universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Scheduling worker to jobs with a fixed sequence and time windows under the consideration of workload-balancing“

verfasst von / submitted by

Jessica Agbo, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz.

I want to thank

- My professor **Univ.-Prof. Dr. Karl Dörner** for his support
- My colleague **Mag. Karl Schneeberger** for useful inputs throughout the master thesis
- My colleague **Georg Fröhlich, BSc** for useful input concerning programming issues
- My partner **Michael Gerdenitsch** for useful input concerning formatting and correct reading
- **Mag. Viktoria Mahlendorf, MBA** and **Ing. Dipl. Ing. Kurt Mahlendorf, MSc BSc** for correct reading
- My mother **Marina Fally** for supporting me in raising my two children and giving me the strength to pass my master

Abstract

This work considers worker scheduling problems to jobs, which have a given sequence, are fixed scheduled to a machine and have tight hard time windows. The aim is to minimize the usage of workers and enable work-load balancing, which refers to the equal distribution of workload. Furthermore the minimization of long waiting times (breaks) in-between the jobs is taken under consideration. For the workload balancing two mathematical formulations are presented, a basic and a random construction heuristic are invented and the cheapest insertion method is used as improvement heuristic. At last a further heuristic is implemented to take care of the workload-balancing.

Zusammenfassung

Diese Arbeit behandelt die optimale Zuteilung von Mitarbeitern zu Aufgaben, die eine vorgegebene Reihenfolge und harte Zeitfenster haben sowie fix zu einer Maschine zugeteilt sind. Das Ziel ist die Minimierung der Mitarbeiter, wobei auf die Ermittlung einer ausgewogenen Verteilung der Arbeitsbelastung geachtet wird. Des Weiteren werden lange Wartezeiten (Pausen) zwischen den Aufgaben minimiert. Für die Berücksichtigung der gleichverteilten Arbeitsbelastung werden zwei mathematische Modelle vorgestellt, zwei Konstruktionsheuristiken – eine simple Heuristik und eine basierend auf Zufallsvariablen – wurden implementiert und die "cheapest insertion" Methode wurde zur Verbesserung der Konstruktionsheuristik angewandt. Zuletzt wurde eine weitere Heuristik entwickelt, um die Ausgewogenheit der Arbeitsbelastung zu gewährleisten.

Table of content

List of figures

List of tables

List of abbreviations

CVRP- capacitated vehicle routing problem

GAP- generalized assignment problem

ILL- Industrie-Logistik-Linz

LNS- Large Neighborhood Search

TSP- travelling salesman problem

VRP- vehicle routing problem

VRPTW- vehicle routing problem with time windows

WSPTW- worker scheduling problem with fixed job sequence and time windows

1. Introduction

Due to digitalization and upcoming themes like the “Industry 4.0” or the “Internet of Things”, there is need for efficient computer programs and automated systems for a variety of tasks including logistical planning to ensure competitiveness and productivity. The future of logistics control systems goes towards central control and the automatization of processes, therefore this work contributes to the central control of worker scheduling problems.

This master thesis is done in cooperation with the logistics provider Industrie-Logistik-Linz (ILL), which currently schedules one worker per machine due to the lack of central planning. The workers are assigned to different tasks from the refilling of the pallets to the storing and packaging. The current schedule is inefficient, since the workload of some workers is quite low-meaning that they stand nearby the machine waiting for the next task to be ready to be done.

With the centralized planning the optimal usage of the workers should be ensured which includes the increase in workload per worker and the reduction of workers needed. Furthermore the workload of the workers should be balanced in a fair way and the breaks in-between the tasks assigned to a worker should be minimized to assure that there is also a balance in the time schedule of the tasks. This has the advantage that there is a puffer in case the tasks need longer than planned, as this leads to smaller breaks between many tasks.

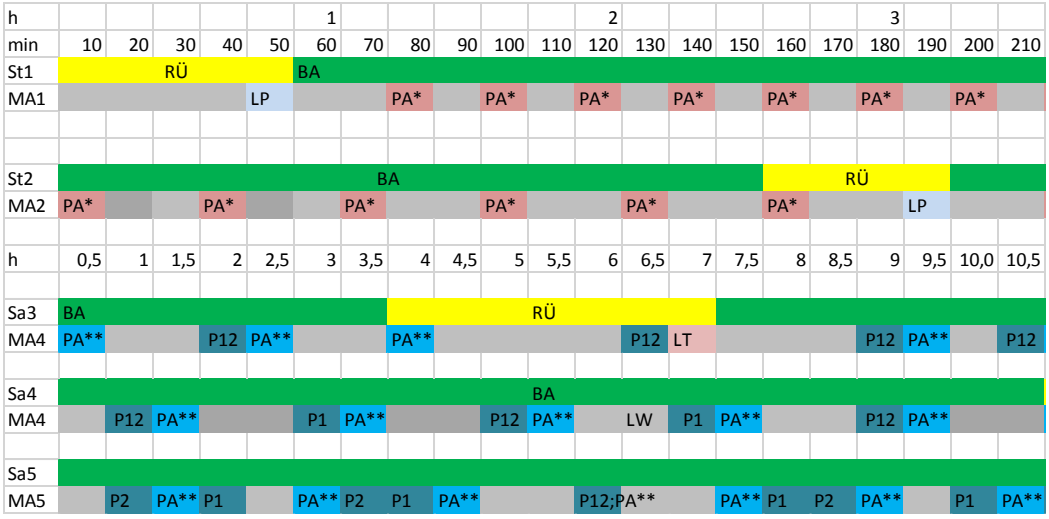

Figure 1: machine and worker view

Figure 1 illustrates the machine view (first bar) and the worker view (second bar) on a small example with the aim to give an overview of the complete problem. The machine view (ST= punching machine, SA= welding machine), shown on the upper bar shows that the machine is almost the whole time working (BA) or setup (RÜ) for the next assignment. The setup is not done by the general workers that are scheduled to the tasks, so it is not taken under consideration when solving the worker scheduling problem. In the worker view (MA) the grey fields are empty times (gaps), where the worker does not have a task to fulfill, whereas the colored fields are tasks to fulfill.

The remaining work is as follows: Chapter 2 is composed of a literature review on worker scheduling problems, a summary of vehicle routing problems and heuristics to solve the problem, as well as an introduction to multi-objective optimization. Chapter 3 introduces the basic mathematical model for the considered problem and suggests two extensions for the consideration of workload-balancing. A random construction heuristic, an improvement heurictic based on cheapest insertion as well as a workload-balance heuristic are suggested in chapter 4. In chapter 5 the computational results are shown based on self-generated random test-data and chapter 6 ends with a conclusion.

2. Literature review

The second chapter involves a literature review on models and algorithms, on which this work is based, as well as a summary of the recent literature.

2.1. Vehicle routing formulations

The worker scheduling problem at hand is solved as a vehicle routing problem (VRP) with the extension of time windows (VRPTW), therefore the basic VRP model is presented as well as the formulation of the VRPTW. Other extensions are also taken under consideration.

The VRP is among the most studied optimization problems of transportation logistics with the aim to find optimal routes to serve all customers i with given vehicles k. It was introduced in 1959 by Dantzig and Ramser under the name "the truck dispatching problem". (Dantzig and Ramser 1959). The VRP with only one vehicle, which has to find a least-cost solution including all nodes is called the travelling salesman problem (TSP). It is noticeable that the VRP is also often referred to as the vehicle routing and scheduling problem., e.g. in El-Sherbeny (2010) This formulation shows that many scheduling problems are formulated as a VRP.

In subchapter 2.1.1 the basic vehicle routing problem is introduced, which is extended with time windows in chapter 2.1.2. The last subchapter about vehicle routing models (2.1.3) mentions other commonly used extensions.

2.1.1. Vehicle routing problem

The main formulation considers the distribution of goods from its depot to its customers and back to the depot. It is formulated on a complete graph with the vertex set V and the arc set A. The arcs can be directed or indirected, depending on the symmetry. In case that the distance from customer i to customer j is the same as from j to i for every i and j, the distance matrix is symmetric and the arcs are undirected. Usually the triangle inequality is assumed to be satisfied, which means that it is not attractable to differ from the direct link between two vertices i and j. In mathematical terms it is described as follows:

$$c_{ik} + c_{kj} \geq c_{ij} \; \forall i, j, k \; \in V$$

As shown in the upper formulation, instead of the distance matrix a cost matrix can also be taken under consideration. The vehicles might be fixed or unlimited and therefore defined through the considerations of the requirements. All vehicles start at the same depot and are identical in the basic model. The demand is assumed to be deterministic, known in advance and cannot be split. (Toth and Vigo 2001)

The VRP is formulated as an integer linear- programming model with the binary decision variables x_{ij}, which are equal to 1 if the arc from customer i to customer j is traversed and 0 otherwise and the depot is situated at node 0. The following sets and pararmeters are used:

N set of nodes, start depot 0
K set of vehicles
c_{ij} travel cost on arc (i, j)

The two-index formulation of the VRP based on Vigo and Toth (2001) is as follows:

VRP

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

subject to:

$$\sum_{i \in N} x_{ij} = 1 \qquad \forall\, j \in N \setminus \{0\} \tag{2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall\, i \in N \setminus \{0\} \tag{3}$$

$$\sum_{i \in N} x_{0i} = K \tag{4}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \qquad \forall\, S \subseteq N \setminus \{0\}, S \neq 0 \tag{5}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i,j \in N \qquad (6)$$

The objective function (1) aims to minimize the variable travel cost. The constraints (2) and (3) ensure that one arc enters and leaves each customer, therefore it is made sure that every customer is visited exactly once. Constraint (4) imposes that all vehicles leave the depot. It is noticeable that this formulations forces to use all the available vehicles. There are other formulations that deal with a unfinite number of available vehicles, which can be looked up at Toth and Vigo (2001). The so called "capacity cut constraints" take care of the subtour elimination. The number of arcs that at least need to leave the subset is a lower bound and can be calculated as :

$$r(S) = \left\lceil \sum_{i \in S} \frac{d_i}{C} \right\rceil$$

C is the vehicle capacity and d_i the demand of customer i (Dongarra 1998). Since the constraints have exponential cardinality many other subtour elimination constraints have been invented. Well known are the constraints by Miller, Tucker and Zemlin published in 1960. Achutan, Caccetta and Hill (1996) also invented subtour elimination constraints and used an algorithm based on branch and bound to solve the problem to optimality.

The remaining constraints (6) ensure that the decision variables are binary. This formulation is for the asymmetric case and can as well be used for a symmetric VRP, anyway it is not as efficient as the symmetric VRP formulation using an edge-notation that can be looked up at Toth and Vigo (2001). The presented VRP formulation is not the only one, further formulations such as the set partitioning formulation can be looked up at Toth and Vigo (2001). The VRP is an NP-hard problem (and therefore all extensions to the problem are as well NP-hard), for which reason many heuristics and metaheuristics have been developed.

2.1.2. Vehicle routing problem with time windows

Another important mathematical model for this work is the vehicle routing problem with time windows (VRPTW). The general formulation is extended by hard time windows during which the service needs to start. Violations are not allowed in comparison to the soft time windows, which incur penalty cost in case of a violation. The vehicle is allowed to arrive before the beginning of the time window, but then needs to wait with the service until the start of the time window.

Additional decision variables b_{ik} are needed, which declare the starting time of the service at customer i with vehicle k. The time windows $[e_i, l_i]$ are declared as the earliest starting time and the latest possible starting time for the service. Note that in this formulation, there is a start depot at node 0 as well as an end depot at node n+1. For a good overview the sets, parameters and decision variables are shown below:

N set of nodes, start depot 0, end depot n + 1

K set of vehicles

c_{ijk} travel cost on arc (i, j) of vehicle k

t_{ijk} travel time on arc (i, j) of vehicle k

d_i demand of customer i

C_k capacity of vehicle k

T_k maximum tour length of vehicle k

s_i service duration at customer i

e_i earliest start time of service at node i

l_i latest start time of service at node i

<u>decision variables</u>:

$$x_{ijk} = \begin{cases} 1 \text{ if arc (i, j) is traversed by vehicle k} \\ 0 \text{ otherwise} \end{cases}$$

b_{ik} time of vehicle k beginning service at node i

The mathematical model of the VRPTW is formulated with three indices as follows (adapted from El-Sherbeny, 2010):

VRPTW

$$\min \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ijk}x_{ijk} \tag{6}$$

Subject to:

$$\sum_{k\in K}\sum_{j\in N} x_{ijk} = 1 \qquad \forall i \in N \setminus \{0, n+1\} \tag{7}$$

$$\sum_{j\in N, j\neq 0} x_{0jk} = 1 \qquad \forall k \in K \tag{8}$$

$$\sum_{i\in N} x_{i,n+1,k} = 1 \qquad \forall k \in K \tag{9}$$

$$\sum_{j\in N} x_{ijk} = \sum_{j\in N} x_{jik} \qquad \forall i \in N\,\{0, n+1\}, k \in K \tag{10}$$

$$b_{jk} \geq b_{ik} + s_i + t_{ij} - T_k(1 - x_{ijk}) \qquad \forall i, j \in N, k \in K \tag{11}$$

$$b_{n+1,k} - b_{0k} \leq T_k \qquad \forall k \in K \tag{12}$$

$$e_i \leq b_{ik} \leq l_i \qquad \forall i \in N, k \in K \tag{13}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in N, k \in K \tag{14}$$

The objective (6) is transport cost minimization as in the basic VRP model. Node i needs to be left exactly once (7), so that every customer is visited once by a vehicle. Every vehicle must leave the start depot and return to the end depot. (8-9) Due to these constraints a vehicle that is

not used drives straight from the start depot to the end depot without visiting any customer. The flow conservation constraints (10) ensure that a customer that is visited is left again. The inequality constraints (11) declare that if vehicle k goes from customer i to customer j, then it cannot arrive at customer j before the starting time of service at customer i plus the traveltime. The complete tour of vehicle k may not exceed its maximum tour length (12). The beginning of the service at node i must be within its time windows (13)- therefore the combination of constraints (13) and (11) ensures the starttime feasibility.

The model presented is a basic version of the VRPTW. Another possible extension with time windows might be the usage of soft time windows, which incurre a penalty in case of violations. This makes the problem formulation and solution computation more difficult. Another possiblity regarding time windows is the implementation of multiple time windows per customer. (El-Sherbeny 2010). Load constraints might also be added to the model. Desaulniers, Lavigne and Soumis (1997) considered waiting time cost, which are incurred when a vehicle arrives at a customer before the beginning of the time window.

2.1.3. VRP extensions

An unlimited number of possible extensions exists, out of which the common used ones are presented in this subchapter.

When capacity restrictions of the vehicles are taken under consideration the model is called capacitated vehicle routing problem (CVRP). The VRP with Backhauls extends the CVRP in a way that the customer set is divided into two subsets- the “linehaul customers”, who demand the delivery of a certain quantity of a product- and the “backhaul customers”, who require a pickup of a certain number of a product. The basic model for the VRP with delivery and pickup considers each customer with a given demand of a product to be picked up as well as delivered (e.g. coke glas bottles) (Toth and Vigo 2001). Precedence constraints are given such that the pickup must be before the delivery, which is visualized in figure 2 based on Ghiani, Laporte and Musmanno (2013, p361).

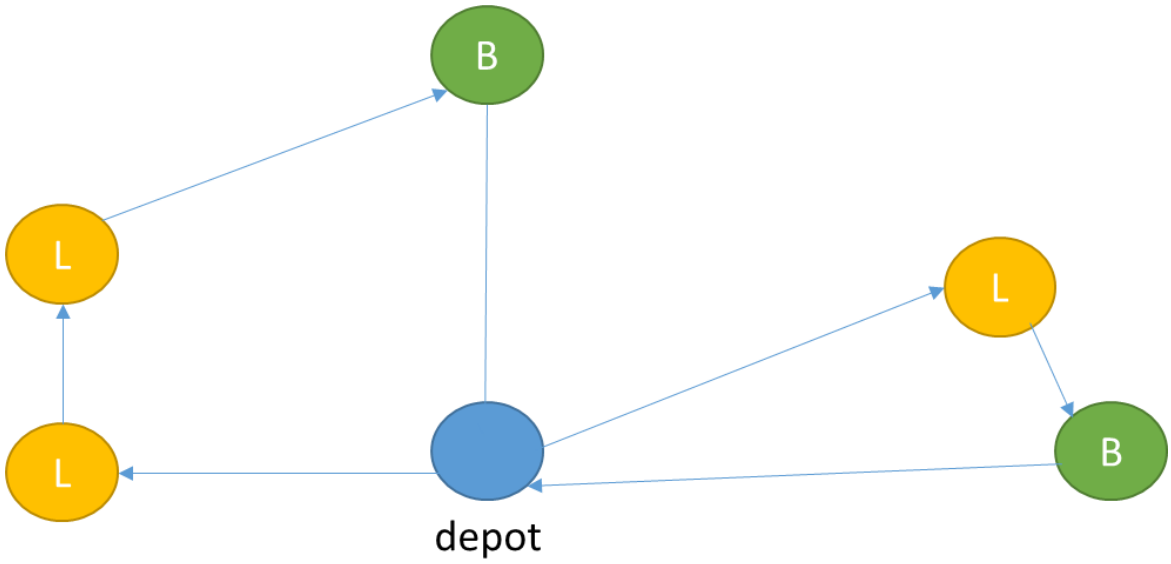

Figure 2: VRP with pickup and delivery

The Dial-a-Ride Problem consists of the determination of routes and scheduling customers, who have pickup and drop-off requests. The aim is to schedule as many people as possible. This model is often used for the transportation of elderly people. (Cordeau and Laporte, 2002)

Considering time aspects in addition to those of chapter 1.2.2 the travel time might also vary during the day. (Ghiani, Laporte and Musmanno 2013)

Furthermore a heterogenous fleet can be considered with vehicles that have different capacities, different types, different costs, or might need different travel times. (El-Sherbeny 2010)

In adaption of real-life scenarios instead of one depot serving all the customers there might be multiple depots. In Desaulniers, Lavigne and Soumis (1997) each depot has its own fleet, which satisfy the complete demand of the customers together.

The customers' demand might also be allowed to be satisfied by serveral vehicles in case of capacity restrictions, in this case the model is called VRP with split deliveries. The multiple use of vehicles is considered in Bräysy et al (2008).

Another version of the VRP assumes that the demand is stochastic and can be generated by some distribution function. Jalel, Adnan and Habib (2015) published a paper on the dynamic vehicle routing problem. The clients demands are not known in the beginning, but exposed after some decision have been made.

2.2. Heuristics for the vehicle routing problem

Since the VRP and its extensions are NP-hard problems, good heuristics are needed for the computation within reasonable time. These can be classified into the "classic heuristics", which are composed of construction heuristics and improvement heuristics, as well as metaheuristics, which aim is to improve an initial feasible solution by overcoming a local optima. The construction heuristics can again be subchategorized in "constructive heuristics", where a feasible solution is generated gradually and "two-phase heuristics", which has two components- the route construction and the clustering of nodes to routes. Nowadays many construction heuristics already have improvement steps within the algorithm, so the boundry between those two is blurred. (Toth and Vigo 2001)

This chapter outlines some well known heuristics with a focus on the ones being used in this work. The first three subchapters explain the basic constructive heuristic- the cheapest insertion procedure, which was used in this work and summarizes other algorithms (mainly two-phase heuristics). In the last subchapter 2.2.3 some important metaheuristic procedures are mentioned. All the heuristics are explained for the basic VRP-model, but can easily be adopted for its extensions, e.g. for the consideration of time windows a feasibility check needs to be added.

2.2.1. Construction heuristic: cheapest insertion

In this work the cheapest insertion method is applied as an improvement heuristic, although it is initially a construction heuristic. The heuristic is often referred to as "sequential insertion heuristic" and the basic idea is to determine the cheapest insertion for every vertex and every position. Moles and James sequential algorithm uses two parameter μ and λ for the selection of the insertion rules with the determination of the insertion cost:

$$\alpha(i,k,j) = c_{ik} + c_{kj} - \lambda c_{ij}$$
$$\beta(i,k,j) = \mu c_{0k} - \alpha(i,k,j)$$

The upper equation calculates the insertion costs- if vertex k is inserted between i and j, the according costs are calculated similarly to the savings cost. The equation beneath puts the distance of the vertex to be inserted to the depot under consideration. With the pararmeters the

insertion rules are chosen- for example with $\lambda = 1$ and $\mu = 0$, the vertex to be inserted is the one with the minimum extra distance.

An unassigned vertex k is chosen and the route (0,k,0) is built. Then for all unassigned vertices, the cheapest insertion (the minimum out of the $\alpha(i,k,j)$) between all vertices of the current route is determined and excerced. Afterwards the route is improved by a 3-opt. This procedure is done until no feasible insertion is possible. Then the next single-route is constructed with a vertex not yet assigned to a route and the next cheapest insertion movements are selected. The algorithm stops when all vertices are assigned. (Mole and Jameson 1976)

The sequential insertion algorithm by Chrisofides, Mingozzi and Toth (1979) uses a two phase procedure alternating between a sequential route construction and a parallel route construction. In the route construction phase, an initial route is created and then in the parallel route construction phase the unassigned vertices are assigned to the existing routes. If no feasible insertion is possible, a next initial route is constructed and so on. Within this phases the routes are always improved with a 3-opt algorithm.

A problem of the cheapest insertion approach is that the last generated routes are usually of poor quality, since the last customers to assign are often disparsed. For this reason Potvin and Rousseau (1993) developed a heuristic that simultaneously constructs routes.

2.2.2. Other construction heuristics

Between the 1960s and the 1990s many heuristics were developed to solve the VRP problem. An excerpt of well known construction heuristics is given in this subchapter. Most of the heuristics are so called “two-phase heuristics”, whereas the two phases are composed of the forming of clusters and the route construction. There are two types of two-phase heuristics: the route first- cluster second heuristics and the cluster first- second heuristics.

Savings algorithm

The Clarke and Wright savings algorithm that was published in 1964 is among the most famous heuristics for the VRP. (Toth and Vigo 2001) First of all single routes (0,i,0) are created for

every node and the savings of merging two routes are computed. The savings from every node to another node are calculated as follows:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}$$

The savings determine how much cheaper it is to connect i with j compared to the prior situation (single routes). Afterwards the savings list is sorted in nonincreasing order. There are two versions of the algorithm: the parallel version and the sequential version. The parallel version begins with the biggest saving s_{ij} of the savings list and tries to merge these routes in case that a feasible merge is possible. Then the next biggest saving is considered and so on. The sequential version considers each route. The biggest saving to merge the considered route with another route is then taken under consideration. In case that a feasible merge is possible it is fulfilled. This routine is repeated until no feasible merge is possible for the current route. Then the next route is chosen for the procedure. (Clarke and Wright 1964)

According to Toth and Vigo (2001) the parallel version dominates the sequential one. The savings heuristic has the disadvantage that good routes are mainly constructed at the beginning, but the lost constructed routes are not that promising.

Cluster first-route second heuristics

In the Fisher and Jaikumar algorithm, a generalized assignment problem (GAP) is solved. First of all seed vertices are initialized, the allocation costs to the seed vertices are calculated (similar to the savings in the savings algorithm) and the GAP is solved to distribute the vertices to the clusters. Afterwards the routes within the cluster are constructed by solving a TSP. Note that for this heuristic the number of vehicles needs to be fixed. (Toth and Vigo 2001)

The Sweep Algorithm is a simple heuristic that can only be applied on planar instances. In the heuristic clusters are formed by rotating in a circle beginning at the depot and assigning the vertices to the cluster. If the capacity is not enough to assign a vertex to a cluster, a new cluster is opened. The routes are then again constructed within the clusters by solving a TSP. (Toth and Vigo 2001) This was extended in the petal algorithm by Baliski and Quandt (1964). The

approach is to generate a set S of various routes, out of which a few are selected by solving a set partitioning problem:

$$\min \sum_{k \in S} d_k x_k$$

Subject to:

$$\sum_{k \in S} a_{ik} x_k = 1 \qquad \forall\, i\ \{0\}$$

$$x_k \in \{0,1\} \qquad \forall\, k \in S$$

The binary decision variable x_k is equal to one if route k is chosen in the solution. The parameter d_k gives the distance or the costs of the street. Various forms of the sweep and petal algorithm have been proposed since then, for example in Foster and Ryan (1976) or Renaud, Boctor and Laporte (1996).

Route first- cluster second heuristics

The heuristics first construct an infeasible giant TSP- tour and then divide this tour into feasible vehicle routes (clustering). This part is often solved with the shortes path algorithm by Dijkstra (1959).

2.2.3. Improvement heuristics

As the name identifies the improvement heuristic tries to enhance the feasible solution by searching for a better one in its neighborhood, which is "a set of solutions that can be generated with a single modification of" the solution. (El-Sherbeny 2010) This method is called local search, since it is only able to find the local optimum, which might also be the global optimum. The improvement heuristics are subdivided into single-route improvements, which only consider changes within a route and multiroute improvements.

Single-route Improvements

Single-route improvements are procedures that were developed for the TSP, but can as well be applied on the VRP for the improvement of each route separately.

The λ-opt exchange was introduced by Lin (1965). The idea is to remove λ edges and replace these by λ additional edges that reconstruct a feasible route. When an improvement is detected, it is implemented and the procedure starts afresh. There are two possible ways for the implementation: best fit and first fit. Meanwhile the first-fit option implements straight the first improvement that can be found and continues with the next iteration, the best-fit tries out all possible exchanges and takes the best improvement found for the next iteration. The heuristic stops when no improvement can be found, then the found solution is a local optimum. The most applied λ-opt heuristics are the 2-opt and the 3-opt. (Toth and Vigo 2001) For computational reasons, limited versions of this procedure have been introduced as the Or-opt (Or, 1976), which removes 1, 2 or 3 consecutive vertices and replaces them at another place and the 4*-opt (Renaud, Boctor and Laporte, 1996), which considers good connections between a chain with two edges and another one with at most k edges.

This procedures are implemented within lots of construction heuristics to immediately improve the current solution. (Toth and Vigo 2001)

Multiroute-improvements

The improvement of a feasible solution of the VRP is usually more promising if movements between different routes are considered. Several authors have developed multiroute improvement heuristics with different exchange schemes. To mention a few- Thompson and Psaraftis (1993), Van Bredam (1994) and Stewart and Golden (1984) have developed well used exchange schemes. The improvement schemes of Van Bredam are shortly described and illustrated beneath:

String cross: “two strings of vertices are exchanged by crossing two edges of two routes”

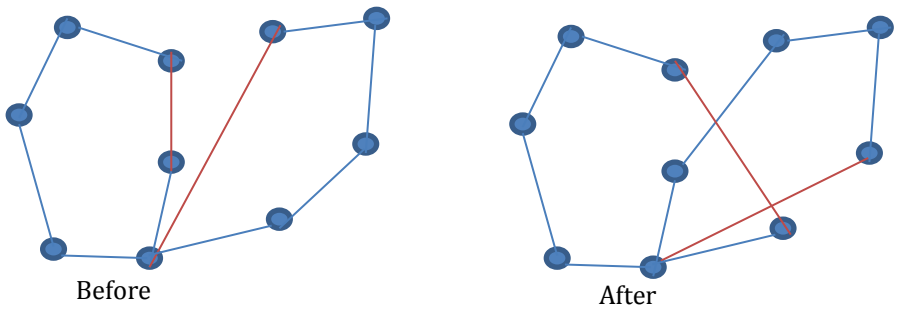

Figure 3: string cross

String exchange: “two strings of at most k vertices are changed between two routes”

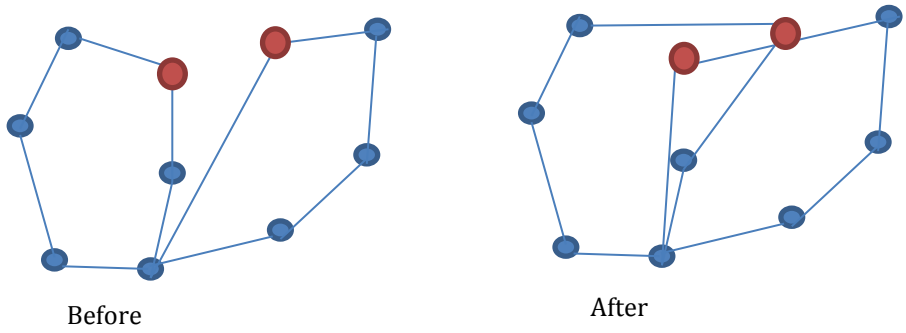

Figure 4: string exchange

String relocation: k concesutive vertices are moved to another route, where k is usually 1 or 2

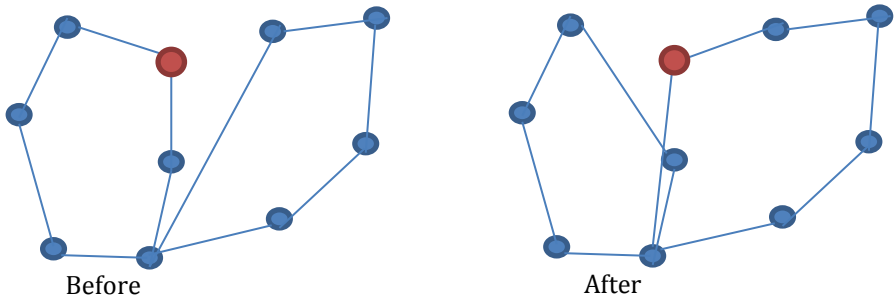

Figure 5: string relocation

String mix: The string exchange and the string relocation are taken under consideration, out of which the best movement is chosen.

Thompson and Psaraftis (1993) invented a heuristic with a b-cyclic k transfer. A number of customers k from every route are shifted to the next route of the cyclic permutation, whereas b is the number of considered routes.

2.3. Multiobjective optimization

The basic models presented in chapter 2.1 only have a single objective, which is cost (or distance) minimization. However, in real life there are usually more objectives, e.g. economic requirements or pollution concerns that must be taken under consideration, therefore multi-

objective optimization procedures are needed. This subchapter discusses therefore some important solution techniques for the multi-objective optimization.

For the multi-objective optimization, there is not a single optimal solution, but a complete Pareto front, which is a set of efficient solution. A solution is Pareto optimal "if all other vectors $x \in S$ have a higher value for at least one of the objective functions". (Carmia and Dell'Olmo 2008) It can be observed that all the solutions on the Pareto front dominate the other possible solutions in the grey solution space. The complete Pareto front can hardly be computed due to its exponential size.

After the generation of the Pareto-front the decision maker decides which option to choose. Anyway, in real-life problems it is difficult for the decision maker to have the complete information correctly. (Carmia and Dell'Olmo 2008)

The extended worker scheduling problem considered in this work has three objectives: the minimization of workers, the equal distribution of the tasks as well as gap minimization. For this case the three objectives do not have the same priority. The focus is on the worker minimization, meanwhile the other objectives should also be taken under consideration without increasing the worker costs.

Two well used techniques for the multiobjective are the scalarization technique and the ϵ-constraints method, which are shortly introduced. More sophisticated methods are mentioned at the end of this subchapter.

The scalarization Technique

The method for multi-objective optimization that is also referred to as weighted-sum method is a widely often approach, where the various objective functions are combined into a single one. The mathematical formulation for this method is given below:

$$\min \sum_{i=1} y_i f_i(x)$$

$$\sum_{i=1} y_i = 1$$

$$y_i \geq 0, \quad \forall i$$
$$x \in S$$

The objectives i go from {1..n} and S is the set of constraints. The complete objective function weights the single objectives f_i with the weights y_i, which in sum must equal to one. Furthermore the weights are non-negative, so in case that a weight is 0, this part of the objective function is not considered. The decision maker is determined to choose appropriate weights that correspond to the relative importance of the objective funtion. (Carmia and Dell'Olmo 2008) This method was chosen in this work.

ϵ- constraints Method

Another method to solve multi-objective optimization problems is the ϵ- constraints method. One main objective is chosen and optimized, meanwhile the other objectives are transformed into constraints. These resulting constraints should not exceed a certain value. (Carmia and Dell'Olmo 2008) In mathematical terms the method can be described as follows:

$$\min f_j(x)$$
$$f_i(x) \leq \epsilon_i, \forall i$$
$$x \in S$$

The upper bounds ϵ_i for the constraints need to be chosen adequately. In contrast to the scalarization technique, the ϵ- constraints method "is able to achieve efficient points in a non-convex Pareto curve". (Carmia and Dell'Olmo 2008)

Other methods

Goal Programming is a different method that in contrast to other methods views the problem from the goal perspective. The aim is to reach a certain targeted goal, rather than aiming to optimize. Various variants exist such as the multi-criteria goal programming, the lexigographical goal programming and the interactive goal programming, which can be looked up at T'Kindt and Billaut (2006).

In the multi-level programming method the seperate objectice functions are sorted according to its importance with the most important being at the first place. Then the minimizers of the first objective are found, then of the second and so on. This procedure finds an optimal point of the Pareto front. The method computes good results if the hierarchical order among the objectives is meaningful. (Carmia and Dell'Olmo 2008)

2.4. Worker scheduling literature review

Worker assignment problems were first considered in research in the 19th century. (Dolgui, et al. 2017). Various versions can be found in literature that deal with the allocation of workers to shifts, operations or workstations. There is no preamble for worker assignment problems, but there exist rather various terms. Some important worker scheduling problems definitions are staff scheduling, workforce scheduling or workforce planning, labour or personnel scheduling as well as rostering problems (Rocha, Oliveira and Carravilla 2012). Since the beginning of the 21st century constraints were added continuously, such as -capacity and resource constraints as well as constraints for timetabling and rostering (Dolgui, et al. 2017).

The case when workers have different speed to fulfill a task is considered in the Assembly Line Worker Assignment and Balancing Problem introduced by Miralles in 2007. This problem has particularly been used for the integration of disabled people to assembly lines. (Araújo, Costa and Miralles 2012). But it is also of relevance for a general scheduling problem to assembly lines, which "are manufacturing systems in which a product is assembled progressively in workstations by different workers or machines, each executing a subset of the needed assembly operations (or tasks)" (Moreira, et al. 2015). An example for an assembly line scheduling problem in production is given in a case study by Battaia et al (2015), who researched on a mixed-model assembly line in the automotive industry.

A worker assignment problem that considers uncertain and worker-dependend processing times on an assembly line is considered in Moreira et al (2015). The aim is to assign the heterogenous worker in a way that the number of stations is minimized.

A recent paper published in 2017 by Dolgui et al studied a paced assembly line, which is able to produce different products. The processing time of the operations depends on the number of workers that are assigned to this operation. The aim is to minimize the number of workers assigned simultaneously.

Another assembly line problem takes worker qualifications and operations requirements into account. The workforce has hierarchical order in a way that a higher qualified worker is able to fulfill all the task a lower worker can do and can therefore represent him, but not vice versa. (Sungur and Yavuz 2015)

Corominas et al (2008) differentiates between permanent and temporary workers with the aim to minimize the number of temporary workers.

In many cases the fulfillment of the operations' demands experience and skills, which needs to be considered in the problem. (Mutlu, Polat and Supciller 2013) A recent review of worker assignment problems with heterogeneous workers is given in De Brueckner et al (2015).

An assembly line with various workstations whose operations require different amounts of workers that are fixed is studied in Camm et al (2008). The objective is to minimize the workforce.

The workload is often distributed inequal to the workstations. Therefore the balancing of workload to workstations has been studied in literature. (Venkatesh 2008) Diverse metaheuristics, such as a variable neighborhood search (Polat et al 2016) or a genetic algorithm (Mutlu, Polat and Supciller 2013) were developed to solve the worker assignment and balancing problem.

As can be observed in the literature research, the tasks usually have precedence relations and are often assigned to a workstation beforehand. The consideration of fixed time windows for the tasks, which are then assigned to workers was not found in literature. Therefore this work contributes to the research of worker assignment problems with fixed time windows and workload balancing considerations.

Due to my knowledge, there is no such literature that considers the basic case with equally skilled workers.

3. Mathematical Models

The mathematical models are based on the VRPTW and aim to minimize the worker costs, therefore the number of workers used should be as small as possible. Within a given time period a given number of tasks located at different machines need to be done within a given time frame by as less workers as possible. This must be done in a way that all the operational constraints are satisfied. Note that the maximum number of workers that might be used is implicitly given by the number of machines, since it must always be possible that one worker fulfills the tasks on one machine.

In chapter 3.1 the basic worker scheduling model is introduced, meanwhile in chapter 3.2 the model is extended to ensure workload-balancing including the avoidance of big gaps (empty time between tasks where worker is not scheduled).

3.1. Basic model

The worker scheduling problem with a fixed job sequence and time windows (WSP) is formulated as an integer problem on a complete graph. Each task is considered as a node, which has to be visited and has a given service time, which is assumed to be known and the same for every worker.

The sequence of the tasks is given and the assignment of the tasks to the machine was made a priori. In case that task i needs to be before task j on the same machine then its hard time window is before the following task without any overlap. A task needs to be fulfilled by one worker and the processing time is deterministic and the same for every worker, who are homogeneous, so it is assumed that every worker has the same skills and is able to fulfill all tasks. If a worker is assigned to a task, the task needs to be completed before the worker can move to another task. A worker cannot fulfill more than one task simultaneously. The travel time from one task to another occurs only if the worker has to change the machine. The workers have to fulfill one task after another within their given hard time windows.

Since the model is formulated as a VRPTW, an artificial depot at node 0 and an enddepot at the last node n+1 is introduced. The resulting scheduling problem consists of determining a start time for each task and an assignment of workers to the task in order that the number of workers is minimized.

The parameters and decision variables are as follows:

N set of tasks , start task (“depot”) 0, end task (“enddepot”) n+1

K set of workers

h_k worker cost

e_i earliest start time of task i

l_i latest start time of task i

St_i service time of task i

tt_{ij}travel time from task i to task j

M big number

decision variables

$$z_k = \begin{cases} 1 \text{ if worker k is used} \\ 0 \text{ otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 \text{ if worker does job j after i} \\ 0 \text{ otherwise} \end{cases}$$

st_{ik} time worker k begins service at task i

The mathematical model for the WSPTW is formulated as follows:

WSPTW

$$\min \sum_k h_k z_k \tag{1}$$

Subject to:

$$\sum_k \sum_j x_{ijk} = 1 \qquad \forall\, i \in N, i \neq 0, j \in N, k \in K \tag{2}$$

$$\sum_{i,i\neq n+1} x_{i,n+1,k} = z_k \qquad \forall k \in K \qquad (3)$$

$$\sum_{j,j\neq 0} x_{0jk} = z_k \qquad \forall k \in K \qquad (4)$$

$$\sum_{j,j\neq i,j\neq 0} x_{ijk} = \sum_{j,j\neq n+1,j\neq i} x_{jik} \qquad \forall i \in N, \forall k \in K \qquad (5)$$

$$\sum_{i}\sum_{j} x_{ijk} \leq z_k M \qquad \forall k \in K \qquad (6)$$

$$st_{jk} \geq st_{ik} + St_i + tt_{ij} - M(1 - x_{ijk}) \qquad \forall k \in K, i \in N, j \in N, i \neq j \qquad (7)$$

$$e_i \leq st_{ik} \qquad \forall i \in N, k \in K \qquad (8)$$

$$l_i \geq st_{ik} \qquad \forall i \in N, k \in K \qquad (9)$$

$$z_k \in \{0,1\}, x_{ijk} \in \{0,1\} \qquad \forall i \in N, j \in N, k \in K \qquad (10)$$

$$st_{ik} \geq 0 \qquad \forall i \in N, k \in K \qquad (11)$$

The objective function (1) aims to minimize the worker cost, which implies the minimization of the worker. In the first constraints (2) it is ensured that every task is done exactly once by one worker. Every worker that is used in the solution needs to leave the depot (4) and return to the depot (3). Furthermore these constraints guarantee that the worker is only assigned to one "tour".The flow conservation constraints (5) ensure that after fulfilling a task the worker leaves the task and goes to the next task or the depot- so that one complete tour is developed. A task can only be assigned to a worker that is used in the solution (6). Constraints (7) take care of the starttime feasibility. If task j is scheduled right after task i for worker k, then its starttime needs to be feasible, which implies that it must be at lowest when task i is finished plus the traveltime

from i to j. Otherwise (if task j is not scheduled after i) the problem is relaxed. The starttime of a task needs to be within the time window (8 and 9). The remaining constraints state that x_{ijk} and z_k are binary constraints and the starttime is non-negative.

As the VRPTW, the WSPTW is also NP-hard and has exponential cardinality. Since the job sequence is fixed and the time frame is hard, many decision variables can be eliminated. The small example in figure 6 illustrates 7 tasks that need to be assigned on two machines. The number of x_{ijk} variables for this problem is 7*7*2=98 (without counting the depot). Indeed many decision variables are unnecessary, it is for example not possible to go from task 7 to task 1, 2, 3, 5 or 6. So the only necessary decision variables from task 7 are x_{741} and x_{742}. Furthermore it does not make sense to go from a task to itself, which is usually taken care of in the constraints, but the variables would not need to be created at first place. With the elimination of the unnecessary variables, the number of x_{ijk} variables for this example can be reduced to 40.

time	1	2	3	4	5	6
Machine 1	task 1	task2		task3		task4
Machine 2		task5	task6		task7	

Figure 6: illustration of decision variable elimination

It needs to be notified that there are many optimal solutions concerning the assignment of the tasks. The only costs that are considered in the objective function are the worker cost. It does therefore not make a cost difference to which worker which task is assigned. There is, for example the possibility that one worker is only assigned to one task, meanwhile another worker has a much higher workload. For this reason the model needs to further consider workload balancing.

3.2. Extensions with workload-balancing and gap minimization

In this subchapter the worker scheduling problem is extended by the consideration of workload balance. This implies that every worker should have about the same workload, which means that the sum of service times of the tasks scheduled to one worker should be as equal as possible.

A further extension is the minimization of the gaps (empty time) from the beginning of the runtime till the first assigned task, between the tasks, and after the last task till the end of the run time. The reason for this extension is that there should be a balance of assigned tasks- a worker should not be assigned to many tasks in the beginning and then have a long empty time, but more short breaks are efficient. In addition this consideration ensures that there is a small buffer in between the tasks, so there are no shortcomes in case that a task takes unexpectatly long.

The method to solve the resulting multiobjective optimization problem that was chosen is the integration of weighted multiple objectives into one single objective function. The reason for the choice is that the main goal is the minimization of the worker, which is assigned to a high weight and the other two objectives have minor priority.

For the workload balancing, two mathematical models were formulated. The model considered in chapter 3.2.1 aims to minimize the maximum workload, whereas the model presented in chapter 3.2.2 has the objective of minimizing the average workload deviation from the average service time per worker.

3.2.1. Minimizing the maximum workload and the maximum gap

Workload-balancing should be reached by minimizing the maximum workload of a worker. The gap extensions are equal in both model extensions and presented in this subchapter.

The parameter w is a given weight to integrate the different objectives into one objective function. The following decision variables need to be added for the formulation of the extended model:

w_k workload of worker k

W_{max} maximum workload

G_k^{max}maximum gap of worker k

G_{max} maximum gap

Whereas gap is the empty time, where the worker is waiting to do the next task- so the time he neither fulfills a task nor goes to the next task. The objective function for this problem needs to be adopted to:

$$\min \; w \sum_k h_k z_k + \left(1 - \frac{w}{2}\right) W_{max} + \left(1 - \frac{w}{2}\right) G_{max}$$

This multiobjective function aims to minimize the maximum workload of a worker and the maximum gap in addition to the main worker minimization goal.

Furthermore the following additional constraints to the constraints (2-11) of the basic WSPTW need to be added to the model:

$$W_k = \sum_i \sum_j x_{ijk} St_i \qquad \forall k \qquad (12)$$

$$W_{max} \geq W_k \qquad \forall k \qquad (13)$$

$$G_k^{max} \geq st_{jk} - st_{ik} - ST_i - tt_{ij} - M(1 - x_{ijk}) \qquad \forall i, j, k \qquad (14)$$

$$G_{max} \geq G_k^{max} \qquad \forall k \qquad (15)$$

$$G_k \geq 0 \qquad \forall k \qquad (16)$$

Constraints (12) calculate the workload of each worker as the sum of the service times of the assigned tasks. The maximum workload, which is minimized in the objective function, is bigger than the workload of each worker (13). Constraints (14) determine the maximum gap per

worker between task i and j in case that task j is scheduled after task i for worker k. The maximum gap G_{max}, which is minimized in the objective function, is the biggest gap among the G_k^{max}decision variables. The last additional constraints (16) ensure the non-negativity of the gap per worker.

3.2.2. Minimizing the average workload deviation and the maximum gap

The second model extension aims to minimize the summed up workload deviations of all used workers from the average service time per used worker. This implies that the workload of all the workers in use is about the same, since it approaches the average service time. In addition to the basic model the following parameters are added:

P set of possible worker levels
$W_k$workload of worker k
ST_p^{avg} average service time per worker for level p
w weight

The amount of workers that are assigned to tasks is not known before solving the model. Therefore the set P, which is the number of used workers, is introduced. The average service time per worker is calculated beforehand for different levels. In addition to the decision variables of the basic model the following decision variables are added:

W_k^{abs} absolute workload deviation from median service time
$y_p$binary decision variable decading whether level p worker are used
G_k^{max}maximum gap between two tasks of worker k
G_{max} maximum gap between two tasks

The decision variables for the gaps are the same as in the min max extension model. The binary decision variables y_p determine the number of workers used- if, for example y_p=1, there are 5 workers used in the solution. The objective function of this model is:

$$\min\ w\sum_k h_k z_k + \left(1-\frac{w}{2}\right)\sum_k W_k^{abs} + \left(1-\frac{w}{2}\right)G_{max}$$

The first main part of the objective function minimizes the worker cost. The second part minimizes the summed up workload deviations and the third part minimizes the maximum gap. The weights are chosen in a way that the focus is on the first part. The additional constraints to the basic model are:

$$W_k^{abs} \geq \mathrm{ST}_\mathrm{p}^{\mathrm{avg}} - W_k - M * (1 - y_p) - M * (1 - z_k) \qquad \forall k \in K, p \in P \tag{17}$$

$$W_k^{abs} \geq W_k - \mathrm{ST}_\mathrm{p}^{\mathrm{avg}} - M * (1 - y_p) - M * (1 - z_k) \qquad \forall k \in K, p \in P \tag{18}$$

$$\sum_p y_p = 1 \tag{19}$$

$$\sum_p p y_p = \sum_k z_k \tag{20}$$

$$G_k \geq st_{jk} - st_{ik} - ST_i - tt_{ij} - M(1 - x_{ijk}) \qquad \forall i \in N, j \in N, k \in K \tag{21}$$

$$G_{max} \geq G_k \qquad \forall k \in K \tag{22}$$

$$G_k \geq 0 \qquad \forall k \in K \tag{23}$$

$$y_p \in \{0,1\} \qquad \forall p \in P \tag{24}$$

Constraints (17) and (18) determine the absolute workload deviation from the average service time per worker, in case that p worker are used. In case that worker k is not used or the number of worker does not equal p, the constraints are relaxed. The number of workers needs to be fixed to one value, therefore only one level p can be used (19). The level p must be equal to the sum of used workers, which is ensured in constraint (20). The constraints for the gaps (21-23) are the same as in the first extension model. Finally, the decision variables for the number of workers y_p are binary (24).

4. Heuristics

Since the WSPTW and its extensions is NP-hard, heuristics were developed. In 4.1 two construction heuristis are presented, which are improved with a cheapest insertion heuristis introduced in chapter 4.2. Afterwards a further heuristic shifts the task in a way that workload-balancing and gap minimization is reached. The latter heuristic is included in chapter 4.3.

4.1. Construction heuristics

Both construction heuristics that are presented in this chapter generate feasible starting solutions, which are afterwards improved with a variation of the cheapest insertion heuristic.

The first construction heuristic simply assigns one worker to one machine, therefore all the tasks on one machine are assigned to the same worker. This is the actual situation at ILL, which is of cause very costly and results in low workload per worker.

The second construction heuristic randomly chooses one task out of the set of next tasks, which is then assigned to the cheapest possible worker. More precisely the heuristic uses the following sets:

S… set of schedulable tasks
S^n… set of next tasks
W… set of workers sorted according to its cost in ascending order

S is denoted the set of schedulable tasks and contains the next tasks to be assigned to the workers. The set S^n is used to determine the next tasks to be inserted in the set S. The heuristis works as follows:

put first task of each machine in S

do

{

randomly choose a task out of S &assign to cheapest worker of W possible

add task after recently assigned task to S^n

if (starttime of task in S^n < starttime of one task in S)-> move task to S

if (S is empty)-> move all tasks from $S^n$to S

}

until S^n& S are empty

The heuristic terminates when all tasks are assigned. The tasks are assigned to the cheapest worker possible, which means that the worker does not have a task scheduled that finishes before the latest possible starttime. In comparison to the first simple construction heuristic, the random heuristic hardly uses all the worker, but leaves out the most expensive ones.

4.2. Improvement heuristic

After the computation of the construction heuristic, the feasible solution is improved by a cheapest insertion heuristic. The main idea is to reduce the number of used workers by moving its tasks to the other assigned workers.

The first step in the heuristic is to choose the worker, whose assigned tasks should be moved to other used tasks. Usually the worker is chosen according to its cost, so the most expensive worker with assigned tasks is eliminated if possible. Another possible criteria, that was implemented is to choose the worker with the least assigned tasks. A worker is only chosen once for elimination.

```
n=0
do
{
choose worker to eliminate
for every task in eliminate worker:
        {
        For every used worker and every position
                {
                 determine gap between inserted task and task before, if feasible insertion possi-
                 ble
                }
                Store best insert place and starttime updates
        }
        if all tasks can be moved→ do changes and eliminate worker
n++
}
until n=N
```

Then for every task of the worker to be eliminated the best place to move is determined. This is done by first checking if a feasible insertion before, between or after assigned tasks of a used worker is possible. An insertion is feasible if the task to be inserted can start after the task before and the task after the inserted task can still be scheduled in a feasible way within its time window. In case of a feasible possible insertion the gap between the task before the task to be inserted and the latter is calculated as follows:

Latest starttime (inserttask)- [starttime(task before)+ servicetime(task before)+traveltime to inserttask}]

The gaps, insert places and starttime updates are stored and in case that all tasks of the worker to be eliminated can be moved to other used worker, the movements are fulfilled. The heuristic stops when a given number of iterations N is reached.

4.3. Workload balancing and gap minimization heuristic

At last the workload balancing and gap minimization heuristic is used to improve the cheapest insertion heuristic regarding the workload and gap. The idea of the heuristic is still to use a variant of the cheapest insertion heuristic. In general the goal of the heuristic is to move tasks in a way that an equal workload is reached. The description of the heuristic is as follows:

```
n=0
do
{
select highest worker and lowest worker
for every task i of highest worker
        {
        if ( insertion feasible && gap not increased for lowest worker)
        move i to lowest worker
        else if (insertion feasible && gap not increased for 2ndlowest worker)
        move i to 2nd lowest worker
        else if else if (insertion feasible && gap not increased for 2ndlowest worker)
        move i to 3rd lowest worker
        else break;
        }
n++
}
while n<=N
```

In general the heuristic aims to move a task of the worker with the highest workload to the worker with the lowest workload. The move is only done if a feasible move is possible (the feasibility criteria is the same as in the cheapest insertion heuristic described in the last sub-chapter) and the new gap that is developed when moving a task from the worker with the highest workload is not greater than the gap than the new gap of the worker with the lowest workload. In case that with this criteria the movement of a task from the worker with the highest workload to the worker with the lowest workload is not possible, the movement is tried to the worker

with the second lowest workload. If that is still not possible the movement is lastly tried to the worker with the third lowest workload. The heuristic terminates after a given number of iterations.

5. Computational Results

The following sub-paragraphs discuss and compare the different mathematical models and diverse combinations of heuristics according to various criterias. The data generation is described in the first subchapter, chapter 5.2 compares the basic mathematical model with the various combinations of the proposed heuristics and subchapter 5.3 considers the workload balancing extensions.

5.1. Generation of test data

Since there is no real data, test data for this work was generated in C++. Based on an Excel table of real data, the data was generated randomly with the assumption of a normal distribution for the times of the diverse tasks, such as packaging, bringing a pallet, putting a full pallet to the storage place, removing empty pallets and so on. The diverse subtasks were cumulated into a pickup or a delivery task. The working times of the machines are again assumed to be normal. In this work the test data is generated based on the ILL case with two different machines, a welding machine and a cutting machine, which have quite different distributions.

Since the machines are close to each other, the traveltime from one machine to another is assumed to be two and from one task to another on the same machine it is zero.

The time windows are tight with a span of six minutes, meaning that the latest starting time of a task is six minutes after its earliest starting time. The idea is that some tasks need to be done urgently since it is costly when the machine is out of work. Nevertheless some tasks, as the storage of the full pallets actually have more time, since the machines have two nests.

Also the worker costs are generated randomly, with 2000€ being the lowest possible cost. It is clear that the most expensive worker is never used, meanwhile the cheapest worker is always used. This is of low importance, since the number of wokers needed and the task assignment to the workers is the required information. The tasks can easily be moved to another worker without any effort. In contrast when the model is extended with a quality matrix, which indicates which worker is able to do which task, the assignment of the tasks to the worker matters.

5.2. Heuristic vs optimal solution basic model

The Gantt chart of the current situation at ILL, which is also used as construction heuristic 1, is illustrated in figure 7 and shows the inefficiency of using one worker per machine. One bar represents a task, which has to be fulfilled. The first number in the bar is the task number and the second number indicates the machine, on which the task needs to be done. The color of the bars represent the machines, e.g. all bars in yellow need to be fulfilled on machine 3.

In this example 38 tasks need to be fulfilled on 8 machines within 6 hours. The workload is the time the worker is busy with fulfilling a task (and walking which is not the case for this construction heuristic) and is given next to the labeling of the worker. It can be observed that worker 1 has the highest workload with 44% and is assigned to 10 tasks, meanwhile worker 6 has the lowest workload with only 5% and is only assigned to 2 tasks. This illustration demonstrates the high potential in reducing the number of workers needed. Furthermore the workload should be more balanced.

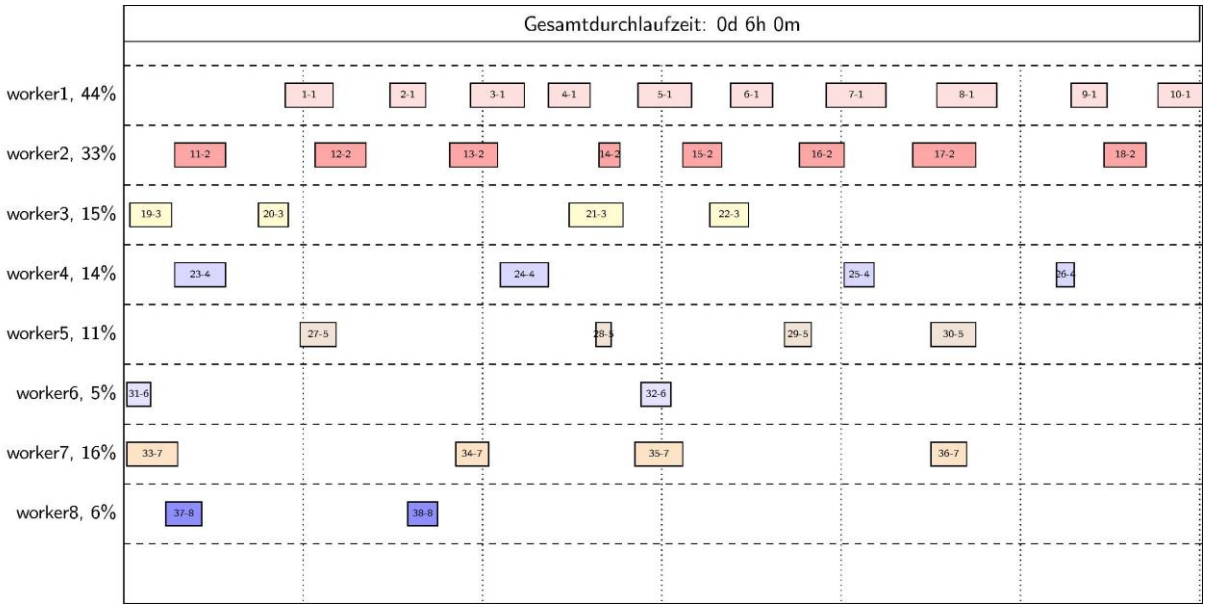

Figure 7: Gantt chart current situation ILL & basic construction heuristic

In comparison the same instance was solved to optimality by using the basic model, which is illustrated in figure 8. It can be observed that only 4 out of 8 available workers are needed to perform the required tasks. Furthermore the complete fulfillment of the tasks takes 6 minutes

longer. Anyway, the basic model does not take the workload under consideration, therefore it variates between 24% and 50%. Worker 7 has almost no breaks in the middle of the observed time, whereas the gap of worker 3 between his first and second assigned task is very large.

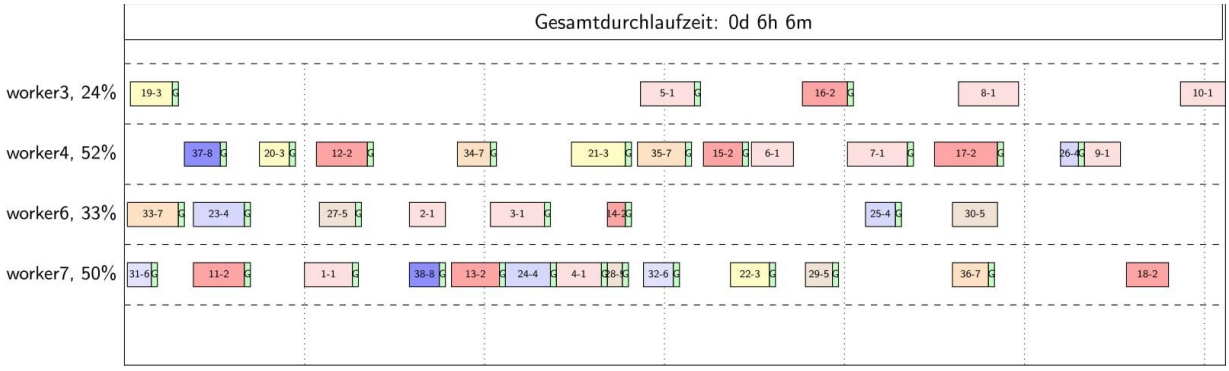

Figure 8: Gantt chart optimal solution

125 computations were made with the number of tasks varying between 24 and 291. In Table 1 the solution quality of the basic construction heuristic, the random construction heuristic and its extensions with the cheapest insertion improvement heuristic is compared to the optimal solution. The results show that the combination of the random construction heuristic with the cheapest insertion improvement heuristic yield the best resulte. Though, in about 20 % of the results the heuristic got stuck at a local optimum. For that reason a metaheuristic should be applied afterwards. The computation times are compared in table 2.

The solutions quality given in worker costs is compared between the optimal solution, the basic construction heuristic, the random heuristic as well as its application with the improvement cheapest insertion heuristic. The last rows demonstrate the gap to the optimal solution for the construction heuristics and its improvement. Table 3 shows the average gap of the 123 instances as well as the minimum and maximum gap. An impressive observation is that the current solution of ILL (basic construction heuristic) has an average gap of 53%, so with these instances half of its worker cost could be reduced. The best heuristic solutions are obtained by applying the random construction heuristic and improving it with the cheapest insertion. By applying the improvement heuristic to construction1, the heuristic gets more often trapped in a local optimum. The optimal solution was found at least once with every method.

It can be observed that the heuristic happens to get stuck at a local optimum, so the solutions might again be improved by a metaheuristic, e.g. a variable neighborhood search that destroys part of the solution and repairs it again.

	solution quality (cost)					solution gap (%)			
	math. model	heuristics				heuristics			
# tasks	optimal	b_constr	ran_constr	b_constr+cheap	ran_constr+cheap	b_constr	ran_constr	b_constr+cheap	ran_constr+cheap
151	15423,5	17683,4	15423,5	15423,5	15423,5	14,65231627	0	0	0
153	15510,1	17772,9	17772,9	15510,1	15510,1	14,58920316	14,58920316	0	0
153	15510,1	17772,9	17772,9	15510,1	17772,9	14,58920316	14,58920316	0	14,58920316
153	15510,1	17772,9	15510,1	15510,1	15510,1	14,58920316	0	0	0
153	10906	17627,8	15364	10906	10906	61,63396296	40,8765817	0	0
155	10951,4	17765,4	15469,6	10951,4	10951,4	62,22035539	41,25682561	0	0
155	10951,4	17765,4	13204,5	10951,4	10951,4	62,22035539	20,57362529	0	0
155	13213,2	17666	15431,4	17666	13213,2	33,6996337	16,7877577	33,6996337	0
155	13213,2	17666	15431,4	17666	15431,4	33,6996337	16,7877577	33,6996337	16,7877577
160	13148,7	17755,3	15440,7	15440,7	13148,7	35,03464221	17,43138105	17,43138105	0
163	8699,54	17703,6	13155,8	8699,54	8699,54	103,500415	51,22408771	0	0
167	8633,5	17524,1	15288,6	10834,3	8633,5	102,9779348	77,08461227	25,49139978	0
174	15592,3	17843,9	15592,3	15592,3	15592,3	14,440461	0	0	0
175	15558,4	17825,5	17825,5	17825,5	15558,4	14,57154977	14,57154977	14,57154977	0
175	11057,2	17865,7	13315,9	13315,9	11057,2	61,57526318	20,42741381	20,42741381	0
177	12948,7	17406	15175,6	15175,6	12948,7	34,42276059	17,19786542	17,19786542	0
180	10711,8	17337,2	15122,7	10711,8	10711,8	61,8514162	41,17795329	0	0
181	13242	17791,3	15487,8	15487,8	13242	34,35508231	16,95967377	16,95967377	0
181	13037	17533,2	17533,2	13037	17533,2	34,4879957	34,4879957	0	34,4879957
184	8757,05	17755,5	13193,5	8757,05	10972	102,7566361	50,66146705	0	25,29333508
184	11037,8	17797,8	15519,7	11037,8	11037,8	61,2440885	40,60501187	0	0
187	8677,6	17722,8	17722,8	8677,6	10883,3	104,2361943	104,2361943	0	25,41831843
191	8677,6	17722,8	13153,5	10883,3	10883,3	104,2361943	51,57992993	25,41831843	25,41831843
192	10782,9	17504,8	15207,1	10782,9	10782,9	62,33851747	41,02977863	0	0
192	10782,9	17504,8	15207,1	10782,9	10782,9	62,33851747	41,02977863	0	0
204	17365,1	17365,1	17365,1	17365,1	17365,1	0	0	0	0
212	12951,5	17394,4	17394,4	12951,5	12951,5	34,30413466	34,30413466	0	0
212	13284,1	17892,9	15563,4	17892,9	13284,1	34,69410799	17,15810631	34,69410799	0
212	13251,5	17752,4	15496,7	13251,5	15496,7	33,96521149	16,94298759	0	16,94298759
213	10977,1	17822,7	15529,1	13239,1	10977,1	62,36255477	41,46814732	20,60653542	0
214	13016,9	17478,7	17478,7	17478,7	13016,9	34,27697839	34,27697839	34,27697839	0
214	13216,1	17701,9	17701,9	17701,9	13216,1	33,94193446	33,94193446	33,94193446	0
218	13101,1	17589,9	17589,9	17589,9	15344,4	34,26277183	34,26277183	34,26277183	17,12298967
222	8696,36	17530,1	15291,3	13085,3	10886,6	101,5797414	75,83563698	50,46870185	25,18570988
227	13018,6	17570,2	15290,6	17570,2	15290,6	34,96228473	17,45195336	34,96228473	17,45195336
235	13294	17873,7	15575,8	13294	13294	34,44937566	17,1641342	0	0
235	13294	17873,7	15575,8	13294	13294	34,44937566	17,1641342	0	0
235	15423,4	17684,7	17684,7	17684,7	17684,7	14,66148839	14,66148839	14,66148839	14,66148839
235	12982,6	17394,9	15179,3	12982,6	15179,3	33,98625853	16,92033953	0	16,92033953
235	13178,8	17747,5	15449,6	13178,8	13178,8	34,66704101	17,23070386	0	0
236	13294	17873,7	15575,8	13294	13294	34,44937566	17,1641342	0	0
241	8772,97	17743,5	15487,7	11000,7	13238,7	102,251917	76,53884602	25,39311088	50,90328589
243	13309,6	17874,5	15584,5	17874,5	15584,5	34,29780008	17,09217407	34,29780008	17,09217407
246	13036,2	17550,8	17550,8	13036,2	13036,2	34,63125758	34,63125758	0	0
248	15218,4	17546,3	17546,3	15218,4	15218,4	15,29661462	15,29661462	0	0
261	11063	17873,5	17873,5	11063	11063	61,56105939	61,56105939	0	0
262	13025,9	17509,4	17509,4	17509,4	15237,3	34,41988653	34,41988653	34,41988653	16,97694593
267	15295,2	17592,8	17592,8	17592,8	15295,2	15,02170616	15,02170616	15,02170616	0
270	12946,6	17437,9	15177,6	17437,9	12946,6	34,69096133	17,23232354	34,69096133	0
278	8612,2	17476,9	12990,7	10797,3	10797,3	102,9318873	50,84066789	25,37214649	25,37214649
291	10978	17818,1	15526,6	10978	10978	62,30734196	41,43377664	0	0

b_constr... basic construction heuristic
ran_constr... random construction heuristic
cheap... cheapest insertion improvement heuristic

Table 1: solution quality heuristics versus optimal solution basic model

	computation time (seconds)		
	math. model	heuristics	
# tasks	optimal	b_constr+cheap	ran_constr+cheap
151	105,231	11,842	6,463
153	89,805	8,009	2,13
153	73,024	14,958	10,243
153	91,009	7,533	1,87
153	84,628	29,496	14,283
155	137,327	36,478	88,535
155	103,394	33,787	13,873
155	93,692	30,126	20,822
155	98,549	7,73	10,006
160	132,418	15,538	9,128
163	92,366	70,846	12,742
167	115,347	49,06	26,073
174	128,527	35 ,822	22,452
175	130,16	18,841	25,833
175	189,914	36,416	24,642
177	181,923	21,472	30,606
180	220,486	44,124	33,258
181	235,687	34,208	25,49
181	238,146	72,17	17,911
184	156,615	80,487	35,953
184	169,511	64,772	14,422
187	187,765	118,258	48,377
191	270,849	105,29	55,34
192	257,049	57,667	26,284
192	208,468	53,294	24,945
204	242,613	17,914	3,936
212	430,396	108,286	22,18
212	194,452	18,984	4,019
212	649,032	73,868	12,129
213	602,356	68,61	45,161
214	494,287	31,881	16,557
214	290,943	52,317	7,258
218	1139,15	59,413	14,026
222	443,248	67,611	54,316
227	438,924	4,363	4,679
235	694,01	110,934	20,597
235	694,01	110,934	20,597
235	628,867	48,301	14,963
235	734,548	104,347	35,379
235	689,141	83,963	15,662
236	686,403	119,156	44,588
241	481,904	211,734	91,493
243	921,277	88,638	21,626
246	1052,04	152,963	69,473
248	1588,47	125,765	18,165
261	1574,75	170,583	118,797
262	1675,85	5,833	10,856
267	1551,54	24,71	9,589
270	1718,77	56,398	16,515
278	945,673	150,644	18,911
291	1452,45	307,932	70,058
b_constr… basic construction heuristic			
ran_constr… random construction heuristic			
cheap… cheapest insertion improvement heuristic			

Table 2: computation time heuristics versus optimal solution basic model

gaps

	b_constr	b_constr+cheap	ran_constr+cheap
avg	0,532	0,311	0,079
min	0	0	0
max	1,712	1,042	0,509

b_constr… basic construction heuristic
ran_constr…random construction heuristic
cheap… cheapest insertion improvement heuristic

Table 3: avg, min and max gaps basic problem

Figure 9 illustrates the number of iterations a gap was within a certain percentage according to the legend. Again the random construction with the cheapest insertion performed best, since it was able to find the optimal solution in 79 instances.

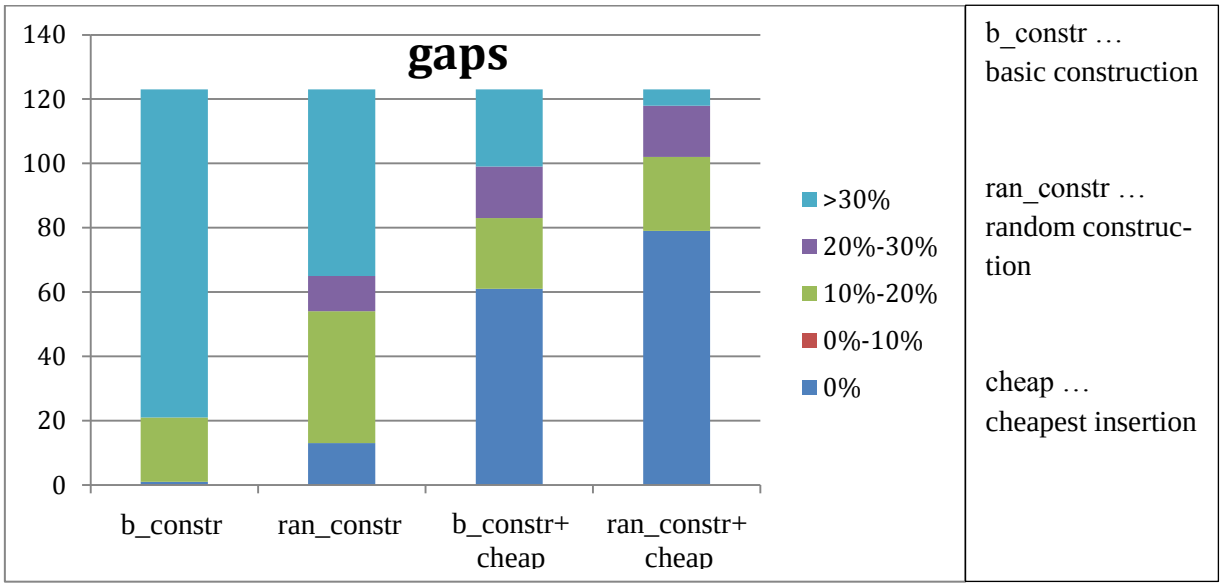

Figure 9: gaps basic problem

The computation time of construction1 and the random construction heuristic is so small that it is neglicable. The highest computation time was around half an hour to obtain the optimal solution for an instance with 270 tasks. The highest computation time of basic construction+ cheapest insertion was 308 seconds for an instance with 291 tasks, for the random construction heuristic+ cheapest insertion the highest computation time was 139. The reason why the same insertion heuristic takes longer when it is applied to construction1 is that more workers need to be eliminated. The computations time depending on the number of tasks is illustrated in figure

10. It can be observed that the computation time rises drastically from about 150 tasks. Also the computation time is not proportional to the number of tasks, since it takes the branch and bound algorithm different times to prove optimality.

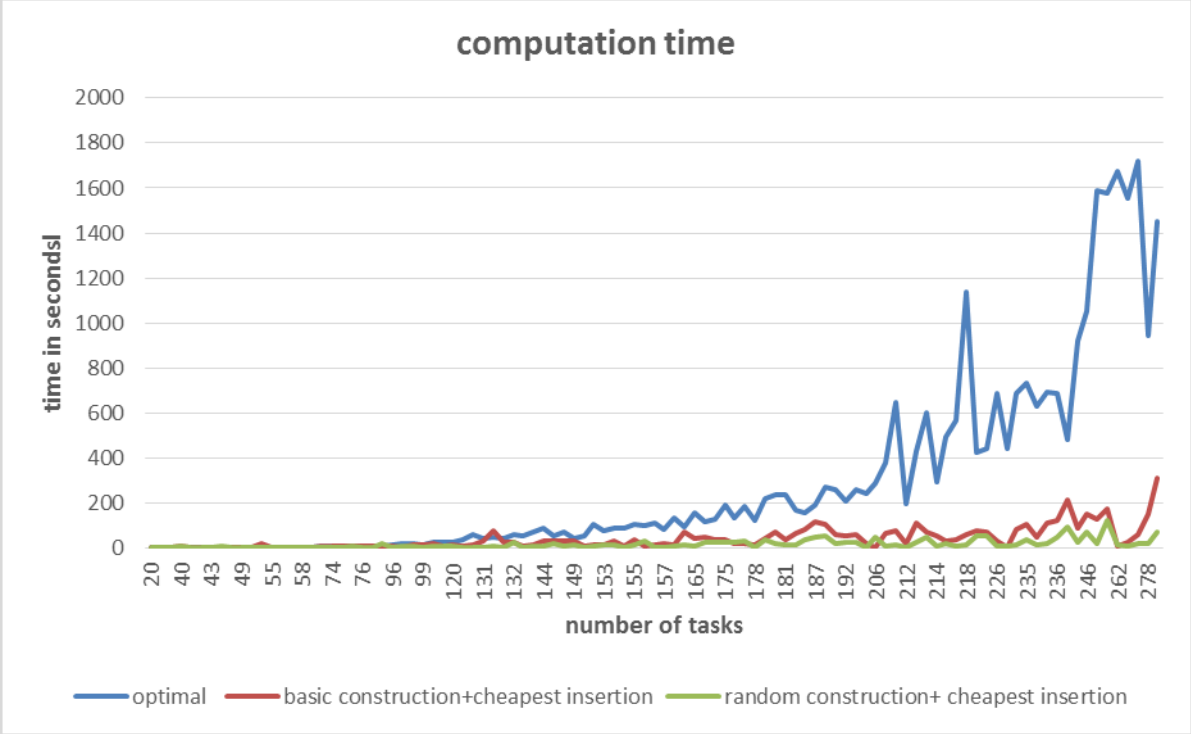

Figure 10: computation times basic problem

5.3. Results workload-balancing

The two extensions of the basic mathematical model were implemented and compared to the results of the workload-balancing heuristic. Since the worker cost minimization has a high priority over workload-balancing its weight was chosen to 0.99. The remaining 1 percent was shared among the other two parts of the objective function.

The far best results are obtained by the minimization of the average workload. The following Gantt chart in figure 11 illustrates the solution for the same instance of the Gantt charts in the prior subchapter. It can be observed that the same 4 workers are used, but the workload is distributed almost exactly equal. Furthermore, there are no such big gaps (breaks) as in the result of the basic model.

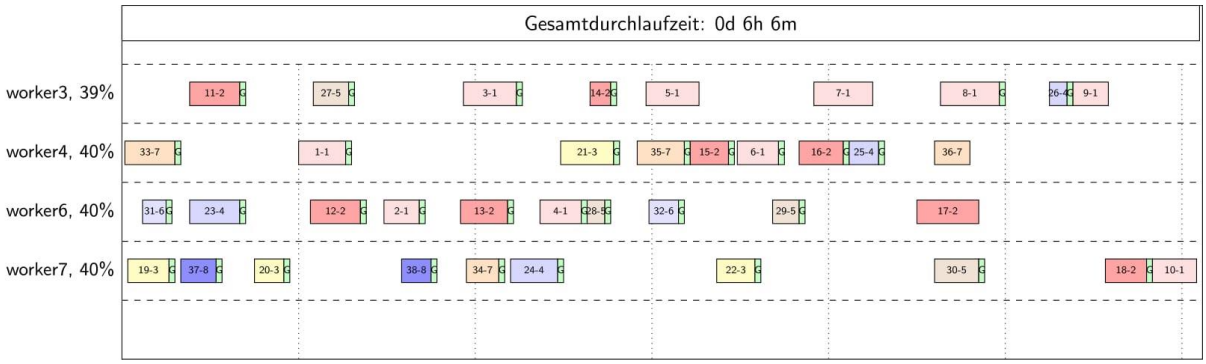

Figure 11: Gantt chart Minimum Average Workload optimization model

In comparison the random construction heuristic with the cheapest insertion improvement and the workload-balancing heuristic, illustrated in figure 12, also yields good results. The workload variates between 35% and 45% and there are no big gaps.

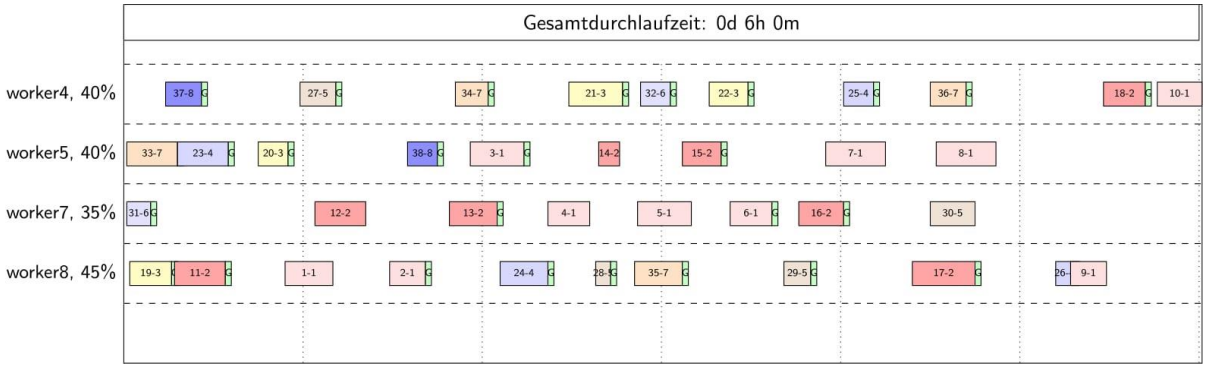

Figure 12: Gantt chart Random Construction+ Cheapest Insertion+ Workload Balancing

Unfortunately this model has a high number of binary decision variables and therefore its computation time is very large. As can be looked up in table 4, the computation time for the assignment of 58 tasks is already above 2 hours. For the second workload extension model, that aims to minimize the maximum workload of a worker, the computation time for the assignment of 132 tasks is 109 minutes. For this reasons the remaining 70 computations were made without the optimal models.

computation time					
num tasks	optimal	optMinMax	optAv	b_constr+cheap+WL	ran_constr+cheap+WL
20	0,34	0,579	1.653	0,336	0,343
33	0,86	2,61	6,816	0,942	1,208
38	1,454	3,035	214,47	0,517	1,263
40	1,553	6,583	31,805	5,612	4,933
43	1,684	10,119	707,433	0,467	0,105
43	1,673	10,143	707,919	0,466	0,766
43	2,321	40,352	168,75	1,905	1,362
43	1,878	41,722	163,861	1,912	4,719
46	1,715	29,42	39,759	0,774	0,708
49	2,126	30,852	129,308	2,183	1,307
49	2,026	29,603	126,291	2,214	0,061
52	2,554	24,538	83,397	18,23	1,672
55	3,31	16,462	672,12	3,902	1,218
55	2,951	14,865	665,366	3,909	3,44
58	3,605	104,849	7545,48	1,856	2,725
58	3,393	166,8		3,034	3,042
58	3,441	168,672		2,964	1,123
66	4,689	77,045		6,853	3,61
74	6,964	1147,31		6,051	2,201
76	6,678	354,276		4,845	6,045
76	6,716	354,389		4,765	1,477
84	9,213	227,252		5,343	3,107
88	10,079	493,94		0,133	16,588
96	14,343	1421,72		6,311	2,447
99	16,096	1768,17		13,001	4,128
112	24,067	2058,33		17,754	3,045
132	46,856	6595,67		69,517	0,885
num tasks... number of tasks					
optimal... basic mathematical model					
optMinMax... mathematical model extension 1: minimizing the maximum workload and the maximum gap					
optAv... mathematical model extension 2: minimizing the average workload deviation and the maximum gap					
b_constr+cheap+WL... basic construction heuristic+ cheappest insertion+ workload balancing heuristic					
ran_constr+cheap+WL... random construction heuristic+ cheapest insertion+ workload-balancing heuristic					

Table 4: computation time (in seconds) workload balancing

The quality of the workload-balancing condition is checked by calculating the deviation between the minimum and the maximum workload by computation and method, which is shown in table 5. At the bottom the average deviation over the number of iteration per method is computed. The far best results are unsurprisingly found with the minimum average workload model. The results also show that the second model extension as well as the workload balancing heuristic perform an improvement concerning workload balancing in comparison to the corresponding solutions of the basic model and the construction and improvement heuristics.

difference between min and max workload									
iteration	optimal	optMinMax	optAv	b_constr	b_constr+cheap	b_constr+cheap+WL	ran_constr	ran_constr+ cheap	ran_constr+cheap+WL
1	31	9	2	40	65	2	49	49	12
2	22	15	2	42	47	4	22	19	4
3	33	12	2	28	39	4	37	18	5
4	37	32	1	21	21	9	57	57	8
5	33	12	2	28	39	4	38	20	9
6	28	18	1	39	50	5	26	15	3
7	8	37	2	19	19	15	38	25	8
8	8	37	2	19	19	15	62	62	14
9	40	4	1	27	36	5	27	13	7
10	40	6	2	30	46	6	31	16	4
11	40	4	1	27	36	5	49	28	3
12	17	14	2	31	58	7	46	20	5
13	17	14	2	31	58	7	41	19	3
14	25	51	1	25	25	8	47	26	10
15	19	5	2	30	29	8	36	39	8
16	19	5		14	26	7	52	52	23
17	18	2		27	38	7	34	23	1
18	30	6		15	24	17	50	50	21
19	17	4		32	43	14	36	21	5
20	23	2		4	8	3	57	31	10
21	12	2		14	15	8	54	27	3
22	17	4		32	43	14	45	32	2
23	41	3		27	12	2	39	14	3
24	30	11		8	24	5	43	43	6
25	30	11		8	24	5	52	38	7
26	19	3		10	18	13	58	58	11
27	25	6		15	19	3	52	52	20
...	...			...	...	...	...	...	...
average	**18,04**	**12,18518519**	**1,6667**	**21,31683**	**25,01980198**	**9,574257426**	**41,792079**	**32,03960396**	**13,11881188**
optimal... basic mathematical model									
optMinMax... mathematical model extension 1: minimizing the maximum workload and the maximum gap									
optAv... mathematical model extension 2: minimizing the average workload deviation and the maximum gap									
b_constr... basic construction heuristic									
ran_constr...random construction heuristic									
cheap... cheapest insertion heurisitc									
WL.. Workload-balancing heuristic									

Table 5: deviation between min and max workload

Another way to check on the solution quality is the calculation of the sum of the deviations per worker of the average workload, which is given in table 6. It can again be observed that the far best result are obtained by the minimum average workload model. The mathematical extensions considering workload-balancing as well as the workload balancing application are marked in blue. Also under this point of view, the workload- balancing heuristic yields an improvement.

	sum workload deviation								
	optimal	optMinMax	optAv	b_constr	b_constr+cheap	b_constr+cheap+WL	ran_constr	ran_constr+ cheap	ran_constr+cheap+WL
	40	11	3	74	71	2	55	55	13
	27	19	3	75	55	4	28	22	5
	40	18	2	63	59	5	54	25	7
	59	39	3	47	47	16	108	108	15
	40	18	2	63	59	5	44	29	11
	45	25	2	82	56	7	40	20	4
	16	60	5	43	43	35	74	34	12
	16	60	5	43	43	35	105	105	26
	54	5	2	60	44	6	63	17	8
	50	8	2	73	64	8	53	20	5
	54	5	2	60	44	6	101	37	6
	23	18	2	70	71	8	86	27	6
	23	18	2	70	71	8	47	25	5
	43	92	3	43	43	24	77	51	14
	26	7	4	65	40	9	47	57	9
	38	7		27	52	16	108	108	58
	27	3		60	54	12	71	30	2
	57	10		31	43	27	135	135	58
	18	4		67	72	29	66	30	6
	41	3		8	20	9	117	64	23
	20	4		28	30	10	97	47	7
	18	4		67	72	29	107	36	3
	49	4		64	14	2	57	19	4
	55	20		17	43	8	83	83	12
	55	20		17	43	8	119	80	10
	24	4		21	34	21	91	91	23
	35	11		33	33	7	84	84	45
	...			...	...	7	...	...	...
min	7	3	2	8	8	0	22	7	2
max	59	92	5	82	81	67	150	140	120
avg	29,17821782	18,40740741	2,8	45,87129	42,20792079	18,73267327	85,069307	57,73267327	29,55445545
optimal... basic mathematical model									
optMinMax... mathematical model extension 1: minimizing the maximum workload and the maximum gap									
optAv... mathematical model extension 2: minimizing the average workload deviation and the maximum gap									
b_constr... basic construction heuristic									
ran_constr...random construction heuristic									
cheap... cheapest insertion heurisitc									
WL.. Workload-balancing heuristic									

Table 6: sum of deviation of average workload

With the workload-balancing extension, the gap (break) minimization per worker from the start of the computation to the first task, between the assigned tasks and from the last assigned task to the end of the computation should also be taken under consideration.

There is a trade-off between the workload-balancing and the gap minimization, so both criteria cannot be perfectly satisfied. Therefore, the priority is set on workload- balancing, which is considered as more important for the fairness consideration of the workers. Anyway, some changes do not affect the workload-balancing put improve the gap- minimization criteria, which were then made. The corresponding maximum gaps for the same computations are shown in table 7. It can be observed that the average maximum gap even increased when applying the

workload-balancing heuristic on the basic construction+ cheapest insertion heuristic, due to the workload-balancing priority.

max gap(break)									
iteration	optimal	optMinMax	optAv	b_constr	b_constr+cheap	b_constr+cheap+WL	ran_constr	ran_constr+ cheap	ran_constr+cheap+WL
1	86	65	66	106	34	72	34	34	54
2	88	77	70	99	43	72	108	53	53
3	102	131	90	183	67	104	103	103	85
4	129	191	133	158	158	164	202	202	169
5	102	131	90	183	67	104	152	134	84
6	154	93	80	165	59	135	138	69	50
7	93	180	82	179	179	179	92	92	71
8	93	180	82	179	179	179	72	72	107
9	87	48	81	183	139	102	121	103	48
10	205	118	96	206	206	156	156	156	156
11	87	48	81	183	139	102	146	146	84
12	98	125	107	182	131	101	132	84	78
13	98	125	107	182	131	101	86	86	86
14	88	94	77	141	141	141	133	133	122
15	137	85	145	203	101	119	134	134	66
16	146	192		165	165	165	495	495	329
17	147	135		210	139	143	170	170	142
18	183	139		154	132	172	107	107	221
19	118	113		172	143	183	148	100	87
20	139	134		277	182	182	152	152	173
21	195	182		180	180	180	189	189	274
22	118	113		172	143	183	132	132	136
23	183	63		182	97	97	213	120	75
24	276	161		179	162	134	134	134	186
25	276	161		179	162	134	126	126	214
26	132	155		165	160	160	433	433	156
27	169	192		212	212	212	264	264	173
	...			...	...	...	...	...	...
average	233,19	127,074074	92,47	207,68	198,05	212,69	380,3	295,95	263,91
optimal... basic mathematical model									
optMinMax... mathematical model extension 1: minimizing the maximum workload and the maximum gap									
optAv... mathematical model extension 2: minimizing the average workload deviation and the maximum gap									
b_constr... basic construction heuristic									
ran_constr...random construction heuristic									
cheap... cheapest insertion heurisitc									
WL.. Workload-balancing heuristic									

Table 7: maximum gap

6. Conclusion

In conclusion this work shows the high potential of a centralized planning in comparison to the current situation of scheduling one worker fixed on one machine. The presented mathematical models involved the main characteristics in the ILL case and might be extended in further research. The proposed heuristics for the worker scheduling problem yield good results for big instances and might further be improved by a metaheuristic.

There are several possible extensions that might be considered. The worker might be heterogeneous, such that a qualification matrix that stores the information, which worker is able to do which operation needs to be developed. Furthermore, dynamic operation times that for example depend on the number of scheduled workers or on the workers' skills. Other worker requirements, such as lunch breaks and maximum working times per day can also be taken under consideration.

A problem of the proposed centralized planning might be that problems arise, if worker do not feel responsible for a machine. If, for example, a problem occurs on a machine, it might not be immediately realized and nobody might feel responsible in finding a solution. A solution to this problem is more control and a better observation of the workers.

7. References

Achuthan, N.R., L. Caccetta, and S.P. Hill. 1996. "A new subtour elimination constraint for the vehicle routing problem." *European Journal of Operational Research 91* 573-586.

Araújo, F.F.B., A.M. Costa, and C. Miralles. 2012. "Two extensions for the ALWABP: Parallel stations and collaborative approach." *International Journal of Production Economics 140* 755-770.

Baker, E., and J. Schaffer. 1989. "Solution improvement heuristics for the vehicle routing and scheduling problem with time windows constraints." *American Journal of Mathematics and Management Sciences 6* 261-300.

Balinski, M., and R. Quandt. 1964. "On an integer program for a delivery problem." *Operations Research* 300-304.

Battaïa, O., D. Xavier, A. Dolgui, J. Hagemann, A. Horlemann, S. Kovalev, and S. Malyutin. 2015. "Workforce minimization for a mixed-model assembly line in the automotive industry." *International Journal Production Economics* 489-500.

Bräysy, Olli, Michel Gendreau, Geir Hasle, and Arne Løkketangen. 2008. *A Survey of Heuristics for the Vehicle Routing Problem Part I: Basic Problems and Supply Side Extensions.* Norwegian Open Research Archives.

Camm, J.D., M.J. Magazine, G.G. Polak, and G.S. Zaric. 40 (8). "Scheduling parallel assembly workstations to minimize a shared pool of labor." *ILE Transactions* 749-758.

2008. "Chapter 2: Multi- objective Optimization." In *Multi- objective Management in Freight Logistics. Increasing Capacity, Service Level and Safety with Optimization Algorithms*, by M. Carmia and P. Dell'Olmo, 11-36. Springer.

Christofides, N., A. Mingozzi, and P. Toth. 1979. "The vehicle routing problem." In *Combinatorial Optimization*, by N., Mingozzi, A. Christofides, P. Toth and C. Sandi, 315-338. Chichester, UK: Wiley.

Clarke, G., and J.V. Wright. 1964. "Scheduling of vehicles froma a central depot to a number of delivery points." *Operations Research 12* 568-581.

Cordeau, Jean-Francoise, and Gilbert Laporte. 2003. "The Dial-a-Ride Problem (DARP): Variants, modelling issues and algorithms." *Quarterly Journal of the Belgian, French and Italian Operations Research Societies 1, issue 2* 89-101.

Corominas, A., R. Pastor, and J. Plans. 2008. "Balancing assembly line with skilled and unskilled workers." *Omega 36 (6)* 1126-1132.

Dantzig, G.B., and J.H. Ramser. 1959. "The truck dispatching problem." *Management Science* 6-80.

De Bruecker, P., J. Van den Bergh, J. Beliën, and E. Demeulemeester. 2015. "Workforce planning incorporating skills: state of the art." *European Journal of Operational Research* 1-16.

Desaulniers, G., J. Lavigne, and F. Soumis. 1997. "Multi-depot vehicle scheduling problem with time windows and waiting cost." *European Journal of Operation Research 111* 479-494.

Dijkstra, E.W. 1959. "A note on two problems in connection with graphs." *Numerische Mathematik* 269-272.

Dolgui, A., S. Kovalev, M. Y. Kovalyov, and S. Malyutin. 2017. "Optimal workforce assignment to operations of a paced assembly line." *European Journal of Operational Research* 1-12.

Dongarra, J.J. 1998. *Performance of various computers using standard linear equations software.* Technical Report, Knoxville: University of Tennessee.

El-Sherbeny, Nasser A. 2010. "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods." *Journal of King Saud University* 123-131.

Foster, B.A., and D.M. Ryan. 1976. "An integer programming approach to the vehicle scheduling problem." *Operations Research 27* 109-124.

Ghiani, Gianpaolo, Gilbert Laporte, and Roberto Musmanno. 2013. "Managing Freight Transport." In *Introduction to Logistics Systems Management, Second Edition,* 318-432. Wiley Series in Operations Research and Management Science.

Jalel, Euchi, Yassine Adnan, and Chabchoub Habib. 2015. "The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach." *Swarm and Evolutionary Computation Vol. 21* 41-53.

Lin, S. 1965. "Computer solutions of the travelling salesman problem." *Bell System Technical Journal 44* 2245-2269.

Miller, C.E., A.W. Tucker, and R.A. Zemlin. 1960. "Integer programming formulations and traveling salesman problems." *Journal of the ACM 7* 326-329.

Mole, R.H., and S.R. Jameson. 1976. "A sequential route-building algorithm employing a generalized savings criterion." *Operational Research Quaterly 27* 503-511.

Moreira, M.C., J.-F. Cordeau, A.M. Costa, and G. Laporte. 2015. "Robust assembly line balancing with heterogenous workers." *Computers & Industrial Engineering, 88* 254-263.

Mutlu, Ö., O. Polat, and A.A. Supciller. 2013. "An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-2." *Computers & Operations Research 40 (1)* 418-426.

Or, I. 1976. *Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking.* Ph.D. dissertation, Northwestern University, Evanson: Separtment of Industrial Engineering and Management Sciences.

Polat, O., C.B. Kalayci, Ö. Mutlu, and S.M. Gupta. 2016. "A two-phase variable neighborhood search algorithm for assembly line worker assignment and balancing problem typ-2: an industrial case study." *International Journal of Production Research, 54* 722-741.

Potvin, J., and J. Rousseau. 1993. "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows." *Journal of Operational Research Society 46* 331-340.

Renaud, J., F.F. Boctor, and G. Laporte. 1996. "A fast composite heuristic for the symmetric traveling salesman problem." *INFORMS Journal on Computing 8* 134-143.

Renaud, J., F.F. Boctor, and G. Laporte. 1996. "An improved petal heuristic for the vehicle routing problem." *Journal of Operational Research Society* 329-336.

Rocha, M., J.F. Oliveira, and M.A. Carravilla. 2012. "Quantitative Approaches on Staff Scheduling and Rostering in Hospitality Management: An Overview." *American Journal of Operations Research* 137-145.

Shaw, P. 1998. "Using constraint programming and local search methods to solve vehicle, Volume 1520 of Lecture notes in computer science." *n CP-98 (Fourth International Conference on Principles and Praxis of Constrained Programming.* 417-431.

Sungur, B., and Y. Yavuz. 2015. "Assembly line balancing with hierarchical worker assignment." *Journal of Manufacturing Systems, 37(1)* 290-298.

T'Kindt, V., and J.-C. Billaut. 2006. *Multicriteria Scheduling.* Heidelberg: Springer-Verlag Berlin Heidelberg.

Toth, Paolo, and Daniele Vigo. 2001. *The vehicle routing problem.* SIAM.

Venkatesh, J. 2008. "Evaluation of performance measures for representing operational objectives of a mixed model assembly line balancing problem." *International Journal Production Research* 6367-6388.