Technical Report

Mathematical modelling of the quality-based order assignment problem

Jacqueline Schmitt, Florian Hahn, Jochen Deuse

02/2018

technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

Part of the work on this technical report has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project B3.

Speaker: Prof. Dr. Katharina Morik
Address: TU Dortmund University
Joseph-von-Fraunhofer-Str. 23
D-44227 Dortmund
Web: http://sfb876.tu-dortmund.de

1 Introduction

The increasing global competition forces companies to reduce their production costs and increase the quality of their products at the same time. Due to individualized customer needs, there can be numerous customer requirements to the products that need to be fulfilled to ensure customer satisfaction. Therefore, many companies established a quality management (QM) system, which aims for continuous improvement of performance regarding system, process, and product quality. Basic concepts and requirements for QM systems can be found in the ISO 9000 standards series. A main principle hereby is the customer orientation so that individualized customer needs can be considered within the design of internal quality testing gates [4].

To ensure quality producing companies implement quality control circuits which vary in size depending on their application area. Product-related control circuits can be quite large because they only appear at quality gates which are mostly at the end of the value chain. To decrease product-related control circuits and reduce the amount of value adding to defective products, process-integrated quality monitoring needs to be established. In this context, the application of quality prediction with machine learning algorithms is highly promising [21]. The prediction of the anticipated final product quality allows to derive control decisions quite early in the process chain.

One possible control decision is the assignment of (intermediate) products to customer orders based on the comparison of their predicted/expected quality and customer requirements. Control points are established along the procution chain where control decisions are derived based on quality prediction. If the product does not fulfil the requirements of the currently assigned customer order, a mathemtical optimization model will provide suggestions for reassigning the products in order to minimize selected production logistic KPIs, e.g. delivery date deviation or average processing time.

Within this technical report we present two approaches to model the product to customer order assignment problem (PCO-AP) mathematically as a 0,1 assignment problem (0,1-AP) and generalized assignment problem (GAP).

The second chapter presents the mathematical modelling as well as miniature examples to demonstrate the approach. The third chapter presents a short introduction to algorithms for solving the assignment problem including a small example again. The fourth chapter concludes this reports and gives an outlook to further research within this concept.

2 Mathematical modelling of the product to customer order assignment problem (PCO-AP)

2.1 Problem statement

Suppose that for each customer order arriving, a production order to manufacture the desired product is placed. Customer requirements do not always directly relate to the functionality of a product. Therefore, each customer may require a product of a specific, but not unified, quality. The quality of products may also vary because of various reasons, like inhomogenous quality of raw materials or process variations. Quality predictions are used to derive insights on the expected final quality along the production chain in order derive control decisions at an early stage.

Every company measures their performance through production logistics KPIs. From a customer point of view, delivery date deviation seems to be a key factor for customer satisfaction.

The problem now is to determine the assignment of products to customer orders which results in the least overall delivery date deviation.

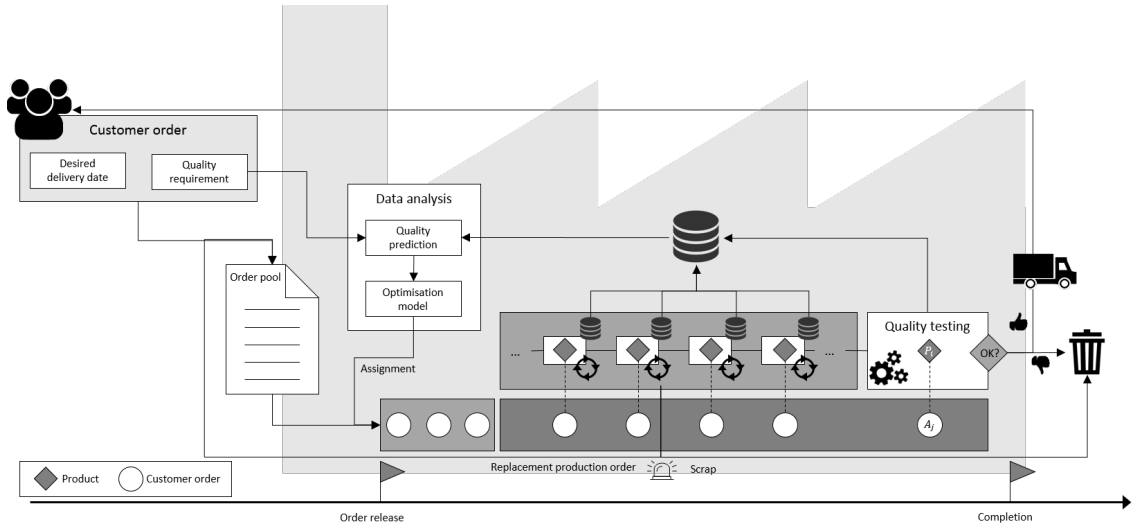

Figure 1: Overall concept [20]

2.2 Notation

For consistency the notation of the presented models is adapted to the problem of assigning products to customer orders based on their predicted final quality. In the most general form, the index sets are referred to as *agents* and *tasks* [18].

Table 1 shows different notations for the assignment problem found in literature (see for example [15, 18, 19]) and how these have been adpated for the assignment problem of products to customer orders based on their predicted quality. Detailed descriptions of the different parameters will be given at the respective point within this paper.

Table 1: Adapted notation for the assignment problem of products and customer orders

Index	Original notation	Adapted notation
$i \in I$	Agents, individuals	Customer orders
$j \in J$	Tasks, jobs	Products
$Q = (q_{ij})$	Qualification matrix	Quality matrix
$C = (c_{ij})$	Cost matrix	Production logistics KPI
$P = (p_{ij})$	Required resource	Predicted quality of product j (p_j)
$b_i > 0$	Available amount of resource	Required quality of order i
x_{ij}	Decision variable	

2.3 Model description as 0,1 assignment problem (0,1-AP)

The first presented modelling approach in the event of nominal quality criteria follows the formulation of the assignment problem used by Kuhn in 1955 as a 0,1 assignment problem [15].

(0,1-AP) Problem statement Suppose that n *agents* $(I = 1, ..., n)$ are available for n *tasks* $(J = 1, ..., n)$ and that a *qualification matrix* $Q = (q_{ij})$ is given, where $q_{ij} = 1$ if agent i qualifies for task j and $q_{ij} = 0$ otherwise. The question of the 0,1 assignment problem then is:

What is the largest number of tasks that can be assigned to qualified agents, such that not more than one task is assigned to each agent and vice versa?[15]

Modelling of (PCO-AP) as (0,1-AP) Within this modelling approach the qualification matrix $Q = (q_{ij})$ indicates whether a product j fulfills the quality requirements of an order i, so that $q_{ij} = 1$ and $q_{ij} = 0$ otherwise. Therefore, Q can also be referred to as the *quality matrix*.

This combinatorial optimization problem can be formulated as a linear program (LP) as follows:

$$(0,1\text{-}AP) \qquad \text{Maximize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_{ij} \tag{1}$$

$$\sum_{i\in I} x_{ij} = 1 \qquad \text{f.a. } j \in J \tag{2}$$

$$\sum_{j\in J} x_{ij} = 1 \qquad \text{f.a. } i \in I \tag{3}$$

$$\text{subject to} \quad x_{ij} \in \{0,1\} \qquad \text{f.a. } i \in I, j \in J \tag{4}$$

where $x_{ij} = 1$ if product j is assigned to customer i and $x_{ij} = 0$ otherwise.

Example 1 The following miniature example is constructed based on [15] with minor adjustments to illustrate the modelling of (PCO-AP) as a 0,1 assignment problem.

Four customers (denoted by $i = 1, 2, 3, 4$) require four products (denoted by $j = 1, 2, 3, 4$). Without loss of generality, we assume that customer orders arrive in ascending order and that the production orders are generated with identical enumeration (When order 1 arrives, the production of product 1 is initiated, ...). In this example, the product quality is evaluated nominally by the categorie A,B, and C, whereas A represents the best achievable and C the minimum required quality category so that A > B > C with > meaning "... is better than ...". The quality requirements of the customers are given according to table 2 and the predicted quality of the products are given according to table 3.

Table 2: Customer requirements

Customer	Quality requirement
1	C
2	A
3	B
4	A

Table 3: Predicted quality of products

Product	Predicted quality
1	C
2	B
3	B
4	A

A customer will accept all products that have at least his required quality. Accordingly, a customer that requires quality A will only accept products with quality A, a customer that requires quality B will accept products with quality A and B, and a customer that only requires quality C will accept any of the products (with quality A, B, and C).

Therefore,

$$\text{Customer} \begin{cases} 1 \\ 2 \\ 3 \\ 4 \end{cases} \quad \text{will accept product(s)} \begin{cases} 1,2,3 \text{ and } 4 \\ 4 \\ 2,3 \text{ and } 4 \\ 4 \end{cases}$$

This information can be presented effectively by the quality matrix

$$Q = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

in which rows stand for customers and columns for products. If a customer accepts a product, the respective entry is marked by a 1 and if he does not accept the product by a 0.

The question is now what is the largest number of customer orders that can be satisfied with the existing products? Thereby we need to take into account that each product can only be assigned to one customer order and vice versa. In terms of the matrix Q the question is what is the largest number of 1's that can be chosen from Q with no two chosen from the same row or column?

Since this example is quite small, a feasible and even optimal assignment can be derived easily without high computational effort. Looking at the rows and columns of Q it becomes obvious that customers 2 and 4 will only be satisfied if they are assigned to product 4. Customer 3 will be satisfied with products 2,3, and 4 and customer 1 will accept all four products. Therefore, the largest number of customers satisfied is 3, see coloured 1's in examples Q_1, Q_2 and Q_3.

$$Q_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad Q_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad Q_3 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Eventhough there are more possibilities for feasible assignments, it is impossible to improve the assignment in terms of increasing the number of satisfied customers. Therefore, one replacement order to fulfil the remaining customer order has to be generated and the product that cannot be assigned to any customer order has to be rejected as scrap or buffered for further orders.

From the industrial application point of view there are multiple approaches on how to determine which of the optimal solutions to choose.

From the product perspective we would reject a product with less value added rather than a product which is (almost) finished. Therefore, if multiple optimal solutions exist, we

would choose the solution where the product with least value added is not assigned to a customer order. Since we assumed that the production of the products started according to their index, we would prefer to reject product 3 rather than product 2 because the value added is assumed to be less due to its later start (e.g. Q_2).

From the customer point of view, customer satisfaction is highly related to on-time delivery. Hence, we would try to reduce delivery date deviations. For this example, we assume that the customer does not complain about an early delivery but will most likely not accept a delayed delivery. Accordingly, we would try to fulfil the customer orders that have been in the system the longest and assign a replacement order to the newest. In this example, we would place a replacement order for customer order 4 and assign the remaining orders to the products in any feasible way (e. g. Q_3).

Generally, there is a variety of production lgistic KPIs that are taken into account when deriving control decisions. To include these KPIs in the objective of the optimization a cost matrix has to be included within the objective function which leads to the problem formulation as a generalized assignment problem, presented in the next subsection.

2.4 Model description as generalized assignment problem

In this subsection a formulation of the assignment problem of intermediate products and customer orders as a generalized assignment problem (GAP) appears. This formulation of the problem is needed when quality criteria are metric (instead of nominal) and other production logistics KPIs should be considered in the objective of the optimization.

The generalized assignment problem is a generalization of the ordinary assignment problem of linear programming in which multiple assignments of tasks to agents are limited by some resource available to the agents [19].

(GAP) Problem statement Suppose that a set of agents $I = \{1, 2, ..., n\}$ is available for a set of tasks $J = \{1, 2, ..., n\}$ and that a performance rating matrix $R = (r_{ij})$ is given, where r_{ij} are positive integers for all i and j. An assignment consists of the choice of one task j for each agent i such that no task is assigned to two different agents and each agent is assigned exactly one task. The question of the GAP therefore is:

For which assignment is the sum of the performance ratings largest? [15, 19]

Within linear programming the performance ratings r_{ij} can be displayed in matrix form. A set of elements of a matrix are called *independent* if no two of them lie in the same row or column [18]. Respectively, the aim is to choose a set of n independent elements of the matrix R so that the sum of these elements is maximum.

The respective formulation of the GAP as a maximaziation problem of the overall performance rating is as follows:

$$(GAP_{Max}) \qquad \text{Maximize} \quad \sum_{i \in I} \sum_{j \in J} r_{ij} x_{ij} \tag{5}$$

$$\text{subject to} \quad \sum_{i \in I} x_{ij} = 1 \qquad \text{f.a. } j \in J \tag{6}$$

$$\sum_{j \in J} x_{ij} = 1 \qquad \text{f.a. } i \in I \tag{7}$$

$$x_{ij} \in \{0,1\} \qquad \text{f.a. } i \in I, \text{ f.a. } j \in J \tag{8}$$

Parenthesis: Transformation from maximization to minimization problem

Since most industrial applications focus on minimizing KPIs, like costs or processing times, it is useful to model the assignment problem as a minimzation instead of a maximization problem.

The maximization problem stated above can be transformed into a minimization problem. Therefore, let r be the maximum value of all r_{ij}, so that $r = \max_{i,j} r_{ij}$, and let $c_{ij} = r - r_{ij}$. Accordingly, the equivalent problem is to choose a set of n independent elements of the matrix $C = (c_{ij})$ such that the sum of these elements is minimum [18].
c_{ij} can be assumed to be the *costs* incurred if agent i is assigned to task j.

In addition to the 0,1-assignment problem, the generalized assignment problem consideres p_{ij} to be the *resource required* by agent i to do task j and $b_i > 0$ to be the amount of *resource available* to agent i. Therefore, the additional condition

$$\sum_{j \in J} p_{ij} x_{ij} \leq b_i \qquad \text{f.a. } i \in I \tag{9}$$

needs to be added to the problem formulation above. The natural interpretation of the decision variable again is

$$x_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to task } j, \\ 0 & \text{otherwise.} \end{cases}$$

Modelling of (PCO-AP) as (GAP) The products again represent the tasks $J = \{1, 2, ..., n\}$ and the customer orders the agents $I = \{1, 2, ..., n\}$. The costs c_{ij} can be any production logistics KPI, e. g. processing time or delivery date deviation. This is a major advantage compared to the (0,1-AP) approach because different objectives within the production logistics context can be considered which allows for greater applicability and scalability of the overall concept for different industrial applications.

The required resources p_{ij} that an agent i requires to perform task i can be interpreted as the predicted quality of product j. Since the predicted quality does not depend on the

assigned customer order i, the index i can be omitted (leading to p_j). In this context, b_i can be interpreted as the required quality of customer order i. Since the quality of the assigned product has to be greater or equal to the requirement, the "$\leq$" constraints 9 need to be transformed into "$\geq$" constraints. From a mathematical point of view, both sides of the inequalitis can be multiplied by -1 to invert the relation of both sides.

Accordingly, a mathematical formulation of the generalized assignment problem is [19]:

$$(GAP_{Min}) \qquad \text{Minimize} \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{10}$$

$$\text{subject to} \quad \sum_{j \in J} p_j x_{ij} \geq \tilde{b}_i \qquad \text{f.a. } i \in I \tag{11}$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \text{f.a. } j \in J \tag{12}$$

$$\sum_{j \in J} x_{ij} = 1 \qquad \text{f.a. } i \in I \tag{13}$$

$$x_{ij} \in \{0, 1\} \qquad \text{f.a. } i \in I, \text{ f.a. } j \in J \tag{14}$$

As one can see in constraints (11), this approach requires numerical quality information. As in most industrial applications, nominal quality information are gained by classifying numerical values using treshold values. This preprocessing step can be omitted and the original measured values can be used for quality prediction and the optimisation of the assignment instead.

The (GAP) as stated above consists of $3n + n^2$ constraints and n^2 decision variables. In addition, (GAP) is NP-hard [10] and considerable research has been done to find effective enumeration algorithms to solve problems of reasonable size to optimality (see chapter 3).

In general, as well as in this specific application, GAP can be reduced to a simple assignment problem (AP) which is a special case of GAP [19]. The advante of the AP compared to GAP is that it can be converted into a non-NP-hard problem which can be solved optimally within polynomial time.

When each constraint (11) of the GAP is scaled by dividing both sides of the inequality by the right-hand side value to obtain

$$\sum_{j \in J} k_{ij} x_{ij} \leq 1 \tag{15}$$

where

$$k_{ij} = \frac{p_j}{b_i}, \tag{16}$$

then (AP) is a special case of (GAP) in which $k_{ij} = 1$ for all $i \in I, j \in J$ [19].

In the context of (PCO-AP), $k_{ij} = 1$ f. a. i, j implies that the quality requirements of all customer orders and the quality of all products are identical. To ensure that individual quality requirements are fulfilled by assigned products, these constraints can be included in the calculation of the cost matrix C.

The entries of the matrix can thereby be modified as follows:

$$\tilde{c}_{ij} = \begin{cases} c_{ij} & \text{if quality requirement of order } i \text{ is fulfilled by product } j \text{ ,} \\ M \text{ or } - & \text{otherwise, depending on the selected solution algorithm.} \end{cases}$$

whereas c_{ij} is the original entry of the cost matrix C, representing the contribution to the selected KPI by the assignment of product j to order i and M is a very big number to ensure these entries are not part of the optimal solution. Instead of M the entries could also be left empty indicating that such an assignment is not feasible.

However, if there is no feasible solution such that the existing products are not sufficient to fulfil all customer order requirements, a replacement order has to be generated and one product has to be rejected. From the mathematical point of view, this requires to add another virtual product, representing the replacement production order, and a fictive customer representing the rejection as scrap to the problem. Therefore, the cost matrix $\tilde{C}$ is of size $(n+1) \times (n+1)$ with entries

$$\tilde{c}_{ij} = \begin{cases} c_{ij} & \text{if quality requirement of order } i \text{ is fulfilled by product } j \\ & \text{for } i, j = 1, ..., n, \\ M \text{ or } - & \text{otherwise for } i, j = 1, ..., n, \\ a_{ij} & \text{for } i = 1, ..., n, j = n+1 \text{ and } i = n+1, j = 1, ..., n \end{cases}$$

whereas there are different options to define suitable values for a_{ij}.

1. From the customer perspective, e.g. when minimizing delivery date deviation, the column $n+1$ holds the values $\hat{c}_{ij}$ which contribute to the objective function if a customer order is fulfilled by the replacement order. Whereas the rejection of any product as scrap (row $n+1$) is not of particular interest. Therefore,
   $$a_{ij} = \begin{cases} \hat{c}_{ij} & \text{for } i = 1, ..., n, j = n+1 \\ 0 & \text{otherwise} \end{cases}$$
2. From the production point of view, it might be of greater interest to reject products with least added value or to minimize the average order processing time. In the both cases, the product with the least progress would be rejected such that
   $$a_{ij} = \begin{cases} \hat{c}_{ij} & \text{for } i = n+1, j = 1, ..., n \\ 0 & \text{otherwise} \end{cases}$$
   with $\hat{c}_{ij}$ increasing along the production progress of the products.
3. Other values of a_{ij} highly depend on the selected objective as well as individual preference within the optimization process.

The simple assignment problem can accordingly be formulated as follows:

$$(AP) \qquad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\tilde{c}_{ij}x_{ij} \tag{17}$$

$$\text{subject to} \quad \sum_{j\in J}x_{ij} = 1 \qquad \text{f.a. } i \in I \tag{18}$$

$$\sum_{i\in I}x_{ij} = 1 \qquad \text{f.a. } j \in J \tag{19}$$

$$x_{ij} \in \{0,1\} \qquad \text{f.a. } i \in I, \text{ f.a. } j \in J \tag{20}$$

The problem (AP) could be considered as a mixed integer programming, which is NP-hard in general [22]. Noting that the constraint matrix is totally unimodular, the 0-1 constraints (20) can be ignored, so that the problem (AP) is reduced to a linear programming, which is polynomial solvable [7] and leads to the final formulation of (PCO-AP) as:

$$(PCO-AP) \qquad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\tilde{c}_{ij}x_{ij} \tag{21}$$

$$\text{subject to} \quad \sum_{j\in J}x_{ij} = 1 \qquad \text{f.a. } i \in I \tag{22}$$

$$\sum_{i\in I}x_{ij} = 1 \qquad \text{f.a. } j \in J \tag{23}$$

$$x_{ij} \geq 0 \qquad \text{f.a. } i \in I, \text{ f.a. } j \in J \tag{24}$$

Example 2 In this example we will extend example 1 and model the (PCO-AP) as a simple assignment problem. Again, the four customers (denoted by $i = 1, 2, 3, 4$) require four products (denoted by $j = 1, 2, 3, 4$). We now assume that the quality classes A,B, and C from example 1 had been derived through treshold values from metric measurement data.

Table 4: Quality tresholds

Quality Class	Tresholds
A	Q $\geq$ 0,8
B	0,6 $\leq$ Q $<$ 0,8
C	0,4 $\leq$ Q $<$ 0,6
Scrap	Q $<$ 0,4

Assuming the treshold classification given in table 4, the original metric quality requirements of the customers were according to table 5 and the predicted quality of the products was according to table 6.

Table 5: Customer requirements

Customer	Quality requirement
1	0,4
2	0,8
3	0,6
4	0,9

Table 6: Predicted quality of products

Product	Predicted quality
1	0,5
2	0,7
3	0,7
4	0,9

Using this quality information we construct the auxiliary matrix $Q = (q_{ij})$ where $q_{ij} = 1$ if the quality of product i fulfills the requirements of customer j, and $q_{ij} = 0$ otherwise.

$$Q = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The objective of this example is the minimization of delivery date deviations, an important KPI from the customer point of view as stated before. We assume that a production order in the original assignment (without loss of generality $i = j$) is placed for each customer order such that the delivery will be just on time. The calculation of the delivery date deviations depend on the underlying production setting and will therefore be not included in this paper. We assume the delivery date devations to be given as follows:

$$C = \begin{pmatrix} 0 & 10 & 20 & 30 \\ 10 & 0 & 10 & 20 \\ 20 & 10 & 0 & 10 \\ 30 & 20 & 10 & 0 \end{pmatrix}$$

The cost matrix $\tilde{C} = (\tilde{c}_{ij})$ will now be constructed as follows:

$$c_{ij} = \begin{cases} c_{ij}, & \text{if } q_{ij} = 1 \text{ for } i,j = 1,...,4 \\ - & \text{if } q_{ij} = 0 \text{ for } i,j = 1,...,4 \\ 0 & \text{for } i = 1,...,5, j = 5 \\ 50 - j \cdot 10 & \text{for } i = 5, j = 1,...,4 \end{cases} \tag{25}$$

Accordingly, the cost matrix $\tilde{C}$ is defined as

$$\tilde{C} = \begin{array}{c} \begin{array}{ccccc} P1 & P2 & P3 & P4 & PE \end{array} \\ \begin{pmatrix} 0 & 10 & 20 & 30 & 40 \\ - & - & - & 20 & 30 \\ - & 10 & 0 & 10 & 20 \\ - & - & - & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} C1 \\ C2 \\ C3 \\ C4 \\ \text{Scrap} \end{array}$$

The solution of the problem (PCO-AP) is an assignment of products to customer orders with least overall delivery date deviation. In terms of the matrix $\tilde{C}$ the task is to find 5 entries, where there is no more than one entry in each row or column, such that the sum of entries is minimum. To solve the assignment problem optimally and with least computational effort, several approaches had been developed. The next chapter provides an overview of algorithms and presents the most popular polynomial time algorithm for solving the simple assignment problem, the Kuhn-Munkres algorithm. We will also demonstrate the algorithm to solve this example and discuss the result from the industrial application point of view.

3 Algorithms for solving the assignment problem

Algorithms for solving the assignment problem are especially the Kuhn-Munkres algorithm and Shortest Augmented Path approaches (see [5, 6, 8]). Other approaches can for example be found in [2].

3.1 Kuhn-Munkres algorithm

The Kuhn-Munkres is one of the most popular polynomial time algorithms for solving the simple assignment problem. The algorithm was developed and published in 1955 by Kuhn [15]. The algorithm is also called "Hungarian method" because it is largely based on the ealier work of two Hungarian mathematicians in 1916 [14] and 1931 [9]. Theoretically, the Kuhn-Munkres algorithm is guaranteed to reach the global optimum [7].

There are two versions of implementations of the algorithm - matrix and graph [7]. The matrix formulation of the problem as presented in chapter 2 can be transferred into the graph version without loss of generality.

From a graph theoretical point of view the assignment problem can be considered as the maximum matching problem or bipartite graph [12]. The following paragraph will provide some graphtheoretical definitions that will be needed to understand and conduct the algorithm.

Graphtheoretical definitions
A *bipartite graph* $G = (V, E)$ consists of two disjoint and independent sets of *vertices* X and Y, so that $V = X \cup Y$ and $X \cap Y = \emptyset$. Each *edge* in a bipartite graph connects a vertex in X to one in Y, so that $E \in X \times Y$. The edges (x, y) between vertex $x \in X$ and a vertex $y \in Y$ have a *weight* $w(x, y)$.

The *neighbourhood* of a vertex v is the set $J_G(v)$ including all vertices sharing an edge with v. The *neighborhood* of a set S is the set $J_G(S)$ which includes all vertices that share an edge with a vertex in S.

Let $l(x)$ be the *label* for each vertex in the graph, which can also be interpreted as dual variables of the problem, where each label of a vertex corresponds to its only matching constraint [7]. A feasible labelling is a function $l : V \to \mathbb{R}$ that satisfies the condition $l(x) + l(y) \geq w(x, y), \forall x \in X, \forall y \in Y$.

Considering a *matching* $M(M \subseteq E)$, vertex y is called *matched* if it is a vertex in M, otherwise it is called *free*.

Let G_l denote the subgraph of G that contains all edges where $l(x)+l(y) = w(x, y)$. G_l is a *spanning subgraph* of G that includes all vertices from G. G_l only includes those edges from the bipartite matching which allow the vertices to be perfectly feasible [7].

Theoretically, if M^* is a *perfect matching* in the equality subgraph G_l, then M^* is a *maximum-weighted matching* in G. It can be shown that there is no perfect matching with greater weight than M^* so that M^* is a maximal perfect matching, and the Kuhn-Munkres algorithm is guaranteed to reach the global optimum [7].

The numerical time complexity of the algorithm is $O(n^3)$, with n elements in X (=tasks) and n elements in Y (=agents).

Original Kuhn-Munkres algorithm

Input: A bipartite graph $G = (V, E)$ and weights $w(x, y)$

Output: Optimal perfect matching M

Step 1: Generate initial labelling l and matching M in G_l, go to 2.

Step 2: If M perfect: STOP.

Otherwise pick free vertex $u \in X$. Set $S = u, T = \emptyset$, go to 3.

Step 3: If $J_l(S) = T$: update labels (forcing $J_l(S) \neq T$)

$$\alpha_l = \min_{s \in S, y \notin T} l(x) + l(y) - w(x, y)$$

$$\hat{l}(v) = \begin{cases} l(v) - \alpha_l, & v \in S \\ l(v) + \alpha_l, & v \in T \\ l(v), & \text{otherwise} \end{cases}$$

Go to 4.

If $J_l(S) \neq T$: choose $y \in J_l(S) \backslash T$, go to 4.

Step 4: If y free: $u - y$ is an augmenting path. Augment M and go to 2.

If y matched, assuming to z: extend the alternating tree:

$S = S \cup z$, $T = T \cup y$. Go to 3.

Algorithm 1: The original Kuhn-Munkres algorithm

Application of the Kuhn-Munkres algorithm to example 2 First, we have to define the example given in matrix form from the graph-theoretical point of view. The graph consists of vertices $V = C \cup P$, where C are the customer orders and P are the products. Since an entity can whether be an order or a product but not both, the required condition $C \cap P = \emptyset$ is fulfilled. The edges (c, p) represent all feasbile assignments between products and customer orders (all entries of $\tilde{C}$ that are not "$-$"). The weight $\tilde{w}(c, p)$ of the edges are the delivery date deviations as given in $\tilde{C}$.

Since the original formulation of the Kuhn-Munkres algorithm delivers a maximal matching, we have to transform our minimization into a maximization problem. Therefore, we transform the weights of the edges as follows:

$$w_{ij} = (\max_{i,j} \tilde{w}_{ij}) - \tilde{w}_{ij} \tag{26}$$

Hence, the graph representation of our example is according to figure 2.

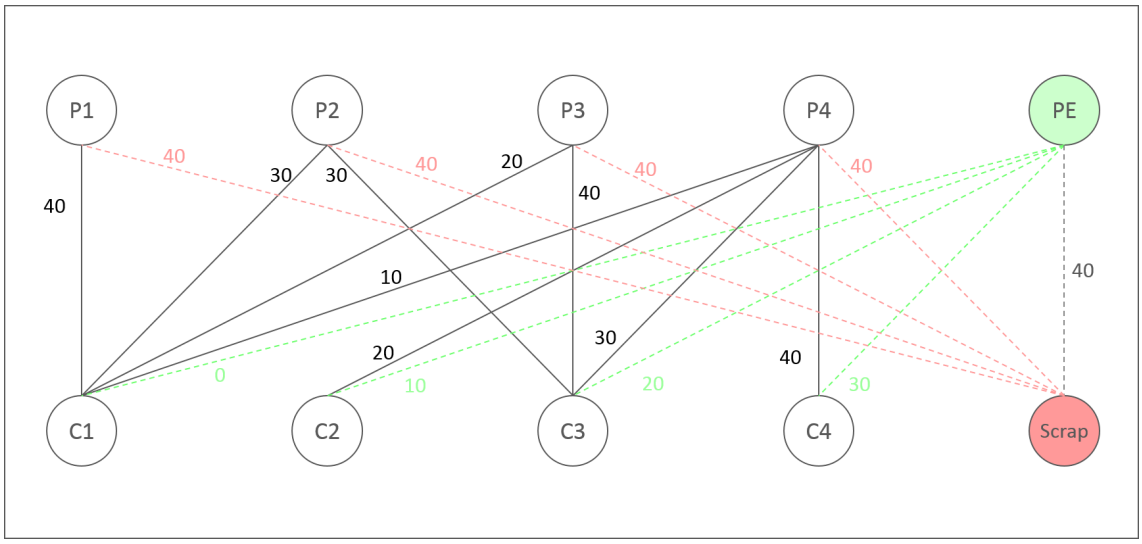

Figure 2: Graph with weights $w(c, p)$

We only plotted the edges that are feasible for our example. But as the algorithm requires the graph to be complete, we assume all other edges to be present and weighted with $w(c, p) = 0$ even if not included in figure 2.

Now we will apply the Kuhn-Munkres algorithm to our example.

Step 1: Generate initial labelling l and matching M in G_l
First, we set $l(p) = 0$ for all $p \in P$. We then define $l(c)$ as the maximum value of all edges connecting c to vertices in P. The spanning subgraph G_l includes all vertices but only those edges where $l(c) + l(p) = w(c, p)$ (see figure 3).

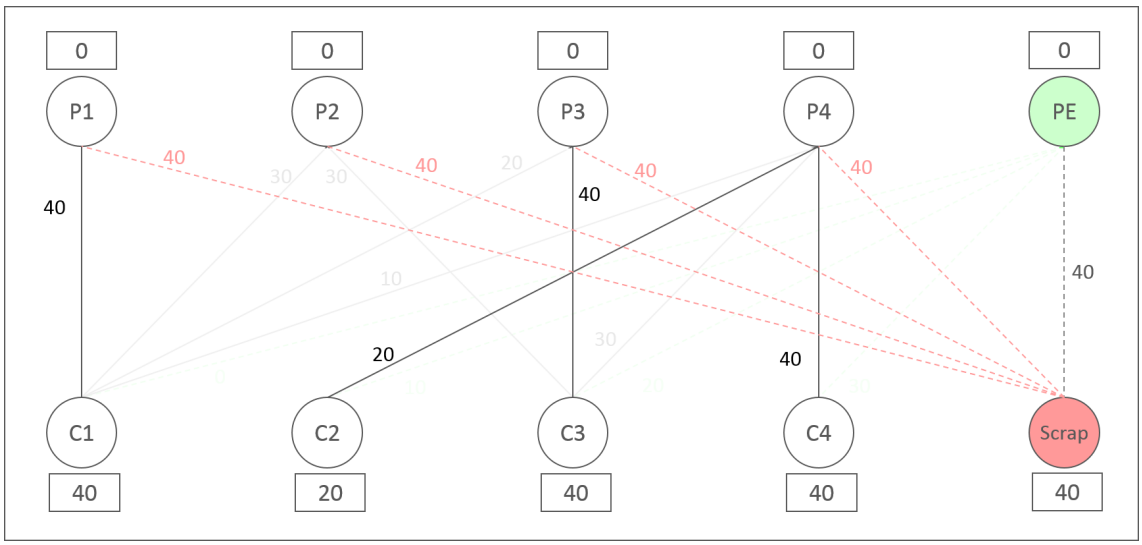

Figure 3: Initial labelling and spanning subgraph

We then perform an initial matching as shown in figure 4.

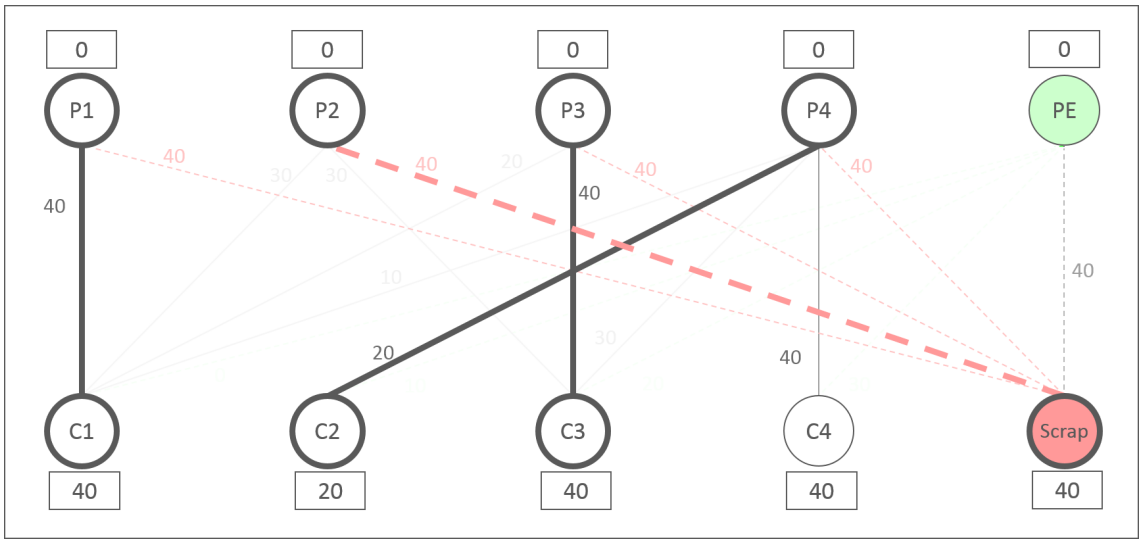

Figure 4: Initial matching

Step 2: Since M is not perfect (because PE and $C4$ are not matched), we now pick the free vertex $C4 \in C$ and set $S = \{C4\}$ and $T = \emptyset$.

Step 3: We determine $J_l(S) = \{P4\} \neq T$ and choose $P4 \in J_l(S) \backslash T$.

Step 4: Because $P4$ is matched to $C2$, we extend the alternating tree $S = \{C2, C4\}$ and set $T = \{P4\}$.

Step 3: Now it holds that $J_l(S) = \{P4\} = T$, we need to update the labels. Therefore we determine $l(c) + l(p) - w(c, p)$ for all $c \in S, p \notin T$ to find their minimum.

Table 7: Calculation of $(l(c) + l(p) - w(c, p))$ for all $c \in S, p \notin T$

	P1	P2	P3	PE
C2	20	20	20	10
C4	40	40	40	10

Accordingly,

$$\alpha_l = \min_{c \in S, p \notin T} l(c) + l(p) - w(c, p) = 10$$

and we update the labels

$$\hat{l}(v) = \begin{cases} l(v) - 10, & C2, C4 \\ l(v) + 10, & P4 \\ l(v), & \text{all other vertices} \end{cases}$$

add the edges where $l(c) + l(p) = w(c, p)$ now holds to the subgraph and keep the former matching (see figure 5)

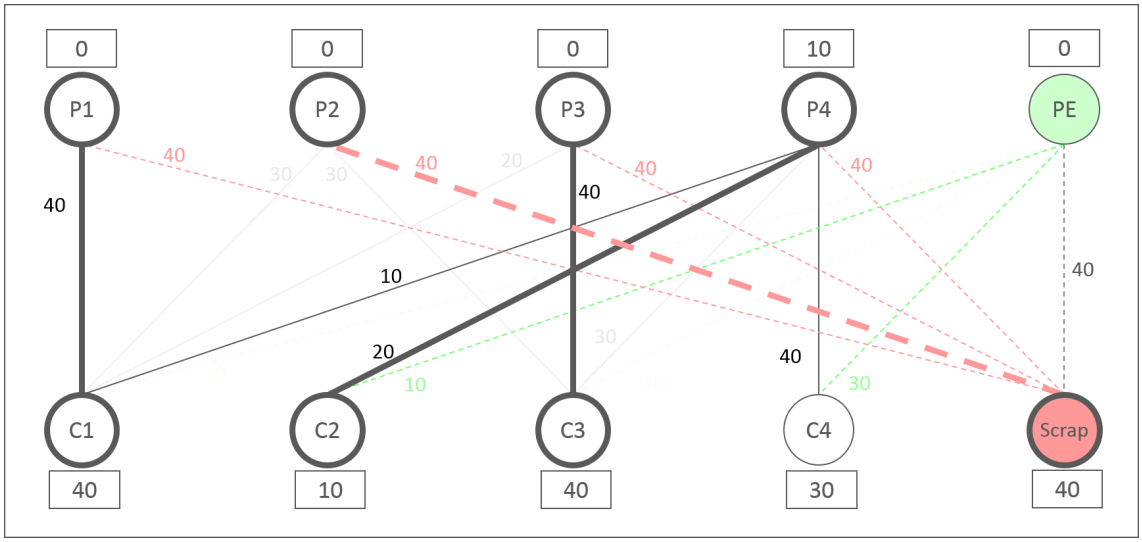

Figure 5: Updated labels with former matching and new edges in G_l

The neighborhood of S is now $J_l(S) = \{P4, PE\} \neq T$, so that we choose $PE \in J_l(S) \backslash T$.

Step 4: Because PE is free, we add the edge $(C4, PE)$ to M.

Step 2: As we can see in figure 6 the matching M is perfect and we have therefore found an optimal solution with an objective value of 170.

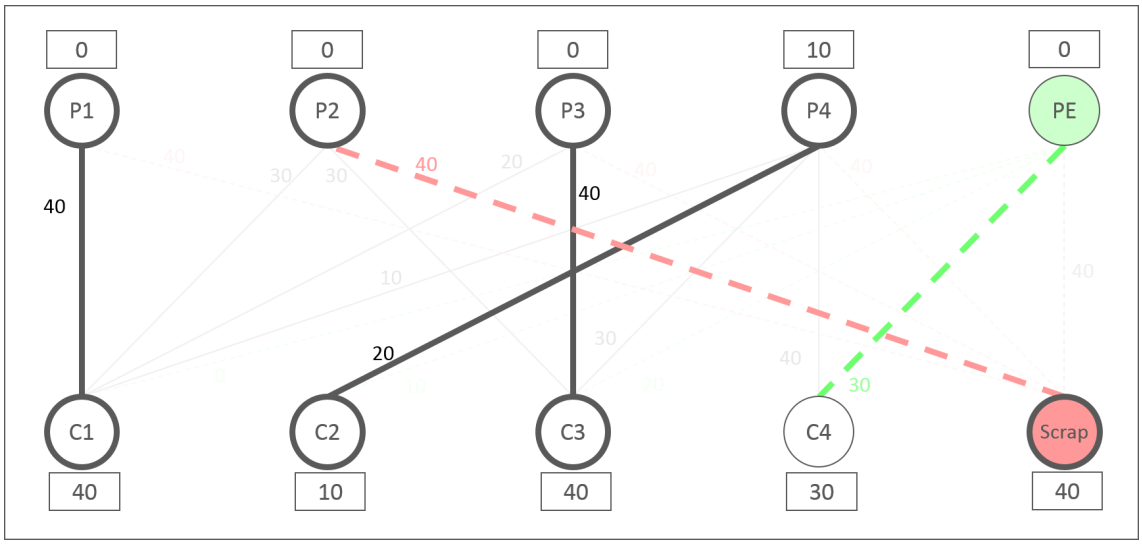

Figure 6: Perfect matching

Because we converted the costs of our original example to obtain a maximization problem, we now need to determine the respective target value in terms of delivery date deviations. Assigning the products to customer orders like shown in figure 6, will result in delivery date deviations of 40 time units (calculated by summing up the repective entries of $\tilde{C}$).

3.2 Brief overview of other algorithms

Various researchers investigated the Kuhn-Munkres algorithm and presented modifications and improvements. Jonker et al. [13] presented three approaches TODO, READY, and SCAN which speed up for sparse problems. Bertsekas et al. [3] investigated three synchronous and asynchronous implementations of the Kuhn-Munkres algorithm to reduce computational time cost. Cui et al. [7] proposed a sparsity based (sKM) and a parallel (pKM) Kuhn-Munkres modification. While sKM greatly improves the computational performance, pKM solves the assignment problem with considerable accuracy loss.

Eventhough, the Kuhn-Munkres is the most commonly used algorithm to solve the assignment problem, other approaches were investigated. Ross and Soland [19] present a branch and bound algorithm that solves the generalized assignment problem by solving a series of binary knapsack problems to determine the bounds. More details can be found for example in [1, 11, 16, 17, 19, 23]

4 Conclusion and Outlook

In our current research we investigate into quality-based order assignment. This paper presents the approach to model the problem from a mathematical point of view as a 0,1- and as a general assignment problem. To solve the assignment problem, we selected the Kuhn-Munkres algorithm in its original form with numerical time complexity of $O(n^3)$.

Next steps will include the coupling of quality prediction and order assignment and the more detailed definition of required system interfaces to be able to integrate the concept into industrial process control.

References

[1] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems. In *New Trends in Systems Theory*, pages 105–112. Springer, 1991.

[2] D. P. Bertsekas. *Linear Network Optimization: Algorithms and Codes.* The MIT Press, 1991.

[3] D. P. Bertsekas and D. A. Castañon. Parallel asynchronous hungarian methods for the assignment problem. *ORSA Journal on Computing*, 5(3):261–274, 1993.

[4] S. Brugger-Gebhardt. *Die DIN EN ISO 9001:2015 verstehen.* Springer Fachmedien Wiesbaden, 2016.

[5] R. E. Burkard and U. Derigs. *Assignment and Matching Problems: Solution Methods with FORTRAN-Programs.* Springer Berlin Heidelberg, 1980.

[6] G. Carpaneto, S. Martello, and P. Toth. Algorithms and codes for the assignment problem. *Annals of Operations Research*, 13(1):191–223, dec 1988.

[7] H. Cui, J. Zhang, C. Cui, and Q. Chen. Solving large-scale assignment problems by kuhn-munkres algorithm. *AME2*, 2016.

[8] U. Derigs. *Programming in Networks and Graphs*. Springer Berlin Heidelberg, 1988.

[9] J. Egerváry. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38(1931):16–28, 1931.

[10] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, sep 1986.

[11] A. Frank. On kuhn's hungarian method – a tribute from hungary. *Naval Research Logistics (NRL)*, 52(1):2–5, 2005.

[12] F. Glover. Maximum matching in a convex bipartite graph. *Naval Research Logistics Quarterly*, 14(3):313–316, 1967.

[13] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, dec 1987.

[14] D. König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77:453–465, 1916.

[15] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, mar 1955.

[16] H. W. Kuhn. On the origin of the hungarian method. *History of mathematical programming*, pages 77–81, 1991.

[17] G. A. Mills-Tettey, A. Stentz, and M. B. Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. 2007.

[18] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32 – 38, March 1957.

[19] G. T. Ross and R. M. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8(1):91–103, dec 1975.

[20] J. Schmitt, M. Wiegand, and J. Deuse. Qualitätsbasierte auftragszuordnung - zuordnung von zwischenprodukten zu kundenaufträgen auf basis von qualitätsprognosen. *ZWF online*, 04 2018.

[21] M. Wiegand, M. Stolpe, J. Deuse, and K. Morik. Prädiktive prozessüberwachung auf basis verteilt erfasster sensordaten. *Automatisierungstechnik*, 64(7):521–533, 2016.

[22] L. A. Wolsey. *Integer programming*. Wiley-Interscience, New York, NY, USA, 1998.

[23] H. Zhu, M. Zhou, and R. Alkins. Group role assignment via a kuhn–munkres algorithm-based solution. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(3):739–750, 2012.