



RICARDO EULER¹, RALF BORNDÖRFER², TIMO STRUNK,
TUOMO TAKKULA

ULD Build-Up Scheduling with Dynamic Batching in an Air Freight Hub

¹  0000-0001-5112-4191

²  0000-0001-7223-9174

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30 84185-0
Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de
URL: <http://www.zib.de>

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

ULD Build-Up Scheduling with Dynamic Batching in an Air Freight Hub

Ricardo Euler¹, Ralf Borndörfer¹, Timo Strunk², and Tuomo Takkula²

¹ Zuse Institute, Takustraße 7, 14195 Berlin, Germany,

² Ab Ovo Germany GmbH, Prinzenallee 9, 40549 Düsseldorf, Germany

Abstract. Air freight is usually shipped in standardized unit load devices (ULDs). The planning process for the consolidation of transit cargo from inbound flights or locally emerging shipments into ULDs for outbound flights is called build-up scheduling. More specifically, outbound ULDs must be assigned a time and a workstation subject to both workstation capacity constraints and the availability of shipments which in turn depends on break-down decisions for incoming ULDs. ULDs scheduled for the same outbound flight should be built up in temporal and spatial proximity. This serves both to minimize overhead in transportation times and to allow workers to move freight between ULDs. We propose to address this requirement by processing ULDs for the same outbound flight in batches.

For the above build-up scheduling problem, we introduce a multi-commodity network design model. Outbound flights are modeled as commodities; transit cargo is represented by cargo flow volume and unpack and batch decisions are represented as design variables. The model is solved with standard MIP solvers on a set of benchmark data. For instances with a limited number of resource conflicts, near-optimal solutions are found in under two hours for a whole week of operations.

Keywords: Logistics, Airline Applications

1 Introduction

Air freight is usually shipped in standardized unit load devices (ULD). Often these ULDs are routed through a hub airport. As they frequently contain freight for multiple destinations, they need to be unpacked (break-down) and reconsolidated (build-up). An intricate scheduling problem thus arises at the hub airport: Outbound ULDs need to be scheduled for reconsolidation in time for their departure while respecting constraints imposed by the availability of workstations and workforce. The amount of available shipments in turn is a function of break-down decisions for inbound ULDs subject to similar resource constraints.

Since many shipments cannot be stacked arbitrarily and also often come in odd shapes, it is desirable to build up multiple ULDs destined for the same flight simultaneously and in spatial proximity in order to facilitate better packing options. An easy model of proximity is a partition of the workstations into groups.

We refer to a set of identical ULDs for the same flight scheduled at the same time in the same workstation group as a batch. In general, it is not allowed to keep shipments that do not fit into an ULD in the build-up area. Instead, they have to be moved back to the warehouse. Hence, a welcome side effect of build-up in batches is a reduction in the number of movements necessary between the warehouse and the build-up area. From a modeling perspective, considering batches instead of individual ULDs reduces the amount of variables to consider, since outgoing ULDs of the same type (e.g. a container or pallet) are indistinguishable and need no longer be represented individually. Inbound ULDs, however, can be distinguished by their freight and also do not benefit from being deconsolidated in batches. Hence, they are not treated as such. We call the resulting scheduling problem the build-up scheduling problem with dynamic batch building (BSP).

Build-up scheduling (without batches) is categorized as one step of the sequential air cargo load planning problem in [3], which also contains a comprehensive literature review of related problems. The authors survey three modeling approaches for the scheduling of personnel for ULD build-up [7,5,6]. Among these, [6] also schedules workers for break-down operations. However, build-up and break-down demand are parameters and not interdependent in their model. All of these models only consider personnel scheduling and do not take individual ULDs, batches or workstations into account. A variant of build-up scheduling without batches is studied in [2]. The same author also introduced the benchmark instances [8] on which we base our computational study. Recently, [4] studied the problem of scheduling both personnel and batch build-ups under constraints on the availability of workstations. Their model treats the creation of batches from incoming cargo as a preprocessing step such that batches appear as jobs with a definite release time, dead-line and resource consumption. Here, workstations are not split into groups. Our approach differs from both [4] and [2] in several key aspects. First, we do not consider explicit personnel constraints but assume these to be implicitly given by the availability of workstations. Secondly, we also consider break-down processes and, thirdly, we aim to maximize the size of batches in workstation groups. To the best of our knowledge, this is the first work to incorporate dynamic batch building and interdependent build-up and break-down scheduling into a single model.

Despite the name, BSP is difficult to classify using classical scheduling notation (see e.g. [1]). BSP consists of two parallel processor scheduling problems connected by cargo flow constraints. Here, the value of a build-up job in the objective function depends on the amount of freight made available by finished break-down jobs. Note however, that this is not a precedence relationship. In fact, outbound ULDs might be constructed even if few or no relevant inbound ULDs are unpacked and maximizing the amount of cargo placed in an outbound ULD is part of the objective function.

2 A Multi-Commodity Network Design Model with Edge Activity

BSP can be addressed using a network design approach. Let $T = (t_1, \dots, t_{|T|})$ be a discretized time horizon for which ULD build-ups and break-downs need to be scheduled. We are given a set of ULD types V with capacities $c_v \in \mathbb{N}$ and build-up times b_v for all $v \in V$. We denote the set of departing flights by K . Each departing flight $k \in K$ has a departure time δ_k , a freight demand d_k , a (financial) cost of one unit of unshipped cargo l_k and a number of pre-planned ULDs $p_{v,k}$ for all $v \in V$. Each ULD requires a workstation for consolidation. We aggregate workstations that are close to each other into disjoint workstation groups W . The number of workstations in a group $w \in W$ is its capacity c_w . Inbound ULDs I are already assigned a type and an inbound flight in the input data. Therefore, we directly assign each of them a break-down duration β_i , a freight volume $\lambda_{i,k}$ for all outgoing flights $k \in K$ and an arrival time α_i .

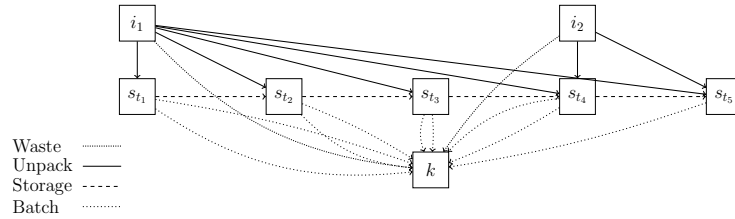


Fig. 1. The time-expanded cargo flow network of the build-up scheduling problem for two inbound ULDs and one departing flight. Here, i_1 and i_2 are sources and k is a sink. The time horizon consists of five time points. Build-up, break-down durations and departure and arrival times of flights are factored into the construction of the graph and determine the presence of edges.

We define a candidate batch decision $b \in B$ to be a five-tuple $(t_b, n_b, k_b, v_b, w_b)$ where $n_b \in \mathbb{N}$ is the number of ULDs in the batch, $t_b \in [0, \delta_{k_b} - b_{v_b}] \cap T$ the starting time of build-up, $v_b \in V$ the ULD type used, $k_b \in K$ the outgoing flight and $w_b \in W$ the assigned workstation group. An ULD unpack decision $u \in U$ is a tuple (t, i) with $i \in I$ and $t \in [\alpha_i, t_{|T|}] \cap T$.

Now, BSP can be formalized as finding a set of decisions $D = \bar{B} \cup \bar{U}$ with $\bar{B} \subset B$ and $\bar{U} \subset U$ such that no flight k is assigned more ULDs of type v than $p_{k,v}$, no more than c_w ULDs are scheduled for any workstation group $w \in W$ at any given point in time, the general storage capacity is never exceeded and no inbound ULD i is unpacked more than once while both minimizing the amount of unshipped cargo and maximizing the average batch size. Minimizing freight losses is essential for customer satisfaction, while maximizing batch size has organizational benefits. Hence, we treat the BSP as a single-objective problem using a parametrization that prioritizes loss avoidance over batch building.

To solve BSP, we propose a time-expanded fixed-charge multi-commodity network design model in which the arcs represent unpack and batch decisions or the storage unit. Consider the network $\mathcal{N} = (I \cup S \cup K, A)$, where nodes in $S = \{s_{t_1}, \dots, s_{t_{|T|}}\}$ represent the storage facility at various time points. Then, we define the arc set A as follows. For each unpack decision $u = (t, i) \in U$ an arc $(i, s_{t+\beta_i})$ is created. For each batch decision $b = (t_b, n_b, k_b, v_b, w_b) \in B$, an arc (s_{t_b}, k_b) is created if $t_b + b_{v_b} \leq \delta_{k_b}$. Note that this introduces multi-arcs and that the multiarcs between s_t and k represent all possible batches one can start building for k at time point t . Arcs $(s_{t_j}, s_{t_{j+1}})$ are introduced for all $t_j \in \{t_1, \dots, t_{|T|-1}\}$ representing cargo kept in the storage facility during the time interval $[t_j, t_{j+1}]$. Finally, arcs (i, k) are added for all $i \in I, k \in K$. Cargo that is routed along these arcs is considered unscheduled and penalized with high weights. We write $\mathcal{A} \subset A$ with $\mathcal{A} := B \cup U$. The set \mathcal{A} are the design arcs, while arcs in $A \setminus \mathcal{A}$ are always active. An example of the resulting network can be found in Figure 1. In our MIP formulation, all batches that differ only in their size correspond to columns that are multiples of each other. To mitigate this degeneracy, we introduce *activity* variables and reduce batches $b \in B$ to four-tuples $b = (t_b, k_b, v_b, w_b) \in B$.

Then, the build-up scheduling problem can be formulated as the following MIP:

$$\min \sum_{a \in \mathcal{A}} w_a x_a + \sum_{k \in K} \sum_{a \in \mathcal{A}} w_a^k f_a^k \quad (1)$$

$$\text{s.t.} \quad \sum_{k \in K} d_k f_a^k \leq c_a \quad \forall a \in A \setminus \mathcal{A} \quad (2)$$

$$\sum_{k \in K} d_k f_a^k \leq c_a y_a \quad \forall a \in \mathcal{A} \quad (3)$$

$$\sum_{a \in \delta^+(v)} f_a^k - \sum_{a \in \delta^-(v)} f_a^k = \gamma_v^k \quad \forall k \in K \forall v \in V \quad (4)$$

$$f_a^k \leq x_a \quad \forall k \in K \forall a \in \mathcal{A} \quad (5)$$

$$\sum_{a \in \mathcal{A}} \alpha_a^r y_a \leq L^r \quad \forall r \in R \quad (6)$$

$$y_a \leq M_a x_a \quad \forall a \in \mathcal{A} \quad (7)$$

$$x_a \in \{0, 1\} \quad \forall a \in \mathcal{A} \quad (8)$$

$$y_a \in [0, M_a] \cap \mathbb{Z} \quad \forall a \in \mathcal{A} \quad (9)$$

$$f_a^k \in [0, 1] \quad \forall a \in \mathcal{A} \forall k \in K. \quad (10)$$

Here, f_a^k is the amount of cargo for flight k passed along an arc $a \in \mathcal{A}$, x_a indicates whether a is active and y_a represents the number of ULDs constructed on a . Depending on the arc type, constraints (2) and (3) impose bounds on storage or ULD capacity. Constraints (4) ensure flow conservation with $\gamma_v^k := \lambda_{i,k}/d_k$ if $v \in I$, $\gamma_v^k := 1$ if $v \in K$ and $\gamma_v^k := 0$ otherwise. Note that $\sum_{i \in I} \gamma_{i,k} = 1 \forall k \in K$. Constraints (5) ensure that flow only passes through active arcs. Finally, constraints (6) summarize resource limits on active arcs. These are: Ensuring that for each $i \in I$ only one unpack arc is active, that batch activity for flight $k \in K$ and ULD type $v \in V$ is smaller than $p_{k,v}$ and finally that for all $t \in T$ and $w \in W$ the workstation utilization is at most c_w . By introducing costs on the slack of the second type of resource constraints, penalties for planned but offloaded ULDs can be introduced. Constraints (7) limit the activity of active arcs. Here, for a

batch arc $b \in B$ we have $M_b = \min\{c_{w_b}, p_{v_b, k_b}, \lceil \frac{d_{k_b}}{c_{v_b}} \rceil\}$. Note that $M_u = 1$ for all $u \in U$. In the objective function, we set $w_a^k = d_k l_k$ if $a = (i, k) \in I \times K$ and $w_a = 0$ otherwise. Also $w_a = 1$ if $a \in B$ and $w_a = 0$ otherwise. Hence, we aim to minimize the number of batches in order to maximize the average batch size. As a consequence, late build-up is incentivized.

3 Computational Study

We based our computational study on the data set provided in [2,8]. The data set consists of shipments extracted from anonymized real-world booking data that is randomly assigned to a real flight schedule of one week with 82 outbound flights. In cooperation with our industry partner Ab Ovo Germany GmbH, we augmented this data as follows. We created 28 time horizons from the week consisting of all possible combinations of between one and seven consecutive days. In the original data set each shipment has a release time at which it becomes available. We grouped shipments with similar arrival times together to form an inbound ULD using a randomly drawn ULD type. The ULD's arrival time is its earliest shipment's release time minus its break-down time. Workstation groups are not part of the data set. We modeled these around settings which appeared sensible to us and our industry partner. We created three different set-ups with 12, 24 and 48 workstations partitioned in groups of six. Hence, the testset consists of 84 instances in total. We did not apply any storage capacity or other

#WS	Off	Cap	Opt	Free		Enum		Activ			
				Gap	t	Opt	Gap	t	Opt	Gap	t
12ws	•	66	1	2.16	6943.85	0	13.12	7205.23	0	4.78	7201.80
24ws	•	66	28	0.00	13.31	28	0.00	156.96	25	0.01	2229.53
48ws	•	66	28	0.00	22.97	28	0.00	152.62	28	0.00	1501.74
12ws	•	90	4	1.83	6174.67	0	13.17	7206.93	0	5.84	7202.44
24ws	•	90	28	0.00	58.55	28	0.00	151.91	28	0.00	137.02
48ws	•	90	28	0.00	202.59	28	0.00	129.86	28	0.00	117.86
12ws	◦	66	7	4.29	5405.67	3	12.65	6447.57	3	7.28	6511.30
24ws	◦	66	28	0.00	8.85	5	0.04	5926.12	8	0.03	5468.70
48ws	◦	66	28	0.00	15.27	5	0.07	6011.70	7	0.05	5644.11
12ws	◦	90	28	0.00	204.73	5	15.26	6128.94	4	8.72	6255.31
24ws	◦	90	28	0.00	44.97	6	1.55	5966.61	5	1.53	5959.77
48ws	◦	90	28	0.00	147.67	4	2.02	6444.38	4	1.87	6193.93

Table 1. Computational results. *Activ* and *Free* refer to the formulation as defined in Section 2 with and without batch costs, respectively. *Enum* refers to a standard network design formulation with batch costs. For all 12 scenarios defined by the number of workstations (*#WS*), offload penalties (*Off*) and usable ULD capacity (*Cap*), we report the number of instances solved to optimality (*Opt*), the average gap (*Gap*, %) and the average run time (*t*) in seconds. Run times (of the MIP solver) were capped at 7200s.

restrictions to ULD break-down in this study. Thus, the problem is reduced to a variant of the problem studied in [2] without personnel constraints but using the additional batching objective and workstation groups. We investigated scenarios both with offload penalties for unscheduled outbound ULDs calculated following [2] and without. In line with [2], we assumed that 66% of nominal

ULD capacity is available. As this resulted in severe overbooking of some flights, we additionally investigated scenarios with 90% capacity. We implemented the MIP model of Section 2 with (*Activ*) and without nonzero (*Free*) costs for batch variables. Additionally, we implemented a standard network design formulation without activity variables, where batch sizes are explicitly enumerated (*Enum*). All implementations were carried out in Python3 using Gurobi 9.1.2. All tests were conducted on a Dell PowerEdge M620v3. Results are reported in Table 1. We found that BSP becomes easier when more workstations are available. If 48 workstations are present, this effect diminishes, most likely due to the increased number of integer variables. BSP is generally easier when minimizing the number of batches is not part of the objective. This effect is distinctly more pronounced in scenarios without offload penalties. In general, offload penalties result in more instances solved to optimality and lower run times. We believe this to be due to a reduction in dual degeneracy. Without penalties, scheduling additional empty ULDs has no effect on the objective function, which can lead to the presence of a high number of optimal solutions. Comparing *Enum* and *Activ*, we find that introducing activity variables gives mixed results. *Activ* reports a lower gap in all but one scenario. In scenarios without offload penalties, *Activ* solves three more instances to optimality and reports lower run times in four out of six scenarios. When offload penalties are applied, however, *Activ* reports significantly higher run times than *Enum* for ULD capacities of 66% while being slightly faster for ULD capacities of 90%. These scenarios differ mainly in the amount of offloaded freight. The reason for this variation in performance is not yet well understood.

In conclusion, the BSP proves challenging especially when workstation resources are scarce and offloads are not penalized. The uneven performance of the activity-based formulation warrants further investigation. Future work will also include the scheduling of break-down operations.

References

1. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Handbook on scheduling: from theory to applications. Springer Science & Business Media (2007)
2. Brandt, F.: The air cargo load planning problem. Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2017). DOI 10.5445/IR/1000075507
3. Brandt, F., Nickel, S.: The air cargo load planning problem - a consolidated problem definition and literature review on related problems. *European Journal of Operational Research* **275**(2), 399–410 (2019). DOI <https://doi.org/10.1016/j.ejor.2018.07.013>.
4. Emde, S., Abedinnia, H., Lange, A., Glock, C.H.: Scheduling personnel for the build-up of unit load devices at an air cargo terminal with limited space. *OR Spectrum* **42**(2), 397–426 (2020). DOI 10.1007/s00291-020-00580-2.
5. Nobert, Y., Roy, J.: Freight handling personnel scheduling at air cargo terminals. *Transportation Science* **32**(3), 295–301 (1998). DOI 10.1287/trsc.32.3.295.
6. Rong, A., Grunow, M.: Shift designs for freight handling personnel at air cargo terminals. *Transportation Research Part E: Logistics and Transportation Review* **45**(5), 725–739 (2009). DOI <https://doi.org/10.1016/j.tre.2009.01.005>.

7. Yan, S., Chen, C.H., Chen, M.: Stochastic models for air cargo terminal manpower supply planning in long-term operations. *Applied Stochastic Models in Business and Industry* **24**(3), 261–275 (2008). DOI <https://doi.org/10.1002/asmb.710>.
8. ACLPP Instances. <https://github.com/fbrandt/ACLPP>. Commit: 3516c2b