixed in a complete execution an RF heuristic, which happens maximally once per execution of the RF heuristic. The addition of Equations (21) and (22) will choose an arc $(i, j)$ to be binary, such that the RF heuristic can continue. Now, if the VRP problem is extremely tight, it can still occur that the RF heuristic will not return a feasible solution. As we never set our binary variables to 0, this can only be the case when there are not enough vehicles to execute all determined routes. For example, in the most extreme case, we could have fixed all binary variables $x_{i,depot}$ to 1 for all $i$ except one node, but this one leftover node can not be added to any existing routes, because vehicle capacity would be exceeded. In this case we would need one extra vehicle to get a feasible solution, therefore we need to keep enough vehicles at our disposal.

## 3.4 Optimizing Relax-and-Fix

As can be seen in the Section 3.1, the Relax-and-Fix heuristic needs a binary variable order as input over which the heuristic executes. To get good objective values, we want to group the variables which correlate the most in windows. In the case of a lot sizing problem, the window moves over the time index for production setup variables. This means that for a window size of five, five chronologically consecutive setup variables are kept binary, while the rest of the variables are fixed/relaxed. This has given satisfactory results, since setup decisions directly prior to periods of high demand can produce a surplus of the product for the period with high demand for low holding costs. Therefore it is a good idea to move the window along these indices. In the case of the knapsack problem, there is no natural ordering of the variables. The variables are indexed by a simple ID, which could easily have been switched in order. The grand mathematical problem would not change if index 2 and 3 were to be switched, but the execution of the

RF heuristic would change, because a different variable would be fixed during a different subproblem solve. This means that a different order will result in a different heuristic solution. Therefore, we aim to find a subproblem solving sequence which gives the best results in the RF heuristic. If there is a pattern in the solving sequence, one can use this pattern to generate fast solutions to similar knapsack problems. Finding a pattern in the solving sequence is the focus of the first approach.

Furthermore, additional parameters can be of importance for the standard Relax-and-Fix than just the solving sequence. Also, not only the window size and the increment size, which are explained in Section 3.1, are of importance. For example, the window parameter does not have to be a fixed size, this gives us a number of parameters:

1. Number of subproblems
2. Number of binary variables per subproblem
3. Subproblem solving sequence

This gives us the idea that perhaps it is better to change the Relax-and-Fix such that it can roam more freely through the subproblems. Some subproblems are computationally more complex than others, so the amount of binary variables is smaller, while the easier variables can be part of a larger window of binaries. While in the first approach the emphasis of the improvement is on the solving sequence of the RF heuristic, in this second approach the emphasis of improvement is on the subproblem size parameter.

The described methods will be performed on a specific version of the Lot Sizing Problem (LSP) and a version of the Vehicle Routing Problem (VRP). In the next section, we will elaborate on the first approach. Afterwards, the second approach will come to part.

### 3.4.1 Varying Order Relax-and-Fix

It is logical that Relax-and-Fix has led to good results for the lot sizing problem in the past, because there is a natural ordering of variables. If production only takes a single day, then the quantity to produce in the first day of the month has almost nothing to do with the production quantity at the end of the month. Because those quantities do

not correlate much, it does not matter if some variables are fixed or relaxed and only a part of the problem is solved with binaries. In the coming part we will try to take advantage of that idea, by creating a heuristic which searches for orderings of variables which generate better solutions when solved using Relax-and-Fix. Perhaps any binary mathematical problem formulation can be heuristically solved using RF, given that the RF heuristic is executed in a smartly chosen sequence of variables.

To determine new and better sequences of variables, we propose a local search algorithm, which together with the RF heuristic we call the Varying Order Relax-and-Fix heuristic (VORF):

**Algorithm 2** Local Search for Optimizing the Solving Sequence of Relax-and-Fix

```
1: procedure VARYING ORDER LOCAL SEARCH(window,increment,order,problem,maxTime)
2:     bestObjective ← ∞
3:     while total computation time < maxTime do
4:         objective ← RELAX-AND-FIX(window, increment, order, problem)
5:         if objective < bestObjective then
6:             bestObjective ← objective
7:             bestOrder ← order
8:         end if
9:         order ← CHANGEORDER(order, parameter)
10:    end while
11:    return order, objective
12: end procedure
```

As input the algorithms asks for a *window* and *increment* parameter, which have to be chosen by the user. In this approach, the subproblems have a constant size, and only differ in terms of which exact variables are binary. Further more, an initial ordering of variables is requested. A good initial ordering can decrease total computation time, but a random order should reach similar objective values eventually when using this approach. Also, we have noted *problem* as input, this can be any mathematical problem with binary variables. The algorithm starts by setting the best objective value to infinity. Each time a better order of binary variables is found, the terms *bestObjective* and *bestOrder* are

updated. The overall performance of the algorithm is solely dependant on finding better orders of variables. Therefore the ChangeOrder($\cdot$) function will be explained in detail.

The ChangeOrder($\cdot$) function changes the order of the variables. A state is defined as a variable sequence over which the RF heuristic is executed. A neighbour of a state is defined as a sequence where $k$ variables have changed place. For $k$=1, if the current order is $\{1, 2, 3, 4\}$ and a neighbour needs to be selected, it is possible that 4 is selected to move to index 1. The selection of a variable to be replaced is done randomly using a discrete uniform distribution, as well as selecting the index where it moves to. The resulting neighbour which is tested is $\{4, 1, 2, 3\}$. If a neighbour has a better objective value than the current best objective value, the *bestObjective* and *bestOrder* are updated.

At the start of the algorithm, it is relatively easy to find better orders of binary variables. Therefore, we take large steps to a better solution by changing the place of a fraction $p$ of the total binary variables, so $k = p \times |N|$, were $|N|$ is the number of nodes in the problem. If no better solutions are found for a number $r$ of iterations, the number of order mutations per iteration decreases to $k = p \times p \times |N|$, to pin down the order to a local optimum. The stopping criterium of the algorithm is set to a certain amount of time $maxTime$.

### 3.4.2 Dynamic Subproblem Relax-and-Fix

To determine the subproblem size parameters of the Relax-and-Fix heuristic, we construct a heuristic that explores the best ways to define the subproblems of the RF heuristic. Here, a subproblem is defined as a single solve of a mathematical problem, with a number of variables being fixed, binary and relaxed.

In this approach, we do not look at the solving sequence, but it is assumed that there is always some value in the initial ordering of the variables. This approach puts the emphasis on finding the best way to define each sequential subproblem. We call this approach the Dynamic Subproblem Relax-and-Fix Heuristic (DSRF heuristic).

The DSRF heuristic exploits the idea that a larger subproblem always gives a better

solution, if no overlap between subproblems is taking into account. This is easily explained when looking at the extreme cases. When the RF heuristic is executed with a single subproblem, where all Relax-and-Fix variables are binary, then we simply end up with the initial mathematical problem, of which the solver will calculate the optimal solution, given enough time. The idea is to start the algorithm with small subproblems, and merge these smaller subproblems into larger ones as long as the solver's computation time of the subproblem is relatively small. This results in an algorithm which lengthens subproblems until their computation time is too large to solve within reasonable time. When the maximum computation time is reached, the algorithm terminates and returns the best obtained result. The pseudocode is given as:

**Algorithm 3** Dynamic Subproblem Size Relax-and-Fix for minimization

1: **procedure** DYNAMIC SUBPROBLEM($order$,$join$,$maxTime$,$maxSolverTime$)
2: $window \leftarrow 1$
3: $increment \leftarrow 0$
4: $bestSolution \leftarrow$ RELAX-AND-FIX($order, window, increment$, Mathematical Problem)
5: **while** total computation time $< maxTime$ **do**
6: **for** every subproblem **do**
7: $Solution =$ SOLVE($subproblem, maxSolverTime$)
8: $runtime(i) \leftarrow$ runtime of subproblem
9: **end for**
10: $n \leftarrow join * \#subproblems$
11: **for** subproblem $i$ **do**
12: $combinedRuntime(i, j) = runtime(j) + runtime(i)$
13: $j \leftarrow i$
14: **end for**
15: **for** $n$ of the smallest combined runtimes **do**
16: subproblem $i \leftarrow$ MERGE$(i, j)$
17: **end for**
18: **end while**
19: **return** $Solution$
20: **end procedure**

The first part of the algorithm builds an initial solution for the mathematical problem using a *window* size of 1 and an *increment* size of 0. These are the smallest possible subproblems. In each subproblem, only a single variable is kept binary. In the next iteration, this variable is fixed and the next in order is kept binary.

The second part of the algorithm will improve the RF heuristic by merging adjacent subproblems together. After running the RF heuristic for every subproblem, the computation time of the solver for each subproblem is saved. For each pair of adjacent subproblems $i$ and $j$, their joint computation time is calculated and saved. Then, a fraction $join$ of all subproblems will be merged with an adjacent subproblem.

The subproblems which will be merged, are determined by picking the lowest $join \times |N|$ $(i, j)$ pairs from the joint runtimes. This means that in the first iteration, subproblems with a single binary variable will merge to a subproblem with two binaries. After all merges are complete, the RF heuristic will run again with the newly created subproblems, while maintaining their initial order. This process continues until the maximum computation time of the DSRF heuristic is exceeded, or when the maximum solver time of a single subproblem is exceeded, or when the optimal solution is reached.

The advantage of the VORF in comparison to the DSRF, is that the results possible show a pattern in the variables. This can be of great value, as this pattern can be used over and over again to solve problems of the same kind, while you only have to determine this pattern once. For example, in a Vehicle Routing Problem (VRP) it could be very benificial to have a variable order where subsequent variables indicate locations which are very close together. For a binary knapsack problem, it may be found that ordering the items according to their utility/weight ratio could be beneficial. The DSRF heuristic also has this potential, but its results are more problem specific, as it is very hard to know beforehand which variables are computationally the hardest. Though, we suspect that performing the DSRF on the resulting order of the VORF could further improve overall solutions.

# 4 Numerical Results

In this part we present the results of our methods. Both the VORF and the DSRF heuristic are executed over two mathematical programming formulations, namely the Multi-Level Capacitated Lot Sizing Problem (MLCLSP) and the Capacitated Vehicle Routing Problem (CVRP). First, the results on the MLCLSP are given in part 4.1. Afterwards, the results of the performance of our methods on the CVRP are elaborated on. For each formulation we present results on the VORF heuristic, the DSRF heuristic and the combined VORF and DSRF heuristic.

For collecting our results we used the Eclipse IDE, while programming was done with the Java language. All mathematical problems were solved using IBM ILOG CPLEX version 12.10.0. The Eclipse Version which is used is 4.13.0 with Java version 1.8.0. All code was executed using a Windows 10 64-bit OS. The hardware specifications contain 12 GB of RAM and an Intel i7-2600 CPU with 3.40 GHz computing frequency.

## 4.1 Multi-Level Capacitated Lot Sizing Problem

The MLCLSP instances are the same as used by Toledo et al. (2015) in their RF and Fix-and-Optimize heuristic. This data was gathered for the article of Akartunali and Miller (2009) and can be found at *MULTILSB* (2015). The MLCLSP with backlogging is a formulation which is useful for comparison of Relax-and-Fix methods, since multiple papers have used the RF heuristic to solve lot sizing problems. The MLCLSP formulation we use is the same a used by Toledo et al. (2015), and is given below:

| | Letter | Description |
|---|---|---|
| Sets | $J$ | Set of products |
| | $T$ | Set of periods |
| | $M$ | Set of resources |
| | $F$ | Set of product families |
| | $\delta(j)$ | Set of immediate successors of product $j$ |
| | $\Delta$ | Set of end products |
| Parameters | $a_{mj}$ | Time necessary to produce one unit of product $j$ on machine $m$ |
| | $B_{jt}$ | Upper bound for lot-size of product $j$ in period $t$ |
| | $bc_j$ | Backlogging cost of product $j$ |
| | $C_{mt}$ | Total capacity of machine $m$ in period $t$ |
| | $D_{jt}$ | Primary demand of product $j$ in period $t$ |
| | $h_j$ | Holding cost per unit of product $j$ in one period |
| | $p_{jf}$ | 1 if product $j$ belongs to family $f$ |
| | $r_{jk}$ | Quantity of product $j$ necessary to produce one unit of product $k$ |
| | $st_{mf}$ | Setup time for product family $f$ on machine $m$ |
| Variables | $x_{jt}$ | Lot-size of product $j$ in period $t$ |
| | $w_{ft}$ | Setup variable of family $f$ in period $t$ |
| | $i_{jt}$ | Stock holding quantity of product $j$ in period $t$ |
| | $b_{jt}$ | Backlogging quantity of product $j$ in period $t$ |

$$\min \sum_{j=1}^{J} \sum_{t=1}^{T} (bc_j \cdot b_{jt} + h_j \cdot i_{jt}) \tag{23}$$

s.t.

$$i_{jt-1} + b_{jt} + x_{jt} = i_{jt} + b_{jt-1} + D_{jt} \qquad \forall j \in \Delta, \forall t \in T \tag{24}$$

$$i_{jt-1} + x_{jt} = i_{jt} + \sum_{k \in \delta(j)} r_{jk} \cdot x_{kt} \qquad \forall j \in (J \setminus \Delta), \forall t \in T \tag{25}$$

$$\sum_{j=1}^{J} a_{mj} \cdot x_{jt} + \sum_{f=1}^{F} st_{mf} \cdot w_{ft} \leq C_{mt} \qquad \forall m, t \tag{26}$$

$$x_{jt} \leq wft \cdot B_{jt} \qquad \forall j, f, t | p_{jf} = 1 \tag{27}$$

$$x_{jt}, i_{jt}, b_{jt} \geq 0 \qquad \forall j, t \tag{28}$$

$$w_{ft} \in \{0, 1\} \qquad \forall f, t \tag{29}$$

As can be seen in Equation (23), the objective of the MLCLSP is to minimize backorder costs and holdings costs. For all end products, this has to be done while taking into account that inventory, demand and backorders are in balance over all periods (Equations (24)). For all non-end products, inventory and internal demand should be balanced (Equations (25)). Furthermore, no machine can exceed their capacity. The weight of a product contains a product specific parameter and a family specific parameter, which together form the total weight. The capacity constraint is denoted in Equations (26). Then, in Equations (27) we make sure that production of a product $j$ can only occur if a setup for its product family $f$ has taken place. Lastly, the domains of the variables are set in Equations (28) and (29).

The advantage of MLCLSP with backlogging over other lot sizing problem formulations is that it is always feasible because of the backlogging being allowed. If we never produce any items to fulfill any demand, all demand will be recorded as backlogging costs. Next to that, it is applicable to real life problems, since it is both multi-level and capacitated. The problem is multi-level in the sense that end products need to be assembled by certain items, which would again need to be assembled before that on their own. The capacitated part means that the machines/resources which produce the end products can only produce as much as their respective capacity.

In the coming Section we present the results of the VORF on the MLCLSP formulation.

### 4.1.1 Varying Order Relax-and-Fix

For the execution of the VORF heuristic we need an *increment* and a *window* parameter. Toledo et al. (2015) have determined that a window size of 40 an an increment size of 8 variables (80% overlap) gives satisfying results with respect to quality and time. Furthermore, we present results with a small window of 2 and an increment of 1. By reducing the window size to 2, we expect that particular orders of variables show exceptionally well, because all binary variables in any subproblem are directly adjacent to each other. If it is important that two variables are binary in the same subproblem, then it is obviously easier to spot that these belong together if they are directly adjacent to each other. Or, the other way around, it is be hard to see a particular order if two variables in the same subproblem are still 40 variables apart. If the window is larger, we anticipate that an improved order is more likely to be found, but it is expected to be harder to pinpoint why this order performs better.

The MLCLSP data from *MULTILSB* (2015) can be subdivided into 4 groups, all with different characteristics. Set 1, 3 and 4 contain data for 16 periods of demand, while set 2 contains data for 24 periods of demand. All sets contain 30 instances of demand with differing rates of variability for a total of 78 products of which 6 products are end products. The 72 products which are not end products are required for the assembly of the end products. For our results, we use only instance number 30 for each set, because this demand set has the highest variability, and therefore we expect the order of binary variables to be more important than in instances with low variability. If our methods do not return a logical order to the instances of high variability, one can assume they do not return a logical order to instances with low demand variability. But if a logical order is found, it can be of value to test instances with lower variability. Furthermore, the sets 1, 3 and 4 differ from each other in terms of resource utilization and backorder costs. The resource utilization factor determines how much of the capacity of the machines has to be used for it to fulfill all demand. For all demand sets the resource utilization factor is larger than 1. Therefore in all solutions backlogging will occur.

At the start of the heuristic, the ordering of the variables is randomized. From there on the order changes with a single variable at a time to obtain betters results. The

objective value of the initial ordering and of the obtained ordering is reported in Table 2.

Table 2: Improvement of the VORF heuristic over 10 hours runtime

| Demand set | Window | Increment | Initial Objective | Eventual Objective | Improvement |
|---|---|---|---|---|---|
| 1 | 2 | 1 | 77,456 | 74,470 | 3.86% |
| 1 | 40 | 8 | 74,001 | 70,863 | 4.24% |
| 2 | 2 | 1 | 93.566 | 81,847 | 12.52% |
| 2 | 40 | 8 | 77,828 | 72,458 | 6.90% |
| 3 | 2 | 1 | 611,689 | 586,058 | 4.19% |
| 3 | 40 | 8 | 603,880 | 463,391 | 23.25% |
| 4 | 2 | 1 | 478,073 | 293,588 | 38.59% |
| 4 | 40 | 8 | 313,664 | 271,835 | 13.34% |

As can be seen in Table 2, choosing a smart ordering of variables can improve the solution quality by up to 38.59% for some instances. This suggest that the order in which the RF heuristic is executed is very important for the results. The improvement percentages are very different for each instance. This is due to the fact that the random order with which we start can already be a quite well performing order, but in other cases, as can be seen, there is much more improvement to be made.

In Table 3 we present objective values for 4 instances of the MLCLSP, both for multiple parameters. We compare them to a lower bound provided by *MULTILSB* (2015) and to a solution of a periodwise RF heuristic. Periodwise means that variables are ordered in such a way that setup variables for period 1 are fixed before period 2 and so on.

Table 3: Objective values returned by running the VORF algorithm for 10 hours, compared to lower bounds and periodwise RF.

| Demand set | Window | Increment | Objective Value | Lower Bound | Periodwise RF |
|---|---|---|---|---|---|
| SET1/30 | 2 | 1 | 74,470 | 68,836 | 70,863 |
| SET1/30 | 40 | 8 | 70,863 | 68,836 | 70,863 |
| SET2/30 | 2 | 1 | 81,847 | 67,870 | 71,407 |
| SET2/30 | 40 | 8 | 72,458 | 67,870 | 71,407 |
| SET3/30 | 2 | 1 | 586,058 | 235,921 | 399,609 |
| SET3/30 | 40 | 8 | 463,391 | 235,921 | 399,609 |
| SET4/30 | 2 | 1 | 293,588 | 235,921 | 263,064 |
| SET4/30 | 40 | 8 | 271,835 | 235,921 | 263,064 |

As can be seen in Table 3, the objective values returned by the VORF heuristic come very close to the values of a single iteration of a periodwise RF heuristic. Therefore we know that the VORF heuristic found orders that were almost as good as the periodwise order.

As we expect the variable order to converge to a periodwise order, we define a descriptive statistic which can confirm this. The descriptive statistic is a deviation statistic. It counts the total deviation of a variable to its expected location in the variable order according to the periodwise ordering. For example, if we have an expected order of $\{1, 2, 3, 4\}$ and the resulting order is $\{1, 4, 3, 2\}$, then the deviation statistic equals 4, as the 4 and the 2 have switched places, while they were originally 2 spaces apart. As the data set has 11 families and 16 periods, the expected order is constructed as follows:

$$\{1_1, 1_2, \ldots, 1_{11}, 2_1, \ldots, 2_{11}, \quad \ldots \quad , 16_1, 16_2, \ldots, 16_{11}\} \tag{30}$$

We acknowledge that this statistic does not give a complete view of any logical order in the variable sequence, so the resulting orders for SET3 have been provided in Appendix A and other resulting orders can be provided by the author. The deviation statistic of the variable sequence we start with and the deviation statistic of the variable order we end up with are presented in Table 4.

Table 4: Deviation from a periodwise ordering

| Demand set | Window | Increment | Initial Deviation | Eventual Deviation |
|---|---|---|---|---|
| SET1 | 2 | 1 | 928 | 976 |
| SET1 | 40 | 8 | 928 | 976 |
| SET2 | 2 | 1 | 2224 | 2104 |
| SET2 | 40 | 8 | 2224 | 2102 |
| SET3 | 2 | 1 | 928 | 958 |
| SET3 | 40 | 8 | 928 | 936 |
| SET4 | 2 | 1 | 928 | 996 |
| SET4 | 40 | 8 | 928 | 970 |

As we can see, the VORF heuristic does not converge towards a periodwise order in any way, shape or form. For all datasets but SET2, we see that the resulting variable order is even further away from a periodwise variable order than when the heuristic started. Since we already know that the objective value of the VORF heuristic does tend to the objective value of the periodwise RF heuristic, we expect that there is some common characteristic to the random orders found by the VORF heuristic and the periodwise order which determines the comparable objective value, but it is unclear what this common characteristic is.

### 4.1.2 Dynamic Subproblem Relax-and-Fix

The DSRF heuristic constructs larger subproblems out of the smallest possible subproblems in the RF heuristic. By only adding variables to subproblems which have a short computation time, it creates a subproblem composition of the RF heuristic which produces good results in a relatively short amount of time. The DSRF heuristic does not change the order of the variables, as the VORF does. Therefore, DSRF gives the best results when a initial ordering is used which already works well with the standard RF heuristic. That is why we expect the DSRF to develop satisfactory results to the ML-CLSP.

To execute the DSRF heuristic, we need to determine the *join* input parameter. By

trial and error, we conclude that a *join* parameter of 0.2 gives satisfactory results. This means that each iteration, 20% of the subproblems are merged with an adjacent subproblem. If we merge too many subproblems every iteration, we might merge subproblems which are already fairly hard, which we do not want. Merging too few subproblems at once will results in an increase of computation time. If your objective is to find the best distribution of variables over the subproblems, the *join* parameter can be set smaller, as computation time does not matter. However, for practical application we do not want a computational intensive algorithm, therefore it is set to 0.2. Furthermore, we always merge at least one subproblem and no overlap of subproblems is considered when performing the DSRF.

The data sets on which we execute the DSRF heuristic are again demand instance 30 of all 4 different datasets, since it is the dataset with the highest variability in demand. We expect the high variability to show patterns relatively well. The objective values of the DSRF heuristic are presented in Table 5, together with the LP relaxation bound found by the CPLEX solving software. Optimal values are indicated by an asterisk (*).

Table 5: Initial and eventual objective values of the DSRF heuristic on MLCLSP instances after running for 5 minutes

| Dataset | Periods | Initial Objective | Eventual Objective | LP bound |
|---|---|---|---|---|
| SET1/30 | 16 | 71,766 | 70,864* | 68,836 |
| SET2/30 | 24 | 72,314 | 71,408* | 67,870 |
| SET3/30 | 16 | 597,274 | 404,108 | 235,921 |
| SET4/30 | 16 | 325,151 | 262,381 | 235.921 |

For SET1 and SET2 the table reports objective values of the DSRF heuristic which equal the optimal solution. This is due to the fact that the instances are solvable within 5 minutes when all variables are binary at once. For SET3 this is not the case. After 5 minutes the RF heuristic of SET3 and SET4 were divided into 2 subproblems, which constructed a solution which was logically superior over the initial subproblem decomposition.

To get an insight at the decomposition of the subproblems we pause the DSRF heuristic when a decomposition of 10 subproblems is made. The variable sequence we use is the periodwise sequence, such that all variables in period 1 are fixed at first, followed by 2, 3, 4 etc. For the 4 datasets the subproblem sizes of the 10 subproblems are reported in Table 6.

Table 6: Window sizes of the DSRF heuristic when it has built up to 10 subproblems

| Dataset | Periods | Size of subproblem $\{1, 2, \ldots, 10\}$ |
|---|---|---|
| SET1/30 | 16 | $\{5, 22, 13, 12, 13, 1, 16, 32, 23, 39\}$ |
| SET2/30 | 24 | $\{29, 9, 33, 32, 25, 12, 33, 22, 51, 18\}$ |
| SET3/30 | 16 | $\{20, 14, 16, 12, 6, 20, 14, 16, 18, 38\}$ |
| SET4/30 | 16 | $\{5, 22, 22, 16, 15, 11, 13, 11, 20, 41\}$ |

As we can see, the subproblem sizes are fairly evenly distributed. Only the last or next to last subproblems seem to be of larger binary size. This can be explained by the fact that, for the last subproblem, all variables outside of the window are fixed. While, for the first subproblem, all variables outside the window are relaxed and thus their value needs to be determined. In the latter case, the subproblem is practically larger, because more variables need to be determined.

## 4.2 Capacitated Vehicle Routing Problem

The CVRP instances we used are from the freely available VRP library of the University of Bologna (*VRPLIB*, n.d.). The data in their VRP library was provided by multiple authors, but we use only those of Christofides et al. (1981), as they are relatively small, but large enough to draw conclusions for our methods. We have chosen to use the smaller instances for our results, since they are less computationally complex and since we already know that the RF heuristic can find feasible solutions for larger instances substantially faster than the solver. Furthermore, it is easier to notice any orders in the variables, since the data set is smaller and easier to look at. The data sets contain at most 26 nodes, all with differing demands and differing numbers of vehicles. For each instance the vehicles capacity differs, but within each instance all vehicles have the

same capacity. There is no previous literature on performing any form of a Relax-and-Fix heuristic on the CVRP. The lack thereof is probably because there is no natural ordering in the variables of the CVRP and because of the infeasibility issues with the CVRP. Because of this, it is an interesting mathematical problem to perform our VORF heuristic on. If the VORF heuristic manages to find an ordering which performs well in comparison to other methods, and as such produces objective values relatively close to known lower bounds, then the technique could possibly be expanded to any problem without a natural ordering. The formulation we implement is the same as the two-index vehicle flow formulation defined by Munari et al. (2017). We have already noted the formulation in Section 3.2, Equations (4)-(10).

### 4.2.1 Varying Order Relax-and-Fix

As has been discussed in Section 3.2, the standard VORF heuristic of Algorithm 2 does not generally result in feasible solutions for the CVRP. Therefore we introduce two methods to counteract the infeasibilities, namely the penalty formulation of Section 3.3.1 and the Adjusted RF of Section 3.3.2. The penalty formulation relies completely on the inputted variable sequence for returning feasible solutions. The adjusted RF ignores a part of the RF heuristic, as it only fixes variables to 1, to build a (near) feasible solution faster. For our penalty formulation, we set the $M$ parameter to 10,000, such that infeasibilities are easily observable in the objective values. In the VORF heuristic, we set the initial $p$ to 0.5, which determines the number of replacements that are made when a neighbour needs to be built, so we replace half of the binary variables in the order to obtain a neighbour. The $r$ parameter is set to 50. This determines that the $p$ fraction is updated if no objective value improvements are found for 50 iterations. The $p$ fraction is then updated to $p \times p$.

The objective values of both our methods are presented in Table 7. In this table, $N$ and $V$ represent the number of nodes and available vehicles in the dataset, respectively. $C$ denotes the vehicle capacity and $W$ and $I$ are our window and increment parameters.

Table 7: Objective values of the solver and the Penalty VORF and Adjusted VORF heuristics after running for 2 hours

| Dataset | $N$ | $V$ | $C$ | $W$ | $I$ | Optimal | Penalty VORF | Adjusted VORF |
|---|---|---|---|---|---|---|---|---|
| E016-03m | 16 | 3 | 90 | 2 | 1 | 278.73 | 339.47 | 299.68 |
| E016-03m | 16 | 3 | 90 | 40 | 8 | 278.73 | 321.12 | 296.70 |
| E026-08m | 26 | 8 | 48 | 2 | 1 | 606.3 | 150,445.86 | 740.38 |
| E026-08m | 26 | 8 | 48 | 40 | 8 | 606.3 | 60,610.57 | 708.41 |

As we can see, the penalty VORF heuristic attains feasible results for only the smallest instances. For instance E026-08m and a $W$ of 2 and $I$ of 1, it reports an objective value of over 150,000. This indicates that at most 150 constraints are not satisfied. It is possible that the penalty VORF heuristic will find feasible solution eventually, as it did for the smaller instances, but we can not guarantee it.

The Adjusted VORF outperforms the penalty VORF in all of our instances. This is because we always attain a near-feasible result to the CVRP using the adjusted VORF. It can only happen that more vehicles are deployed than there are available. A disadvantage of the adjusted RF is that it takes a significantly longer time to create a solution in comparison to the penalty VORF. For the penalty RF of problem E026-08m, using a window size of 2 and an increment of 1, a single iteration took on average 11.21 seconds. The adjusted RF however took on average 4.8 minutes to complete an iteration. Many more subproblems need to be solved in the adjusted RF, because only the nonzero variables are fixed, so we iterate all subproblems multiple times. Next to this, the order of the variables has less value, because the order of all variables which are 0 does not matter, since they are kept relaxed in the next subproblem. In the worst case, we can have that no binary variables are ever set to 1, and that we have to force a variable to be fixed in each iteration. The instance of set E026-08m with $W = 2$ and $I = 1$ never improved its solution in the 2 hours that we ran the heuristic. This is partly because not many iterations were tried, only 25 in 2 hours. But still, since the starting order is just a random order, it can be expected that any of the 24 next orders does perform better than the random starting order.

Of course, the actual solutions found by both the Penalty RF and the Adjusted RF can always be infeasible. For the Penalty RF, we can not even guarantee that there is a variable order which results in a feasible solution, as it is very likely that fixing variables to 0 or 1 while keeping others relaxed results in some relaxed variables not being able to take on a binary value later on in the heuristic. For the Adjusted RF, we need to have enough vehicles such that it is always possible to go directly from a depot to a node and vice versa, otherwise we can not guarantee a feasible solution.

For the Adjusted RF heuristic, it sometimes happens that no variable is set to 1 for a complete RF iteration. In that case, a variable is forced to be 1 using the Equations (21) and (22). This situation actually happens quite a lot. For example, for data set E016-03m it happens on average about 9 times for each RF iteration. Furthermore, each of these forced fixes have a whole RF iteration prior to it where no fixed variables are added to the solution. This means that, next to the computation time of the adjusted VRP formulation itself, the solver also has accumulated the computation time of the multiple normal RF iterations, without fixing extra variables. This makes the adjusted RF heuristic computationally much harder than the normal RF heuristic. Additionally, the forcing of variables can cause infeasibilities in terms of the amount of routes that are made. Many times, more vehicles are used than there are available. In Table 8 we report the number of vehicles that are used by the solutions of the adjusted RF heuristic can be found.

Table 8: Number of vehicles used in the solution of the adjusted RF heuristic

| Dataset | $N$ | $V$ | $C$ | $W$ | $I$ | Vehicles Used |
|---|---|---|---|---|---|---|
| E016-03m | 16 | 3 | 90 | 2 | 1 | 4 |
| E016-03m | 16 | 3 | 90 | 40 | 8 | 4 |
| E026-08m | 26 | 8 | 48 | 2 | 1 | 12 |
| E026-08m | 26 | 8 | 48 | 40 | 8 | 11 |

As we see, in all instances the number of employed vehicles exceeds the amount of available vehicles. This is easily explained, as the adjusted RF heuristic can solve these

problems in the relaxed CVRP using partial visits. As has been explained before, using partial routes can satisfy the vehicle constraint, but when these partial routes are forced to be binary, the vehicle constraint can no longer be satisfied. Thus, we need an extra vehicle. Besides, the datasets are extremely tight in terms of their capacity constraints. The total demand of all nodes is only a little bit lower than the total capacity of all vehicles together. For E016-03m the total demand equals 258 and the total capacity of the 3 vehicles equals 270. For E026-08m, demand equals 367 and the total capacity of the 8 vehicles is 384. For a less tight formulation, the adjusted RF heuristic does find feasible solutions in terms of vehicle availability.

Table 9 presents the resulting variable order deviation statistics of the VORF heuristic for both the penalty RF and the adjusted RF. Again we use the deviation statistic to spot movements in the variable orders, but as this statistic can hide information, the complete initial and resulting variable sequences of data set E016-03m are also given in Appendix B. For the CVRP we create a distance wise deviation statistic, as we expect the VORF heuristic to give the best results when longer arcs are handled first, as they produce the largest amount of cost. The deviation statistic is this time defined as the absolute deviation in distance from the expected order. We denote the distance of a variable in $m$ metres. For example, if the expected order is $\{9.5m, 6.5m, 5m, 4m, 3.5m\}$ and the resulting order is $\{3.5m, 6.5m, 5m, 4m, 9.5m\}$, then the deviation statistic equals 12, as 9.5 and 3.5 are both 6 metres away from their expected distance.

Table 9: Deviation statistic of the initial variable ordering compared to the resulting order of the Penalty RF and the Adjusted RF

| Dataset | $N$ | $V$ | $C$ | $W$ | $I$ | Initial | Penalty RF | Adjusted RF |
|---|---|---|---|---|---|---|---|---|
| E016-03m | 16 | 3 | 90 | 2 | 1 | 3,501 | 3,273 | 3,253 |
| E016-03m | 16 | 3 | 90 | 40 | 8 | 3,501 | 3,261 | 3,301 |
| E026-08m | 26 | 8 | 48 | 2 | 1 | 12,462 | 11,673 | 11,689 |
| E026-08m | 26 | 8 | 48 | 40 | 8 | 12,462 | 11,622 | 11,696 |

As we can see, the deviation statistic decreases in comparison to the initial deviation statistic. This means that the resulting variable orders start moving towards a variable

sequence were long distance arcs are handled before short distance arcs. This indicates that an initial variable order where variables with the highest cost are handled first, can result in better objective values than random orders. Though, the change in deviation statistic seems to be very small.

### 4.2.2 Dynamic Subproblem Relax-and-Fix

The DSRF heuristic performs best if a known and well performing variable order is put in over which the RF heuristic can execute. As we have not found a well performing variable order for the CVRP formulation, the value of the DSRF heuristic is not as great. Furthermore, the DSRF heuristic can not be executed using the standard RF heuristic. Again, we need to make use of the adjusted RF heuristic as reported in Section 3.3.2. This results in a completely different DSRF heuristic, which we shortly explain.

Normally, after each performance of the RF heuristic, the computation times of each subproblem of the RF heuristic are evaluated. But, since the RF heuristic can not be executed normally, not all variables in a subproblem are fixed once they have been binary. This means that it can happen that a part of the binary variables in a subproblem are fixed, and a part is relaxed. It can be that a subsequent RF iteration is necessary because a feasible solution has not yet been found. In that case the computation time of the same subproblem as in the previous iteration, which has not changed in size, can decrease substantially because some of its variables are fixed. Thus, we can again add extra variables to this subproblem, which improve solution quality.

Sadly we do not know a variable sequence which will guarantee satisfactory results. Still, we can use the descending distance ordering, as found in Section 4.2.1, to obtain varying subproblems compositions of this order. For the execution of the DSRF heuristic we use some larger problem instances than for the VORF, as the CPLEX solver can optimally solve the instances we used for the VORF heuristic fairly fast. As for the MLCLSP, the $p$ fraction is set to 0.2. We report the objective values we found in Table 10. Here, the initial objective value found represent the solution of the DSRF where all subproblems contain a single binary variable. The resulting objective contains the solutions of the DSRF after running for 5 minutes, or the optimal value, indicated by an asterisk, if this

was attained.

Table 10: Objective values of the DSRF heuristic on the CVRP using the descending distance variable order, maximum runtime of 5 minutes.

| Dataset | $N$ | $V$ | $C$ | Initial Objective | Resulting Objective | Optimal |
|---|---|---|---|---|---|---|
| E016-03m | 16 | 3 | 90 | 321.92 | 278.73* | 278.73 |
| E026-08m | 26 | 8 | 48 | 716.05 | 716.05 | 606.3 |
| E041-14h | 41 | 14 | 60 | 1026.46 | 1026.46 | 861.79 |

As we can see, the DSRF heuristic results in good solutions for smaller instances. But for E026-08m, only 7 solution where created during 5 minutes and it was the same each time. This is because the heuristic starts with subproblems of binary size 1. Therefore, very many subproblems need to be solved. This takes up a lot of time, although there are not many binary variables to take into account. The DSRF heuristic also reiterates all subproblems if no feasible solution has been found, which takes extra computation time. This is due to the fact that no variables are fixed to 1 for many iterations. Every time this happens, we force a variable to be 1 using auxiliary binary variables. These binary variables also add a significant amount to the computation time for larger problems, since the amount of variables increases exponentially in the number of nodes. In terms of computation time, it is very important that binary variables get fixed and are not made relaxed, but sadly this is not possible for the CVRP. Therefore we did not continue with the DSRF heuristic for the CVRP, since it is too computationally intensive.

# 5 Conclusion

In the world of operations research computational challenges are common practice. That is why a sound choice of parameters for any method is of paramount importance. In this paper, we introduced methods to improve the Relax-and-Fix heuristic. We first defined the parameters of the Relax-and-Fix heuristic in the broadest sense of the word, such that subproblems could be dynamic and variable sequences are not fixed. Then, we used metaheuristics to improve these parameters.

The VORF heuristic aimed to find variable orders for the RF heuristic which result in good objective values. For the MLCLSP formulation, good objective values were found, but the variable order which resulted in these objective values did not follow any order whatsoever. Therefore we conclude that the periodwise variable order is still the best performing variable order to the MLCLSP. Furthermore, the DSRF heuristic can be used for determining the subproblem sizes. Many of our instances were solvable while keeping all variables binary. But for larger problems, this was not the case and the DSRF heuristic could improve the eventual objective value by increasing subproblem sizes until the solver finds them too hard to solve. For this, one only needs to input the maximum amount of time the solver may be solving a subproblem. For the DSRF heuristic, it is important that a mathematical formulation will always result in feasible solutions when variables are no longer relaxed. If this is not the case, the heuristic can become computationally intensive.

Additionally, we broadened the use of the RF heuristic to more than just Lot Sizing Problems. For the RF heuristic to be executed on other problem formulations, we needed to take care of any infeasibilities that could happen. The CVRP formulation is such a problem which encounters infeasibilities when using the RF heuristic. These infeasibilities were solved in our penalty method by adding variables to the formulation, which check if the original constraints are satisfied or not, and if not, these variables add a penalty to the objective value. In the second method we adjusted the RF heuristic such that it only sets decision variables to 1. The first method had a very hard time finding feasible solutions, but did manage to do so for our smallest instances. This method is also the most broadly applicable, as for any problem formulation it is possible to add penalty variables to the constraints to create a formulation which is always feasible. The second

method could always find a near-feasible solution in a reasonable amount of time, but the value of the variable order was decreased due to the fact that we only fix variables to 1. This meant that the VORF heuristic was not able to find better variable sequences easily. Furthermore, extra binary variables were added in the adjusted RF heuristic. The effect of these variables is unclear and has to be further researched to check if the method performs well on of broader scale of VRP's.

For the MLCLSP our methods could not find a better variable ordering than the periodwise ordering which is common in practice. For the CVRP, the results hinted towards a descending distance variable order, but further research needs to be done to confirm this.

# References


Akartunali, K., & Miller, A. J. (2009). A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*. doi: 10.1016/j.ejor.2007.11.033

Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*. doi: 10.1007/BF01589353

De Oliveira, L., De Souza, C. C., & Yunes, T. (2014). Improved bounds for the traveling umpire problem: A stronger formulation and a relax-and-fix heuristic. *European Journal of Operational Research*. doi: 10.1016/j.ejor.2013.12.019

Dillenberger, C., Escudero, L. F., Wollensak, A., & Zhang, W. (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*. doi: 10.1016/0377-2217(94)90074-4

Gonźalez, P. H., Simonetti, L. G., De Jesus Martinhon, C. A., Santos, E., & Michelon, P. Y. P. (2014). An improved relax-and-fix algorithm for the fixed charge network design problem with user-optimal flow. In *Icores 2014 - proceedings of the 3rd international conference on operations research and enterprise systems.*

Helber, S., & Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*. doi: 10.1016/j.ijpe.2009.08.022

Mohammadi, M., Fatemi Ghomi, S. M., Karimi, B., & Torabi, S. A. (2010). Rolling-horizon and fix-and-relax heuristics for the multi-product multi-level capacitated lotsizing problem with sequence-dependent setups. *Journal of Intelligent Manufacturing*. doi: 10.1007/s10845-008-0207-0

*MULTILSB.* (2015). doi: 10.15129/252b7827-b62b-4af4-8869-64b12b1c69a1

Munari, P., Dollevoet, T., & Spliet, R. (2017). *A generalized formulation for vehicle routing problems.* Retrieved from `http://arxiv.org/abs/1606.01935`

Noor-E-Alam, M., & Doucette, J. (2012). Relax-and-fix decomposition technique for solving large scale grid-based location problems. *Computers and Industrial Engineering*. doi: 10.1016/j.cie.2012.07.006

Pedroso, J. P., & Kubo, M. (2005). Hybrid tabu search for lot sizing problems. In *Lecture*

*notes in computer science.* doi: 10.1007/11546245_7

Ribeiro, C. C., Santos, T. d. A., & de Souza, C. C. (2019). Multicast routing under quality of service constraints for vehicular ad hoc networks: mathematical formulation and a relax-and-fix heuristic. *International Transactions in Operational Research*. doi: 10.1111/itor.12605

Roshani, A., Giglio, D., & Paolucci, M. (2017, 7). A relax-and-fix heuristic approach for the capacitated dynamic lot sizing problem in integrated manufacturing/remanufacturing systems. *IFAC-PapersOnLine*, *50*(1), 9008–9013. doi: 10.1016/j.ifacol.2017.08.1580

Soler, W. A., Santos, M. O., & Akartunalı, K. (2019). MIP approaches for a lot sizing and scheduling problem on multiple production lines with scarce resources, temporary workstations, and perishable products. *Journal of the Operational Research Society*. doi: 10.1080/01605682.2019.1640588

Stadtler, H. (2003). Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*. doi: 10.1287/opre.51.3.487.14949

Toledo, C., da Silva Arantes, M., Hossomi, M., França, P., & Akartunalı, K. (2015, 10). A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. *Journal of Heuristics*, *21*(5), 687–717. doi: 10.1007/s10732-015-9295-0

*VRPLIB.* (n.d.). Retrieved from `http://or.dei.unibo.it/library`

Wolsey, L. A. (1998). *Integer Programming.* Wiley.

# A MLCLSP SET3 Variable Orders

**Initial order (F: Family, T, Time period):**

[F:16 T:9] [F:7 T:12] [F:4 T:13] [F:5 T:2] [F:14 T:6] [F:16 T:13] [F:4 T:14] [F:7 T:6] [F:3 T:11] [F:7 T:10] [F:3 T:10] [F:6 T:5] [F:16 T:6] [F:12 T:8] [F:14 T:11] [F:15 T:6] [F:11 T:3] [F:6 T:14] [F:16 T:8] [F:6 T:10] [F:3 T:3] [F:13 T:1] [F:15 T:7] [F:12 T:9] [F:6 T:6] [F:7 T:14] [F:15 T:1] [F:4 T:4] [F:6 T:1] [F:5 T:13] [F:15 T:8] [F:16 T:16] [F:15 T:15] [F:13 T:10] [F:13 T:2] [F:4 T:7] [F:13 T:16] [F:6 T:3] [F:12 T:3] [F:11 T:14] [F:14 T:4] [F:3 T:15] [F:3 T:1] [F:15 T:3] [F:14 T:1] [F:11 T:7] [F:11 T:12] [F:3 T:6] [F:6 T:9] [F:4 T:10] [F:16 T:4] [F:3 T:5] [F:16 T:7] [F:11 T:6] [F:3 T:7] [F:12 T:1] [F:14 T:15] [F:11 T:15] [F:7 T:7] [F:7 T:8] [F:14 T:9] [F:4 T:15] [F:12 T:13] [F:15 T:5] [F:5 T:5] [F:12 T:7] [F:15 T:16] [F:16 T:1] [F:7 T:15] [F:13 T:8] [F:3 T:14] [F:15 T:11] [F:13 T:13] [F:14 T:3] [F:12 T:14] [F:11 T:5] [F:4 T:9] [F:13 T:6] [F:15 T:12] [F:7 T:1] [F:16 T:10] [F:3 T:12] [F:12 T:16] [F:11 T:10] [F:4 T:6] [F:11 T:8] [F:4 T:11] [F:12 T:15] [F:5 T:14] [F:4 T:12] [F:5 T:6] [F:7 T:13] [F:5 T:1] [F:14 T:5] [F:12 T:4] [F:13 T:11] [F:14 T:2] [F:15 T:10] [F:12 T:12] [F:6 T:16] [F:14 T:14] [F:5 T:3] [F:14 T:16] [F:15 T:9] [F:14 T:12] [F:3 T:8] [F:15 T:13] [F:13 T:12] [F:7 T:16] [F:14 T:8] [F:11 T:11] [F:5 T:12] [F:7 T:5] [F:11 T:1] [F:4 T:8] [F:16 T:14] [F:5 T:4] [F:16 T:11] [F:15 T:14] [F:5 T:15] [F:5 T:11] [F:5 T:7] [F:5 T:8] [F:15 T:4] [F:7 T:9] [F:13 T:5] [F:4 T:2] [F:11 T:13] [F:6 T:11] [F:11 T:16] [F:16 T:15] [F:11 T:4] [F:16 T:2] [F:6 T:13] [F:14 T:7] [F:3 T:2] [F:6 T:12] [F:14 T:10] [F:13 T:7] [F:14 T:13] [F:5 T:9] [F:16 T:5] [F:12 T:11] [F:6 T:15] [F:15 T:2] [F:12 T:5] [F:13 T:3] [F:3 T:4] [F:13 T:9] [F:4 T:3] [F:3 T:13] [F:12 T:10] [F:16 T:3] [F:13 T:4] [F:12 T:2] [F:7 T:11] [F:4 T:5] [F:5 T:10] [F:6 T:4] [F:4 T:16] [F:6 T:2] [F:7 T:3] [F:6 T:8] [F:3 T:16] [F:16 T:12] [F:13 T:15] [F:7 T:4] [F:5 T:16] [F:12 T:6] [F:4 T:1] [F:11 T:9] [F:3 T:9] [F:13 T:14] [F:6 T:7] [F:7 T:2] [F:11 T:2]

**Resulting order (Window 2, Increment 1) :**

[F:12 T:3] [F:13 T:14] [F:14 T:6] [F:4 T:1] [F:3 T:14] [F:6 T:16] [F:14 T:15] [F:15 T:1] [F:6 T:12] [F:12 T:15] [F:12 T:8] [F:12 T:4] [F:5 T:16] [F:7 T:2] [F:14 T:7] [F:11 T:4] [F:13 T:7] [F:7 T:1] [F:4 T:12] [F:15 T:3] [F:11 T:8] [F:12 T:7] [F:4 T:7] [F:3 T:8] [F:7 T:12] [F:11 T:15] [F:6 T:11] [F:15 T:4] [F:3 T:10] [F:5 T:3] [F:11 T:7] [F:6 T:14] [F:5 T:8] [F:16 T:4] [F:12 T:11] [F:4 T:3] [F:4 T:10] [F:4 T:13] [F:15 T:5] [F:5 T:10] [F:15 T:11] [F:4 T:16] [F:3 T:9] [F:7 T:8] [F:13 T:15] [F:6 T:8] [F:11 T:12] [F:12 T:5] [F:5 T:12] [F:5 T:13] [F:11 T:14] [F:7 T:13] [F:6 T:13] [F:7 T:3] [F:11 T:2] [F:12 T:1]

[F:15 T:12] [F:14 T:11] [F:6 T:6] [F:5 T:15] [F:7 T:9] [F:3 T:11] [F:3 T:7] [F:15 T:15] [F:11 T:5] [F:14 T:16] [F:3 T:1] [F:13 T:8] [F:16 T:16] [F:14 T:12] [F:15 T:6] [F:6 T:2] [F:11 T:10] [F:6 T:15] [F:14 T:3] [F:3 T:16] [F:3 T:5] [F:12 T:14] [F:13 T:10] [F:16 T:12] [F:12 T:9] [F:6 T:9] [F:11 T:3] [F:16 T:1] [F:15 T:2] [F:13 T:11] [F:16 T:3] [F:14 T:9] [F:3 T:3] [F:6 T:5] [F:13 T:6] [F:7 T:7] [F:4 T:4] [F:12 T:13] [F:13 T:16] [F:16 T:9] [F:15 T:8] [F:4 T:9] [F:16 T:14] [F:12 T:2] [F:14 T:2] [F:7 T:14] [F:14 T:14] [F:14 T:1] [F:7 T:5] [F:16 T:10] [F:11 T:6] [F:13 T:4] [F:13 T:12] [F:6 T:1] [F:3 T:6] [F:7 T:15] [F:15 T:10] [F:5 T:4] [F:13 T:9] [F:15 T:9] [F:6 T:10] [F:16 T:13] [F:15 T:13] [F:4 T:15] [F:13 T:1] [F:7 T:16] [F:5 T:11] [F:11 T:13] [F:6 T:4] [F:3 T:13] [F:3 T:12] [F:14 T:10] [F:13 T:2] [F:16 T:11] [F:5 T:7] [F:4 T:6] [F:14 T:8] [F:7 T:11] [F:15 T:16] [F:13 T:13] [F:12 T:12] [F:7 T:10] [F:14 T:13] [F:11 T:11] [F:6 T:7] [F:13 T:5] [F:15 T:14] [F:4 T:5] [F:6 T:3] [F:4 T:14] [F:5 T:2] [F:11 T:16] [F:4 T:8] [F:14 T:5] [F:12 T:10] [F:5 T:1] [F:5 T:9] [F:4 T:2] [F:13 T:3] [F:5 T:14] [F:12 T:6] [F:12 T:16] [F:5 T:5] [F:16 T:7] [F:3 T:15] [F:4 T:11] [F:11 T:1] [F:16 T:6] [F:14 T:4] [F:16 T:5] [F:7 T:4] [F:16 T:2] [F:5 T:6] [F:3 T:4] [F:16 T:8] [F:3 T:2] [F:11 T:9] [F:15 T:7] [F:7 T:6] [F:16 T:15]

**Resulting order (Window 40, Increment 8) :**

[F:12 T:3] [F:14 T:6] [F:3 T:14] [F:6 T:16] [F:14 T:15] [F:15 T:1] [F:6 T:12] [F:12 T:15] [F:12 T:8] [F:12 T:4] [F:5 T:16] [F:7 T:2] [F:14 T:7] [F:11 T:4] [F:15 T:4] [F:4 T:12] [F:13 T:7] [F:7 T:1] [F:15 T:3] [F:11 T:8] [F:12 T:7] [F:4 T:7] [F:7 T:12] [F:11 T:15] [F:6 T:11] [F:3 T:10] [F:5 T:1] [F:5 T:3] [F:11 T:7] [F:16 T:16] [F:16 T:4] [F:11 T:3] [F:12 T:11] [F:4 T:10] [F:4 T:13] [F:15 T:5] [F:5 T:10] [F:3 T:1] [F:4 T:16] [F:3 T:9] [F:7 T:8] [F:13 T:15] [F:6 T:8] [F:7 T:4] [F:12 T:5] [F:5 T:15] [F:5 T:12] [F:5 T:13] [F:7 T:13] [F:6 T:13] [F:7 T:3] [F:16 T:3] [F:7 T:7] [F:11 T:2] [F:15 T:12] [F:15 T:11] [F:12 T:1] [F:3 T:7] [F:14 T:11] [F:6 T:6] [F:16 T:13] [F:7 T:9] [F:16 T:15] [F:3 T:11] [F:3 T:5] [F:15 T:15] [F:11 T:5] [F:14 T:13] [F:14 T:16] [F:16 T:1] [F:13 T:8] [F:14 T:12] [F:4 T:15] [F:15 T:6] [F:6 T:2] [F:11 T:10] [F:6 T:15] [F:14 T:3] [F:3 T:4] [F:13 T:14] [F:12 T:14] [F:13 T:10] [F:16 T:12] [F:12 T:9] [F:6 T:9] [F:12 T:16] [F:15 T:2] [F:3 T:8] [F:13 T:11] [F:14 T:9] [F:7 T:5] [F:3 T:3] [F:6 T:5] [F:5 T:8] [F:13 T:6] [F:4 T:4] [F:12 T:13] [F:13 T:16] [F:16 T:9] [F:15 T:8] [F:13 T:1] [F:13 T:12] [F:4 T:9] [F:16 T:14] [F:12 T:2] [F:7 T:14] [F:14 T:14] [F:14 T:1] [F:16 T:10] [F:11 T:6] [F:13 T:4] [F:6 T:1] [F:3 T:6] [F:7 T:15] [F:15 T:10] [F:7 T:6] [F:5 T:4] [F:13 T:9] [F:15 T:9] [F:15 T:13] [F:5 T:5] [F:5 T:11] [F:11 T:13] [F:6 T:4] [F:3 T:13] [F:13 T:2] [F:15 T:16] [F:3 T:16]

[F:16 T:11] [F:5 T:7] [F:4 T:5] [F:4 T:6] [F:5 T:6] [F:14 T:2] [F:7 T:11] [F:6 T:14] [F:3 T:12] [F:13 T:13] [F:12 T:12] [F:7 T:10] [F:6 T:7] [F:13 T:5] [F:14 T:5] [F:15 T:14] [F:6 T:3] [F:4 T:14] [F:11 T:12] [F:5 T:2] [F:11 T:16] [F:4 T:8] [F:11 T:11] [F:12 T:10] [F:5 T:9] [F:4 T:2] [F:13 T:3] [F:14 T:8] [F:5 T:14] [F:12 T:6] [F:16 T:7] [F:3 T:15] [F:4 T:11] [F:11 T:1] [F:16 T:6] [F:14 T:4] [F:4 T:3] [F:16 T:5] [F:6 T:10] [F:16 T:2] [F:16 T:8] [F:3 T:2] [F:11 T:14] [F:11 T:9] [F:15 T:7] [F:14 T:10] [F:7 T:16] [F:4 T:1]

# B CVRP E016-03m Variable Orders

**Initial order (i: Arc starting point, j, arc ending point):**

[i:5 j:7] [i:9 j:6] [i:6 j:16] [i:12 j:5] [i:12 j:13] [i:14 j:16] [i:11 j:7] [i:8 j:10] [i:3 j:4] [i:9 j:5] [i:12 j:7] [i:3 j:12] [i:1 j:12] [i:13 j:11] [i:2 j:15] [i:13 j:6] [i:11 j:12] [i:12 j:9] [i:16 j:2] [i:11 j:5] [i:12 j:11] [i:4 j:5] [i:1 j:9] [i:1 j:2] [i:7 j:4] [i:3 j:7] [i:13 j:15] [i:4 j:15] [i:8 j:14] [i:16 j:17] [i:13 j:17] [i:2 j:8] [i:6 j:9] [i:11 j:15] [i:10 j:14] [i:9 j:14] [i:14 j:11] [i:3 j:17] [i:10 j:13] [i:1 j:3] [i:2 j:17] [i:1 j:14] [i:8 j:9] [i:9 j:12] [i:13 j:3] [i:5 j:15] [i:13 j:5] [i:1 j:17] [i:11 j:8] [i:16 j:9] [i:9 j:8] [i:4 j:6] [i:13 j:4] [i:7 j:5] [i:8 j:17] [i:4 j:9] [i:7 j:10] [i:7 j:13] [i:8 j:2] [i:14 j:2] [i:8 j:16] [i:9 j:11] [i:13 j:9] [i:10 j:3] [i:5 j:8] [i:3 j:5] [i:10 j:9] [i:4 j:17] [i:6 j:17] [i:14 j:4] [i:9 j:10] [i:2 j:12] [i:10 j:6] [i:10 j:11] [i:2 j:6] [i:9 j:7] [i:6 j:8] [i:7 j:14] [i:9 j:3] [i:12 j:17] [i:16 j:10] [i:8 j:11] [i:12 j:6] [i:14 j:5] [i:6 j:12] [i:9 j:13] [i:15 j:7] [i:6 j:10] [i:2 j:14] [i:1 j:11] [i:2 j:5] [i:15 j:6] [i:6 j:13] [i:14 j:13] [i:11 j:6] [i:8 j:7] [i:15 j:4] [i:16 j:6] [i:12 j:3] [i:7 j:9] [i:4 j:10] [i:2 j:13] [i:15 j:8] [i:14 j:7] [i:7 j:8] [i:8 j:12] [i:11 j:2] [i:5 j:4] [i:12 j:2] [i:10 j:17] [i:1 j:10] [i:14 j:17] [i:9 j:15] [i:7 j:6] [i:5 j:9] [i:8 j:6] [i:4 j:2] [i:16 j:11] [i:7 j:11] [i:3 j:15] [i:13 j:8] [i:15 j:3] [i:16 j:12] [i:15 j:11] [i:14 j:9] [i:8 j:5] [i:15 j:2] [i:11 j:10] [i:16 j:15] [i:2 j:9] [i:6 j:11] [i:4 j:16] [i:8 j:13] [i:4 j:13] [i:4 j:7] [i:15 j:12] [i:3 j:6] [i:5 j:10] [i:12 j:4] [i:1 j:6] [i:4 j:14] [i:15 j:10] [i:6 j:15] [i:4 j:8] [i:1 j:4] [i:15 j:16] [i:6 j:7] [i:10 j:12] [i:3 j:16] [i:5 j:11] [i:16 j:3] [i:12 j:14] [i:5 j:17] [i:15 j:13] [i:14 j:6] [i:5 j:16] [i:11 j:9] [i:10 j:16] [i:7 j:3] [i:3 j:13] [i:5 j:13] [i:7 j:15] [i:12 j:10] [i:2 j:4] [i:16 j:14] [i:2 j:16] [i:13 j:10] [i:13 j:7] [i:15 j:5] [i:3 j:8] [i:6 j:2] [i:15 j:14] [i:4 j:3] [i:3 j:14] [i:11 j:4] [i:7 j:12] [i:1 j:7] [i:12 j:8] [i:14 j:8] [i:6 j:4] [i:11 j:17] [i:9 j:4] [i:2 j:11] [i:1 j:5] [i:13 j:2] [i:11 j:3] [i:1 j:8] [i:13 j:16] [i:13 j:12] [i:6 j:14] [i:10 j:2] [i:6 j:3] [i:12 j:15] [i:10 j:5] [i:4 j:12] [i:1 j:15] [i:5 j:2] [i:3 j:9]

[i:9 j:16] [i:14 j:12] [i:14 j:3] [i:8 j:15] [i:15 j:17] [i:2 j:10] [i:3 j:10] [i:3 j:2] [i:1 j:16] [i:5 j:3] [i:5 j:12] [i:11 j:16] [i:6 j:5] [i:11 j:14] [i:15 j:9] [i:10 j:8] [i:8 j:4] [i:1 j:13] [i:16 j:7] [i:4 j:11] [i:11 j:13] [i:8 j:3] [i:13 j:14] [i:16 j:8] [i:10 j:15] [i:14 j:15] [i:7 j:16] [i:7 j:17] [i:3 j:11] [i:14 j:10] [i:2 j:7] [i:16 j:4] [i:5 j:6] [i:9 j:2] [i:16 j:5] [i:9 j:17] [i:12 j:16] [i:5 j:14] [i:7 j:2] [i:10 j:7] [i:10 j:4] [i:16 j:13] [i:2 j:3]

**Resulting order Penalty VORF (Window 2, Increment 1):**

[i:16 j:11] [i:5 j:7] [i:9 j:6] [i:12 j:3] [i:6 j:16] [i:12 j:5] [i:12 j:13] [i:14 j:16] [i:11 j:7] [i:8 j:10] [i:3 j:4] [i:9 j:5] [i:12 j:7] [i:3 j:12] [i:1 j:12] [i:13 j:11] [i:2 j:15] [i:11 j:12] [i:12 j:9] [i:16 j:2] [i:11 j:5] [i:12 j:11] [i:4 j:5] [i:1 j:9] [i:7 j:4] [i:12 j:10] [i:3 j:7] [i:13 j:15] [i:4 j:15] [i:8 j:14] [i:7 j:16] [i:16 j:17] [i:13 j:17] [i:2 j:8] [i:6 j:9] [i:11 j:15] [i:10 j:14] [i:9 j:14] [i:14 j:11] [i:3 j:17] [i:10 j:13] [i:1 j:3] [i:2 j:17] [i:1 j:14] [i:8 j:9] [i:12 j:16] [i:9 j:12] [i:13 j:3] [i:5 j:15] [i:13 j:5] [i:1 j:17] [i:11 j:8] [i:16 j:9] [i:9 j:8] [i:4 j:6] [i:13 j:4] [i:7 j:5] [i:8 j:17] [i:4 j:9] [i:7 j:10] [i:7 j:13] [i:8 j:2] [i:1 j:15] [i:14 j:2] [i:8 j:16] [i:9 j:11] [i:13 j:9] [i:10 j:3] [i:5 j:8] [i:3 j:5] [i:10 j:9] [i:4 j:17] [i:6 j:17] [i:13 j:6] [i:14 j:4] [i:9 j:10] [i:9 j:16] [i:2 j:12] [i:10 j:6] [i:10 j:11] [i:2 j:6] [i:9 j:7] [i:6 j:8] [i:7 j:14] [i:9 j:3] [i:12 j:17] [i:16 j:10] [i:8 j:11] [i:12 j:6] [i:6 j:12] [i:9 j:13] [i:15 j:7] [i:6 j:10] [i:2 j:14] [i:1 j:11] [i:2 j:5] [i:15 j:6] [i:6 j:13] [i:14 j:13] [i:11 j:6] [i:8 j:7] [i:15 j:4] [i:16 j:6] [i:7 j:9] [i:4 j:10] [i:2 j:13] [i:15 j:8] [i:14 j:7] [i:7 j:8] [i:8 j:12] [i:11 j:2] [i:12 j:2] [i:10 j:17] [i:16 j:4] [i:1 j:10] [i:14 j:17] [i:9 j:15] [i:7 j:6] [i:5 j:9] [i:8 j:6] [i:4 j:2] [i:7 j:11] [i:3 j:15] [i:13 j:8] [i:15 j:3] [i:16 j:12] [i:15 j:11] [i:14 j:9] [i:8 j:5] [i:15 j:2] [i:11 j:10] [i:16 j:15] [i:2 j:9] [i:6 j:11] [i:4 j:16] [i:8 j:13] [i:4 j:13] [i:14 j:5] [i:4 j:7] [i:15 j:12] [i:3 j:6] [i:5 j:10] [i:12 j:4] [i:1 j:6] [i:4 j:14] [i:15 j:10] [i:6 j:15] [i:4 j:8] [i:1 j:4] [i:2 j:3] [i:15 j:16] [i:6 j:7] [i:10 j:12] [i:3 j:16] [i:16 j:3] [i:12 j:14] [i:5 j:17] [i:15 j:13] [i:14 j:6] [i:5 j:16] [i:11 j:9] [i:5 j:14] [i:10 j:16] [i:7 j:3] [i:3 j:13] [i:5 j:13] [i:7 j:15] [i:2 j:4] [i:16 j:14] [i:2 j:16] [i:13 j:10] [i:13 j:14] [i:13 j:7] [i:15 j:5] [i:3 j:8] [i:6 j:2] [i:15 j:14] [i:4 j:3] [i:3 j:14] [i:11 j:4] [i:7 j:12] [i:1 j:7] [i:12 j:8] [i:14 j:8] [i:6 j:4] [i:11 j:17] [i:9 j:4] [i:2 j:11] [i:1 j:5] [i:13 j:2] [i:11 j:3] [i:1 j:8] [i:13 j:16] [i:13 j:12] [i:6 j:14] [i:10 j:2] [i:6 j:3] [i:12 j:15] [i:10 j:5] [i:4 j:12] [i:5 j:2] [i:3 j:9] [i:14 j:12] [i:14 j:3] [i:8 j:15] [i:15 j:17] [i:2 j:10] [i:3 j:10] [i:3 j:2] [i:1 j:16] [i:5 j:3] [i:5 j:12] [i:11 j:16] [i:6 j:5] [i:11 j:14] [i:15 j:9] [i:10 j:8] [i:8 j:4] [i:1 j:13] [i:16 j:7] [i:4 j:11] [i:11 j:13] [i:8 j:3] [i:16 j:8] [i:5 j:11]

[i:10 j:15] [i:14 j:15] [i:7 j:17] [i:3 j:11] [i:14 j:10] [i:2 j:7] [i:5 j:6] [i:9 j:2] [i:16 j:5] [i:9 j:17] [i:7 j:2] [i:1 j:2] [i:5 j:4] [i:10 j:7] [i:10 j:4] [i:16 j:13]

**Resulting order Penalty VORF (Window 40, Increment 8):**

[i:5 j:7] [i:9 j:6] [i:6 j:16] [i:12 j:5] [i:12 j:13] [i:14 j:16] [i:11 j:7] [i:8 j:10] [i:3 j:4] [i:9 j:5] [i:12 j:7] [i:3 j:12] [i:1 j:12] [i:13 j:11] [i:2 j:15] [i:13 j:6] [i:11 j:12] [i:12 j:9] [i:16 j:2] [i:12 j:11] [i:4 j:5] [i:1 j:9] [i:7 j:4] [i:3 j:7] [i:13 j:15] [i:4 j:15] [i:8 j:14] [i:1 j:5] [i:16 j:17] [i:13 j:17] [i:2 j:8] [i:6 j:9] [i:11 j:15] [i:10 j:14] [i:9 j:14] [i:14 j:11] [i:3 j:17] [i:10 j:13] [i:1 j:3] [i:2 j:17] [i:1 j:14] [i:8 j:9] [i:9 j:12] [i:13 j:3] [i:5 j:15] [i:13 j:5] [i:1 j:17] [i:11 j:8] [i:16 j:9] [i:9 j:8] [i:4 j:6] [i:13 j:4] [i:7 j:5] [i:8 j:17] [i:4 j:9] [i:7 j:10] [i:7 j:13] [i:14 j:2] [i:8 j:16] [i:9 j:11] [i:13 j:9] [i:10 j:3] [i:5 j:8] [i:3 j:5] [i:10 j:9] [i:6 j:17] [i:14 j:4] [i:9 j:10] [i:2 j:12] [i:10 j:6] [i:10 j:11] [i:2 j:6] [i:9 j:7] [i:6 j:8] [i:7 j:14] [i:9 j:3] [i:12 j:17] [i:16 j:10] [i:8 j:11] [i:12 j:6] [i:6 j:12] [i:9 j:13] [i:15 j:7] [i:6 j:10] [i:1 j:11] [i:2 j:5] [i:15 j:6] [i:6 j:13] [i:14 j:13] [i:11 j:6] [i:8 j:7] [i:15 j:4] [i:16 j:6] [i:12 j:3] [i:4 j:10] [i:2 j:13] [i:15 j:8] [i:14 j:7] [i:7 j:8] [i:8 j:12] [i:11 j:2] [i:5 j:4] [i:12 j:2] [i:10 j:17] [i:1 j:10] [i:14 j:17] [i:9 j:15] [i:7 j:6] [i:5 j:9] [i:8 j:6] [i:16 j:11] [i:7 j:11] [i:3 j:15] [i:13 j:8] [i:15 j:3] [i:16 j:12] [i:15 j:11] [i:14 j:9] [i:8 j:5] [i:15 j:2] [i:3 j:11] [i:11 j:10] [i:16 j:15] [i:9 j:4] [i:2 j:9] [i:14 j:12] [i:6 j:11] [i:4 j:16] [i:8 j:13] [i:4 j:7] [i:15 j:12] [i:3 j:6] [i:5 j:10] [i:12 j:4] [i:1 j:6] [i:4 j:14] [i:15 j:10] [i:6 j:15] [i:4 j:8] [i:1 j:4] [i:15 j:16] [i:6 j:7] [i:10 j:12] [i:14 j:5] [i:3 j:16] [i:5 j:11] [i:16 j:3] [i:16 j:7] [i:12 j:14] [i:5 j:17] [i:15 j:13] [i:14 j:6] [i:5 j:16] [i:11 j:9] [i:10 j:16] [i:7 j:3] [i:3 j:13] [i:5 j:13] [i:7 j:15] [i:12 j:10] [i:2 j:4] [i:2 j:16] [i:13 j:10] [i:13 j:7] [i:15 j:5] [i:3 j:8] [i:6 j:2] [i:15 j:14] [i:4 j:3] [i:4 j:17] [i:3 j:14] [i:16 j:14] [i:11 j:4] [i:7 j:12] [i:1 j:7] [i:1 j:8] [i:8 j:2] [i:6 j:4] [i:1 j:2] [i:2 j:11] [i:1 j:13] [i:13 j:2] [i:11 j:3] [i:11 j:14] [i:13 j:16] [i:13 j:12] [i:16 j:5] [i:4 j:13] [i:10 j:2] [i:6 j:3] [i:12 j:15] [i:10 j:5] [i:4 j:12] [i:5 j:2] [i:3 j:9] [i:9 j:16] [i:8 j:15] [i:15 j:17] [i:14 j:3] [i:2 j:10] [i:3 j:10] [i:1 j:15] [i:3 j:2] [i:1 j:16] [i:11 j:17] [i:5 j:3] [i:5 j:12] [i:11 j:16] [i:6 j:5] [i:15 j:9] [i:4 j:2] [i:10 j:8] [i:8 j:4] [i:14 j:8] [i:4 j:11] [i:11 j:13] [i:8 j:3] [i:13 j:14] [i:16 j:8] [i:10 j:15] [i:14 j:15] [i:7 j:16] [i:7 j:17] [i:14 j:10] [i:2 j:7] [i:7 j:9] [i:2 j:14] [i:16 j:4] [i:5 j:6] [i:9 j:2] [i:9 j:17] [i:12 j:16] [i:5 j:14] [i:7 j:2] [i:10 j:7] [i:10 j:4] [i:16 j:13] [i:12 j:8] [i:2 j:3] [i:6 j:14] [i:11 j:5]

**Resulting order Adjusted VORF (Window 2, Increment 1):**

[i:5 j:7] [i:9 j:6] [i:6 j:16] [i:12 j:5] [i:12 j:13] [i:14 j:16] [i:11 j:7] [i:8 j:10] [i:3 j:4] [i:9 j:5] [i:12 j:7] [i:3 j:12] [i:1 j:12] [i:13 j:11] [i:2 j:15] [i:11 j:12] [i:12 j:9] [i:16 j:2] [i:11 j:5] [i:12 j:11] [i:4 j:5] [i:1 j:9] [i:1 j:2] [i:7 j:4] [i:3 j:7] [i:13 j:15] [i:4 j:15] [i:8 j:14] [i:16 j:17] [i:13 j:17] [i:2 j:8] [i:6 j:9] [i:11 j:15] [i:10 j:14] [i:9 j:14] [i:14 j:11] [i:3 j:17] [i:10 j:13] [i:1 j:3] [i:2 j:17] [i:1 j:14] [i:8 j:9] [i:9 j:12] [i:13 j:3] [i:5 j:15] [i:13 j:5] [i:1 j:17] [i:11 j:8] [i:16 j:9] [i:9 j:8] [i:4 j:6] [i:13 j:4] [i:7 j:5] [i:5 j:14] [i:8 j:17] [i:4 j:9] [i:7 j:10] [i:7 j:13] [i:8 j:2] [i:14 j:2] [i:8 j:16] [i:9 j:11] [i:13 j:9] [i:10 j:3] [i:5 j:8] [i:3 j:5] [i:10 j:9] [i:4 j:17] [i:6 j:17] [i:14 j:4] [i:13 j:6] [i:9 j:10] [i:2 j:12] [i:10 j:6] [i:10 j:11] [i:2 j:6] [i:9 j:7] [i:6 j:8] [i:7 j:14] [i:9 j:3] [i:12 j:17] [i:16 j:10] [i:8 j:11] [i:12 j:6] [i:14 j:5] [i:6 j:12] [i:9 j:13] [i:15 j:7] [i:6 j:10] [i:2 j:14] [i:1 j:11] [i:2 j:5] [i:15 j:6] [i:14 j:13] [i:11 j:6] [i:8 j:7] [i:15 j:4] [i:16 j:6] [i:12 j:3] [i:7 j:9] [i:4 j:10] [i:2 j:13] [i:15 j:8] [i:14 j:7] [i:7 j:8] [i:8 j:12] [i:11 j:2] [i:5 j:4] [i:12 j:2] [i:10 j:17] [i:1 j:10] [i:14 j:17] [i:9 j:15] [i:7 j:6] [i:5 j:9] [i:8 j:6] [i:4 j:2] [i:16 j:11] [i:7 j:11] [i:3 j:15] [i:13 j:8] [i:15 j:3] [i:16 j:12] [i:15 j:11] [i:14 j:9] [i:8 j:5] [i:15 j:2] [i:11 j:10] [i:16 j:15] [i:2 j:9] [i:6 j:11] [i:4 j:16] [i:8 j:13] [i:4 j:13] [i:4 j:7] [i:15 j:12] [i:3 j:6] [i:5 j:10] [i:12 j:4] [i:1 j:6] [i:4 j:14] [i:15 j:10] [i:6 j:15] [i:4 j:8] [i:1 j:4] [i:15 j:16] [i:6 j:7] [i:10 j:12] [i:3 j:16] [i:5 j:11] [i:16 j:3] [i:12 j:14] [i:5 j:17] [i:15 j:13] [i:14 j:6] [i:5 j:16] [i:11 j:9] [i:10 j:16] [i:7 j:3] [i:3 j:13] [i:5 j:13] [i:7 j:15] [i:12 j:10] [i:2 j:4] [i:16 j:14] [i:2 j:16] [i:13 j:10] [i:13 j:7] [i:15 j:5] [i:3 j:8] [i:6 j:2] [i:15 j:14] [i:4 j:3] [i:3 j:14] [i:11 j:4] [i:7 j:12] [i:1 j:7] [i:12 j:8] [i:14 j:8] [i:6 j:4] [i:11 j:17] [i:9 j:4] [i:2 j:11] [i:1 j:5] [i:13 j:2] [i:11 j:3] [i:1 j:8] [i:13 j:16] [i:13 j:12] [i:6 j:14] [i:10 j:2] [i:6 j:3] [i:12 j:15] [i:10 j:5] [i:4 j:12] [i:1 j:15] [i:5 j:2] [i:3 j:9] [i:9 j:16] [i:14 j:12] [i:14 j:3] [i:8 j:15] [i:15 j:17] [i:2 j:10] [i:3 j:10] [i:3 j:2] [i:1 j:16] [i:5 j:3] [i:5 j:12] [i:11 j:16] [i:6 j:5] [i:11 j:14] [i:15 j:9] [i:10 j:8] [i:8 j:4] [i:1 j:13] [i:16 j:7] [i:4 j:11] [i:11 j:13] [i:8 j:3] [i:13 j:14] [i:16 j:8] [i:10 j:15] [i:14 j:15] [i:7 j:16] [i:7 j:17] [i:3 j:11] [i:14 j:10] [i:2 j:7] [i:16 j:4] [i:5 j:6] [i:9 j:2] [i:16 j:5] [i:9 j:17] [i:12 j:16] [i:7 j:2] [i:10 j:7] [i:6 j:13] [i:10 j:4] [i:16 j:13] [i:2 j:3]

**Resulting order Adjusted VORF (Window 40, Increment 8):**

[i:5 j:7] [i:9 j:6] [i:6 j:16] [i:12 j:5] [i:12 j:13] [i:14 j:16] [i:11 j:7] [i:8 j:10] [i:3 j:4] [i:9 j:5] [i:12 j:7] [i:3 j:12] [i:1 j:12] [i:13 j:11] [i:11 j:16] [i:2 j:15] [i:13 j:6] [i:11 j:12]

[i:12 j:9] [i:16 j:2] [i:11 j:5] [i:12 j:11] [i:4 j:5] [i:1 j:9] [i:1 j:2] [i:7 j:4] [i:3 j:7]
[i:13 j:15] [i:4 j:15] [i:8 j:14] [i:16 j:17] [i:13 j:17] [i:2 j:8] [i:6 j:9] [i:11 j:15] [i:10 j:14]
[i:9 j:14] [i:14 j:11] [i:3 j:17] [i:10 j:13] [i:1 j:3] [i:2 j:17] [i:1 j:14] [i:8 j:9] [i:9 j:12]
[i:13 j:3] [i:5 j:15] [i:13 j:5] [i:1 j:17] [i:11 j:8] [i:16 j:9] [i:9 j:8] [i:4 j:6] [i:13 j:4]
[i:7 j:5] [i:8 j:17] [i:4 j:9] [i:7 j:10] [i:7 j:13] [i:8 j:2] [i:6 j:10] [i:14 j:2] [i:8 j:16]
[i:9 j:11] [i:13 j:9] [i:10 j:3] [i:5 j:8] [i:3 j:5] [i:10 j:9] [i:4 j:17] [i:6 j:17] [i:14 j:4]
[i:9 j:10] [i:2 j:12] [i:10 j:6] [i:10 j:11] [i:2 j:6] [i:9 j:7] [i:6 j:8] [i:7 j:14] [i:9 j:3]
[i:12 j:17] [i:16 j:10] [i:8 j:11] [i:12 j:6] [i:14 j:5] [i:6 j:12] [i:9 j:13] [i:15 j:7] [i:2 j:14]
[i:1 j:11] [i:2 j:5] [i:15 j:6] [i:6 j:13] [i:14 j:13] [i:11 j:6] [i:8 j:7] [i:15 j:4] [i:16 j:6]
[i:12 j:3] [i:7 j:9] [i:4 j:10] [i:2 j:13] [i:15 j:8] [i:14 j:7] [i:7 j:8] [i:8 j:12] [i:11 j:2]
[i:5 j:4] [i:12 j:2] [i:10 j:17] [i:1 j:10] [i:14 j:17] [i:9 j:15] [i:8 j:15] [i:7 j:6] [i:5 j:9]
[i:8 j:6] [i:4 j:2] [i:16 j:11] [i:7 j:11] [i:3 j:15] [i:13 j:8] [i:15 j:3] [i:16 j:12] [i:15 j:11]
[i:14 j:9] [i:8 j:5] [i:15 j:2] [i:11 j:10] [i:16 j:15] [i:2 j:9] [i:6 j:11] [i:4 j:16] [i:8 j:13]
[i:4 j:13] [i:4 j:7] [i:15 j:12] [i:3 j:6] [i:5 j:10] [i:12 j:4] [i:1 j:6] [i:4 j:14] [i:15 j:10]
[i:6 j:15] [i:4 j:8] [i:1 j:4] [i:15 j:16] [i:6 j:7] [i:10 j:12] [i:3 j:16] [i:5 j:11] [i:16 j:3]
[i:12 j:14] [i:5 j:17] [i:15 j:13] [i:14 j:6] [i:5 j:16] [i:11 j:9] [i:10 j:16] [i:7 j:3] [i:3 j:13]
[i:5 j:13] [i:7 j:15] [i:12 j:10] [i:2 j:4] [i:16 j:14] [i:2 j:16] [i:13 j:10] [i:13 j:7] [i:15 j:5]
[i:3 j:8] [i:6 j:2] [i:15 j:14] [i:4 j:3] [i:3 j:14] [i:11 j:4] [i:7 j:12] [i:1 j:7] [i:12 j:8]
[i:14 j:8] [i:6 j:4] [i:11 j:17] [i:9 j:4] [i:2 j:11] [i:1 j:5] [i:13 j:2] [i:11 j:3] [i:1 j:8]
[i:13 j:16] [i:13 j:12] [i:6 j:14] [i:14 j:15] [i:10 j:2] [i:6 j:3] [i:12 j:15] [i:10 j:5] [i:4 j:12]
[i:1 j:15] [i:5 j:2] [i:3 j:9] [i:9 j:16] [i:14 j:12] [i:14 j:3] [i:15 j:17] [i:2 j:10] [i:3 j:10]
[i:3 j:2] [i:1 j:16] [i:5 j:3] [i:5 j:12] [i:6 j:5] [i:11 j:14] [i:15 j:9] [i:10 j:8] [i:8 j:4]
[i:1 j:13] [i:16 j:7] [i:4 j:11] [i:11 j:13] [i:8 j:3] [i:13 j:14] [i:16 j:8] [i:10 j:15] [i:7 j:16]
[i:7 j:17] [i:3 j:11] [i:14 j:10] [i:2 j:7] [i:16 j:4] [i:5 j:6] [i:9 j:2] [i:16 j:5] [i:9 j:17]
[i:12 j:16] [i:5 j:14] [i:7 j:2] [i:10 j:7] [i:10 j:4] [i:16 j:13] [i:2 j:3]