An Integrated Matching and Partitioning Problem with Applications in Intermodal Transport

Erwin Pesch, Dominik Kress, Sebastian Meiswinkel
University of Siegen
Deptartment of Management Information Science
Kohlbettstr. 15, 57068 Siegen, Germany
{erwin.pesch, dominik.kress, sebastian.meiswinkel}@uni-siegen.de

***Abstract*—We introduce a combination of the problem of partitioning a set of vertices of a bipartite graph into disjoint subsets of restricted size and the Min-Max Weighted Matching Problem. The resulting problem has applications in intermodal transport. We propose a mathematical model and prove the problem to be NP-hard in the strong sense. Two heuristic frameworks that decompose the problem into its partitioning and matching components are presented. Additionally, we analyze a basic implementation of tabu search and a genetic algorithm for the integrated problem. All algorithms outperform standard optimization software. Moreover, the decomposition heuristics outperform the classical metaheuristic approaches.**

I. Introduction

In this paper we consider an integrated matching and partitioning problem that is motivated by real world problems arising in intermodal transport. Consider, for example, a rail-road terminal, that mainly serves as an interface in intermodal transport. Here, gantry cranes transship containers between trains and trucks. A typical layout of a rail-road terminal (yard) is schematically depicted in Fig. 1. This yard was presented in details in [1]. Freight trains are parked on parallel transhipment tracks of the yard. Trucks arrive on parallel truck lanes, which are usually separated into a driving lane and a parking lane. In the most modern transshipment yards or MegaHubs there additionally is a fully automated sorting system. Such a sorter consists of shuttle cars that can receive containers close to their initial positions from inbound trains and move them alongside the yard to their target positions. Usually, the yard area is subdivided horizontally into slots of equal size measured in units of the length of a standard railcar (or any other unit). The resulting grid is used to identify the coordinates of any given container in the yard and the coordinates of all parking slots of trucks. The area of the rail-road terminal is divided into several subareas according to the number of cranes (see [2]). A crane is assigned to each area and each crane operates only in the assigned area. Any container transport between subareas is performed using the sorter.

We will make the following simplifications. First, we only consider loaded moves, i.e., moves of cranes carrying containers, which is an assumption that proved to be realistic in previous research. Second, a container travelling through the sorter involves two cranes where the first crane delivers the container to the shuttle car while the second one picks it up again. We drop this assumption and consider only one crane operating in the area from which the container is delivered to the sorter. We will assume the number of containers to be less or equal to the number of parking slots.

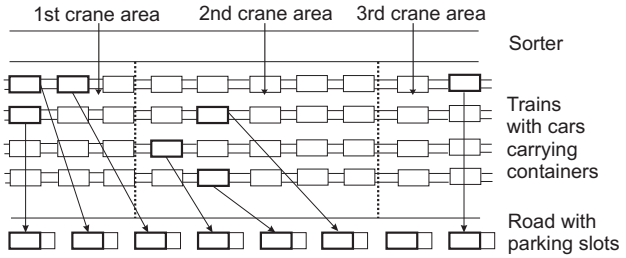

Fig. 1. Schematic layout of a rail-road yard

Let the completion time of train operations be the moment when the last container of the train is transhipped on the truck. Each train has a specified due date. The problem is to schedule container transhipment operations so that the maximum lateness is minimized. Then, given the above simplifications, the described situation within a rail-road terminal can be modelled as a Dedicated Variative Jobs Problem, which is defined as follows (see [3]). Set J of n dedicated jobs is to be processed on m parallel machines. The job set is partitioned in advance into mk subsets $J_{lz}, l = 1,\dots,m, z = 1,\dots,k$, and all jobs from the set $\cup_{z=1}^{k} J_{lz}$ are assigned to machine l. Each job j has a due date d_j and for all $j \in \cup_{l=1}^{m} J_{lz}$ the due date is the same, i.e., $d_j = D_z$. Processing times of jobs are interconnected and they are the subject of the decision making. The choice of the particular processing times is as follows. We are given a complete bipartite graph $G = G(J, V, A)$ with bipartitions J and V, $n = |J| \leq |V|$, and edge set A. For each $j \in J$ and each $v \in V$, edge $(j, v) \in A$ has a weight $c(j, v)$. Each $c(j, v)$, $v \in V$, is a possible processing time for job j. If we have chosen $c(j, v^0)$ as the processing time for some job j then none of $c(j', v^0)$ may be chosen as the processing time for job $j' \neq j$. Thus, any feasible choice of the processing times for the jobs should provide a maximum (by inclusion) matching for graph G. The goal is to choose a processing time for each job in the feasible way and to

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.245

construct a schedule S that minimizes maximum lateness $L_{max}(S) = \max_{j \in J} L_j(S) = \max_{j \in J}(C_j(S) - d_j)$, where $C_j(S)$ is the completion time of the job j in the schedule S. To define a particular schedule, we should choose the processing time for each job and construct a sequence of jobs for each machine.

In terms of the Dedicated Variative Jobs Problem, the described above situation within a rail-road terminal can be modelled as follows. We have m cranes and k trains. Set J is the set of all containers to be transhipped. Subset $J_{lz}, l = 1, \ldots, m, z = 1, \ldots, k$, is the subset of all containers that are situated on the z-th train in the l-th crane zone. Set V is the set of parking slots of the trucks. The positions of containers and parking slots are predefined and known. Moreover, the duration of transporting the particular container to the defined parking slot is known. These transportation times define the weights of the edges. Due date $D_z, 1 \leq z \leq k$, corresponds to the containers of the z-th train.

When we assume the trains in the rail-road terminal to arrive (and leave) in bundles, and thus aim at workload balancing of the gantry cranes, the above mentioned real-world problem can be modelled as the so called MIN-MAX WEIGHTED MATCHING Problem, that has recently been introduced in [4]. This problem can be described as follows. There is a complete weighted bipartite graph $G = G(U, V, E)$ where U and V are the bipartitions and E is the set of edges. Each edge $e = (u, v) \in E, u \in U, v \in V$, has a rational weight $c(e)$. Set U is partitioned into m subsets $U_1, U_2, \ldots, U_m$, the so-called components. For any maximum matching Π, we define the weight of component U_k as the sum of the weights of those edges of the matching with a vertex belonging to this component: $c(U_k, \Pi) = \sum_{u \in U_k, (u,v) \in \Pi} c(u, v)$. The value of Π is the maximum of the weights of the components: $c(\Pi) = \max_{k \in \{1,\ldots,m\}} c(U_k, \Pi)$. The problem is to find a maximum matching of minimum value.

In terms of the Min-Max Weighted Matching Problem, the rail-road terminal example is modelled as follows. Set U denotes the set of all containers to be transshipped. Subset U_k is the set of all containers that are located in the k-th crane area. The set of parking slots is denoted by V. The transportation times define the weights of the corresponding edges. The problem is to balance the workload over the cranes.

Given the strongly NP-hard Min-Max Weighted Matching Problem, the main focus of this paper lies on relaxing the assumption of the components being fixed in advance. In terms of the considered problem at rail-road terminals, this corresponds to considering the decision on the crane areas (i.e. the partitioning decision) as part of the problem (which becomes more adequate when considering crossover cranes that are allowed to pass one another). We will refer to this problem as the PARTITIONING MIN-MAX WEIGHTED MATCHING Problem. It has first been introduced in [5].

Besides the rail-road terminal case, another application of Partitioning Min-Max Weighted Matching arises at small to medium sized sea ports where containers are handled by reach stackers as represented in Fig. 2. These terminals can include large long-term storage areas and additional temporary storage areas (or marshaling-areas, [6], [7]). The latter areas aim at improving the performance of the terminals by inducing short turnaround times of vessels when distances to long-term storage areas are relatively large. When a vessel arrives at a berth at the terminal, containers are unloaded by quay cranes and are then stored in a temporary storage area that is located next to the berth. Containers that leave the terminal by ship are moved to the temporary storage area during previous idle times. An application of Partitioning Min-Max Weighted Matching arises, when considering the process of emptying or refilling the temporary storage area using reach stackers during idle times. We assume the vehicles to be "fast" if they are unloaded and "slow" if they are loaded. This is a common assumption when considering container movements [2] and is reasonable when noting that "reach stackers [in comparison to straddle carriers] are less stable in the forward direction as the machines will fall forward when breaking in an emergency, particularly if the load is being carried high for visibility reasons" [8]. We can then restrict ourselves to considering laden movements of the vehicles only. We are faced with the problem of assigning each container in the temporary storage area to an empty slot in the long-term storage area (or vice versa) and assigning the corresponding container movements to a limited number of available vehicles. Naturally, the objective is to perform all movements as fast as possible.

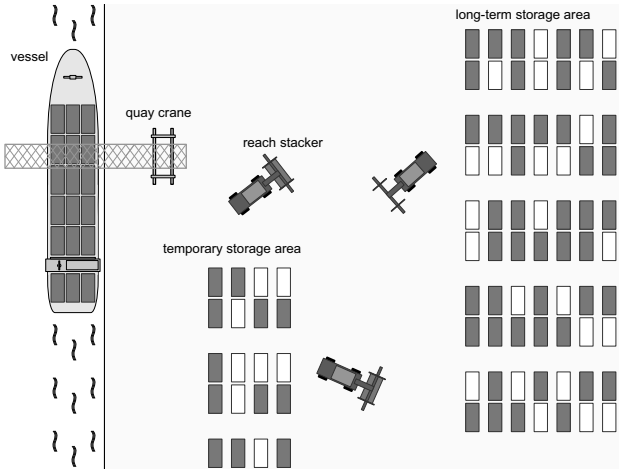

Fig. 2. Schematic layout of a reach stacker based terminal (see [5])

This paper proceeds as follows. In Sections II and III (see also [5]), we present a detailed problem description along with a mathematical model of Partitioning Min-Max Weighted Matching and a proof of the problem's NP-hardness, respectively. Section IV is concerned with describing heuristic approaches for solving Partitioning Min-Max Weighted Matching. On the one hand, we are concerned with heuristics that decompose the problem into its matching and partitioning components. On the other hand, we consider classical metaheuristic approaches for the integrated problem.

The heuristics are analyzed in a computational study in Section V. The paper closes with a summary in Section VI.

II. Problem Definition

Let $G(U, V, E)$ be a weighted bipartite graph with bipartitions U and V and edge set E. The elements of U and V are indexed $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$, respectively. Assume $n_1 \leq n_2$. A weight $c(e) = c_{uv} \in \mathbb{Q}_0^+$ is associated with each edge $e = (u, v) \in E$ of G. Define a matching as a set $M \subseteq E$ of pairwise nonadjacent edges and a maximum matching as a matching having the largest possible size $|M|$ amongst all matchings on G. Throughout the paper, we will assume that, for any given bipartite graph, there exists a maximum matching Π with $|\Pi| = n_1$. As in [4], given a partitioning of U into m disjoint subsets, $U_1, U_2, \ldots, U_m$, the value of a maximum matching Π is defined to be $c(\Pi) := \max_{k \in \{1,\ldots,m\}} \{\sum_{u \in U_k, (u,v) \in \Pi} c_{uv}\}$. Then Partitioning Min-Max Weighted Matching can formally be defined as follows: Find a partitioning of the vertex set U into m (potentially empty) disjoint subsets, $U_1, U_2, \ldots, U_m$, with at most $\bar{u}$ elements in each subset, and a maximum matching Π on G, such that the value of Π is minimum amongst all maximum matchings over all possible partitionings of U.

We define the following binary variables:

$$x_{ij} := \begin{cases} 1 & \text{if } (i,j) \in \Pi, \\ 0 & \text{else,} \end{cases} \qquad \forall\, (i,j) \in E \tag{1}$$

and

$$y_{ik} := \begin{cases} 1 & \text{if } i \in U_k, \\ 0 & \text{else,} \end{cases} \qquad \forall\, i \in U, k \in \{1, \ldots, m\}. \tag{2}$$

Then a nonlinear mathematical model for Partitioning Min-Max Weighted Matching is as follows:

$$\min_{\mathbf{x},\mathbf{y}} \max_{k \in \{1,\ldots,m\}} \left\{ \sum_{i \in U} \sum_{j \in V} c_{ij} y_{ik} x_{ij} \right\} \tag{3}$$

s.t.

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall\, i \in U, \tag{4}$$

$$\sum_{i \in U} x_{ij} \leq 1 \qquad \forall\, j \in V, \tag{5}$$

$$\sum_{k=1}^{m} y_{ik} = 1 \qquad \forall\, i \in U, \tag{6}$$

$$\sum_{i \in U} y_{ik} \leq \bar{u} \qquad \forall\, k \in \{1, \ldots, m\}, \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, (i,j) \in E, \tag{8}$$

$$y_{ik} \in \{0, 1\} \qquad \forall\, i \in U,\, k \in \{1, \ldots, m\}. \tag{9}$$

The objective function (3) minimizes the value of the maximum matching over all possible partitionings and all maximum matchings. Constraints (4)–(5) are well known maximum matching constraints (recall that $n_1 \leq n_2$). Constraints (6) enforce every vertex $u \in U$ to be an element of exactly one partition U_k, $k \in 1, \ldots, m$. Constraints (7) restrict the number of vertices in each partition to be at most $\bar{u}$. The domains of the variables are defined by (8)–(9).

Model (3)–(9) can easily be linearized by introducing additional variables $z_{ijk} \geq 0$ for all $i \in U$, $j \in V$ and $k \in \{1, \ldots, m\}$:

$$\min_{\mathbf{x},\mathbf{y}} \quad c \tag{10}$$

s.t.

constraints (4)–(9),

$$c \geq \sum_{i \in U} \sum_{j \in V} c_{ij} z_{ijk} \quad \forall\, k \in \{1, \ldots, m\}, \tag{11}$$

$$z_{ijk} \geq y_{ik} + x_{ij} - 1 \;\forall\, i \in U,\, j \in V,\, k \in \{1, \ldots, m\}, \tag{12}$$

$$z_{ijk} \geq 0 \qquad \forall\, i \in U,\, j \in V,\, k \in \{1, \ldots, m\}. \tag{13}$$

III. Computational Complexity

In [3] the authors proved that problem Dedicated Variative Jobs is NP-Hard even if $m = 1$ and finding an approximate solution to problem Dedicated Variative Jobs with an approximation ratio less than 2 is strongly NP-hard problem. Moreover Min-Max Weighted Matching is NP-hard in the strong sense and finding an approximate solution to problem Min-Max Weighted Matching with an approximation ratio less than 2 is strongly NP-hard, see [4]

Note that Partitioning Min-Max Weighted Matching is not a straight forward generalization of Min-Max Weighted Matching in the sense that, given an instance I of Min-Max Weighted Matching, we need only "copy" the corresponding bipartite graph, fix a set of Partitioning Min-Max Weighted Matching parameters to specific values, and solve the resulting Partitioning Min-Max Weighted Matching instance to receive an optimal solution to I. More generally, it is not trivial to construct an instance of Partitioning Min-Max Weighted Matching that (when solved to optimality) is guaranteed to result in the partitioning given in I and thus to provide an optimal solution to I. Thus we cannot simply conclude that Partitioning Min-Max Weighted Matching is NP-hard from the NP-hardness of Min-Max Weighted Matching.

A formal proof of strong NP-hardness of Partitioning Min-Max Weighted Matching, which we will now provide, is more naturally based on a reduction of a classical partitioning problem than on a matching problem.

We will consider the decision problem related to Partitioning Min-Max Weighted Matching:

Definition 1 (Partitioning Min-Max Weighted Matching - D)**.** *Given a weighted bipartite graph as defined in Section II. Does there exist a partitioning of the vertex set U into m (potentially empty) disjoint subsets, $U_1, U_2, \ldots, U_m$, with at most $\bar{u}$ elements in each subset, and a maximum matching Π on G, such that the value of Π is no larger than a given $\omega \in \mathbb{Q}^+$?*

The proof is based on a reduction of the strongly NP-complete 3-Partition Problem [9]:

Definition 2 (3-Partition)**.** *Given a set A of $3k$ elements, a bound $B \in \mathbb{Z}^+$, and sizes $s(a) \in \mathbb{Z}^+$ for all $a \in A$ such*

that $B/4 < s(a) < B/2$ and such that $\sum_{a\in A} s(a) = kB$. Is there a partitioning of A into k disjoint subsets $A_1, \ldots, A_k$ such that, for $1 \leq i \leq k$, $\sum_{a\in A_i} s(a) = B$?

Theorem 1. *Partitioning Min-Max Weighted Matching-D is NP-complete in the strong sense.*

Proof: Obviously Partitioning Min-Max Weighted Matching-D $\in$ NP because for any partitioning and maximum matching it can be checked in polynomial time if the corresponding value is no larger than ω.

Now consider an arbitrary instance I of 3-Partition and construct a complete bipartite graph G as an instance J of Partitioning Min-Max Weighted Matching-D with $U = A$ and vertex set V such that $|V| = |U|$. Furthermore, for each $u \in U$, define the weights of all edges incident to u in G to be equal to the corresponding element's size $s(a)$, $a \in A$. Set $m = k$, $\bar{u} = 3$, and $\omega = B$. Now note that, given an arbitrary partitioning $U_1, \ldots, U_m$ of U, every maximum matching has the same value. We are left with the task of showing that there exists a partitioning $U_1, \ldots, U_m$ of U and a maximum matching Π with a value of no more than ω if and only if the answer to I is yes (we will say that I "has a solution" if its answer is yes and call a corresponding partitioning of A a "solution").

Suppose we are given a solution $A_1, \ldots, A_k$ to I. Construct a partitioning $U_1, \ldots, U_m$ of U such that subset U_i contains all vertices that correspond to elements of A_i for all $i = 1, \ldots, m$, and consider any maximum matching on G. It is immediately implied that the value is ω.

Suppose we are given a partitioning $U_1, \ldots, U_m$ of U and a maximum matching Π such that its value is no more than ω. As $\bar{u} = 3$ and $|U| = 3m$, we necessarily have $|U_i| = 3$ for all $i = 1, \ldots, m$. Now note that, as the sum of the edge weights of any maximum matching on G equals $m\omega$ and the value of Π is at most ω, we have $\sum_{u\in U_i,(u,v)\in\Pi} c_{uv} = \omega$ for all $i = 1, \ldots, m$. Hence, the subsets of A that correspond to the given partitioning of U establish a solution to I. ■

We conclude:

Corollary 1. *Partitioning Min-Max Weighted Matching is NP-hard in the strong sense.*

IV. Heuristic and Metaheuristic Approaches for Solving Partitioning Min-Max Weighted Matching

Partitioning Min-Max Weighted Matching can be considered as a combination of a matching and a partitioning problem. Hence, a natural way of constructing heuristics for Partitioning Min-Max Weighted Matching is to decompose the problem into its matching and partitioning components, solve the resulting problems separately (either exact or heuristically), and combine the resulting solutions to a solution of Partitioning Min-Max Weighted Matching. Obviously, we can construct different heuristics by changing the order of solving the separate stages. We will consider such "decomposition approaches" in Section IV-A. For details, see [5].

On the other hand, one can try to solve Partitioning Min-Max Weighted Matching directly ("integrated approaches"), for example by applying metaheuristic approaches such as tabu search [10], [11] or genetic algorithms [12]. In Section IV-B we will briefly discuss how to adapt these approaches to the Partitioning Min-Max Weighted Matching Problem.

A. Decomposition Approaches

Partition-Match heuristics assume that there is a vertex weight associated with each element of vertex set U and, in the first stage, partition U into no more than m components with at most $\bar{u}$ elements in each component, such that the maximum weight of the components is as small as possible. Here, the weight of a component is defined as the sum of the weights of the vertices of the component. When considering the objective of minimizing the maximum weight of the components, we refer to this problem as the Restricted Partitioning (RP) Problem. Based on the proof of Theorem 1, it is easy to see that RP is NP-hard in the strong sense. In the next stage of a Partition-Match heuristic, we drop the weights of the vertices. Given the components resulting from the first stage, we now need to determine a maximum matching of small value. Hence, we are faced with an instance of Min-Max Weighted Matching. Solving this problem results in a maximum matching, that we may use for restarting the overall procedure by solving RP with each vertex weight being equal to the weight of the edge of the maximum matching that is incident to the very vertex. Summing up, Partition-Match heuristics proceed as follows:

1) Generate or modify vertex weights.
2) **Partitioning stage**: Solve the resulting Restricted Partitioning Problem.
3) Drop vertex weights.
4) **Matching stage**: Solve the resulting Min-Max Weighted Matching Problem.
5) If there has been no improvement of the best known solution for 20 iterations, then stop. Otherwise go to step 1.

We initialize a Partition-Match heuristic by generating vertex weights that are used for determining the first partitioning of the vertex set U. We tested several strategies of generating these weights. As they all performed similarly in computational tests, we decided on applying the average weight of all edges incident to vertex u for each vertex $u \in U$.

Match-Partition heuristics first consider the classical problem of finding a maximum matching of minimum weight, i.e. minimum sum of the weights of all edges of the matching, which we refer to as the Min-Sum Weighted Matching (MSWM) Problem. Hereafter, we proceed in analogy to our approach in Partition-Match heuristics, i.e. by defining vertex weights based on the maximum matching and solving RP. Finally, we modify the edge weights of the bipartite graph to restart the procedure by solving MSWM on the modified graph:

1) **Matching stage**: Solve the (resulting) Min-Sum Weighted Matching Problem.
2) Generate or modify vertex weights.

3) **Partitioning stage**: Solve the resulting Restricted Partitioning Problem.
4) If there has been no improvement of the best known solution for 20 iterations, then stop.
5) Modify edge weights and go to step 1.

Concerning the modification of edge weights in step 5, we apply a simple transformation strategy, that increases the weight of exactly one edge $e \in E$ in each iteration. The selection of edge e depends on the current solution of the Partitioning Min-Max Weighted Matching input-instance: Select the component with the largest weight. Among the current edges of the matching that are incident to vertices of this component, select the one with largest weight. Then, set the weight of this edge to a large value (we set the weight to $10^2 \cdot c_{\max}$, where $c_{\max}$ is the largest edge weight of the input graph, in our computational tests) to reduce the probability that it will be part of the solution of the altered MSWM instance. After λ iterations of the overall procedure, restore the original edge weight of edge e. In preliminary computational tests, we have seen that $\lambda = \lceil 0.1 \cdot n_1 \cdot n_2 \rceil$ is a reasonable value. Note that the perturbed edge weights are only used for the matching problem in stage one. The vertex weights assigned in stage two are based on the original weights of the input graph.

We construct multiple variants of Partition-Match and Match-Partition heuristics by combining different methods of solving the corresponding subproblems:

- A heuristic algorithm for solving RP, denoted by RPH. As RP is similar to the well known MULTIPROCESSOR SCHEDULING Problem, that has been extensively studied in the literature (see, for example, the literature review [13]), our approach adapts ideas presented in [14] in its first stage that constructs a feasible solution to RP. In a second stage, we apply a local search on the obtained solution.
- The well known $\mathcal{O}(n^3)$ Hungarian algorithm [15], [16] for MSWM (where $n := \max\{n_1, n_2\}$).
- An approximate algorithm by [4], denoted by BPS, and a simple regret heuristic, denoted by REG, for solving Min-Max Weighted Matching. Both methods apply an additional local search stage, denoted by LS. While [4] show BPS to be an adequate method for solving Min-Max Weighted Matching alone, we implemented the rather simple REG heuristic motivated by the fact that, when calling a Partition-Match heuristic, we have to solve multiple instances of Min-Max Weighted Matching. Hence, in order to reduce running times of the overall procedure, a simple heuristic seems to be promising. Its name derives from the fact that it is based on the regret principle (cf., for instance, [17]).

This results in two variants of the Partition-Match framework, PM_{BPS} (applying BPS and LS for solving Min-Max Weighted Matching) and PM_{REG} (applying REG and LS for solving Min-Max Weighted Matching), and two variants of the Match-Partition framework, combining the Hungarian algorithm and RPH with (MP) and without (MP_{LS}) an additional local search stage.

B. Integrated Approaches

1) Tabu Search: Tabu search is a well known metaheuristic due to Fred W. Glover [11]. We assume the reader to be familiar with its basic concepts and refer to [10] for a detailed introduction. In the context of Partitioning Min-Max Weighted Matching, we apply tabu search in its basic form. In order to do so, we define the neighborhood $N(s)$ of a given solution s of Partitioning Min-Max Weighted Matching to consist of three disjoint sets $N(s) = M(s) \uplus S(s) \uplus V(s)$. Each set contains all feasible solutions that can be constructed based on s by performing one specific type of transformation:

- $M(s)$ contains all feasible solutions that can be constructed by applying a MATCH transformation. A MATCH transformation alters a solution s by replacing two edges, (u_i, v_j) and (u_k, v_l), that are part of the maximum matching of s, with (u_i, v_l) and (u_k, v_j). The edges are chosen such that u_i is part of a component of the partitioning of s with largest weight.
- $S(s)$ contains all feasible solutions that can be constructed by applying a SWAP transformation. A SWAP transformation alters the partitioning of vertex set U in solution s by swapping two vertices from different components. More precisely, we determine a partition U_{max} of maximal weight in s and a partition U_{min} (with $U_{min} \neq U_{max}$) of minimal weight in s. Afterwards we move one vertex $u_i \in U_{max}$ from U_{max} to U_{min} and one vertex $u_j \in U_{min}$ from U_{min} to U_{max}.
- $V(s)$ contains all feasible solutions that can be constructed by applying a MOVE transformation. Here, we determine U_{min} and U_{max} as before and move one vertex $u_i \in U_{max}$ from U_{max} to U_{min}.

2) A Genetic Algorithm: Genetic algorithms are well known evolutionary search algorithms due to John H. Holland [18]. As in Section IV-B1, we assume the reader to be familiar with the basic concepts of genetic algorithms and refer to [12] for an introduction. In this section, we will define an appropriate representation of a solution of Partitioning Min-Max Weighted Matching, as well as a mutation operator, a selection operator, and a crossover operator that our implementation of a basic genetic algorithm for Partitioning Min-Max Weighted Matching is based upon.

Our representation of a solution s of Partitioning Min-Max Weighted Matching is based on fixed sizes $|U_k|$ of the components $k = 1, \ldots, m$ of the partitioning and two lists, L_U and L_V, containing a permutation of U and V, respectively. Let $L_U(i)$ denote the i-th element of L_U. The connection of the partitioning of s and the list L_U is $U_i = \{L_U(j) \mid \sum_{k=1}^{i-1} |U_k| < j \leq \sum_{k=1}^{i} |U_k|, j \in \mathbb{N}\}$, $i = 1, \ldots, k$. The maximum matching M of s is represented by $M = \{(L_U(i), L_V(i)) | i \in \{1, \ldots, n_1\}\}$. To overcome the drawback of the sizes of the components of the partitioning being fixed, one may restart the genetic algorithm several times with different sizes that are determined randomly.

The initial population is generated randomly. The selection operator randomly selects two individuals from the current generation and adds the one with the smaller objective function value to the subsequent generation. Afterwards this

solution is mutated or crossed with an other solution with a given probability until the desired population size is reached.

Given two solutions s_1 and s_2, the crossover operator constructs two new solutions, s'_1 and s'_2. This is realized by generating two random numbers $1 \leq p_1 < p_2 \leq n_1$ and proceeding as follows. Let L_{U_1} and L_{V_1} denote the lists representing solution s_1 and L_{U_2} and L_{V_2} the lists representing s_2. Then the list L'_{U_1} representing the new solution s'_1 is

$$L'_{U_1}(i) = \begin{cases} L_{U_2}(i) & \text{if } p_1 < i \leq p_2, \\ L_{U_1}(i) & \text{else.} \end{cases} \tag{14}$$

The lists L'_{V_1}, L'_{U_2} and L'_{V_2} are generated in the same way. If necessary, a simple repair operation is performed to obtain feasible solutions after having called the crossover operator. The mutation operator randomly selects a pair of elements in L_U and interchanges these elements. Similarly, an interchange operation is performed in L_V.

V. Computational Results

In order to assess the performance of the algorithms introduced in Section IV, we ran extensive computational tests. All computational tests were performed on an Intel Core i7 mobile CPU at 2.8GHz and 8GB of RAM, running Windows 7 64bit. All algorithms were implemented in C++ (Microsoft Visual Studio 2010). We used IBM ILOG CPLEX in version 12.5 with 64bit.

In order to get a comprehensive overview of the performance of our algorithms, we use different strategies of generating instances. All instances are defined on bipartite graphs with $n_1 = n_2$, since the case $n_1 \neq n_2$ trivially reduces to the case $n_1 = n_2$ by adding edges of zero weight. We define $n := n_1 = n_2$. The first strategy of generating instances is inspired by [4]. We refer to the instance sets generated by this strategy as $BPS70$ and $BPS80$. We generate n^2 rational numbers uniformly distributed in the interval $[1, 1000]$ and a complete bipartite graph with vertex sets $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. We sort the numbers in non-decreasing order and denote the resulting list by L. Hereafter, we consecutively assign these numbers (as edge weights) to the edges in the following manner: We start with vertex v_1 and assign the first $\lfloor 0.8n \rfloor$ (in case of $BPS80$) or $\lfloor 0.7n \rfloor$ (in case of $BPS70$) elements of L to the edges $(u_1, v_1), (u_2, v_1), \ldots, (u_{\lfloor 0.8n \rfloor}, v_1)$ ($BPS80$) or $(u_1, v_1), (u_2, v_1), \ldots, (u_{\lfloor 0.7n \rfloor}, v_1)$ ($BPS70$) and remove them from L. For the remaining edges being incident to vertex v_1, we randomly choose and assign elements of L. Once an element of L has been assigned, we remove it from L. The procedure is repeated for all remaining vertices $v \in V$, $v \neq v_1$. The second strategy of generating instances constructs complete bipartite graphs and randomly determines and assigns edge weights uniformly distributed in the interval $[1, 1000]$. We refer to instances generated by this strategy as $RAND$. Besides $BPS70$, $BPS80$, and $RAND$, all of which are based on complete bipartite graphs, we generate instances defined on non-complete (or sparse) bipartite graphs with $|E| = \lceil 0.7n^2 \rceil$ or $|E| = \lceil 0.8n^2 \rceil$. We refer to them as $SPARSE70$ and $SPARSE80$, respectively. When generating the edges, we guarantee that there exists at least one feasible solution for the resulting Partitioning Min-Max Weighted Matching instance (cf. restrictions (4)). Again, edge weights are determined randomly and uniformly distributed in the interval $[1, 1000]$.

For each strategy of generating instances, $BPS70$, $BPS80$, $RAND$, $SPARSE70$, and $SPARSE80$, we construct instances based on different parameter settings. We vary the size of the bipartitions from 10 to 170, i.e. $n = 10, 30, \ldots, 170$. Furthermore, we set m to 2, and (if larger than 2) to $\lfloor 0.04n \rfloor$, $\lfloor 0.08n \rfloor$ or $\lfloor 0.125n \rfloor$. Similarly, we set $\bar{u}$ to $\lceil \frac{n}{m} \rceil$, $\left\lfloor \lceil \frac{n}{m} \rceil + \frac{1}{3} \left(n - \lceil \frac{n}{m} \rceil \right) \right\rfloor$, or n. For each combination of parameters and each generation strategy, we construct five Partitioning Min-Max Weighted Matching instances.

Let F^* be the value of the objective function (10) of the best solution found by a specific heuristic. Then, we rate the quality of this heuristic with the ratio $\frac{F^*}{F^{best}}$, where F^{best} is the best objective function value among the objective function values of the solutions obtained by any of the heuristics and the best solution found by CPLEX (model (10)–(13); preliminary tests proved CPLEX to perform better for the linearized model than for directly feeding it with model (3)–(9)) within a time limit of 180 seconds. Concerning the performance of CPLEX, only instances with a size of $n = 10$ could be solved to optimality within this time limit. In general we found the objective function values to be not competitive when compared with the ones determined by the heuristics, which was especially obvious for $RAND$, $SPARSE70$, and $SPARSE80$ instances. The results for $BPS70$ and $BPS80$ instances were better. However, the objective function values determined by CPLEX were (on average) still up to 600% larger than the ones determined by the best heuristic.

A. Decomposition Approaches

As described in Section IV-A, we implemented REG to solve Min-Max Weighted Matching mainly in order to speed up Partition-Match heuristics in comparison to using BPS. This motivation is illustrated in Fig. 3, which compares the average runtimes of REG and BPS on the $RAND$ instances where the partitioning has been fixed randomly with an additional lower bound on the size of each partition. For each n, the figure depicts the average runtimes over the results of all parameter settings. As to be expected, the corresponding solution qualities are better for the BPS heuristic, see Fig. 4. For this figure, F^{best} is set to the best objective function value among both heuristics.

Unfortunately, the same effects arise when embedding REG and BPS into the Partition-Match framework, i.e. PM_{REG} is not competitive in terms of solution quality. Except for PM_{REG}, however, all heuristics perform very well, even for large values of n; for details see [5]. We find that both, MP and MP_{LS}, outperform PM_{BPS} in terms of solution quality. However, PM_{BPS} still performs at a high level. While, for all heuristics, runtimes increase roughly quadratically in n, we find that PM_{BPS} requires by far the most computational effort (resulting from the effect

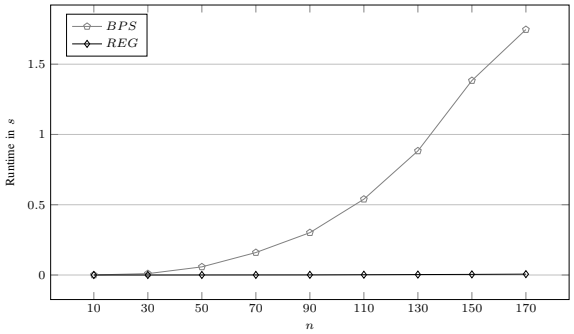

Fig. 3. Runtime comparison of REG and BPS on $RAND$ instances

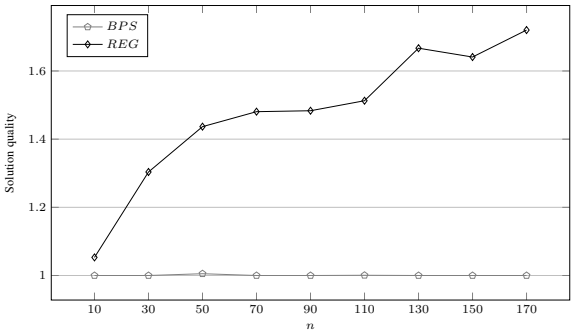

Fig. 4. Solution quality comparison of REG and BPS on $RAND$ instances

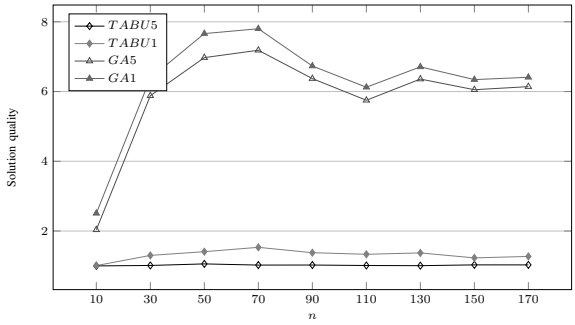

Fig. 5. Solution quality of metaheuristic approaches on $RAND$ instances

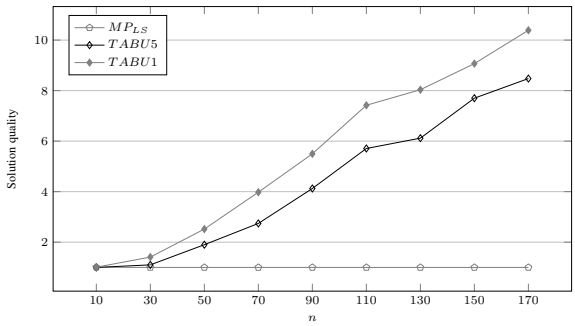

Fig. 6. Tabu search solution quality on $RAND$ instances in comparison to best decomposition approach MP_{LS}

seen in Fig. 3). This fact stands against the gap in qualities of PM_{REG} and PM_{BPS}. PM_{REG}, MP, and MP_{LS} are quite similar in terms of runtimes.

Based on these insights, it seems advisable to choose the Match-Partition framework when facing an instance of Partitioning Min-Max Weighted Matching. While MP and MP_{LS} perform equally well on the $RAND$, $SPARSE70$, and $SPARSE80$ instances, MP seems to be less effective for $BPS70$ and $BPS80$ instances. MP_{LS}, however, does not have this handicap. Even in the case of comparison with optimal solutions, the quality ratio of MP_{LS} is less than 1.0011. For all considered values of n, the runtime of MP_{LS} is smaller than 6 seconds. The average runtime for $n = 170$ is around 1.5 seconds. The influence of including LS on runtimes is (on average) less than a second for all n and all types of instances. Thus it is advisable to always apply MP_{LS} when facing a random instance of Partitioning Min-Max Weighted Matching.

B. Integrated Approaches

As indicated in Section IV-B, we have implemented different versions of the considered metaheuristics. $TABU1$ and $GA1$ refer to the basic tabu search procedure and the genetic algorithm as described in Section IV-B1 and IV-B2, respectively. $TABU5$ corresponds to calling the tabu search procedure starting from five different, randomly chosen solutions, and returning the best solution. Similarly, $GA5$ returns the best solution out of five runs of the genetic algorithm, starting with different, randomly chosen initial populations and sizes of components of the partitioning (see Section IV-B2).

We ran comprehensive preliminary computational tests to determine the remaining parameters for the tabu search algorithm and the genetic algorithm. As a result of these tests we set the length of the tabu list to $max(5, \frac{n_1}{10})$ and the stopping criterion to 500 iterations without improvement for the tabu search procedure. For the genetic algorithm we set the population size to 10, the crossover probability to 0.6, the mutation probability to $\frac{1}{2\,n_1}$, and the stopping criterion to 500 iterations without improvement.

Fig. 5 gives an overview of the performance of the considered metaheuristics solely based on the $RAND$ instances. Note that, in order to measure the quality of the algorithms, F^{best} in this case is set to the best objective function value among the objective function values of the solutions obtained by any of the integrated approaches. Clearly, $TABU1$ and $TABU5$ outperform the genetic algorithms. Furthermore, considering the best solution out of five runs seems to have a similar impact, i.e. a significant improvement of solution quality, in both metaheuristic frameworks.

It is an interesting question, whether the metaheuristic approaches can provide high quality results when comparing them with the best "decomposition approach". As mentioned in Section V-A, MP_{LS} has proven to be most effective out of these approaches when it comes to solving random problem instances. Hence, in Fig. 6 we compare the solution quality of $TABU1$ and $TABU5$ with MP_{LS} on the $RAND$ instances. Unfortunately, tabu search seems to be inferior

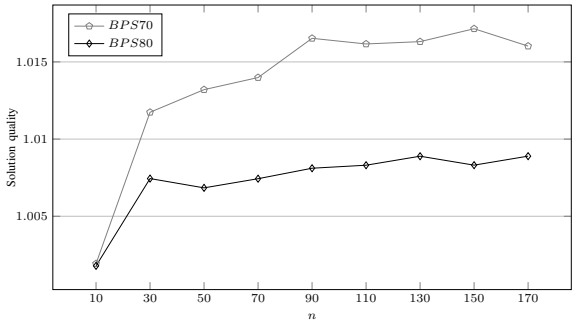

Fig. 7. $TABU5$ solution quality on $BPS70$ and $BPS80$ instances

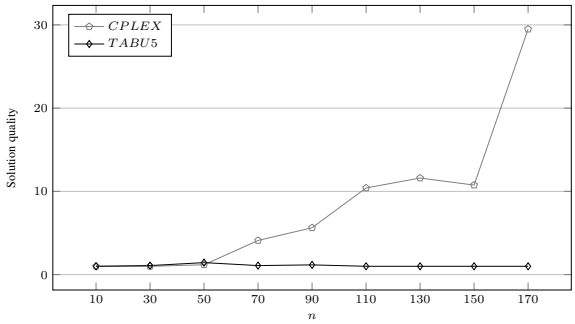

Fig. 8. $TABU5$ solution quality on $RAND$ instances in comparison to CPLEX

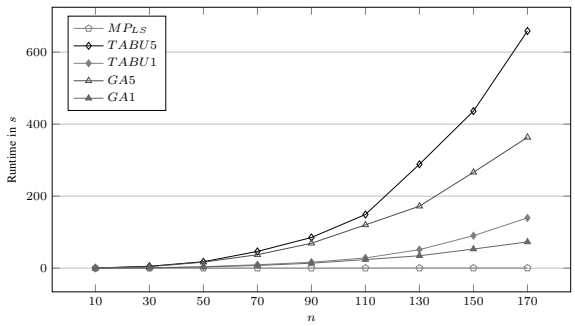

Fig. 9. Runtime comparison for $RAND$ instances

when compared with MP_{LS}.

However, when focussing on $BPS70$ and $BPS80$ instances, $TABU5$ proves to be competitive, as can be seen in Fig. 7. Additionally, it is worth mentioning that tabu search will outperform IBM ILOG CPLEX 12.5. Fig. 8 illustrates this fact by comparing the performance of CPLEX and $TABU5$ on the $RAND$ instances.

Finally, in Fig. 9 we compare the runtimes of our metaheuristic approaches and MP_{LS} on the $RAND$ instances. While runtimes increase roughly quadratically in n for all heuristics, both tabu search and the genetic algorithm require more computational effort than MP_{LS}.

VI. Summary

In this paper we have introduced an integrated matching and partitioning problem, the Partitioning Min-Max Weighted Matching Problem, with applications in intermodal transport. We have presented a mathematical formulation of the problem and proven its strong NP-hardness. Moreover, we have presented different heuristic and metaheuristic approaches for solving Partitioning Min-Max Weighted Matching. More specifically, we have implemented a tabu search and a genetic algorithm for solving Partitioning Min-Max Weighted Matching and we have found that tabu search outperforms the genetic algorithm in terms of solution quality. However, heuristic approaches that make use of the structure of the problem at hand by decomposing it into its matching and partitioning subproblems, perform better than the metaheuristic approaches under consideration. Nevertheless, all heuristic approaches outperform a standard commercial solver (CPLEX), which has shown to not perform competitive in computational tests.

References

[1] N. Boysen, M. Fliedner, F. Jaehn, and E. Pesch, "A survey on container processing in railway yards," *Transport. Sci.*, vol. 47, no. 3, pp. 312–329, 2013.

[2] N. Boysen and M. Fliedner, "Determining crane areas in intermodal transshipment yards: The yard partition problem," *Eur. J. Oper. Res.*, vol. 204, no. 2, pp. 336–342, 2010.

[3] M. Barketau, E. Pesch, and Y. Shafransky, "Scheduling dedicated jobs with variative processing times," *J. Comb. Optim.*, vol. in press, 2015.

[4] ——, "Minimizing maximum weight of subsets of a maximum matching in a bipartite graph," *Discrete Appl. Math.*, vol. in press, 2015.

[5] D. Kress, S. Meiswinkel, and E. Pesch, "The partitioning min-max weighted matching problem," *Eur. J. Oper. Res.*, vol. 247, no. 3, pp. 745–754, 2015.

[6] P. Preston and E. Kozan, "An approach to determine storage locations of containers at seaport terminals," *Comput. Oper. Res.*, vol. 28, no. 10, pp. 983–995, 2001.

[7] E. Kozan and P. Preston, "Genetic algorithms to schedule container transfers at multimodal terminals," *Int. T. Oper. Res.*, vol. 6, no. 3, pp. 311–329, 1999.

[8] Isoloader. (2012) When to choose a straddle and when to choose a forklift. [Online]. Available: http://www.isoloader.com/downloads/

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[10] F. Glover and M. Laguna, *Tabu Search*. Dordrecht: Kluwer, 1997.

[11] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.

[12] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge: MIT Press, 1996.

[13] B. Chen, C. N. Potts, and G. J. Woeginger, "A review of machine scheduling: Complexity, algorithms and approximability," in *Handbook of Combinatorial Optimization, Vol. 1*, D.-Z. Du and P. M. Pardalos, Eds. Boston: Kluwer, 1999, pp. 1493–1641.

[14] C.-Y. Lee and J. D. Massey, "Multiprocessor scheduling: Combining LPT and MULTIFIT," *Discrete Appl. Math.*, vol. 20, no. 3, pp. 233–242, 1988.

[15] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logist. Quat.*, vol. 2, no. 1-2, pp. 83–97, 1955.

[16] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia: Siam, 2009.

[17] W. Domschke and A. Scholl, *Grundlagen der Betriebswirtschaftslehre*, 3rd ed. Berlin: Springer, 2005.

[18] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.