The Institute for Integrating Statistics in Decision Sciences

Technical Report TR-2014-1
January 16, 2014

The Destination-Loader-Door Assignment Problem for Automated Package Sorting Centers

Ahmad I. Jarrah
Department of Decision Sciences
The George Washington University, USA

Xiangtong Qi
Industrial Engineering and Engineering Management
Hong Kong University of Science and Technology, Hong Kong

Jonathan F. Bard
Graduate Program in Operations Research and Industrial Engineering
The University of Texas, USA

The Destination-Loader-Door Assignment Problem for Automated Package Sorting Centers

Abstract

This paper presents a new model and solution procedure for a problem that arises in configuring package sorting centers that perform multiple automated sorts per day. For a given set of loading bays, the first objective is to assign destinations to consecutive doors so that the number of changes of destination-to-door assignments from one sort to the next is minimized. The second and third objectives are to minimize the number of loaders who work the doors and to evenly distribute the volume of packages assigned to each loader. A variety of constraints vastly complicates these assignments and leads to a mixed-integer programming (MIP) model, which we significantly strengthened with structurally derived cuts. A novel feature of the formulation is the use of pattern variables to represent the door assignments. Taking a multiobjective programming approach, solutions are obtained by solving a series of MIPs, each addressing one of the three objectives. The effectiveness of our approach is demonstrated using data provided by a well-known package carrier for 24 workcenters in four facilities. An additional contribution is a complexity analysis of the destination-to-door and the loader-to-door assignment subproblems. Both are shown to be strongly NP-hard. We also examine special cases of the loader subproblem and develop polynomial time algorithms for them.

Key words. transshipment; package carriers; postal services; automated sortation; multi-criteria optimization; cross-docking; mixed-integer programming; workforce planning.

1. Introduction

Freight transportation carriers that provide long haul services typically operate national networks consisting of regional hubs, local terminals, and a sizable fleet of vehicles. One major differentiator between the various carriers in the industry is the degree of automation used during the sortation process at the regional hubs. For example, less-than-truckload (LTL) consolidators sort freight at their hubs predominantly using manual labor and forklifts in essentially one continuous stream. The main impediment to automating material handling operations for such companies is the high variability in the shape, size and weight of the shipped items. In contrast, package carriers such as DHL, FedEx and UPS, and the package divisions of many postal services, have installed sophisticated automated sorting systems in their regional hubs that enable them to efficiently process inbound packages and sort them to their down line destinations. Our focus here is on planning the layout and processing requirements for the automated sorting facilities of package carriers, with the context specifics and associated datasets obtained from the ground network of a well-known international carrier.

When a package is picked up, it is first delivered to the local terminal, typically in the late afternoon, and processed overnight. If its final destination is in the local terminal's service area, it is delivered the next day; if not, it is dispatched to a regional hub where it is sorted and then shipped to either another regional hub, or to the final local terminal from which it is then delivered to its final destination. A typical package may be sorted at one, two, or even three hubs before reaching its recipient. The path it takes through the network and its arrival date depend primarily on its designated service standard.

The carrier's hubs use sophisticated material handling equipment. At induction, packages are off-loaded from the inbound trailers and placed on a multi-tier, multi-spur conveyor system. A sorting facility has multiple outbound loading blocks consisting of 20 to 30 doors, which we shall refer to as workcenters. Each workcenter serves a subset of the destinations for which the hub serves as a transshipment point. A "primary" sort is performed at a very high speed and is used to direct the unloaded packages to the various workcenters. This is followed by a much more detailed "secondary" sort that further directs each package to a specific outbound loading door within a workcenter. The two sorts are done in a continuous flow manner without intermediate manual intervention. Once sorted, all outbound packages are loaded onto trailers and dispatched.

Over the day, a hub may receive 20,000 or more packages/hr that are ultimately shipped to one of roughly 150 destinations. Because packages are quite varied in size and shape, effective loading of outbound trailers from the sorting hub is a time consuming and challenging task. Accordingly, the outbound loading doors often represent binding resources that have to be managed efficiently. Packages arrive at the hub throughout the day in batches that vary as to the destinations being served, and the amount of flow to each of these destinations. In order to effectively use the outbound loading doors, the sortation is performed in multiple "shifts" with pre-defined time windows and modified destination-to-door assignments. For the package carrier studied, four shifts of roughly equal length define a day. A 1-hour break separates each shift and is used to swap or dispatch trailers and reconfigure the doors, if necessary. The break also serves as a buffer for extending the sorts if necessary on the actual days of operation. Our focus is on the design problem associated with operating the hub during the secondary sort. When planning how the facility will operate, two interrelated, shift-dependent decisions must be made that affect both cost and efficiency. The first concerns the assignment of destinations to doors and

the second, the assignment of loaders to doors for each of the four shifts. Once these decisions are made, they generally remain in force for months, but are adjusted seasonally as well as when there is a noticeable change in demand.

The corresponding problem is highly combinatorial but simplifies somewhat due to the design and layout of the material handling system. In particular, it decomposes by workcenter but remains difficult due to a series of practical and physical constraints and the need to address the following three objectives in a hierarchical manner: (i) minimize the number of changes in destination-to-door assignments ("switches") from one shift to the next, (ii) minimize the number of workers, called *loaders*, required to load the trailers, and (iii) balance the workload amongst the loaders. The first objective is critical because modifying destinations at doors requires significant managerial oversight, and is prone to error and delays, as explained shortly.

With these objectives in mind, the main contribution of this paper is the presentation of a new model for the integrated *destination-loader-door assignment problem* (DLDAP) and an efficient hierarchical optimization scheme for finding high-quality solutions. A second contribution centers on the determination of the theoretical complexity of the individual destination-door and loader-door assignments subproblems, which can be extended to any two-tier assignment problem with similar characteristics. Finally, we also provide polynomial-time algorithms for the latter subproblem for two simplified cases where the destination order on each sort is specified.

In Section 2, we provide an overview of related literature, while in Section 3, we describe the DLDAP in more detail focusing on a specific facility. This is followed in Section 4 with our complexity results for the two subproblems that define the DLDAP, and a summary discussion of the complexity of two special cases of the loader-door assignment subproblem (LDAP). In Section 5 the full mixed-integer programming model for the DLDAP is presented, followed in Section 6 with the details of our hierarchical solution procedure that treats each objective in sequence. Constraints are added to the second and third problems to limit the degree to which optimal values previously obtained can be relaxed. In Section 7 we highlight the computational results for the DLDAP, and show that optimal or near-optimal solutions for 24 representative instances reflecting current practice can be obtained in reasonable time. We close with an assessment of the approach and suggestions for future research. Appendix A provides additional details on the complexity analysis.

2. Related Literature

In previous work, we investigated equipment requirements at U.S. Postal Service mail processing and distribution centers (P&DCs) and developed weekly operational schedules (Jarrah et al. 1994a, Zhang and Bard 2005, 2006). In a complementary effort, we designed procedures for structuring the permanent workforce (Jarrah et al. 1994b, Bard et al. 2003) and for providing weekly updates with respect to overtime and the use of casual labor to meet fluctuating demand (Wan and Bard 2007). In this paper, we concentrate on planning destination and loader assignments for the workcenters at the automated sorting hubs of package delivery carriers. The logistics required for package delivery are different in their details from those required for mail handling, primarily due to the significant differences in the physical characteristics of packages and regular mail.

While this is the first paper to address the DLDAP, there is some related published research. The closest is the work related to cross-docking for package delivery or less-than-truckload (LTL) operations. Werners and Wülfing (2010) addressed the assignment of package groupings to staging "endpoints" and

outbound loading gates (or doors) within a package sorting facility with the objective of minimizing the overall distances involved in manually transporting packages from the endpoints to the gates. This contrasts with the sorting environment that we are modeling where automated conveyors directly deliver each package to its outbound door instead of an endpoint. The authors used a hierarchical decomposition scheme to obtain solutions while incorporating some robustness considerations. Conceptually, there is a similarity between their problem and ours in that a two-level assignment is involved (package groups to endpoints and gates, and gates to tours). However, our context is significantly different so their modeling approach is not readily applicable. McAree et al. (2006) analyzed possible package sorting facility designs for hubs that primarily depend on the use of forklifts, rather than automated material handling equipment. Using two MIP models they demonstrated that optimized designs can reduce the expected total forklift travel time by 33% in comparison to manually prepared designs.

Bozer and Carlo (2008) studied the problem of making inbound and outbound trailer-to-door assignments in cross-dock facilities. Their objective was to minimize the overall material handling workload in an LTL environment. Solutions were provided with a simulated annealing algorithm for facilities with up to 118 doors. Choy et al. (2011) addressed a cross-docking problem that arises at space-constrained logistics distribution hubs. The difficulty they faced was that the number of incoming trucks during the day exceeded the number of available docks, and that arrivals were random. Their objective was to minimize the waiting time of trucks by coordinating the pickup and delivery sequences of inbound and outbound orders in the storage areas. Solutions were found with a genetic algorithm. At a high level, their problem is reverse of ours in that they assign inbound trucks to doors rather than outbound doors to destinations as we do. However, they only consider a single sort and do not take into account loader requirements. More recently, Liao et al. (2013) proposed and evaluated several metaheuristics for inbound truck sequencing and dock assignment.

McWilliams et al. (2005) also investigated the problem of scheduling a set of inbound trailers to a fixed number of unload docks at freight consolidation terminals. Their objective, though, was to minimize the time span of the parcel transfer operation without regard to trailer movement or the use of manual labor. A simulation-based scheduling procedure that used a genetic algorithm to drive the search for new solutions was proposed and tested on random instances with up to 160 unload bays.

Another area of research that is tangentially related to ours concerns airport gate assignments (e.g., see Dorndorf et al. 2007, Yan et al. 2011). A closer look, though, reveals that the issues are much different than those associated with the DLDAP. Similarly, the work on staff scheduling has an assignment aspect to it but the constraints and objectives are far afield (e.g., see Ernst et al. 2004). For example, Campbell and Diaby (2002) developed an assignment heuristic for allocating cross-trained workers to multiple departments at the beginning of a shift. Each worker had different qualifications with respect to each department. The problem was formulated as a variant of the generalized assignment problem with a concave objective function that measured department preferences.

Abdelghany et al. (2006) consider the airport baggage sorting station assignment problem (ABSSAP), which is concerned with assigning flights to specific piers within baggage handling quads, with the two goals of minimizing the number of crewmembers required and balancing their workloads while adhering to a host of airline operational constraints. A conceptual similarity to our research is in the two-level assignment of crewmembers and flights to piers, which is analogous to the assignment of loaders and destinations to doors. The other similarities are in the common objectives of minimizing and

balancing the workforce, and in the desirability of retaining the same pier assignment for flights that are operated multiple times within the same day (same doors for destinations, in our case). They reduce the two-level assignment to, in effect, a one-level problem by adopting the surrogate objective of reducing the number of piers used at any point in time, which should be conducive to reducing manpower costs. A greedy heuristic was developed to assign flights to piers.

Ascó et al. (2011) also addressed the ABSSAP and studied several objectives including maximization of assignments (this is really a hard constraint but they indicate that in some under-capacitated situations, it may not be possible to achieve a feasible solution, hence leading to flight delays), minimization of some measure of distance cost associated with assignments, maintaining reasonable buffer times between sorts, and workload balance, which is similar to our third objective. The purpose of their paper was to describe and evaluate several constructive heuristics. In Ascó et al. (2012), the authors discussed their evolutionary algorithm for the ABSSAP using the first three objectives. Despite the similarities between the ABSSAP and the DLD assignment problem at a conceptual level, the details and structure differ considerably so that models are not interchangeable. Also, instead of heuristics, we opt to use and solve a novel, strong MIP formulation for the two-level assignment problem with side constraints that accurately reflects the very different physical and operational constraints of the multi-sort parcel environment.

3. Facility Description and Problem Statement

Although no two hubs are identical, they all possess many of the same structural characteristics and have identical setup requirements. This is true across the industry for carriers with automated sorting capabilities. To add definition to the discussion we use a Southwestern hub of a major package carrier as a prototype. A schematic of the facility, which has three wings A, B and C where the secondary sort takes place, is shown in Figure 1. At the far side of A-Wing, there is a stretch of “unload” doors that feed four primary sort conveyors that are identified at the top-left side of the diagram. The primary sort is done at a high level and continues as long as there are arriving vehicles. In all, there are eight secondary workcenters (technically, they are called “sorters” but to avoid confusion, we refer to them as workcenters) denoted by A1, A2, B1, B2, B3, B4, C1, C2 that are located on the remaining sides of the wings. A1 and A2, for example, are on the inner side of A-Wing.

The local city conveyor on the left side of the schematic represents the work area in the hub that services neighboring towns. Those packages destined for local delivery are transported using an urban fleet rather than the long-haul fleet. Adjacent to the facility is a storage yard where the tractors idle until their trailers are ready to depart, and where empty trailers are staged prior to being moved to the loading doors. A group of small, specialized vehicles move the trailers between the yard and doors as needed.

Vehicles start arriving in the late morning from the national lanes as well as from the local areas and continue throughout the day. During each of four 4-hour shifts, the following separate secondary sorts take place.

Sort 1 (*Day* 2:00 – 6:00 pm). Typically the largest volume and a wide cross-section of destinations.

Sort 2 (*Twilight* 7:00 – 11:00 pm). The majority of volume is for local and regional destinations.

Sort 3 (*Midnight* 12:00 – 4:00 am). The focus is on "hot" lanes (primarily next day, and some two-day) with tight service requirements.

Sort 4 (*Sunrise* 5:00 – 7:30 am). The lowest volume mainly headed for terminals close to the hub but includes some distant destinations; rarely requires 4 hours.

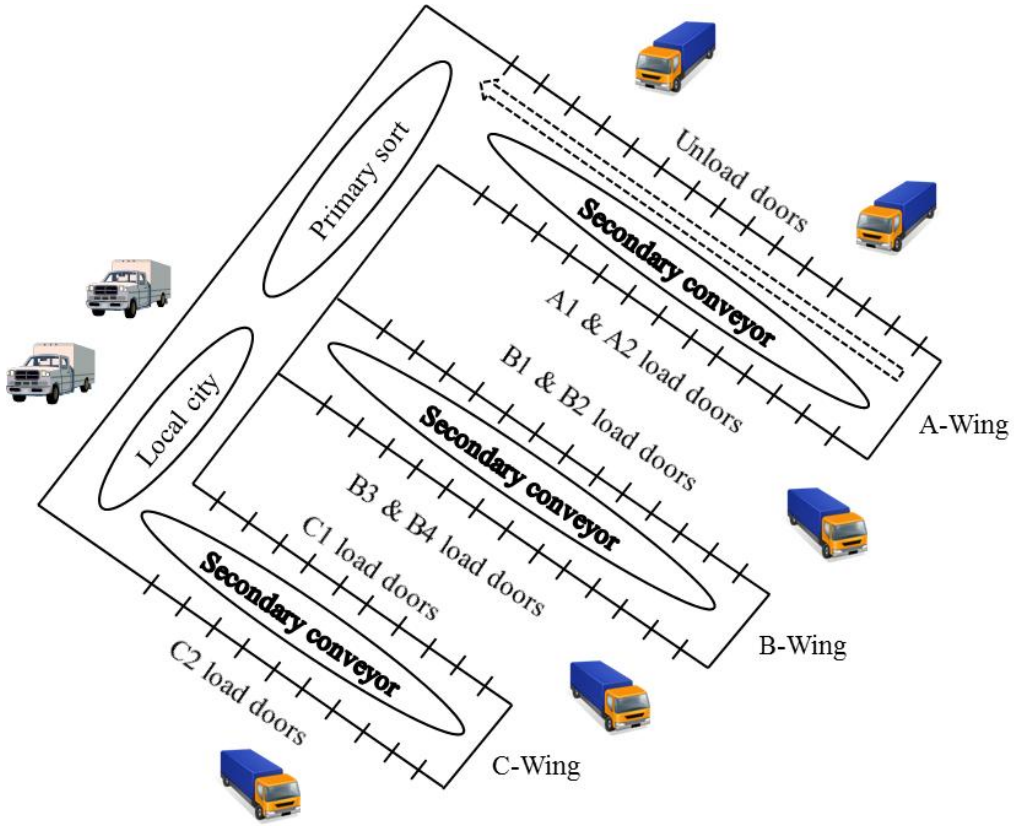

Figure 1. Schematic of representative hub

Each sort is independent of the others but the lineup of doors between sorts is critical. A trailer remains at a door until it is full or when the "cutoff" time required to make service arrives, whichever comes earlier. Because partially full trailers can often be held for further loading in subsequent sorts, it is very important to minimize the number of destinations assigned to a door. If Dallas is assigned to door 1 on sort 1 and Memphis on sort 2, then the Dallas trailer has to be moved either to the yard or to another door on sort 2 and replaced with a Memphis trailer. Although it only takes 5 to 10 minutes to switch out a trailer, it can become a logistical nightmare to move more than a few. To add precision to what we mean by a switch, we have the following.

Definition 1. A *switch* is associated with a single door and occurs when a trailer associated with a particular destination is removed from the door and replaced with a trailer for a different destination.

More importantly, changes in destination assignments are undesirable from a managerial perspective because of their disruptiveness. Switching a destination for a loading door between sorts necessitates entering the modified lineup in the computer system that controls the automated flow of the sorted packages, and communicating the changes to the drivers, loaders, and dock supervisors in a timely fashion to ensure operational integrity. Failures in communication may result in loaders working at the

wrong doors, or even drivers hooking up trailers to the wrong outbound doors, both of which can be time consuming mistakes that require immediate corrective action when detected.

Accordingly, modifying destination-to-door assignments correspond to "planned disruptions," and should not be used unless "necessary." Incorporating switches into the periodic (e.g., monthly) plan generated by the model to reduce the number of loaders is unrealistic. When the actual day-of-operation arrives, the flows are not likely to correspond to the flows used in generating the plan so the switches that were intended for reducing the number of loaders may actually turn out to have a detrimental, rather than beneficial, impact. As such, planned switches are only employed to ensure sufficient loading capacity (i.e., number of doors) exists for each of the destinations and sorts given the planned package flows. On the other hand, in a day-of-operations version of the model, using switches to reduce the number of loaders may be a reasonable approach.

In conclusion, from a managerial point of view, when generating periodic plans for the workcenters it is best to maintain what is called a *universal lineup* where the destination-door assignments remain static across all four sorts. This is illustrated in Figure 2 where doors 1, 2 and 3 are assigned to Dallas and doors 4 and 5 are assigned to Memphis. Notice that on sort 1, for example, doors 3, 4 and 5 are not used.

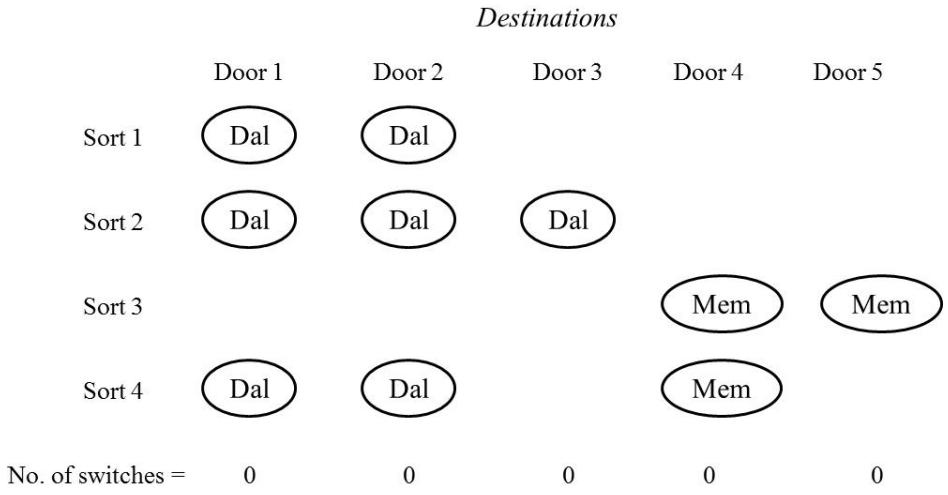

Figure 2. An example of a universal lineup

When a sufficient number of doors are available, a universal lineup is always possible. Limited capacity and increasing demand, however, conspire against this type of configuration. When space is tight, a lineup similar to the one shown in Figure 3 is more likely, and in fact, may be the only feasible option. Nevertheless, there is some advantage in reducing the number of empty doors and making the lineup more compact that relates to the use of loaders. This is discussed in subsequent sections. For planning purposes, each loader can handle a maximum of V^{max} packages/hr but this value decreases as the number of doors he is assigned increases. The productivity relationship is shown in Table 1, which

indicates that the maximum number of doors, call it $\overline{n}^{doors}$, that can be assigned to a loader is 5. The symbol V_n^{max} denotes the maximum processing rate of a loader when he works *n* doors.

Table 1. Productivity relationship for door assignments

	Doors/loader, *n*				
Parameters	1	2	3	4	5
Packages/hour, V_n^{max}	450	400	375	350	325
Productivity, ρ_n	1	0.89	0.83	0.78	0.72

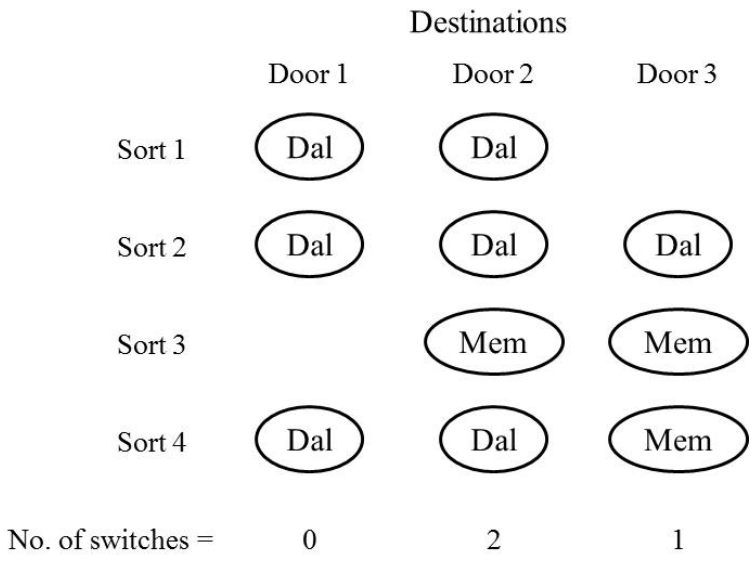

Figure 3. A lineup with destination-door switches between sorts

Two related constraints arise out of practical considerations. The first is that the destination-door assignments on each sort must be consecutive. This is illustrated in Figures 2 and 3 where there are no intervening destinations between the Dallas and Memphis doors. Empty doors within the door lineup for the same destination are permitted though, but only under limited circumstances. Such an arrangement may be desirable if it reduces the number of loaders required to handle the overall flow on a sort. Consider another example where on sort 1 Memphis has a normalized flow of 0.5 and is assigned to door 1, Dallas has a flow of 0.8 and is assigned to door 3, and Chicago has a flow of 0.4 and is assigned to door 5. Doors 2 and 4 are closed but are used for Dallas on sort 2 along with door 3. Now, if a loader can handle at most two doors, then three loaders are needed. However, if 0.35 of Dallas' flow is shifted to door 2 and the remainder is shifted to door 4, then only two loaders are needed.

The second restriction is that loaders must be assigned consecutive doors, again up to a maximum of $\overline{n}^{doors}$. Empty doors in sequence are counted towards $\overline{n}^{doors}$. Note that the consecutive door requirement is common to the facility plans of both package and LTL carriers.

We take as input the number of packages per hour sent to each secondary workcenter from the primary sorters. The planning is based on monthly averages. It is assumed that there is no interaction between the secondary workcenters so that each can be optimized independently. Additional assumptions follow.

1. All load docks are identical and can accommodate any trailer.
2. All loading is done manually by identical loaders.
3. No trailer can be preempted once a sort starts.
4. The number of doors assigned to any destination on any sort is no greater than the maximum required on any of the four sorts.
5. Each destination for which packages exist during a sort must be assigned at least one door. If more than one door is assigned, then they must be consecutive.
6. A door that is not used during a sort is not included when counting switches between sorts (e.g., see Figure 2, door 1).

Table 2 provides input data for the first workcenter at the Dallas, Texas hub. The flow is given in average packages/hr for October 2011. Of the 12 destinations listed, most have flow on all four sorts. The 13th destination labeled "Shipment integrity" corresponds to a door assigned to trap packages for a specific customer so they can all be shipped and delivered together. For example, a customer may have placed several orders with Staples and wishes that they all be delivered at the same time on a particular day. Because these doors don't affect the analysis, they are removed in a preprocessing step. The bottom row in the table indicates the minimum number of loaders needed for the workcenter and is obtained by dividing the total flow by the loader productivity V^{max} (= 450) and rounding up. We call this the *bin packing* lower bound.

Table 3 gives the number of doors needed by sort for each of the 12 destinations. The values were obtained by dividing the flow in Table 2 by 450 and rounding up. For example, the flow to Florence, SC on sort 1 is 1005 packages/hr, implying that at least $\lceil 1005/450 \rceil = 3$ doors are required. Subsequently, we use the symbol $\underline{n}_{ds}^{doors}$ for the destination *d*, sort *s* entry. The bottom row of the table indicates the minimum number of doors needed by sort but without regard to the number of switches that would be incurred if these values were used. In fact, this workcenter has 21 doors; in Section 5 we show that a universal lineup is possible with only 17 doors.

In the next section, we provide some complexity results for our problem. To recap, the DLDAP seeks to assign destinations and loaders to doors in a fashion that minimizes the number of switches and required loaders, while adhering to the following constraints:

- Each destination is assigned to a contiguous set of doors
- Each loader is assigned to a contiguous set of doors
- Each door can be used by at most one destination and serviced by at most one loader
- A maximum loading capacity applies per loader as indicated in Table 1
- The selected loaders can collectively process the package flows for all the destinations

4. Complexity Issues

The DLDAP is a composition of two assignment problems with side constraints. Because the basic assignment problem is polynomially solvable it is interesting to ask whether the DLDAP is as well, and if

not, whether it can be solved with a pseudopolynomial-time algorithm. As we now show, the answer is no for either case.

Table 2. Input data for a secondary workcenter at the Dallas hub

			Flow (packages/hr)			
			Sort 1	Sort 2	Sort 3	Sort 4
No.	Destination	Designation	Day	Twilight	Midnight	Sunrise
1	Augusta, GA	309	75	67	90	11
2	Houston, TX	772	12	27	87	54
3	Hartford, CT	61	51	209	374	96
4	Florence, SC	295	1005	649	295	16
5	Roanoke, VA	240	444	466	183	6
6	Fort Worth, TX	760	33	116	329	259
7	Hickory, NC	286	239	0	0	0
8	Memphis, TN	381	37	116	267	429
9	Greensboro, NC	273	920	564	328	10
10	Dallas (local), TX	753	64	129	381	331
11	Irving, TX	752	12	38	102	25
14	Toledo, OH	436	0	0	187	0
15	Shipment integrity	8002	0	0	0	0
		Total =	2892	2382	2624	1236
		Minimum no. loaders needed =	7	6	6	3

Table 3. Door requirements for the flow in Table 2

			No. of doors needed per sort, $\underline{n}_{ds}^{doors}$			
			Sort 1	Sort 2	Sort 3	Sort 4
No.	Destination	Designation	Day	Twilight	Midnight	Sunrise
1	Augusta, GA	309	1	1	1	1
2	Houston, TX	772	1	1	1	1
3	Hartford, CT	61	1	1	1	1
4	Florence, SC	295	3	2	1	1
5	Roanoke, VA	240	2	2	1	1
6	Fort Worth, TX	760	1	1	1	1
7	Hickory, NC	286	1	0	0	0
8	Memphis, TN	381	1	1	1	2
9	Greensboro, NC	273	3	2	1	1
10	Dallas (local), TX	753	1	1	1	1
11	Irving, TX	752	1	1	1	1
14	Toledo, OH	436	0	0	1	0
15	Shipment integrity	8002	0	0	0	0
		Total =	15	13	11	10

Proposition 1. The destination-door assignment problem (DDAP) of minimizing the number of trailer switches over two or more sorts without consideration of loaders is NP-hard in the strong sense.
Proof. It will be shown that given an instance of 3-PARTITION, which Garey and Johnson (1979) indicate is strongly NP-hard, an instance of DDAP with two sorts can be constructed in pseudopolynomial time.

3-PARTITION: Given $n = 3m$ positive integers $a_1,\ldots,a_n$ satisfying $\sum_{j=1}^{n} a_j = mB$ integer and $B/4 < a_j < B/2$ for $j = 1,\ldots,n$, is there a partition of $N = \{1,\ldots,n\}$ into m subsets $S_1,\ldots,S_m$ such that $\sum_{j\in S_i} a_j = B$ for $i = 1,\ldots,m$? (Note that each S_i must contain exactly three elements from N.)

Without loss of generality, we can assume $a_j \geq 3$ for $j = 1,\ldots,n$. In the DDAP that we will construct, also assume that there are two sorts, $6m$ destinations divided into 3 groups and $m(B+2)$ doors. Let d_k denote destination k.

Sort 1 has the following destinations:

Group 1: d_k with total flow requiring 2 doors, $k = 1,\ldots,2m$; (i.e., $d_1, d_2,\ldots,d_{2m}$)

Group 2: d_{2m+k} with total flow requiring $B-2$ doors, $k = 1,\ldots,m$; (i.e., $d_{2m+1}, d_{2m+2},\ldots,d_{3m}$)

Sort 2 has the following destinations:

Group 1: d_k with total flow requiring 1 door, $k = 1,\ldots,2m$; (i.e., $d_1, d_2,\ldots,d_{2m}$)

Group 3: d_{3m+k} with total flow requiring a_k doors, $k = 1,\ldots,3m$; (i.e., $d_{3m+1}, d_{3m+2},\ldots,d_{6m}$)

The key to the proof centers on showing that the $3m$ destinations in Group 3 can be partitioned into blocks of three destinations each (where a block is equivalent to the set S_i in 3-PARTITION), such that the sum of the flow associated with the three destinations requires B doors.

Let $H = mB$ be the threshold value for switches. The question associated with the DDAP is whether there exists a door assignment such that the total number of switches is no more than H (actually equals H as shall be demonstrated shortly).

Proposed schedule. Doors are partitioned into m consecutive blocks, where each block j has the same structure with $B + 2$ doors. Blocks 1 and 2 are shown in Figure 4.

	Doors													
	Block 1							Block 2						
	r_1	r_2	r_3	…	r_B	r_{B+1}	r_{B+2}	r_{B+3}	r_{B+4}	r_{B+5}	…	r_{2B+2}	r_{2B+3}	r_{2B+4}
Sort 1	d_1	d_1	d_{2m+1}	…	d_{2m+1}	d_2	d_2	d_3	d_3	d_{2m+2}	…	d_{2m+2}	d_4	d_4
	↓						↓	↓						↓
Sort 2	d_1	3 destinations from Group 3, total flow requiring B doors					d_2	d_3	3 destinations from Group 3, total flow requiring B doors					d_4

Figure 4. Desired schedule for DDAP complexity proof

We now observe the following:

(1) As configured, destinations d_1 and d_2 (of Group 1) for Sort 1, Block 1, each requires two doors, and d_{2m+1} (of Group 2) requires $B - 2$ doors. The total door requirement for Sort 1, Block 1 is then $B + 2$.

For Sort 2, Block 1, d_1 and d_2 (of Group 1) each requires 1 door, while three destinations are chosen from Group 3 with total flow that adds up to B. Finding three such destinations for each block is always possible when 3-PARTITION holds. The total door requirement for Sort 2, Block 1 is $B + 2$. Similar logic applies to *Block* 2.

(2) Suppose that 3-PARTITION has a feasible solution $S_1,\ldots,S_m$. Then we can construct a door assignment as shown in Figure 4. In particular, for the three elements in each subset S_j, we put the three corresponding Group 3 destinations in block j, which exactly fills the B doors in the middle of the block (e.g., $r_2,\ldots,r_{B+1}$ for Block 1). It can be verified that the total number of switches is H. Specifically, there is a switch between doors assigned to the Group 3 destinations in Sort 2.

(3) Suppose that there is a feasible door assignment where the total number of switches is no more than H. Then we can make the following claims regarding such a schedule.

Claim 1: *Each Group 1 destination has to be assigned in one of two patterns, as illustrated in Figure 4 by d_1 and d_2.* Specifically, during Sort 2, each Group 1 destination has to use one of the doors used by the same destination during Sort 1, i.e., there can be no switches involving Sort 2 doors assigned to a Group 1 destination. This is true because any feasible schedule must use all doors in both sorts, so any assignment must have a_k switches for the Sort 2 doors assigned to a Group 3 destination d_{3m+k}. This gives a total of H switches. Any extra switches triggered by a Group 1 destination would lead to more than H switches in total.

Claim 2: *For any two Group 1 destinations, if they use adjacent doors in Sort 1, then they can only be scheduled as pairs, as shown in Figure 4 for d_2 and d_3.* Specifically, they must also use adjacent doors in Sort 2. According to Claim 1, there are four possible cases regarding how d_2 and d_3 are assigned in Sort 2, which can be denoted by (r_{B+1}, r_{B+3}), (r_{B+1}, r_{B+4}), (r_{B+2}, r_{B+4}), and (r_{B+2}, r_{B+3}). Except for the last case which is the object of this Claim, the other three cases allow either one or two doors in Sort 2 between the two destinations d_2 and d_3 to be occupied by some Group 3 destinations. However, this is not possible because each Group 3 destination needs at least three doors ($a_j \geq 3$).

Claim 2 means that at most two Group 1 destinations can be immediately adjacent to each other; the remainder have to be separated by some Group 2 destinations in Sort 1. Due to the fact that there are $2m$ Group 1 destinations and m Group 2 destinations, the only possible assignment is (partially) shown in Figure 4, i.e., in Sort 1, the assignment is

Group 1, Group 2, Group 1, Group 1, Group 2, Group 1,…, Group 1, Group 2, Group 1

Consequently, this leaves m holes in Sort 2 to be filled by Group 3 destinations. In particular, each hole has exactly B doors that must be filled with three Group 3 destinations. This leads to a feasible solution of 3-PARTITION thus proving the result. ■

A natural question that might arise from this result concerns the related problem of minimizing the number of doors when a limit is placed on the number of switches.

Corollary 1. The problem of minimizing the number of required doors given a fixed number of switches is NP-hard in the strong sense.

Proof. Referring to the proof of *Proposition* 1, we change the constructed DDAP instance as

follows: (i) assume that the given number of switches is $H = mB$, and (ii) set the threshold value of the number of needed doors to $m(B+2)$. With these modifications, the arguments in the above proof are still valid. ■

As an alternative, if the problem stated in the corollary is polynomially solvable, then DDAP can also be solved in polynomial time with a binary search over the given number of switches. This would contradict *Proposition* 1.

Proposition 2. The loader-door assignment problem (LDAP) of minimizing the number of loaders required to handle the volume on any sort *s* is NP-hard in the strong sense. Note that for the LDAP, both destinations and loaders are required to be assigned to contiguous doors, and the order of the destinations is not pre-specified.

Proof. Similarly, it will be shown that an instance of 3-PARTITION can be transformed into an instance of the loader-door assignment problem (LDAP) in polynomial time.

Given an instance of 3-PARTITON, we can construct an instance of LDAP in which there are $3m$ destinations each with a flow a_j, $3m$ doors each with a capacity B, and m loaders each having a capacity B and a limit of handling up to $\overline{n}^{doors} = 3$ consecutive doors. In such a problem, each destination will occupy only one door.

We now ask the question: Is there an ordering of the $3m$ destinations (which satisfies $\sum_{j=1}^{n} a_j = mB$ in accordance with the definition of 3-PARTITION) such that each loader is assigned a sequence of consecutive doors whose total volume is at most B? Given that the loaders are indistinguishable, a "yes" answer can be found if and only if the destinations can be partitioned into m subsets such that each subset has exactly $\overline{n}^{doors} = 3$ destinations with their total volume being B, which is a "yes" solution to 3-PARTITION. ■

A look at the proof of *Proposition* 2 indicates that LDAP is computationally intractable when the maximum number of doors a loader can handle $\overline{n}^{doors} \geq 3$. The case with $\overline{n}^{doors} = 2$ is easy when the flow for each destination is no more than the capacity of a loader; it is actually a matching problem and can be solved in polynomial time. The complexity of the general case with $\overline{n}^{doors} = 2$ is open.

A special case of the LDAP arises for the common situation where a facility has only one sort and the sequence of destinations for the sort is pre-specified. Moreover, if the number of doors for the sort is unrestricted, a greedy algorithm can be used to find the minimum number of loaders required. The greedy algorithm and the proof the following proposition can be found in Appendix A.

Proposition 3. For the special case of the LDAP, a polynomial-time greedy algorithm can be used to find the minimum number of loaders to service all the destinations on sort *s*.

Finally, if the number of doors is restricted, a polynomial-time dynamic program derived from the greedy algorithm can be used to obtain the minimum number of loaders. (See Appendix A for details). These cases represent a relaxation of the full problem and so can be used to provide lower bounds on the number of loaders. This is important for daily or weekly replanning when it is desirable to maintain the sequence of destinations on each sort but perhaps add or close some doors.

5. Model Formulation for the DLDAP

In our initial attempt to model the problem we defined binary variables corresponding to the assignment of destinations and loaders to doors on each sort. However, this resulted in an unwieldy formulation with a very large number of variables and very complex sets of constraints required to enforce the door contiguity requirement, restrict loader productivity (which is dependent on the number of consecutive doors assigned to each loader), assign sufficient loader capacity to handle the required flows for the various destinations, and track the changes in the destination-to-door assignments (or switches), among others. The resultant MIP was deemed unpromising and abandoned.

Alternatively, we took a network approach and used door sequence patterns as the basic modeling construct. We begin with the DDAP and develop a set of constraints which ensure that the consecutive door requirement is satisfied on each sort and that a sufficient number of doors are assigned to the destinations on each sort. We then introduce consecutive door constraints for the LDAP, also using patterns as variables, to get an integrated model. For the DDAP, we make use of the following.

Definition 2. A *destination pattern* is a series of consecutive doors that can be used for a destination during a sort.

In the network, there will be one node for each destination *d* and allowable pattern for each sort *s*. Allowable patterns consist of a fixed number of consecutive doors ranging from a minimum of $\underline{n}_{ds}^{doors}$ to the maximum of $\overline{n}_{d}^{doors} = \max\{\underline{n}_{ds}^{doors} : s = 1,\ldots,4\}$, where $\underline{n}_{ds}^{doors}$ is the number of doors required for destination *d* on sort *s* (see Table 3). If $\underline{n}_{ds}^{doors} = 0$, then one pattern will consist of *no* doors for destination *d* on sort *s*.

All allowable destination patterns are enumerated, and for each destination *d*, a subnetwork is set up (see Figure 5) with the following nodes:

- A source node (denoted by 0) with a supply of 1
- Pattern nodes at each of the four sorts with a node corresponding to each pattern
- A sink node (denoted by 5) with a demand of 1

The doors for a pattern are indicated within its node. If a pattern has no doors, an 'X' is used. There are three types of arcs:

- Arcs from the source node to the pattern nodes for sort 1 representing the initial door setup used in sort 1
- Arcs from *each* pattern node in sort *s* (*s* = 1,2,3) to *all* pattern nodes in sort *s* + 1 to model all possible destination pattern changes, or transitions, between two consecutive sorts
- Arcs from pattern nodes in sort 4 to sink node 5

A flow from the source node to the sink node determines a complete set of door assignments for the four sorts for a destination *d*. The full network consists of n^{dest} subnetworks in parallel, where n^{dest} is the total number of destinations served by the hub. Side constraints are needed to ensure that for each door *r* at most one pattern across all destinations is active.

For each destination *d*, the minimum $\left(\underline{n}_{ds}^{doors}\right)$ and maximum $\left(\overline{n}_{ds}^{doors}\right)$ number of doors allowed is subject to the following considerations.

- No more than the maximum number of doors, $\overline{n}_d^{doors}$, is allowed over all sorts. Because doors are a limited resource in most facilities, using more than the maximum for any destination *d* should be avoided even if doing so results in a reduced number of loaders.
- It is useful to allow one door beyond $\left(\underline{n}_{ds}^{doors}\right)$ as that may result in reducing the required number of loaders (see Section 3).
- It may be desirable to assign more doors than $\underline{n}_{ds}^{doors}$ to destination *d* on sort *s* to avoid disruptions arising from changes in the destination-to-door assignments. For example, if $\underline{n}_{d,1}^{doors} = 3$, $\underline{n}_{d,2}^{doors} = 2$, $\underline{n}_{d,3}^{doors} = 2$ and $\underline{n}_{d,4}^{doors} = 3$, then opening three doors for destination *d* during sort 1 and keeping them open during sorts 2 through 4 creates a universal lineup for destination *d*. Additionally, the extra doors available beyond $\underline{n}_{ds}^{doors}$ may result in reducing the number of loaders as we've seen in the example given in Section 2. Accordingly, we set $\overline{n}_{ds}^{doors} = \max_{s':s' \le s} \left\{ \underline{n}_{ds'}^{doors} + 1, \overline{n}_d^{doors} \right\}$.

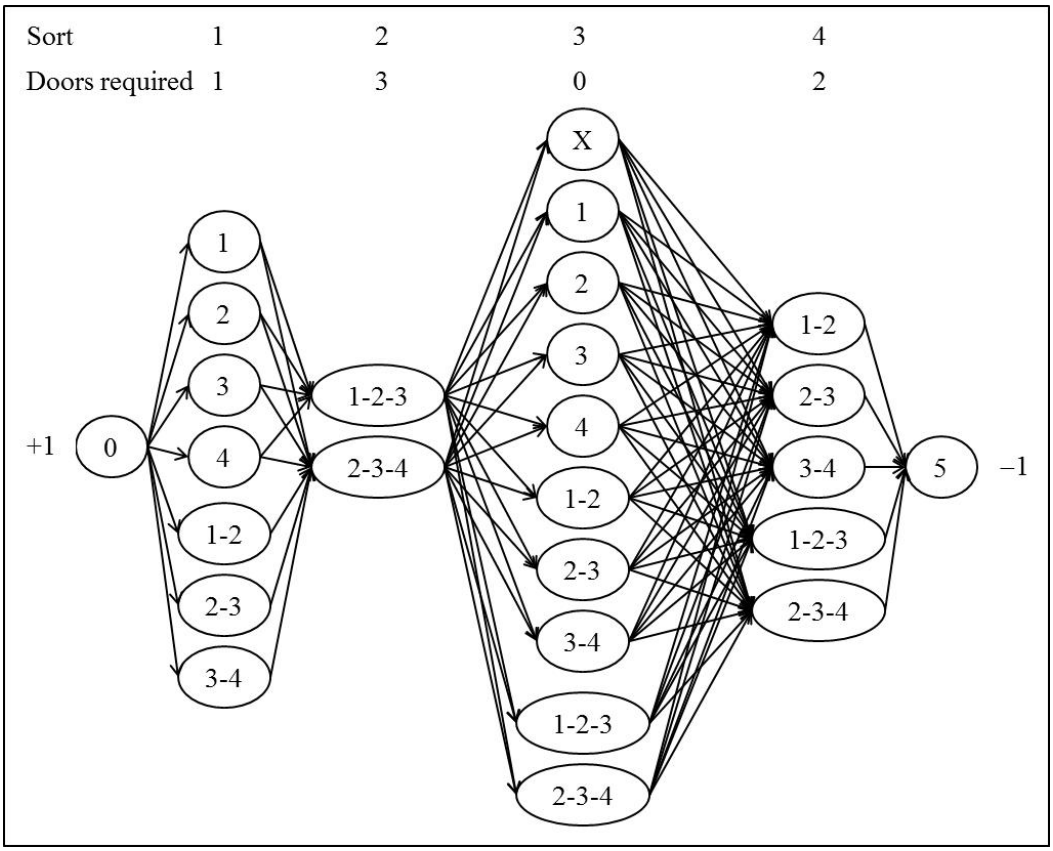

Figure 5. Subnetwork of destination pattern transitions for a single destination

Our primary objective is to minimize the number of switches, that is, the number of times a trailer at a door that is assigned a particular destination has to be replaced by a trailer for a different destination between sorts. Each move requires resources, is time consuming, and can lead to errors. As mentioned, trailers are sometimes placed at the wrong doors, and, if not caught early enough, could lead to a violation of the delivery service standard. When more than a single solution exists for the first objective, we also try to minimize the number of doors opened over all sorts. Our secondary objective is to minimize the number of loaders, and the third is to balance the workload.

The number of switches corresponds to the "cost" of a network arc. We do not count the first time a trailer is placed at a door at the beginning of sort *s* if the door has not been used previously. Given a pattern *p* on sort *s* for destination *d* that can transition into pattern *q* on sort *s* + 1, the number of switches involved in the transition corresponds to the number of doors in pattern *p* that do not appear in pattern *q*. The justification for this statement is that since a switch incurs a cost, no destination will be removed from a door unless the door is required for some other destination. Several examples of switches are depicted in Figure 3. A solution to the full network when the side constraints are included provides a complete assignment of destinations to doors on each sort (i.e., a lineup) with the minimum number of required switches.

Recall that from management's point of view, it is desirable to have a universal lineup where no switches occur over the day. This is only possible when there are a sufficient number (i.e., $\sum_{d=1}^{n^{dest}} \overline{n}_d^{doors}$) of doors in the workcenter.

Definition 3. A *loader pattern* is a fixed number of consecutive doors up to the maximum of $\overline{n}^{doors}$.

A loader can be assigned to one door or to a series of consecutive doors not to exceed $\overline{n}^{doors}$ (typically 5). To illustrate, for a hub with 20 doors, there are 20 one-door loader patterns, {1, 2, …, 20}, 19 two-door patterns, {(1,2), (2,3), …, (19,20)}, 18 three-door patterns, {(1,2,3), (2,3,4), …, (18,19,20)}, and so on. Each pattern is defined to correspond to a unique designation.

With these ideas in mind, we now present the model for the DLDAP making use of the following notation.

Indices

d	destinations
p,q	destination patterns
l	loader patterns
r	doors
s	sorts

Sets

L	loader patterns
L_r	loader patterns that include door r
L_s	loader patterns associated with sort s
P_s	all destination patterns (irrespective of destination) associated with sort s
P_{ds}	destination patterns for destination d on sort s
P_{rs}	destination patterns that cover door r on sort s
R_l	doors associated with loader pattern l

R_p doors associated with destination pattern p

Parameters

n_s^{dest} number of destinations on sort s

n^{doors} number of doors

$\overline{n}^{doors}$ maximum number of doors that can be assigned to a loader

ρ_n loader productivity when n doors are assigned to a loader; maximum productivity occurs when $n = 1$, giving $\rho_1 = 1$

V_{ds} package flow for destination d during sort s (packages/hour)

V^{max} maximum throughput (packages/hour) that can be handled by a loader ($V^{max} = 450$ when he is assigned 1 door)

δ_{pds}^q number of switches involved in transitioning from destination pattern p on sort s to destination pattern q on sort $s + 1$ ($s = 1,2,3$) for destination d

Calculated values

$\underline{n}_{ds}^{doors}$ minimum number of doors needed for destination d during sort s $\left(\underline{n}_{ds}^{doors} = \left\lceil V_{ds} / V^{\max} \right\rceil\right)$

$\overline{n}_d^{doors}$ maximum number of doors needed for destination d over all sorts $\left(\overline{n}_d^{doors} = \max\left\{\underline{n}_{ds}^{doors} : s = 1,...,4\right\}\right)$

Variables

x_{pds} $\begin{cases} 1, & \text{if destination pattern } p \text{ is assigned to destination } d \text{ on sort } s \\ 0, & \text{otherwise} \end{cases}$

t_{pds}^q $\begin{cases} 1, & \text{if pattern } p \text{ for destination } d \text{ on sort } s \text{ transitions to pattern } q \text{ on sort } s+1 \ (s = 1,2,3) \\ 0, & \text{otherwise} \end{cases}$

y_{ls} $\begin{cases} 1, & \text{if loader pattern } l \text{ is selected during sort } s \\ 0, & \text{otherwise} \end{cases}$

u_{rs}^{dest} $\begin{cases} 1, & \text{if door } r \text{ is } \textit{not} \text{ used during sort } s \text{ in some destination pattern} \\ 0, & \text{otherwise} \end{cases}$

u_{rs}^{load} $\begin{cases} 1, & \text{if door } r \text{ is } \textit{not} \text{ used during sort } s \text{ in some loader pattern} \\ 0, & \text{otherwise} \end{cases}$

z_{lrs} fraction of workload assigned to door r during sort s when loader pattern l is selected

The constraints are presented in three parts for the network version of DLDAP: (i) flow balance for DDAP, (ii) side constraints for DDAP to ensure that for each door in a sort at most one destination pattern is selected, and (iii) loader requirements for LDAP. To make the full model more understandable, the x_{pds} variables are explicitly expressed in terms of the transition variables t_{pds}^q in Eqs. (1b) and (1c) although this is not required for sorts 2 and 3. As a consequence, there will be a few redundant equations.

Flow balance constraints

For sort 1, exactly one pattern can be selected for each destination.

$$\sum_{p \in P_{d1}} x_{pd1} = 1,\ d = 1,...,n^{dest} \tag{1a}$$

A given destination pattern p on sort s will transition into exactly one pattern q on sort $s + 1$

$$x_{pds} = \sum_{q \in P_{d,s+1}} t^{q}_{p,d,s+1}\ ,\ d = 1,...,n^{dest},\ s = 1,...,3,\ p \in P_{ds} \tag{1b}$$

A selected pattern q during sort s–1 ($s > 1$) will transition into a given pattern p on sort s.

$$\sum_{q \in P_{d,s-1}} t^{p}_{q,d,s-1} = x_{pds},\ d = 1,...,n^{dest},\ s = 2,...,4,\ p \in P_{ds} \tag{1c}$$

For the last sort ($s = 4$), exactly one pattern can be selected for each destination

$$\sum_{p \in P_{d4}} x_{pd4} = 1,\ d = 1,...,n^{dest} \tag{1d}$$

Network side constraints

For each door on each sort, at most one destination pattern can be selected. If none is selected, then $u^{dest}_{rs} = 1$ in the solution indicating that door r is not used during sort s. The variables u^{dest}_{rs} are used in (1j) below.

$$\sum_{p \in P_{rs}} \sum_{d=1}^{n^{dest}} x_{pds} + u^{dest}_{rs} = 1,\ r = 1,...,n^{doors},\ s = 1,...,4 \tag{1e}$$

Loader constraints

At most one loader pattern can cover a door during each sort.

$$\sum_{l \in L_r} y_{ls} + u^{load}_{rs} = 1,\ r = 1,...,n^{doors},\ s = 1,...,4 \tag{1f}$$

Note that constraints (1f) can be written as inequalities without consequence by removing the "slack" binary variables u^{load}_{rs}. For computational purposes, though, we found it effective to use this variable for branching.

The volume assigned to the doors associated with a loader pattern should not exceed the capacity of a loader.

$$\sum_{r \in R_l} z_{lrs} \le \rho_{|R_l|} y_{ls},\ \forall\ l \in L,\ s = 1,...,4 \tag{1g}$$

The loader capacity serving a particular destination pattern selected on sort s for destination d must be at least as great as the required workload.

$$\sum_{r \in R_p} \sum_{l \in L_r} z_{lrs} \ge \frac{V_{ds}}{V^{max}} x_{pds},\ d = 1,...,n,\ s = 1,...,4,\ \forall\ p \in P_{ds} \tag{1h}$$

The utilization for a loader pattern l on sort s can be calculated as $\sum_{r \in R_l} z_{lrs}$ but only if (1h) is satisfied as an equality when $x_{pds} = 1$. This condition is enforced with the following.

$$\sum_{r\in R_p}\sum_{l\in L_r} z_{lrs} \le \frac{V_{ds}}{V^{max}} x_{pds} + \left|R_p\right|\left(1-x_{pds}\right),\ d=1,...,n,\ s=1,...,4,\ \forall\ p\in P_{ds} \tag{1i}$$

Constraints (1i) say that the flow assigned to the set of doors selected for a particular destination *d* on any sort must not exceed the corresponding workload. For a given sort *s* and destination pattern *p*, when that pattern is not selected to cover destination *d*, then the term in parentheses on the right-hand side of (1i) equals 1 and the constraint is redundant as is (1h). Otherwise, $x_{pds} = 1$ and (1i) places an upper bound on the volume assigned to the doors that are covered by pattern *p*. This forces (1h) and (1i) to be identical equalities.

Note that constraints (1e) imply that a specific destination pattern *p* can be selected for at most one destination on any sort. This allows us to replace (1h) and (1i) respectively with two sets of constraints that are both stronger and far less numerous, as shown below.

$$\sum_{r\in R_p}\sum_{l\in L_r} z_{lrs} \ge \sum_{d\in\left\{1,...,n^{dest}\right\}:\, p\in P_{ds}} \frac{V_{ds}}{V^{max}} x_{pds},\ s=1,...,4,\ \forall\ p\in P_s \tag{1h′}$$

$$\sum_{r\in R_p}\sum_{l\in L_r} z_{lrs} \le \sum_{d\in\left\{1,...,n^{dest}\right\}:\, p\in P_{ds}} \frac{V_{ds}}{V^{max}} x_{pds} + \left|R_p\right|\cdot\left(1-\sum_{d\in\left\{1,...,n^{dest}\right\}:\, p\in P_{ds}} x_{pds}\right),\ s=1,...,4,\ \forall\ p\in P_s \tag{1i′}$$

Constraints (1h′) and (1i′) now include summations on the right-hand side over *all* destinations that share pattern *p*, while constraints (1a) – (1d) ensure that each destination is covered by exactly one pattern on each sort. Furthermore, the following is true.

Proposition 4: Replacing (1h) and (1i) with (1h′) and (1i′) leads to fewer constraints and a tighter feasible region.

Proof. For a given a sort *s*, (1h′) and (1i′) are defined for each *p* in P_s, while (1h) and (1i) are defined for each destination *d* and each *p* in P_{ds}. Depending on the number of destinations, this typically translates to an order of magnitude reduction in the number of constraints. In addition, note that (1h′) has the same left-hand side as (1h); however, the right-hand side of (1h′) is greater than or equal to that of (1h). Similarly, (1i′) and (1i) have the same left-hand side, while the right-hand-side of (1i′) is less than or equal to that of (1i). Hence, (1h′) and (1i′) are stronger than (1i) and (1h), respectively. ■

Flow can only be assigned to a door if it is opened.

$$\sum_{l\in L_r} z_{lrs} \le 1-u_{rs}^{dest},\ r=1,...,n^{doors},\ s=1,...,4 \tag{1j}$$

Variable definitions

$$x_{pds},\ t_{pds}^{q},\ y_{ls},\ u_{rs}\ \in\{0,1\},\ \forall\ d,\ l,\ p,\ r,\ s;\ z_{lrs}\ \in[0,1],\ \forall\ l,r,s \tag{1k}$$

Objective function

Given the hierarchical nature of our objectives, we propose to minimize the weighted sum of the number of switches and the number of loaders, and deal with the lowest priority objective of maximizing loader utilization by solving a separate optimization problem conditioned on the solution to the first. Letting $\alpha_1 >> \alpha_2 >> \alpha_3$ be three nonnegative parameters, the DLDAP objective function is:

$$\text{Minimize } \alpha_1 \sum_{d=1}^{n^{dest}} \sum_{s=1}^{3} \sum_{p \in P_{ds}} \sum_{q \in P_{d,s+1}} \delta_{pds}^{q} t_{pds}^{q} - \alpha_2 \sum_{r=1}^{n^{doors}} \sum_{s=1}^{4} u_{rs}^{dest} + \alpha_3 \sum_{l \in L} \sum_{s=1}^{4} y_{ls} \qquad (1l)$$

Because α_1 is "much" larger than α_2, any algorithm or commercial code used to solve model (1a) – (1*l*) would first minimize the number of switches without regard to the number of doors (or loaders), and then look among alternative optima for the one that minimizes the number of doors. At this point, if multiple optimal solutions still exist, then the third term in (1*l*) is designed to select the one that minimizes the number of loaders.

5.1 Workload balancing

An optimal solution to model (1a) – (1*l*) is somewhat arbitrary with respect to the flow assigned to specific doors even for fixed α values. The only requirements are that each loader be assigned no more than $\overline{n}^{doors}$ doors, his total workload be at most V^{max}, and that all flow for each destination be processed. The first requirement is guaranteed by the specification of the loader pattern set *L* and by constraints (1f), the second by constraints (1g), and the last by (1h′), (1i′) and (1j). Since the number of packages handled by each loader can vary considerably from one solution to the next, we solve an additional optimization problem that attempts to distribute the workload among the loaders more equitably. Several objectives are possible including maximizing the minimum utilization, minimizing the variance of the workload, minimizing the sum of the deviations from the mean workload, and minimizing the maximum negative deviation from the mean. After testing each, the latter provided the best overall results with respect to solution quality and runtime. The corresponding objective is

$$\min \max \left\{ \overline{z} - \sum_{r \in R_l} z_{lrs} : s = 1, \ldots, 4, \; l \in L_s \right\} \qquad (1l')$$

where $\overline{z}$ is the average utilization for the number of loaders found by solving (1a) – (1*l*).

With regard to *Proposition* 2, when the number of loaders is fixed, we have the following.

Corollary 2. Given a fixed number of loaders, the problem of balancing the workload is NP-hard in the strong sense.

The proof is immediate based on the proof of *Proposition* 2 which looks for a schedule with zero (hence minimum) load deviation. In other words, the proof of *Proposition* 2 actually demonstrates the NP-hardness for the feasibility problem with a given number of loaders. So generally speaking, the load balancing problem must be NP-hard since a schedule that balances the workload has to necessarily be feasible for LDAP.

5.2 Tightening the formulation

When solving combinatorial optimization problems it is often advantageous to add cuts to the formulation which remove a portion of the relaxed feasible region. If designed properly, such cuts provide a closer approximation of the convex hull of feasible points and can be critical for obtaining high-quality solutions. We tested half a dozen cuts and found several to be quite effective. The following additional notation is required to describe them.

$\underline{n}_s^{doors}$ = minimum number of doors needed during sort *s*

$\underline{n}_s^{loaders}$ = minimum number of loaders needed during sort s

$NumDoors_{\lambda s}$ = number of doors assigned to loader number λ on sort s

$Thruput_s$ = current volume assigned to loaders on sort s

V_n^{max} = maximum throughput (packages/hour) that can be handled by a loader when assigned to n doors; note that the previously defined V^{max} equals V_1^{max}

Cut 1. Lower bound on the number of loaders by sort.

$$\sum_{l \in L_s} y_{ls} \geq \underline{n}_s^{loaders}, \ s = 1,\ldots,4 \qquad \text{(1m)}$$

The right-hand-side parameter in (1m) can be calculated as follows: $\underline{n}_s^{loaders} = \left\lceil \sum_{d=1}^{n^{dest}} V_{ds} / V^{max} \right\rceil$ which is always less than or equal to $\underline{n}_s^{doors}$, the minimum number of doors required on sort s, where $\underline{n}_s^{doors} = \sum_{d=1}^{n^{dest}} \left\lceil V_{ds} / V^{max} \right\rceil$. When $\underline{n}_s^{loaders} < \underline{n}_s^{doors}$, however, it should be clear that one or more of the $\underline{n}_s^{loaders}$ loaders will be required to service more than one door. We can check to see if the $\underline{n}_s^{loaders}$ loaders, with some or all them assigned multiple doors, can process the input flow given that they are required to span at least $\underline{n}_s^{doors}$ doors. If that turns out to be impossible, then $\underline{n}_s^{loaders}$ can be increased using the procedure below.

`Increase_Loader_Lower_Bound_on_Sort_s`

Step 1. (Initialization) For $\lambda = 1,\ldots, \underline{n}_s^{loaders}$, set $NumDoors_{\lambda s} = 1$ and $Thruput_s = V^{max} \cdot \underline{n}_s^{loaders}$.

Step 2. (Increase door assignments)

For $\kappa = 2,\ldots, \overline{n}^{doors}$

For $\lambda = 1,\ldots, \underline{n}_s^{loaders}$

Put $NumDoors_{\lambda s} \leftarrow NumDoors_{\lambda s} + 1$ and $Thruput_s \leftarrow Thruput_s + V^{max}_{NumDoors_{\lambda s}} - V^{max}_{NumDoors_{\lambda s}-1}$.

If $\sum_{\lambda=1}^{\underline{n}_s^{loaders}} NumDoors_{\lambda s} = \underline{n}_s^{doors}$, then

If $Thruput_s \geq \sum_{s=1}^{4} \sum_{d=1}^{n_s^{dest}} V_{ds}$, stop and report current value of $\underline{n}_s^{loaders}$;

Else, put $\underline{n}_s^{loaders} \leftarrow \underline{n}_s^{loaders} + 1$ and go to Step 1. //increase lower bound

Else, continue.

End λ

If $\kappa = \overline{n}^{doors}$, put $\underline{n}_s^{loaders} \leftarrow \underline{n}_s^{loaders} + 1$ and go to Step 1. //increase lower bound

End κ

The main idea of the lower bound procedure is to gradually increase the number of doors assigned to the loaders while accounting for the accompanying reduction in throughput. As soon as the total number of doors equals $\underline{n}_s^{doors}$, we check to see if the loading capacity of the loaders assigned to the

sort can satisfy the required flow; if not, $\underline{n}_s^{loaders}$ is increased by one loader. The process is repeated until the number of loaders is sufficient to process the overall flow. At Step 1, each loader on sort *s* is assigned one door and the total throughout is set to the number of loaders times the maximum number of packages that each can handle per hour. At Step 2 in the inner λ loop, we incrementally add one door to each loader while simultaneously updating the throughput. After each loader is assigned two doors, we increment the outer loop index κ and begin to assign them three doors, and so on. If and when $\sum_{\lambda=1}^{\underline{n}_s^{loaders}} NumDoors_{\lambda s} = \underline{n}_s^{doors}$, a check is made to see if the throughput is at least as great as the demand. If so, we stop and report $\underline{n}_s^{loaders}$; otherwise, $\underline{n}_s^{loaders}$ is incremented by one and the process starts again from scratch.

As the number of doors assigned to a loader increases, his productivity decreases as specified in Table 1. Therefore, we need to add the lower productivity value $V^{max}_{NumDoors_{\lambda s}}$ to the cumulative throughput parameter, $Thruput_s$, and subtract off the higher value $V^{max}_{NumDoors_{\lambda s}-1}$ to avoid double counting. Finally, if the outer loop terminates with $\kappa = \overline{n}^{doors}$, then each of the loaders has been assigned $\overline{n}^{doors}$, and the condition $\sum_{\lambda=1}^{\underline{n}_s^{loaders}} NumDoors_{\lambda s} = \underline{n}_s^{doors}$ has not been met. In this case, we again increment the number of loaders by one and restart the process.

Cut 2. Direct linkage between loaders and destinations on each sort.

$$\sum_{l \in L_s : R_l \cap R_p \neq \varnothing} y_{ls} \geq \underline{n}_{ds}^{doors} x_{pds}, \; d = 1,...,n^{dest}, \; s = 1,...,4, \; p \in P_{ds} \qquad (1n)$$

The summation on the left-hand side of (1n) is over all the loader patterns associated with sort *s* that have one or more doors in common with destination pattern *p*, as determined by $R_l \cap R_p$. On the right-hand side, when $x_{pds} = 1$ for some sort *s* and destination *d*, the number of loader patterns selected must be at least equal to the number of doors required for destination *d* on sort *s*, which is given by $\underline{n}_{ds}^{doors}$.

Cut 3. The total flow assigned to all the doors for sort *s* must equal the total demand for that sort.

$$V^{max} \cdot \sum_{r=1}^{n^{doors}} \sum_{l \in L_r} z_{lrs} = \sum_{d=1}^{n_s^{dest}} V_{ds}, \; s = 1,\ldots,4 \qquad (1o)$$

The first cut (1m), which is directly derived from the flow data, significantly tightens the number of loaders required when solving the LP relaxation. The second cut (1n) is implied by (1g) – (1i′) but links the *x* and *y* variables directly. Prior to the introduction of (1m) and (1n), it was not possible to obtain feasible solutions for the problem instances we tested in the development phase of the research and, hence, their introduction was critical for obtaining a tractable model. The third cut (1o) simply states that the total workload of the loaders in a sort is exactly equal to the total flow for the sorts and was seen to be of questionable value.

6. Solution Methodology

Model (1) is a large-scale MIP for instances of practical size. For the test data set (Dallas) high quality solutions could not be reliably obtained with Xpress, the high-performance industrial solver we used for

our computations. As a consequence, we developed a three-step sequential approach that is consistent with the preemptive nature of the three objectives described in Section 5.

6.1 Preemptive optimization steps

Step 1. (Determine minimum number of switches and find a feasible solution) Solve the network flow version of DDAP consisting of constraints (1a) – (1e) and (1k) with $\alpha_1 = 1$, $\alpha_2 = 0^+$ (i.e., arbitrarily small) and $\alpha_3 = 0$ in (1*l*) to get a lineup with the minimum number of switches, and then the minimum number of doors. Let the former value be Δ^{min}.

$$\Delta^{min} \cong \text{ Minimize } \quad \alpha_1 \sum_{d=1}^{n^{dest}} \sum_{s=1}^{3} \sum_{p \in P_{ds}} \sum_{q \in P_{d,s+1}} \delta_{pds}^{q} t_{pds}^{q} - \alpha_2 \sum_{r=1}^{n^{doors}} \sum_{s=1}^{4} u_{rs}^{door} \tag{2a}$$

$$\text{subject to } \quad \text{constraints (1a) – (1e) and (1k)} \tag{2b}$$

where the actual value of Δ^{min} is determined by the quadruple summation in the first time in (2a). Also, let $\hat{x}_{pds}$ and $\hat{t}_{pds}^{q}$ (for all *d, p, q, s*) be the optimal values of the lineup and switching variables, respectively. Subject to these values, find a feasible solution to the DLDAP by solving the LDAP below to get the minimum number of loaders conditioned on $\hat{x}$ and $\hat{t}$. Let $\hat{y}(\hat{x},\hat{t})$ be the optimal objective function value.

$$\hat{y}(\hat{x},\hat{t}) = \text{ Minimize } \quad \sum_{l \in L} \sum_{s=1}^{4} y_{ls} \tag{2c}$$

$$\text{subject to } \quad \text{constraints (1f) – (1k) and } x = \hat{x},\ t = \hat{t} \tag{2d}$$

Step 2. (Determine minimum number of loaders subject to bound on switches) Solve model (1a) – (1n) with an additional constraint that bounds the number of switches to Δ^{min} and with $\alpha_1 = 0$, $\alpha_2 = 0$ and $\alpha_3 = 1$ in (1*l*) to obtain the minimum number of loaders. Let y^{min} be the corresponding value and let y_s^{min} is the minimum number of loaders required on sort *s*.

$$y^{min} = \text{ Minimize } \quad \alpha_3 \sum_{l \in L} \sum_{s=1}^{4} y_{ls} \tag{3a}$$

$$\text{subject to } \quad \text{constraints (1a) – (1n)} \tag{3b}$$

$$\sum_{d=1}^{n^{dest}} \sum_{s=1}^{3} \sum_{p \in P_{ds}} \sum_{q \in P_{d,s+1}} \delta_{pds}^{q} t_{pds}^{q} \le \Delta^{min} \tag{3c}$$

Step 3. (Balance workload: minimize the maximum negative deviation from the mean) Compute the average utilization, $\overline{z}$, for the number of loaders found in Step 2; that is, $\overline{z} = \sum_d \sum_s V_{ds} / \left(V^{\max} \cdot y^{min} \right)$. Again, adding the bounding constraint on switches but now replacing (1*l*) with (1*l′*) as the objective, solve the following MIP.

$$\text{Minimize } \quad e \tag{4a}$$

$$\text{subject to } \quad e \ge \overline{z} - \sum_{r \in R_l} z_{lrs} + \left(y_{ls} - 1 \right), \ \forall\ l \in L,\ s = 1,\ldots,4 \tag{4b}$$

$$\sum_{d=1}^{n^{dest}} \sum_{s=1}^{3} \sum_{p \in P_{ds}} \sum_{q \in P_{d,s+1}} \delta_{pds}^{q} t_{pds}^{q} \leq \Delta^{min} \quad (4c)$$

$$\sum_{l \in L} y_{ls} = y_s^{min}, \; s = 1,\ldots,4 \quad (4d)$$

constraints (1a) – (1n), $e \geq 0$ (4e)

Constraints (4b) are a linearization of the "min-max" objective function (1l'). The right-hand side indicates that if loader pattern l is selected for sort s, then the total flow at the corresponding doors $r \in R_l$ is constrained to fall as little as possible below the mean. Pushing up the flow for some loader pattern l_1 has the counterbalancing effect of pushing down the flow for some other pattern l_2 on the same sort. Constraint (4c) again limits the number of switches to no more than the number permitted in Step 1, while constraints (4d) restrict the number of loader patterns (i.e., loaders) on each sort to the minimum number found in the solution to the problem in Step 2. All the original constraints are also included in (4e).

The two problems defined in Step 1 proved very easy to solve with Xpress, but not so for the problems defined in Steps 2 and 3. The computational difficulties we encountered were partially overcome with the enumeration strategies discussed in the next section.

6.2 Branch and bound strategies

When solving the combined models at Step 2, we found that prioritizing the order in which the branching variables were selected greatly improved the results. The order used was as follows: u_{rs}^{dest}, x_{pds}, u_{rs}^{load}, Y_s, y_{ls}, where $Y_s = \sum_{l \in L_s} y_{ls}$ is an integer variable introduced expressly for branching purposes.. By fixing the u^{dest} and x variables first we establish a lineup which makes the loader assignment component of the model much easier to solve. Note that it was never necessary to branch on the t_{pds}^{q} variables in any of the models despite the presence of (3c) or (4c), the implication being that DDAP is very close to pure network flow problem. The same priorities were used at Step 3, except that the Y_s variables are fixed at that point at y_s^{min}.

A second strategy used when solving model (3a) – (3c) was to include an objective function cutoff constraint and then update it dynamically as smaller values of y^{min} were uncovered. We start with the solution $\hat{y}(\hat{x},\hat{t})$ obtained from model (2c) – (2d) and impose the cut $\sum_{s=1}^{4} \sum_{l \in L_s} y_{ls} \leq \hat{y}(\hat{x},\hat{t}) - 1 + \varepsilon$, where $\varepsilon > 0$ is an arbitrarily small constant. When an improvement is found, call it $\hat{y}^{min}$, the term $\hat{y}(\hat{x},\hat{t})$ is replaced with $\hat{y}^{min}$.

6.3 Illustrative results

Using the data in Tables 2 and 3, we solved each of the models outlined in the three-step procedure above. At Step 1, the universal lineup given in Table 4a was obtained for the destination-door assignments. As can be seen, each door handles only a single destination (three-digit number). The minimum number of doors required is 15 but a total of 17 is called for to avoid switches. However, there are many alternative optima, some of which offer the possibility of a reduced number of loaders. In general, it is necessary to permute several of the destination-doors assignments to get an improvement. Implicitly, this is what is happening when models (3) and (4) are solved.

If it were desirable to reduce the number of doors by one in the solution, this could be achieved by shifting a block of assignments to the right or left. For example, for destination 273 on sort 3, door 11 could be moved to door 10 to make room for destination 436 at door 11 without affecting the loader requirements. The number of switches, though, would increase by one.

Given the lineup in Table 4a, the solution to the LDAP gives the loader-door assignments shown in Table 4b along with the package flows in Table 4c. The algorithm for the first cut in Eq. (1m) gave $\underline{n}^{loaders}$ = (8, 7, 7, 4) for the four sorts, or a total of 26 loaders (the optimal number), which is 18.2% above the initial value of 22 and 10.3% below the LDAP solution of 29. The minimum utilization is 0.055 and is associated with loader 26 (sort 4, doors 15 and 16) who handles 22 pkgs/hr. When all loaders are considered, the average utilization is 0.761.

Table 4a. Destination-door assignments for Dallas, Step 1, model (2a) – (2b)

	Door number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sort 1	772	760	61	753	752	381	309	286	273	273	273	---	295	295	295	---	240
Sort 2	772	760	61	753	752	381	309	---	273	273	273	---	295	---	295	240	240
Sort 3	772	760	61	753	752	381	309	---	273	---	273	436	295	---	295	240	---
Sort 4	772	760	61	753	752	381	309	---	---	273	---	---	---	---	295	240	---

Table 4b. Loader-door assignments for Dallas; Step 1, model (2c) – (2d)

	Door number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sort 1	1	1	1	1	1	2	2	2	3	4	5	---	5	6	7	---	8
Sort 2	9	9	9	10	10	10	11	---	11	12	13	---	13	---	14	14	15
Sort 3	16	17	18	19	20	20	21	---	21	---	22	22	22	---	23	23	---
Sort 4	24	24	25	26	26	27	28	---	---	28	---	---	---	---	29	29	---

Table 4c. Flow of packages/hour at each door for assignments in Tables 4a and 4b

	Door number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sort 1	12	33	51	64	12	37	75	239	450	450	20	---	355	450	200	---	444
Sort 2	27	116	209	129	38	116	67	---	29	450	85	---	265	---	384	16	450
Sort 3	87	329	374	381	102	267	90	---	285	---	43	187	78	---	217	183	---
Sort 4	54	259	96	331	25	429	11	---	---	10	---	---	---	---	16	6	---

Utilization: Min = 0.055, Max = 1, Avg = 0.761

Table 5a. Destination-door assignments for balanced solution for Dallas; Step 3, model (4a) – (4e)

	Door number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sort 1	286	753	240	---	273	273	273	309	---	760	381	752	61	772	295	295	295
Sort 2	---	753	240	240	273	273	---	309	---	760	381	752	61	772	295	---	295
Sort 3	---	753	240	240	273	---	273	309	436	760	381	752	61	772	295	---	---
Sort 4	---	753	240	---	---	---	273	309	---	760	381	752	61	772	295	---	---

Table 5b. Loader-door assignments for balanced solution for Dallas; Step 3, model (4a) – (4e)

	Door number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sort 1	1	1	2	---	3	4	5	5	---	5	6	6	6	6	6	7	8
Sort 2	---	9	9	10	10	11	---	12	---	12	12	13	13	13	14	---	15
Sort 3	---	16	16	17	17	---	18	18	18	19	20	20	21	22	22	---	---
Sort 4	---	23	23	---	---	---	24	24	---	24	25	26	26	26	26	---	---

Table 5c. Flow of packages/hour at each door for assignments in Tables 5a and 5b

	Door number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Sort 1	239	64	444	---	450	387	83	75	---	33	37	12	51	12	105	450	450
Sort 2	---	129	246	220	164	400	---	67	---	116	116	38	209	27	199	---	450
Sort 3	---	381	19	164	236	---	92	90	187	329	267	102	374	87	295	---	---
Sort 4	---	331	6	---	---	---	10	11	---	259	429	25	96	54	16	---	---

Utilization: Min = 0.442, Max = 1, Avg = 0.854

At Step 2, solving model (3) reduced the number of loaders to 26 and increased the utilization to 0.869 (results not shown). At Step 3, the solution to model (4) decreased the average utilization slightly to 0.854, bust still 14.2% above the Step 1 solution. The minimum loader utilization now increases to 0.442. The results are given in Tables 5a – 5c. As expected, loaders who are assigned more doors generally process less flow. For example, loader 1 is assigned doors 1 – 2 on sort 1 and handles 303 pkgs/hr. In contrast, loader 6 is assigned doors 11 – 15 on sort 1 and handles 217 pkgs/hr. Regardless of the model solved, though, closing a single door in the lineup, such as door 12 in Table 4a, and shifting the assignments one door to the right would eliminate one door while keeping the number of loaders the same. However, the number of switches would increase by one. This type of tradeoff may be required in highly capacitated facilities.

7. Computational Study

To illustrate the solution methodology we now present results for four representative hubs using average workcenter flow data obtained from a well-known package carrier. All procedures were coded in Mosel and all optimization problems were solved with Xpress (Release FICO Xpress 7.2.1). A 3.16 GHz Xeon 64-bit PC with 16 GBytes of RAM was used for the computations. At Step 1, Xpress always found the optimal solution to the DDAP model (2a) – 2(b) and the LDAP (2c) – (2d) in a matter of seconds so the default stopping criteria were used. For model (3a) – (3c) at Step 2 we allowed up to 40 minutes, and for model (4a) – (4e) at Step 3, we allowed up to 20 minutes.

7.1 Description of input data

The first workcenter at the Dallas hub was used in the development phase of the research. The results were presented in the previous section. For our computational study, we focused on the following four hubs: Chicago (six workcenters), Orlando (five workcenters), Rialto (eight workcenters), and Sacramento (six workcenters). The input data used to initialize the various optimization models are given in Table 6. The first column identifies the hub and the workcenter; the second column specifies the total number of doors available (though not necessarily the number used). Columns 3 and 4 respectively indicate the "bin packing" lower bound on the number of doors required on each sort and the number of outgoing destinations also on each sort. Column 5 gives the bin packing lower bound on the number of loaders needed per sort while column 6 gives the updated lower bound found by the tightening algorithm associated with Eq. (1m). The results indicate that $\underline{n}_s^{loaders}$ is increased by one in virtually all cases. Both bin packing lower bounds were calculated using a maximum door throughput value of $V^{max} = 450$.

Column 7 gives the number of destination-door patterns generated for each sort during model setup, while column 8 gives the number of loader-door patterns generated, which is the same for all sorts. The product of these values is an indication of problem size. The actual numbers of variables and constraints in each instance of model (3) are reported in Appendix B. The last column in Table 6 identifies the total package flow by sort, averaged over the planning month, which can be as low as 517 (Rialto-4 on sort 4) or as high as 5473 (Chicago-5 on sort 1). In the majority of cases, the flow decreases by sort.

Table 6. Input data and preprocessing parameter values

Hub - workcenter	No. doors, n^{doors}	Min. doors/sort, $\underline{n}_s^{doors}$ (s = 1,…,4)	No. destinations/sort, n_s^{dest} (s = 1,…,4)	Min. loaders/sort, $\underline{n}_s^{loaders}$ (s = 1,…,4)	Updated loaders/sort, $\underline{n}_s^{loaders}$ (s = 1,…,4)	No. destination patterns/sort, $\lvert P_s \rvert$ (s = 1,…,4)	No. loader patterns/sort, $\lvert L \rvert$	Average flow/sort, (pks/hr) (s = 1,…,4)
Chicago–1	28	(21,20,18,17)	(17,16,14,12)	(12,12,12,9)	(13,13,12,10)	(55,55,81,81)	130	(5129,5016,5038,3788)
Chicago–2	29	(20,22,20,21)	(17,17,16,16)	(12,11,10,9)	(12,12,11,10)	(57,57,57,84)	120	(4964,4678,4382,3727)
Chicago–3	28	(20,22,21,23)	(16,16,16,16)	(11,11,10,12)	(12,12,11,13)	(81,81,81,81)	130	(4544,4719,4349,4992)
Chicago–4	29	(20,21,20,22)	(17,17,16,16)	(11,11,11,10)	(12,12,12,12)	(57,57,84,84)	135	(4636,4662,4736,4460)
Chicago–5	28	(24,24,21,21)	(18,18,15,17)	(13,12,12,9)	(14,14,13,10)	(80,106,106,106)	130	(5473,5301,5141,3784)
Chicago–6	28	(21,21,20,20)	(16,16,16,16)	(11,11,8,9)	(12,12,9,10)	(81,81,81,81)	115	(4594,4524,3524,3794)
Orlando–1	26	(20,19,19,18)	(15,16,18,18)	(10,9,8,4)	(11,10,9,5)	(75,75,75,75)	110	(4131,3894,3273,1604)
Orlando–2	19	(17,17,13,13)	(17,18,14,14)	(8,8,7,4)	(9,9,7,5)	(37,37,37,37)	85	(3450,3347,2757,1762)
Orlando–3	22	(17,15,12,11)	(11,11,11,11)	(9,9,7,4)	(10,9,7,4)	(61,80,100,100)	75	(3806,3685,2886,1424)
Orlando–4	21	(17,17,12,10)	(10,11,10,10)	(10,9,7,4)	(10,10,8,4)	(56,73,111,111)	85	(4144,3897,2945,1424)
Orlando–5	21	(15,15,10,9)	(9,10,9,9)	(9,9,7,4)	(10,10,7,4)	(59,59,78,78)	70	(3971,3954,2717,1412)
Rialto–1	14	(11,10,9,7)	(8,8,7,7)	(6,7,5,2)	(7,7,5,2)	(25,37,50,50)	50	(2524,2729,1850,628)
Rialto–2	15	(12,12,12,9)	(11,11,11,9)	(6,5,5,2)	(7,6,6,3)	(29,29,29,29)	60	(2493,2211,2151,832)
Rialto–3	14	(10,9,10,6)	(8,8,8,6)	(5,6,5,2)	(5,6,6,3)	(27,27,27,27)	50	(1966,2514,2169,821)
Rialto–4	19	(15,16,13,8)	(11,11,11,8)	(7,8,7,2)	(8,9,8,2)	(37,37,37,37)	70	(3122,3300,2950,517)
Rialto–5	14	(10,9,8,4)	(6,6,6,4)	(6,5,5,2)	(7,6,5,2)	(24,35,60,60)	45	(2620,2230,2050,735)
Rialto–6	19	(16,14,13,7)	(12,12,12,7)	(7,7,7,3)	(8,8,8,3)	(37,37,37,37)	75	(3047,2994,3056,1156)
Rialto–7	26	(22,19,16,13)	(15,15,15,13)	(10,10,10,4)	(11,11,11,5)	(97,120,120,120)	100	(4268,4143,4203,1738)
Rialto–8	26	(18,17,12,9)	(9,9,9,9)	(12,11,9,3)	(13,12,9,3)	(94,117,141,141)	85	(5378,4857,3683,946)
Sacramento-1	22	(15,13,12,11)	(10,10,10,10)	(8,7,8,5)	(9,7,8,6)	(62,82,82,82)	75	(3299,2755,3260,2051)
Sacramento-2	18	(15,13,13,12)	(10,10,10,10)	(8,7,8,5)	(9,8,9,6)	(50,66,66,66)	80	(3520,2960,3365,2027)
Sacramento-3	20	(15,12,12,11)	(10,10,10,9)	(8,7,8,5)	(9,8,8,5)	(57,57,57,57)	80	(3496,3040,3361,1871)
Sacramento-4	18	(15,14,13,13)	(11,11,11,11)	(8,7,7,5)	(9,8,8,5)	(51,51,51,51)	75	(3259,2896,3020,1833)
Sacramento-5	21	(17,16,16,16)	(12,12,12,12)	(9,8,8,6)	(10,9,9,7)	(58,78,78,78)	95	(3668,3586,3567,2631)
Sacramento-6	20	(16,15,14,14)	(12,12,12,11)	(9,7,8,6)	(10,8,9,7)	(56,74,74,74)	85	(3803,3074,3305,2537)

Table 7. Output statistics for the three-step methodology

Hub - workcenter	Total no. doors required	Step 1 (initial solution)				Step 2 (minimum no. loaders)			Step 3 (balanced solution)	
		No. doors used / sort (s = 1,…,4)	Min. no. switches, Δ^{min}	No. loaders / sort, $\hat{y}(\hat{x},\hat{t})$ (s = 1,…,4)	Loader utilization (min, avg, max)	No. doors used / sort (s = 1,…,4)	No. loaders / sort, y_s^{min} (s = 1,…,4)	Loader utilization (min, avg, max)	No. doors used / sort (s = 1,…,4)	Loader utilization (min, avg, max)
Chicago–1	28	(21,20,20,17)	2	(16,13,14,11)	(0.01,0.82,1)	(21,20,20,17)	(14,14,14,10)	(0.29,0.86,1)	(21,20,18,17)	(0.48,0.86,1)
Chicago-2	26	(20,22,21,21)	0	(14,13,12,11)	(0.23,0.84,1)	(20,22,21,21)	(13,13,13,10)	(0.14,0.87,1)	(20,22,22,22)	(0.54,0.87,1)
Chicago-3	28	(20,22,23,23)	0	(12,13,13,15)	(0.09,0.84,1)	(20,23,21,23)	(12,13,12,13)	(0.36,0.89,1)	(20,22,22,23)	(0.67,0.90,1)
Chicago-4	29	(20,21,22,22)	0	(14,13,13,12)	(0.39,0.85,1)	(20,23,21,23)	(14,12,13,12)	(0.21,0.87,1)	(20,21,21,23)	(0.39,0.87,1)
Chicago-5	28	(24,24,22,21)	4	(15,15,14,12)	(0.18,0.83,1)	(24,25,22,21)	(14,15,14,11)	(0.32,0.86,1)	(24,24,23,22)	(0.43,0.87,1)
Chicago-6	25	(21,23,20,20)	0	(14,12,10,11)	(0.22,0.84,1)	(21,22,20,20)	(12,13,10,10)	(0.15,0.89,1)	(20,20,21,21)	(0.69,0.89,1)
Orlando–1	24	(20,21,21,20)	0	(12,12,10,6)	(0.20,0.81,1)	(20,20,21,19)	(11,11,10,5)	(0.41,0.87,1)	(20,21,20,20)	(0.64,0.87,1)
Orlando–2	19	(17,18,14,14)	0	(10,9,10,6)	(0.33,0.79,1)	(17,17,16,13)	(9,9,8,6)	(0.64,0.88,1)	(17,17,16,13)	(0.64,0.88,1)
Orlando–3	17	(17,15,14,11)	0	(11,11,9,6)	(0.10,0.75,1)	(17,16,14,12)	(10,10,8,4)	(0.38,0.90,1)	(17,17,14,13)	(0.76,0.90,1)
Orlando–4	19	(17,17,13,10)	0	(11,11,10,5)	(0.18,0.78,1)	(17,17,14,11)	(10,10,8,4)	(0.65,0.93,1)	(17,18,13,11)	(0.74,0.94,1)
Orlando–5	16	(15,16,12,11)	0	(11,11,8,6)	(0.00,0.80,1)	(15,16,12,10)	(10,10,7,4)	(0.59,0.94,1)	(15,16,13,10)	(0.77,0.93,1)
Rialto–1	12	(11,11,10,8)	0	(7,8,6,3)	(0.10,0.77,1)	(11,10,9,7)	(7,7,5,2)	(0.04,0.89,1)	(11,11,9,7)	(0.59,0.88,1)
Rialto–2	14	(12,12,13,9)	0	(8,8,7,4)	(0.00,0.69,1)	(12,12,13,9)	(7,7,6,3)	(0.46,0.82,1)	(12,12,12,10)	(0.63,0.83,1)
Rialto–3	12	(10,10,11,6)	0	(6,7,6,4)	(0.13,0.77,1)	(10,9,11,6)	(5,7,6,3)	(0.51,0.85,1)	(10,10,10,8)	(0.61,0.86,1)
Rialto–4	16	(15,16,13,8)	0	(9,9,9,3)	(0.10,0.79,1)	(15,16,14,8)	(8,9,8,2)	(0.59,0.89,1)	(15,16,13,9)	(0.63,0.89,1)
Rialto–5	11	(10,9,8,4)	0	(7,6,6,2)	(0.24,0.84,1)	(10,10,9,4)	(7,6,5,2)	(0.15,0.89,1)	(10,9,9,4)	(0.39,0.90,1)
Rialto–6	17	(16,16,15,8)	0	(9,9,9,4)	(0.28,0.80,1)	(16,15,14,7)	(8,8,9,3)	(0.23,0.89,1)	(16,16,17,7)	(0.74,0.91,1)
Rialto–7	22	(22,20,19,13)	0	(11,11,14,7)	(0.12,0.80,1)	(22,21,19,13)	(12,11,13,5)	(0.35,0.85,1)	(22,21,19,13)	(0.60,0.84,1)
Rialto–8	19	(18,17,14,10)	0	(14,12,10,3)	(0.35,0.89,1)	(18,17,14,9)	(13,12,9,3)	(0.63,0.94,1)	(18,17,14,9)	(0.63,0.94,1)
Sacramento-1	17	(15,15,15,11)	0	(10,8,9,7)	(0.14,0.80,1)	(15,13,13,12)	(9,7,6,6)	(0.29,0.91,1)	(15,14,14,11)	(0.46,0.91,1)
Sacramento-2	18	(15,15,14,13)	1	(9,8,10,7)	(0.04,0.84,1)	(15,14,13,12)	(9,8,9,6)	(0.45,0.89,1)	(15,14,14,14)	(0.74,0.90,1)
Sacramento-3	18	(15,14,15,11)	0	(10,9,9,6)	(0.07,0.84,1)	(15,14,13,11)	(9,8,9,5)	(0.62,0.91,1)	(15,14,13,11)	(0.74,0.92,1)
Sacramento-4	17	(15,14,15,13)	0	(9,8,9,4)	(0.10,0.83,1)	(15,14,15,13)	(9,8,8,5)	(0.20,0.89,1)	(15,15,15,13)	(0.52,0.89,1)
Sacramento-5	21	(17,16,16,16)	0	(10,10,10,7)	(0.33,0.88,1)	(17,16,16,17)	(10,9,9,7)	(0.36,0.94,1)	(17,16,16,17)	(0.36,0.94,1)
Sacramento-6	19	(16,15,15,14)	0	(12,9,9,7)	(0.12,0.82,1)	(16,15,14,14)	(10,8,9,7)	(0.38,0.89,1)	(16,15,15,14)	(0.66,0.90,1)

Table 8. Output statistics for computations

Hub – workcenter	Step 2: Model (3) - optimized solution						Step 3: Model (4) – balanced solution				
	No. integer solns	Time to best soln (sec)	Best node	Total B&B nodes	Opt. gap (%)	Run time (sec)	No. integer solns	Time to best soln (sec)	Best node	Total B&B nodes	Run time (sec)
Chicago–1	2	369	14,356	74,532	8.33	2,400	3	529	9,223	22,125	1,200
Chicago–2	1	1	8,943	98,948	8.89	2,400	1	857	36,773	47,572	1,200
Chicago–3	2	22	984	94,105	4.17	2,400	3	296	11,100	43,114	1,200
Chicago–4	1	1,017	63,085	99,139	6.25	2,400	1	607	18,561	33,740	1,200
Chicago-5	2	498	10,819	51,493	5.88	2,400	3	910	14,515	19,468	1,200
Chicago-6	2	20	1,929	99,339	4.65	2,400	5	836	32,437	43,619	1,200
Orlando–1	3	302	39,456	111,675	5.71	2,400	5	967	38,542	46,210	1,200
Orlando–2	2	7	660	112,806	6.67	2,400	0	NA	NA	58,502	1,200
Orlando–3	3	24	7,358	118,977	6.67	2,400	9	220	20,062	78,641	1,200
Orlando–4	4	15	2,956	2,960	0.00	15	5	476	27,067	63,452	1,200
Orlando–5	3	11	3,854	3,856	0.00	11	6	288	33,969	98,240	1,200
Rialto–1	3	4	1,065	1,069	0.00	4	3	11	1,660	16,299	50
Rialto–2	4	63	21,445	126,480	4.55	2,400	7	333	36,752	82,783	1,200
Rialto–3	2	2	201	118,921	0.00	1,031	8	31	7,770	96,257	596
Rialto–4	3	39	10,324	10,328	0.00	39	3	120	15,760	83,768	1,200
Rialto–5	1	1	186	188	0.00	1	1	4	76	907	6
Rialto–6	3	79	21,363	123,896	3.70	2,400	3	1,060	75,763	80,773	1,200
Rialto–7	2	31	5,080	120,384	7.89	2,400	2	65	2,304	47,553	1,200
Rialto–8	2	93	24,473	24,477	0.00	93	0	NA	NA	91,304	1,200
Sacramento-1	4	21	3,621	3,625	0.00	21	1	301	26,106	68,728	1,200
Sacramento-2	1	14	533	536	0.00	14	12	682	30,991	51,431	1,200
Sacramento-3	3	383	46,708	111,240	3.33	2,400	6	853	57,972	71,884	1,200
Sacramento-4	2	54	13,574	13,578	0.00	54	7	856	62,425	74,785	1,200
Sacramento-5	2	1,423	82,365	82,369	0.00	1,423	0	NA	NA	56,691	1,200
Sacramento-6	3	1,353	87,631	87,635	0.00	1,353	5	1,041	54,313	60,045	1,200

NA = not applicable because no integer solution found within allotted time.

7.2 Results

The output statistics for the three-step procedure are summarized in Table 7. The first column again identifies the hub and workcenter and the second column gives the total number of doors used. For Chicago-1, for example, 28 doors are available (Table 6) and 28 were used in the final solution, indicating that this workcenter is at capacity; for Chicago-2, only 26 of the 29 doors were assigned to destinations. The next four columns are associated with Step 1 where the minimum number of switches is found without regard to loader requirements [model (2a) – (2b)], and then the minimum number of loaders for the corresponding door lineup is obtained [model (2c) – (2d)]. The combined results provide a feasible solution.

Column 3 lists the number of doors used per sort, which for Chicago-1 and -2, for example, are always less than the total available. The difference is due to the fact that the flow requirements on each sort rarely require the use of all doors needed to achieve a universal lineup without switches. Column 4 indicates that a universal lineup can be found for all workcenters except three, a surprising result since this is almost never the case in practice (more will be said about this presently). The number of loaders per sort and loader utilization are reported in columns 5 and 6, respectively. The latter statistic averages 0.81 across all 24 workcenters but can be as low as 0.0, signaling an imbalance in workload. Steps 2 and 3 are aimed at improving these measures.

Specifically, the full problem [model (3a) – (3c)] is solved at Step 2 to find the minimum number of loaders while holding the minimum number of switches fixed. The results are reported in columns 7 – 9. In most cases, the number of doors used in a solution remain the same but may increase or decrease by 1 or 2 on some sorts to more evenly distribute the flow associated with the corresponding destinations (compare columns 3 and 7). Recall the simple example in Section 3 which demonstrates that shifting flow among adjacent doors may reduce the number of loaders. Changing the permutation of destinations, along with shifting flow among adjacent doors, usually results in significant reduction in the number of loaders required. This can be seen by comparing the entries in columns 5 and 8 for each workcenter. On average, the number of loaders decreased by 2.72 or 7.2%. Similarly, comparing columns 6 and 9, we see that the average utilization increased from 0.81 to 0.89, or 9.35%.

At Step 3, model (4a) – (4e) is solved in an effort to balance the workloads without increasing the number of switches or loaders. The results are reported in the last two columns of Table 7. For virtually all hubs, the number of doors used changed by plus or minus one on at least one sort compared to the Step 2 solution. The values in columns 8 and 10 are rarely the same across sorts but there is no noticeable pattern. With respect to the minimum utilization, we see a 59.9% increase on average between the Steps 2 and 3 results. The average utilization values, though, are nearly identical, with the slight difference due to the variability in the multi-door loader assignments and associated productivities in the solution; the maximum values are always 1 regardless of model or step.

7.3 Statistics for computations

The statistics associated with the computations are reported in Table 8 for models (3) and (4). For the most part, the columns are self-explanatory. What is noteworthy is that the best solutions were almost always found within a fraction of the time allotted and that the optimality gap at termination averaged 3.07%. Of the 24 instances, 12 had a gap of 0%. The large size of the B&B trees indicates that a vast number of lineups were explored during the search. With respect to Step 3, we were able to find an

improved average workload balance in 21 of the 24 instances (compare columns 9 and 11 in Table 7). As mentioned, this improvement was significant.

7.4 Comparison with current operations

In addition to the optimality gap, a second and third measure of solution quality are the differences between the number of switches and the number of loaders used in practice versus the numbers provided by the solution to model (3a) – (3c). For Chicago-1, for example, all 28 doors are being used currently but the number of switches in the corresponding lineup is 18 compared to 2 in the balanced solution—a remarkable reduction. With respect to the number of loaders, Table 9 highlights the differences, also for Chicago-1. The average reduction is 17.7%.

Table 9. Difference in loader requirements for Chicago-1

Sort	Pkgs/hr	Actual no. loaders	Balanced solution	Gap (%)
1	5129	17	14	17.65
2	5016	16	14	12.50
3	5038	17	14	17.65
4	3788	13	10	23.08

A fourth measure of solution quality is loader utilization. In a similar manner, we compared the number of average packages per hour that the loaders handle in practice with the average number in the balanced solution averaged over all four sorts. The statistics are plotted in Figure 6 again for Chicago-1, where the horizontal axis corresponds to 100-package intervals. The first vertical bar represents the range from 0 to 100 pkgs/hr and the last bar, the abbreviated range from 401 to V^{max} = 450 pkgs/hr. From the figure and with some calculations based on the disaggregated data, we can see that the balanced solution provides a much higher average workload (366 vs. 301 pkgs/hr) and a much smaller range (174 to 450 vs. 95 to 468 pkgs/hr) than current practice. Almost identical results were observed with respect to all four measures for the other workcenters for which current lineup and loader data were available.

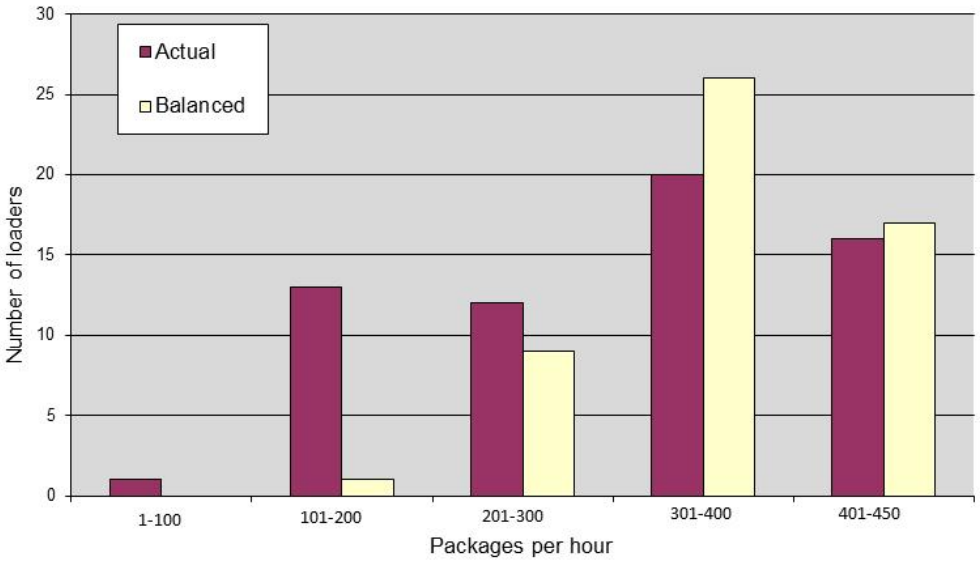

Figure 6. Loader utilization comparison

8. Summary and Conclusions

The characteristics of the destination-loader-door assignment problem at regional package sorting hubs that perform multiple daily sorts offers a variety of modeling issues and algorithmic challenges that have only now come under investigation. In this paper, we present the first formulation for the problem along with an efficient solution procedure that addresses the hierarchical nature of the objectives. The first is to minimize the number of destination switches at the loading doors, the second is to minimize the number of loaders required to handle the flow over four sorts each day, and the third is to balance the workload. To test the procedure, we examined 24 real-world instances associated with an internationally known package carrier that operates over 100 regional hubs throughout the U.S. The results indicated that high-quality solutions could be obtained within a 1-hour time limit in all cases with an average optimality gap of 3%.

An additional contribution of the work was a detailed analysis of the time complexity of the two components of the general problem. We showed that the destination-door assignment subproblem and the loader-door assignment subproblem taken separately are both strongly NP-hard. Nevertheless, for the special case in which the destinations are ordered on each sort, we devised a polynomial time dynamic program to find the minimum number of loaders required.

In the future, we expect to incorporate several additional constraints into the formulation that may be important to some hubs, and to explore other solution techniques. For hubs that have very high fluctuations in demand for some destinations, it may be desirable to assign an empty door to each such destination on one or more sorts to accommodate the overflow. A second way of handling variability is to increase the demand by one or even two standard deviations for those destinations that have high variance to get a more robust solution, or to investigate the feasibility of using stochastic programming approaches. From a management point of view, it may be desirable to restrict the flow at the end doors so that the loaders assigned to them could be easily reassigned as the flow patterns change from one day to the next. For the same reason, it may be better not to assign high flow destinations to adjacent doors. Other constraints are also possible and most, including those just mentioned, can be readily incorporated in our models.

Finally, the model developed for the DLDAP can be adapted for weekly or even daily replanning purposes in response to changes in package volumes. However, that would require additional functionality that limits the changes in the destination-to-door assignments from one week to the next. Preliminary work in this area has shown, for example, that when the destination lineup on each sort is fixed, optimal or near-optimal solutions can be obtained in a few seconds.

References

Abdelghany, A., Abdelghany, K. and Narasimhan, R. (2006). Scheduling baggage-handling facilities in congested airports, *Journal of Air Transport Management*, 12(2), 76-81.

Ascó, A., Atkin, J.A.D. and Burke, E.K. (2011). The airport baggage sorting station allocation problem, *Proceedings of the 5th Multidisciplinary International Scheduling Conference* (MISTA 2011), 419-444, Phoenix, AZ.

Ascó, A., Atkin, J.A.D. and Burke, E.K. (2012). An evolutionary algorithm for the over-constrained airport baggage sorting station assignment problem, L.T. Bui et al. (Eds.): SEAL 2012, LNCS 7673, 32-41.

Bard, J.F., Binici, C. and deSilva, A.H. (2003). Staff scheduling at the United States Postal Service. *Computers & Operations Research*, 30(5), 745-771.

Bozer, Y.A. and Carlo, H.J. (2008). Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions on Design & Manufacturing*, 40(11), 1007-1018.

Campbell, G.M. and Diaby, M. (2002). Development and evaluation of an assignment heuristic for allocating cross-trained workers. *European Journal of Operational Research*, 138(1), 9-20.

Choy, K.L., Chow, H.K.H., Poon, T.C. and Ho, G.T.S. (2011). Cross-dock job assignment problem in space-constrained industrial logistics distribution hubs with a single docking zone. To appear in *International Journal of Production Research*, DOI:10.1080/00207543.2011.581006

Dorndorf, U., Drex,l A., Nikulin, Y. and Pesch. E. (2007). Flight gate scheduling: state-of-the-art and recent developments. *Omega*, 35, 326–334.

Ernst, A.T., Jiang, H. Krishnamoorthy, M. and Sier, D. (2004). Staff scheduling and rostering: a review of applications, methods and models. *European Journal of Operational Research*, 153(1), 3-17.

Garey, M.R. and Johnson, D.S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*, W.H. Freeman, New York.

Jarrah, A.I.Z., Bard, J.F. and deSilva, A.H. (1994a). Equipment selection and machine scheduling in general mail facilities. *Management Science*, 40(8), 1049-1068.

Jarrah, A.I.Z., Bard, J.F. and deSilva, A.H. (1994b). Solving large-scale tour scheduling problems. *Management Science*, 40(9), 1124-1145.

Liao, T.W., Egbelu, P.J. and Chang, P.C. (2013). Simultaneous dock assignment and sequencing of incoming trucks under a fixed outbound truck schedule in multi-door cross docking operations. *Int. J. Production Economics*, 141(1), 212-229.

McAree, P., Bodin, L., Ball, M. and Segars, J. (2006). Design of the Federal Express large package sort facility, *Annals of Operations Research*, 144(1), 133-152.

McWillians, D.L., Stanfield, P.M. and Geiger, C.D. (2005). The parcel hub scheduling problem: A simulation-based solution approach. *Computers & Industrial Engineering*, 49(3), 393–412.

Wan, L. and Bard, J.F. (2007) Weekly scheduling with workgroup restrictions. *Journal of the Operational Research Society*, 58(8), 1030-1046.

Werners, B. and Wolfing, T. (2010). Robust optimization of internal transports at a parcel sorting center operated by Deutsche Post World Net. *European Journal of Operational Research*, 201(2), 419-426.

Yan, S., Tang, C.-H. and Hou, Y.-H. (2011). Airport gate reassignments considering deterministic and stochastic flight departure/arrival times. *Journal of Advanced Transportation*, 45, 304–320.

Zhang, X. and Bard, J.F. (2005). Equipment scheduling at mail processing and distribution centers. *IIE Transactions on Scheduling & Logistics*, 37(2), 175-187.

Zhang, X. and Bard, J.F. (2006). Comparative approaches to equipment scheduling in high volume factories, *Computer & Operations Research*, 37(1), 132-157.

Appendix A. Two Polynomial-Time Algorithms for Restricted Versions of LADP

Consider a restricted version of the loader assignment problem in which the sequence of destinations is given. In addition, assume that a "sufficient" number of doors exist to accommodate the solution that minimizes the number of loaders (more on this shortly). For each sort *s*, assume that the destinations are indexed in accordance with the given sequence of doors. Using slightly different notation for some values than in the main text, let

a_i = given flow for destination i

$\overline{n}^{doors}$ = maximum number of doors that a loader can handle

n_s^{dest} = number of destinations for sort s

V_l^{max} = maximum flow that can be assigned to each loader over l doors; $V_1^{max} \geq V_2^{max} \geq \cdots \geq V_{\overline{n}^{doors}}^{max}$

n_k = number of doors assigned to loader k

b_k = flow assigned to loader k

`Greedy algorithm`

Initialization. Let $i = 1$, $k = 1$, $l = 1$, $n_1 = 0$, $b_1 = 0$.

While ($i \leq n_s^{dest}$), do

If ($n_k = \overline{n}^{doors}$ or $V_l^{max} \leq b_k$), then

Put $k \leftarrow k + 1$, $n_k = 0$, $b_k = 0$, $l = 1$ //start a new loader k with zero flow assigned

If ($V_l^{max} \geq b_k + a_i$), then //current loader has enough capacity; no need to start a new loader

Put $b_k \leftarrow b_k + a_i$, $n_k \leftarrow n_k + 1$, $l \leftarrow l + 1$, $i \leftarrow i + 1$;

Otherwise, // if ($V_l^{max} < b_k + a_i$), then assign k a full load and start a new loader

Put $a_i \leftarrow a_i - (V_l^{max} - b_k)$, $b_k = V_l^{max}$;

Put $k \leftarrow k + 1$, $n_k = 0$, $b_k = 0$, $l = 1$.

End while

Remark: For the case where $V_l^{max} < b_k + a_i$, when we start a new loader, destination i will occupy a new door by design, since it is assumed that loaders are not permitted to share doors. This is where the assumption of a sufficient number of doors is needed because the algorithm uses doors without any restriction.

Proposition 3 (Rephrased). For the given assumptions, `greedy_algorithm` uses the minimum number of loaders to service all the destinations on sort *s*.

The proof is straightforward. Suffice to say it that each loader is assigned as much flow as possible, which is either V_l^{max} or $\overline{n}^{doors}$. The time complexity of the algorithm is $O(m^{max})$, where m^{max} is the maximum number of doors used by the algorithm. This follows because the doors are assigned sequentially and that the assignments for each destination depend only on the known flow, which means that they can be determined in constant time, $O(1)$.

Corollary 1. The value of m^{max} is bounded by $\sum_{i=1}^{n_s^{dest}} \left(1+\lceil a_i / V_1^{max} \rceil\right)$, where the term $\left(1+\lceil a_i / V_1^{max} \rceil\right)$ for each destination *i* is needed since a destination may use one more door than the minimum required.

Proof. By construction, if destination *i* uses $2+\lceil a_i / V_1^{\max} \rceil$ doors, then the $\lceil a_i / V_1^{\max} \rceil$ doors in the middle will be fully loaded, which means that the total flow handled by these middle doors is already a_i. Recal that by construction, partial door allocations can only be made for the first and/or last door associated with a destination Therefore, it is not possible to use $2+\lceil a_i / V_1^{\max} \rceil$ or more doors. ■

This is the case for destination 3 in the following example. Let n_s^{dest} = 7, a_1 = 0.5, a_2 = 0.9, a_3 = 0.2, a_4 = 0.7, a_5 = 0.9, a_6 = 0.9, a_7 = 0.2, $\bar{n}^{doors}$ = 2, V_1^{max} = 1, V_2^{max} = 0.9. Applying `greedy_algorithm` gives the following solution that uses 5 loaders and 10 doors.

Destinations	1	2		3	4	5		6		7
Doors	1	2	3	4	5	6	7	8	9	10
Flow	0.5	0.4	0.5	0.2	0.7	0.2	0.7	0.2	0.7	0.2
Loaders	1		2		3		4		5	

In the solution, destinations 2, 5 and 6 are split into two doors to achieve the minimum of 5 loaders. However, this type of splitting may be regarded as "wasting" doors because it is possible to use only 7 doors where each destination is assigned one door. The tradeoff is that 6 rather than 5 loaders would be needed; destinations 3 and 4 can be combined.

Using the same flow data for the 7 destinations, the more interesting case is when there are 8 doors available. Now we may have a solution with 6 loaders, as shown below.

Destinations	1	2		3	4	5	6	7
Doors	1	2	3	4	5	6	7	8
Volume	0.5	0.4	0.5	0.2	0.7	0.9	0.9	0.2
Loaders	1		2		3	4	5	6

This example shows the tradeoff between the number of doors and loaders. Roughly speaking, it is possible to save loaders by using more doors on a sort. When the number of doors is bounded, though, it may not be possible to achieve the minimum number of loaders associated with the unrestricted case. Therefore, we need to determine when to use more doors than the minimum needed. To this end, we have developed a dynamic programming algorithm.

For the more general problem we are given n_s^{dest} destinations on sort *s* sequentially arranged with corresponding flow that must be allocated to *m* doors, where $m \le m^{max}$. Two decisions must be made: assign destinations to doors and assign loaders to doors. To solve this problem, we need a new concept.

Definition 4. A minimal sub-assignment MSA(*i*, *j*), defined for $i \le j$, is the flow and loader allocation for destinations *i*, *i*+1,…, *j*, where (a) there are a sufficient number of doors available to cover the flow for the

destinations involved, (b) a new loader is started for destination *i*, and (c) each loader, except the last, is assigned maximum flow V_l^{max} given that the loader handles *l* doors.

Now, letting $L(i, j)$ and $R(i, j)$ be the number of loaders and doors needed by MSA(i, j), respectively, we show that the value of these two functions can be determined by applying the `greedy_algorithm` to destinations *i* and *j* only. In the above examples, the `greedy_algorithm` solution for $m = 10$ doors is MSA(1,7) with $L(1,7) = 5$ and $R(1,7) = 10$. [An alternative view of the same solution is MSA(1,3) and MSA(4,7).] The solution for $m = 8$ doors includes five MSAs: MSA(1,3), MSA(4,4), MSA(5,5), MSA(6,6), and MSA(7,7).

Using these concepts, the optimal assignments can be found by enumerating all possible MSAs. The algorithm for doing so is presented below, but first we will prove that there exists an optimal set of assignments formed by consecutive MSAs.

Lemma 1. For the general problem with the destinations in a given sequence and a fixed number of doors, there exists an optimal assignment which is formed by consecutive MSAs.

Proof. We show that any optimal assignment that is not in the form of consecutive MSAs can be so arranged. Suppose that there exists an optimal assignment that is not formed by consecutive MSAs, which implies that there are two adjacent loaders serving the same destination *d* with the former loader's flow less than V_l^{max}. In such a case, we can switch some or all of the second loader's flow to the first loader, either by (i) increasing his flow to V_l^{max} or (ii) making him the last loader serving destination *d*. For (i), the first loader now has maximum flow and becomes part of an MSA. Also, as a consequence of the flow shift, the second loader is only partially occupied. If he shares some flow with the next loader for any destination, then the same procedure can be applied to these two loaders; otherwise, we have an MSA ending with the second loader. For (ii) we have an MSA ending with the first loader. ■

Note that alternate optimal assignments may exist as suggested in the proof that are not formed by consecutive MSAs. For example, the solution when $m = 8$ has MSA(1,3) for the first three destinations. However, a further examination of feasible assignments reveals that there is another assignment with the same 6 loaders and 8 doors, but not in the form of consecutive MSAs; that is,

Destinations	1	2		3	4	5	6	7
Doors	1	2	3	4	5	6	7	8
Flow	0.5	0.3	0.6	0.2	0.7	0.9	0.9	0.2
Loaders	1		2		3	4	5	6

To illustrate the logic of the proof of *Lemma* 1, we see that this solution does not satisfy *Definition* 4 because neither loaders 1 and 2, who are consecutive, are assigned the maximum flow. To put it into the form of consecutive MSAs, we first shift 0.2 from loader 2, door 3, to loader 1, door 2. As a consequence, loader 1 is now assigned the maximum flow and loader 2 has a flow of 0.6. The solution then becomes MSA(1,3) for the first three destinations.

We now present a dynamic programming algorithm for finding optimal assignments for *m* doors. Let $F(i, k)$ be the minimum number of loaders needed to handle the demand from destination *i* to

destination n_s^{dest} using k doors and starting an MSA at destination i. The dynamic programming recursion is

$$F(i,k) = \min_j \{L(i,j) + F(j+1, k - R(i,j)) : j = i, i+1, \ldots, n_s^{dest}\}; i = 1, \ldots, n_s^{dest}; k = i, \ldots, m$$

and the optimal solution is given by $\min\{F(1, n_s^{dest}), F(1, n_s^{dest}+1), \ldots, F(1,m)\}$. To complete the algorithm, we define initial conditions as $F(n_s^{dest}+1, k) = 0$ for $k \geq 0$, and boundary conditions as $F(i,k) = \infty$ for $k < 0$.

To run the dynamic program, we need to calculate all MSA(i,j)'s in advance, which requires a call to `greedy_algorithm` for each MSA(i,j). However, the efficient way of doing this is to calculate MSA(i,i), MSA($i,i+1$),…, MSA(i, n_s^{dest}) in one run, where each MSA($i,j+1$) is calculated based on MSA(i,j) by adding destination $j+1$. Applying this approach, we can calculate all MSA(i,i), MSA($i,i+1$),…,MSA(i, n_s^{dest}) in time $O(m)$, and all MSA(i,j)'s in time $O(n_s^{dest} m)$. Subsequently, the dynamic programming recursion can be run in time $O\left(\left(n_s^{dest}\right)^2 m\right)$, which is thus its complexity, and provides an optimal solution for one sort at a time. Because it only maintains the sequence of destinations and not necessarily the door assignments, the number of switches may increase as each sort is optimized.

Appendix B. Dimensions of MIPs

The instances arising at Step 2 are representative of the size of the MIPs solved in the course of the study. Table 10 reports the number of variables by symbol, the total number of binary variables, and the total number of constraints in model (3a) – (3c). The symbol t^{int} stands for the initial number of transition variables t_{pds}^q in model (2a) – (2b) at Step 1. Once a solution is found giving the minimum number of switches, Δ^{min}, all t_{pds}^q variables are removed that are associated with a greater number of switches than this value.

Table 10. Dimensions for instances of model (3) at Step 2

Hub – workcenter	No. of variables							No. of constraints, m
	t^{init}	x	t	y	u	z	Total binary	
Chicago–1	6,732	2,224	3,006	520	224	1,520	2,968	6,947
Chicago–2	5,582	2,129	1,828	480	232	1,400	2,841	6,649
Chicago–3	5,816	2,010	1,848	520	224	1,520	2,754	6,552
Chicago–4	7,041	2,292	2,238	540	232	1,580	3,064	7,222
Chicago-5	7,915	2,395	3,563	520	224	1,520	3,139	7,615
Chicago-6	6,299	2,099	1,913	460	224	1,340	2,783	6,733
Orlando–1	6,926	2,057	2,026	440	208	1,280	2,705	6,544
Orlando–2	3,684	1,245	1,140	340	152	980	1,737	4,030
Orlando–3	5,140	1,215	1,312	300	176	860	1,691	4,280
Orlando–4	5,992	1,165	1,516	340	168	980	1,673	4,215
Orlando–5	3,953	1,044	1,065	280	168	800	1,492	3,940
Rialto–1	1,817	526	491	200	112	560	838	2,004
Rialto–2	1,826	735	594	240	120	680	1,095	2,502
Rialto–3	1,754	549	511	200	112	560	861	1,960
Rialto–4	2,834	972	795	280	152	800	1,404	3,216
Rialto–5	2,059	431	487	180	112	500	723	1,767
Rialto–6	3,643	1,108	1,018	300	152	860	1,560	3,614
Rialto–7	8,162	1,932	2,095	400	208	1,160	2,540	6,482
Rialto–8	8,907	1,352	2,010	340	208	980	1,900	4,918
Sacramento-1	4,190	1,113	1,110	300	176	860	1,589	3,947
Sacramento-2	3,710	935	1,411	320	144	920	1,399	3,394
Sacramento-3	3,945	1,031	1,086	320	160	920	1,511	3,612
Sacramento-4	2,756	914	814	300	144	860	1,358	3,228
Sacramento-5	4,147	1,180	1,151	380	168	1,100	1,728	4,194
Sacramento-6	3,607	1,093	1,029	340	160	980	1,593	3,889