1 Introduction

This is the technical documentation of our work on the Case Study in Discrete Optimization with the project title Örder Batching". First we give a brief description of the point we started working on. This includes a short explanation of the problem and the definition of variables that we could get or precompute out of the data set we got. In chapter 3 we state the Integer Linear Program we decided on. We state two versions of this model, one assuming an undirected graph and one assuming a directed graph. We then implemented the undirected version. But if one wants to include special cases in the warehouse layout to the problem set up, then the directed version is easier to adapt to such special cases (e.g. one way streets, special aisles for bulky items, ...). In the fourth chapter there is an explanation of how to deal with the subtour elimination constraints. Why this approach works efficiently is explained as theoretical background to the implemented lazy constraint version. The fifth sections is an outlook on how to tackle a certain problem that came up during our project work. The theoretic method and a suggestion for implementation is explained.

2 Preliminaries

2.1 Explanation of the problem

In a warehouse many items are stored. If an order comes in, then the correct item has to be fetched. To avoid unnecessary paths more than one order can be fetched on one round through the warehouse. The problem we treat deals with how the orders should be combined (=batching) and how the path should be gone in each round through the warehouse (=routing), so that the total path length is minimal.

2.2 Input

As input variables we define:

- $N \in \mathbb{N}$ as the number of positions of the different products in the warehouse, 1 is defined as the start and N as the end depot
- $M \in \mathbb{N}$ as the number of possible batches. A batch is a subset of the orders such that the number of picks of these orders does not exceed VOL. (This number will be very large.)
- $A \in \mathbb{N}$ as the number of orders
- $S \in \{0,1\}^{A \times N}$ such that S_i^a is 1 if order a uses position i and 0 otherwise for every $a \in A$ and $i \in N$
- $c_{i,j} \in \mathbb{R}$ as the length of the path between positions i and j for $i, j \in N$
- $v_a \in \mathbb{N}$ as the volume of order a, in our model number of picks in order a. We hereby assume that all items have the same size and we treat picking several items at once as one pick.

- $VOL \in \mathbb{N}$ as the maximal capacity of the trays

2.3 Output

We aim for these variables as our output:

- $y \subset [A]$ as the set of batches with number of picks in $y \leq VOL$. A feasible batch therefore is a subset of the orders such that the number of picks of these orders does not exceed the capacity of the commissioning tray.

- $x_{i,j}^k$ as variables that indicate the path used in each batch with $i, j \in N, k \in M$. This corresponds to the path we go to fetch all orders in each batch.

3 Model: Integer Linear Programm

The following variables are the decisions variables and therefore our output:

- $x_{i,j}^k = \begin{cases} 1, & \text{if the path between } i \text{ to } j \text{ is used in batch } k \\ 0, & \text{else} \end{cases}$

- $y_a^k = \begin{cases} 1, & \text{if order } a \text{ is used in batch } k \\ 0, & \text{else} \end{cases}$

- $B_i^k = \begin{cases} 1, & \text{if position } i \text{ is visited in batch } k \\ 0, & \text{else} \end{cases}$

- $b_k = \begin{cases} 1, & \text{if batch } k \text{ is used} \\ 0, & \text{else} \end{cases}$

3.1 Model undirected graph

This section presents the model using an undirected graph. A description of the objective function and the constraints is included.

$$\min \quad \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j} \cdot x_{i,j}^{k} \tag{1}$$

$$\text{such that} \quad \sum_{l=1}^{i-1} x_{l,i}^{k} + \sum_{j=i+1}^{N} x_{i,j}^{k} = 2 \cdot B_i^k \qquad \forall i \in [N], \forall k \in [M] \tag{2}$$

$$x_{1,N}^{k} = b_k \qquad \forall k \in [M] \tag{3}$$

$$\sum_{i,j \in V, i \leq j} x_{i,j}^{k} \leq |S| - 1 \qquad \forall V \subsetneq [N], |V| \leq VOL + 2, V \neq \emptyset, \forall k \in [M] \tag{4}$$

$$\sum_{a=1}^{A} v_a \cdot y_a^k \leq b_k \cdot VOL \qquad \forall k \in [M] \tag{5}$$

$$\sum_{k=1}^{M} y_a^k = 1 \qquad \forall a \in [A] \tag{6}$$

$$y_a^k \cdot S_i^a \leq B_i^k \qquad \forall i \in [N], \forall k \in [M], \forall a \in [A] \tag{7}$$

$$x_{i,j}^{k} \in \{0,1\} \qquad \forall k \in [M], \forall i \in [N], \forall j \in (i+1), ..., N \tag{8}$$

$$y_a^k \in \{0,1\} \qquad \forall a \in [A], \forall k \in [M] \tag{9}$$

$$b_k \in \{0,1\} \qquad \forall k \in [M] \tag{10}$$

$$B_i^k \in \{0,1\} \qquad \forall i \in [N], \forall k \in [M] \tag{11}$$

Equation (1) is our objective function. The aim is to minimize the total path length of all batches. We assume that the "cost" of a warehouse commission mostly consists of the path length. We neglect other possible cost factors like the number of used batches.
The first constraint (2) ensures that every position has a degree of two, if it is part of the batch. This means that each position has to be visited and left exactly once. (Note: if we deal with a directed graph the direction of the edge matters and we have to define constraints for incoming and outgoing edges separately.) In equation (3) we make sure that after each tour we go from the end depot back to the starting point. Constraint (4) is also known as the subtour elimination. It ensures that there cannot be a circle in any real subset of the nodes. Each batch can only visit a limited number of nodes due to the capacity of the trays. Including the start and end depot at most $VOL + 2$ nodes can be included in one batch. Therefore we can modify the common subtour elimination to only those subsets of N that have a cardinality of at most $VOL + 2$. Still this leads

to exponentially many constraints but it reduces the number of constraints drastically. The next equation (5) ensures that the size of all orders in a certain batch cannot exceed the capacity of the commissioning tray. The feature that every order is processed in exactly one batch is being stated in equation (6) . The connection between batching and routing is made in constraint (7) . It states that if a order which needs to visit a certain position i and is contained in the batch k, then this batch has to visit position i. The rest of the equations are just binary constraints for the decision variables as defined before.

3.2 Model directed graph

The input and decision variables are similar as in section 3.1. The new formalization of the model is as follows:

$$\text{min} \quad \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j} \cdot x_{i,j}^{k} \tag{12}$$

$$\text{such that} \quad \sum_{l=1}^{N} x_{l,i}^{k} = B_i^k \qquad \forall i \in [N], \forall k \in [M] \tag{13}$$

$$\sum_{j=1}^{N} x_{i,j}^{k} = B_i^k \qquad \forall i \in [N], \forall k \in [M] \tag{14}$$

$$x_{N,1}^{k} = b_k \qquad \forall k \in [M] \tag{15}$$

$$\sum_{i,j \in S} x_{i,j}^{k} \leq |S| - 1 \qquad \forall S \subsetneq [N], |S| \leq VOL + 2, S \neq \emptyset, \forall k \in [M] \tag{16}$$

$$\sum_{i=1}^{A} v_a \cdot y_a^k \leq b_k \cdot VOL \qquad \forall k \in [M] \tag{17}$$

$$\sum_{k=1}^{M} y_a^k = 1 \qquad \forall a \in [A] \tag{18}$$

$$y_a^k \cdot S_i^a \leq B_i^k \qquad \forall i \in [N], \forall k \in [M], \forall a \in [A] \tag{19}$$

$$x_{i,j}^{k} \in \{0,1\} \qquad \forall k \in [M], \forall i \in [N], \forall j \in (i+1), ..., N \tag{20}$$

$$y_a^k \in \{0,1\} \qquad \forall a \in [A], \forall k \in [M] \tag{21}$$

$$b_k \in \{0,1\} \qquad \forall k \in [M] \tag{22}$$

$$B_i^k \in \{0,1\} \qquad \forall i \in [N], \forall k \in [M] \tag{23}$$

4 Subtour elimination

Because the subtour elimination constraints are exponentially many, we first want to solve the problem without these constraints. This will yield a optimal solution $x_{i,j}^*$. The aim now is to compute if this optimal solution violates a subtour elimination constraint and if it does which node subset contains the cycle. First we take a closer look at the subtour

elimination constraints:

$$\begin{aligned}
&\sum_{i,j\in S} x_{i,j}^k \leq |S| - 1 \\
\leftrightarrow &2\sum_{i,j\in S} x_{i,j}^k \leq 2|S| - 2 \\
\leftrightarrow &2 \leq 2|S| - 2\sum_{i,j\in S} x_{i,j}^k \\
\leftrightarrow &2 \leq \sum_{i\in S}\sum_{(i,j)\in\delta(i)} x_{i,j}^k - 2\sum_{i,j\in S} x_{i,j}^k \\
\leftrightarrow &2 \leq \sum_{(i,j)\in\delta(S)} x_{i,j}^k + 2\sum_{i,j\in S} x_{i,j}^k - 2\sum_{i,j\in S} x_{i,j}^k \\
\leftrightarrow &2 \leq \sum_{(i,j)\in\delta(S)} x_{i,j}^k
\end{aligned}$$

where $\delta(S)$ denotes the set of edges which are in the $S - V\backslash S$- cut. Therefore a set of edges violates the subtour elimination if and only if

$$\sum_{(i,j)\in\delta(S)} x_{i,j}^k < 2.$$

This is basically a min-cut-problem. If the smallest cut contains less than two edges, then it corresponds to a node set S which violates the subtour elimination constraint. We now construct an ILP describing this min-cut-problem.Let $x_{i,j}^*$ denote an optimal solution of the subtour relaxation of the above stated LP. Let $y_{i,j}$ denote the decision variable such that $y_{i,j} = 1$ if the edge $(i,j) \in \delta(S), 0$ otherwise. and for z_i holds: $z_i = 1$ if $i \in S, 0$ otherwise.

$$\begin{aligned}
\min \sum_{i,j\in[N]} x_{i,j}^* \cdot y_{i,j} \\
\text{such that } y_{i,j} &\geq z_i - z_j \\
y_{i,j} &\geq z_j - z_i \\
z_s &= 1 \\
z_t &= 0 \\
y &\in \{0,1\}^m \\
z &\in \{0,1\}^m
\end{aligned}$$

This can be solved in polynomial time by using the algorithm of Dinic or Ford-Fulkerson. If the objective value of the minimal cut is less than 2, the corresponding node subset S is a violating subtour set.

5 Column generation

Using the upper definition of the model a big problem concerning the size of our decision variables comes up. For instance we expect the decision variable $x_{i,j}^k$ to become of size

approximately $500 \times 500 \times 1500$, where we use the number of orders as an upper bound for the number of batches. As each batch has a limited capacity of VOL, in each batch there are at most VOL entries of $x_{i,j}^k$ not equal to zero. This means we have to save a very sparse and big number of decision variables even tough most of them will be zero. To approach this problem, that leads to running time and memory issues, we use the column generation approach as discussed in the lecture.
The following is a brief outline of how the method of column generation works. There are four steps to perform:

1. Start with the LP-relaxation of the underlying model and calculate its dual. In a very basic case this will look somehow like this:
   primal:

   $$\begin{aligned} &\max c^T x \\ &Ax \leq b \\ &x \geq 0 \end{aligned}$$

   dual:

   $$\begin{aligned} &\min b^T y \\ &A^T y \geq c \\ &y \geq 0 \end{aligned}$$

   Now reduce the primal by only considering a subset of variables e.g. set the majority of variables equal to zero. This corresponds to erasing columns in the constraint matrix A. (We do that mostly by guessing wildly and maybe using some heuristic that guarantees a feasible solution). The choice of variables should still lead to an at least feasible solution. This yields the so called reduced master problem:

   $$\begin{aligned} &\max (c')^T x \\ &A'x' \leq b \\ &x' \geq 0 \end{aligned}$$

2. Now solve this reduced master problem and its corresponding dual to get the optimal primal-dual pair (x', y).

3. Now we check if this solution is alredy optimal for the non-reduced master problem. We do that by using the duality theorem. If the solution of the reduced master problem (x', y) is not optimal, then it has to violate at least one constraint in the dual master problem. If it doesn't violate any constraint in the dual master, then we found an optimal solution of the primal master by setting all variables, that were not considered in the reduced version, to zero. To find the violated constraint we solve the pricing problem:

   $$\begin{aligned} &\max c - a^{(j)} y \\ &a^{(j)} \text{is the j-th column of A} \end{aligned}$$

   Note that this of course is not the reduced matrix A'!
   This pricing step should be solvable in a really efficient and fast way.

4. If the objective value of the pricing problem is positive, we found a violated inequality in the dual master. This corresponds to a column of A, that we have erased in the reduced master problem and therefore it corresponds to a variable not considered in the reduced master. So now we include this very variable and resolve the problem until we get an optimal solution or choose to stop the process.

In order to perform step 1 we need to calculate the dual to the underlying primal program. The following represents the relaxed version of the primal program. Apart from the relaxation it is only a reforumlation to make the calculation of the dual a bit easier:

$$\min \sum_{k=1}^{M}(\sum_{i=1}^{N}\sum_{j=1}^{N} c_{i,j} \cdot x_{i,j}^k + \sum_{a=1}^{A} 0 \cdot y_a^k + 0 \cdot b_k + \sum_{i=1}^{N} 0 \cdot B^k) \tag{24}$$

$$\text{such that} \sum_{j=1}^{N} x_{i,j}^k - 2 \cdot B_i^k = 0 \qquad \forall i \in [N], \forall k \in [M] \tag{25}$$

$$x_{1,N}^k - b_k = 0 \qquad \forall k \in [M] \tag{26}$$

$$-\sum_{a=1}^{A} v_a \cdot y_a^k + b_k \cdot VOL \geq 0 \qquad \forall k \in [M] \tag{27}$$

$$\sum_{k=1}^{M} y_a^k = 1 \qquad \forall a \in [A] \tag{28}$$

$$B_i^k - y_a^k \cdot S_i^a \geq 0 \qquad \forall i \in [N], \forall k \in [M], \forall a \in A \tag{29}$$

$$x_{i,j}^k, y_a^k, b_k, B_i^k \geq 0 \qquad \forall i, j \in [n], \forall a \in [A], \forall k \in [M] \tag{30}$$

We assign the dual variables $\alpha, \beta, \gamma, \delta$ and ϵ to the constraints (25), (26), (27), (28) and (29) respectively. Then the dual is of the following form:

$$\max \sum_{a=1}^{A} \delta_a \tag{31}$$

$$\text{such that } \alpha_i^k \leq c_{i,j} \qquad \forall i, j \in [N], i \neq 1 \wedge j \neq N, \forall k \in [M] \tag{32}$$

$$\alpha_1^k + \beta^k \leq c_{1,N} \qquad \forall k \in [M] \tag{33}$$

$$-2\alpha_i^k + \sum_{a=1}^{A} \epsilon_i^{a,k} \leq 0 \qquad \forall i \in [N], \forall k \in [M] \tag{34}$$

$$-v_a \cdot \gamma^k - \sum_{i=1}^{N} S_i^a \cdot \epsilon_i^{a,k} + \delta_a \leq 0 \qquad \forall a \in [A], \forall k \in [M] \tag{35}$$

$$-\beta^k + VOL \cdot \gamma^k \leq 0 \qquad \forall k \in [M] \tag{36}$$

$$\gamma^k \geq 0 \qquad \forall k \in [M] \tag{37}$$

$$\epsilon_i^{a,k} \geq 0 \qquad \forall i \in [M], \forall a \in [A], \forall k \in [M] \tag{38}$$

We now realised that our model has a special property. We want to illustrate this by an example: For the first step we start by guessing which variables could be zero. So if we

set some b_{k^*} to zero, then with (25) and (26) we also get that $x_{i,j}^{k^*}$ will be zero for all $i, j \in [N]$. Therefore these two inequalities will get trivial. The same happens for (27) and (29). Similar things happen if we erase different columns in the primal. So by erasing one column in the primal we also can erase some rows. In reverse we also generate inequalities by adding one variable to the primal decision variable set. This means if we generate a colunm within the column generation, we also generate new rows in the primal. Then these new rows in the primal correspond to new variables in the dual. So there are dual variables that stay unknown as long as we do not add the corresponding row to the primal constraint matrix. So if we search for a violated inequality in the pricing step then some dual variables are still unknown wich makes it really difficult to find this inequality.

This arising problem can be dealt with with different approaches: the first approach would be to guess the unknown dual variables. The second approach would be to split the LP up into two different programs so that we can apply the column generation to one of those. A natural split up would be to differentiate between the generation of possible batches and the calculation of the corresonding shortest path within these batches. Because this seems to be quite intuitive we decided to follow this idea. Therefore we formulated a program that generates feasible batches and one that calculates the corresponding path. As our objective is to minimize the path length we rate the calculated batches on their effect on the overall path length. We decided to do this iteratively to improve our choice of batches further and further.

First define an objective function to rate the quality of our batches.

$$(f(y_a^k))_\tau = \begin{cases} 0, & \text{if } \tau = 0 \\ C_{\tau-1} \cdot (\sum_{k\in[M]} \sum_{a\in[A]} y_a^k \cdot v_a)^{-1}, & \text{if } \tau \geq 1 \end{cases}$$

Now iteratively for $\tau = 0, ...$

$$\begin{aligned} & \min\ f(y_a^k)_\tau & \\ \text{s. t. } & \sum_{k=1}^{M} y_a^k = 1 & \forall a \in [A] \\ & \sum_{a=1}^{A} v_a \cdot y_a^k \leq \text{VOL} & \forall k \in [M] \\ y_a^k \geq 0 & & \end{aligned}$$

This then yields a solution $(y_a^k)_\tau^*, \forall a \in [A], k \in [M]$. We continue with this solution:

$$\begin{aligned}
&\min \Big(\sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} c_{i,j}\cdot x_{i,j}^k\Big) =: C_\tau \\
\text{s. t. } &\sum_{i=1,i\neq j}^{N} x_{i,j}^k - 2\cdot B_i^k = 0 && \forall i \in [N], \forall k \in [M] \\
&x_{1,N}^k - b_k = 0 && \forall k \in M] \\
&\sum_{a=1}^{A} v_a \cdot (y_a^k)_\tau^* \leq \mathrm{VOL}\cdot b_k && \forall k \in [M] \\
&(y_a^k)_\tau^* \cdot S_i^a \leq B_i^k && \forall i \in [N], \forall a \in [A], \forall k \in [M]
\end{aligned}$$

To stop the iteration we would suggest a criterium like if $C_\tau = C_{\tau-1} - \epsilon$, a time limit or a limited number of iterations. The new objective function $f(y_a^k)$ now measures the average path length per position. This is widely used in practice, really intuitive and therefore a reasonable suggestion. But there are instances where this objective function does not coincide with the minimal total path length. We assume that it would still generate good results in reality.
One can see that the first linear program that creates feasible batches now yields only one dual variable that is dependent one rows in the primal. We now have to guess only this dual variable for the column generation approach. Therefore this spilt up of the LP has reduced the above stated problem but not fully solved it. Nevertheless this spilt up has lead to a really intuitive and nice reformulation of the model that can be easier adapted to the column generation approach.