A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling

Clausen, Tommy

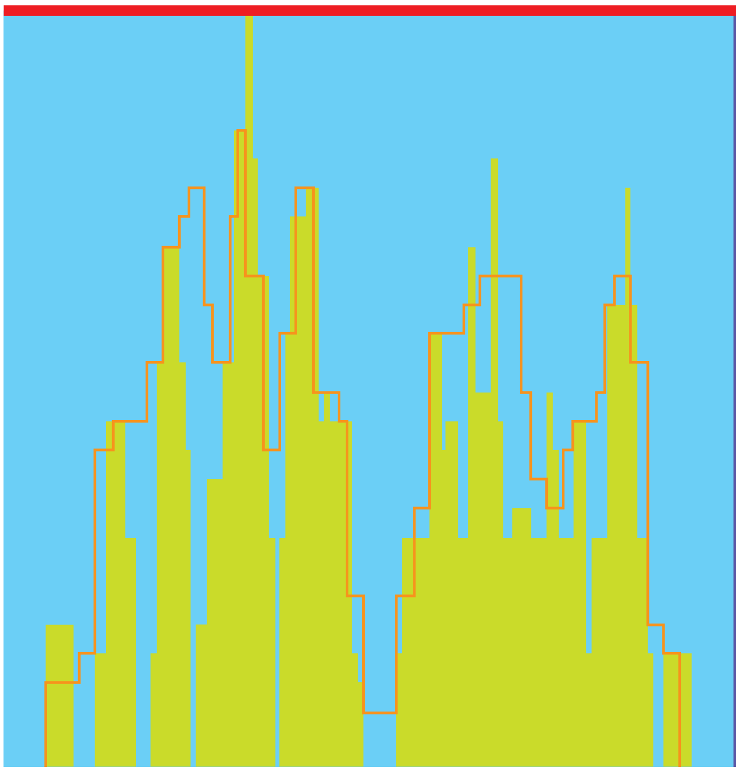
Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Clausen, T. (2010). A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling. Kgs. Lyngby: DTU Management. (DTU Management 2010; No. 7).

A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling



Report 7.2010

DTU Management Engineering

Tommy Clausen

April 2010

DTU Management Engineering
Department of Management Engineering

A Dynamic Programming-Based Heuristic for the Shift Design Problem in Airport Ground Handling

Tommy Clausen[*][†]

April 23, 2010

Abstract

We consider the heterogeneous shift design problem for a workforce with multiple skills, where work shifts are created to cover a given demand as well as possible while minimizing cost and satisfying a flexible set of constraints.

We focus mainly on applications within airport ground handling where the demand can be highly irregular and specified on time intervals as short as five minutes. Ground handling operations are subject to a high degree of cooperation and specialization that require workers with different qualifications to be planned together. Different labor regulations or organizational rules can apply to different ground handling operations, so the rules and restrictions can be numerous and vary significantly. This is modeled using flexible volume constraints that limit the creation of certain shifts.

We present a fast heuristic for the heterogeneous shift design problem based on dynamic programming that allows flexibility in modeling the workforce. Parameters allow a planner to determine the level of demand coverage that best fulfills the requirements of the organization. Results are presented from several diverse real-life ground handling instances.

1 Introduction

The Heterogeneous Shift Design Problem (H-SDP) is concerned with designing a set of work details (shifts) for an organization that specify requirements of different types of employees within the planning period. The shift design problem provide a transformation of the demand from a demand curve (temporal requirements) to more simple shift based requirements. This provides an important tool for organizations to estimate the capabilities of an existing workforce or the need to adjust the workforce to adequately meet the demand.

[*]`tomc@man.dtu.dk`, DTU Management Engineering, Produktionstorvet, Bygn 426, DK-2800 Kgs. Lyngby

[†]WorkBridge A/S, Hauser Plads 18, DK-1127 Copenhagen K

The shifts should be designed such that they allow the organization to satisfy constraints from labor regulations or the capacity of the workforce. Such constraints include the construction of individual shifts, such as shift lengths, the placement and number of relief breaks or the allowed placement of shifts with regards to office hours or similar requirements. Special volume constraints specify a maximum number of shifts within time periods, which may be used to model the size of the workforce and several regulatory constraints such as limits on nights or weekend shifts.

We consider the shift design problem with specific emphasis on the venue of airport ground handling. There is a large number of tasks that must be carried out for every aircraft that lands at an airport, before it is ready for departure. Many airports are extremely busy and are already operating at full or near-full capacity, whilst expecting further growth in passenger numbers in the future. Furthermore, the emergence of low-cost carriers has created additional demands for short aircraft turnaround times, while the hub-and-spoke network structures of larger carriers are creating demand for reduced connection times between airports. The flight arrival or departure times are often placed at periods of high demands or to provide short waiting times for transferring passengers. This means that airport activity can be extremely high during periods of frequent arrivals and departures, whilst other periods of the day are relatively quiet [9].

The demand at airports is usually represented as a demand curve, in which the planning period is divided into a number of equal-sized periods called *time slots*, each having a number of required workers. Arrival and departure times are planned for five minute intervals, yielding a demand interval length of five minutes as well. The combination of short time slots and flight schedules with high variance in activity means that the demand may contain a high level of *irregularity*, where the difference between requirements of two adjacent time slots can be large.

When the demand is irregular, it is unlikely that a set of shifts can cover the demand perfectly, or even come close. Covering a single peak may require a lot of workers, that will be idle before and after the peak. Therefore, it is up to a planner to find a balance between covering the entire demand and minimizing expenditures. This emphasizes the need to consider both staff shortage (*understaffing*) and surplus (*overstaffing*) as part of the desired solution quality. The desired level of *coverage* for an organization depends on the demand and the available workforce.

The work performed within airports is highly specialized. The ability to perform a specific task may require certain security clearances, training or specialized equipment. The requirements may vary greatly across tasks. Different aircraft types may require different types of equipment; different locations in the airport may require different security clearances, and different computer systems (such as check-in terminals) may require different abilities. Many workers may have training in several fields or different certificates, which provide them with capabilities for different partially overlapping areas [10].

The consequence is that shift design for ground handling must handle a heterogeneous demand and workforce. The specialization is modeled by using several demand curves. Similarly, the workforce is divided into groups, each with separate constraints and the ability to cover one or more types of demand.

1.1 Workforce Scheduling

The shift design problem may be viewed as part of a more general problem, the *workforce scheduling* or *staff scheduling* problem. In the workforce scheduling problem, timetables (rosters) are created for workers, such that each worker has a sequence of shifts and rest days that in combination meets a specified demand. Various aspects of the workforce scheduling problem consider different levels of detail and relevant data is available or required at different times. It is therefore common to subdivide the problem into several subproblems that can be solved at different times. One such division into five subproblems (or *stages*) is proposed by Tien and Kamiyama [14]: Stage 1 considers the *temporal manpower requirements*, i.e. demand at each time period or shift. In stage 2 the total manpower requirements are determined. Stage 3 considers blocks of recreation days and Stage 4 combines recreation and work days. Finally, Stage 5 assigns shifts to workers. More recently, Ernst et al. [7] provides a comprehensive survey of workforce scheduling. The survey proposes a taxonomy of different modules of which most workforce scheduling references implement one or more.

In the subdivision of Tien and Kamiyama, *shift design* is part of Stage 1, with some overlap into Stage 2. In the taxonomy of Ernst et al. [7], shift design falls under the area of *demand modeling*, in that it transforms *flexible demand* into *shift based demand*.

There are only a few references in the literature that deal directly with shift design. Musliu et al. [12] present a local search algorithm and show experiments for both random and real-life data from a call-centre. Herbers [8] presents an algorithm based on constraint programming for a task-based demand. DiGaspero et al. [5] presents a local search metaheuristic for the *minimum shift design problem*, where the number of different shifts should be minimized. A construction heuristic based on a min cost flow model is also presented.

Dowling [6] describes a metaheuristic based on local search for staff scheduling at an airport. After each neighborhood iteration, the updated solution is checked for feasibility using an external rule engine. Lau [10] considers the *changing shift assignment problem* where the shift scheduling problem is augmented with constraints that specify feasible transitions between shifts. These *shift change constraints* are used to model airport ground handling problems, where required skills are determined by aircraft type.

To the knowledge of the author, no prior references exist in the literature for the heterogeneous shift design problem.

1.2 Overview

In this paper we describe a fast construction heuristic for the heterogeneous shift design problem. The heuristic emphasizes the ability to balance demand satisfaction versus minimizing excess shifts, while maintaining an even distribution of coverage shortage and surplus. The heuristic uses dynamic programming to iteratively build sequences of shifts to evenly distribute overstaffing and understaffing across the demand period, while satisfying the constraints of the problem that limits the number of shifts created from different groups.

Experiments are performed on real-life data from ground handling operations in major airports around the world. The data is obtained from the WorkBridge PlanManager software product that specializes in workforce scheduling for airport ground handling. The developed algorithm is intended for integration into WorkBridge PlanManager. The experiments show that good solutions can be found for most instances in less than a minute.

The paper is organized as follows: In Sections 2 and 3 we present the H-SDP in detail. Section 4 presents an overview of the solution strategy for iteratively solving relaxations of the H-SDP. In Section 5, we define the 0-1 Shift Design problem as a relaxation of the H-SDP that produces a single sequence of shifts and an algorithm for the 0-1 Shift Design problem based on dynamic programming. Performance considerations are discussed in Section 6. Computational results for diverse problem instances from ground handling are presented in Section 7 and conclusions are presented in Section 8.

2 The Heterogeneous Shift Design Problem

A main characteristic of the shift design problem is that there are conflicting goals, which can be difficult to obtain simultaneously. In this paper, we consider three conflicting goals: Understaffing and overstaffing, and cost. We shall consider understaffing and overstaffing to be most important, and use cost to describe the preferential placement of shifts in cases where coverage is not affected. If the cost is derived directly from the worker's salary, the algorithm will attempt to avoid expensive shifts, such as nights and weekends. The cost can also be used to model the possibility of adding overtime, the expected availability of temp workers or the preferences of the workers.

When the demand is highly irregular, it is impossible to create solutions which are good in terms of both understaffing and overstaffing. Adding shifts to cover peaks in the demand will introduce a lot of overstaffing as well. Often it will be up to the individual planner to decide the optimal balance between understaffing and overstaffing. A good balance between understaffing and overstaffing will depend on the workforce available and the ability to absorb peaked demand into the shifts. If there are few high, thin peaks, the planner may decide not the cover them, in the expectation that the workers will be able to solve them on the day of operation. In other cases, the planner may require that all demand is covered. The correct balance is obtained by allowing the planner to assign weights to the importance of understaffing, overstaffing and cost.

Sometimes peaks in the demand are explicitly removed by a process known as *peak cutting* before planning is done. However, the weighted approach allows the planner to integrate planning and decision making, rather than making the decision *a priori.*

Another focus of shift planning is the distribution of shifts throughout the planning period. When understaffing and overstaffing exists, it should be distributed as much as possible to increase the robustness of the solution. A good distribution of understaffing increases the chance for the workers to cover more demand than scheduled (and thus further reducing understaffing), as well as the likelihood that enough additional workers from a temp agency will be available. Distributed overstaffing decreases the effect of additional work, illness and other disruptions.

3 Basic Notation

Shifts are created for a multi-skilled environment, in which a workload demand may require several skills at once, and a worker may be able to fulfill several types of demand. To model capabilities for workers and demand, we introduce the notion of shift qualifications $q \in Q$ and demand requirements $r \in R$. A requirement (which may include several skills) describes an ability to perform a certain work and a qualification describes all capabilities of a worker following a shift. By using qualifications, shifts are created anonymously, so there is no direct link to the employee that will eventually follow the shift except the implicit expectation that the employee will possess the required skills. In this way, a large degree of flexibility in the workforce is maintained, while the ability to distinguish different employees is preserved.

The shift design problem is specified for at time period, which is subdivided into T smaller time units. Each time unit $t \in [0; T-1]$ has the same duration g (typically 5 minutes) and identifies the time interval $[t \cdot g; (t+1) \cdot g)$. We denote g the *granularity* of the problem. The time period $[0; T)$ is called the planning period and is typically one week or one month.

3.1 The Composition of Shifts

A shift specifies the presence of a single worker and the ability to perform certain types of work, thus covering a portion of the demand. A shift s is defined by a qualification q, a start time t_s, and a length l_s. The qualification q serves two purposes. First it determines the expected capabilities of a worker following shift s. Second, it serves as an index for the rules and regulations valid for the worker, so it is assumed that all workers with qualification q work under the same conditions and constraints.

The shifts are typically not allowed to start at every time t, since for $g = 5$ minutes, the resulting set of shifts would be impossible to manage. The allowed starting times are determined by the *shift granularity* g_q. A starting time t is then valid if $t \mod g_q = 0$. The feasible shift lengths are determined by g_q and minimum and maximum lengths $L_q^{\min}$ and $L_q^{\max}$. Typical shift granularities are 15, 30, or 60 minutes, which for $g = 5$ minutes set $g_q = 3, 6, 12$, respectively.

Additionally, a shift may have several breaks in which the worker is not able to cover demand. These breaks are specified by break rules $b(l_s) = (t^{\min}_{blq}, t^{\max}_{blq}, l_{bq})$ that define valid start positions relative to the shift start and the length of the break l_{bq}. The valid start positions of the break are subject to the shift granularity and the shift length l_s. This allows the break's time window to expand with the shift, and to specify that a break is not relevant for certain shift lengths. A longer shift may for instance require more breaks than a shorter shift. We denote by B_q the set of all break rules for qualification q and by $B = \bigcap_{q \in Q} B_q$ the set of all break rules.

A special kind of break can arise where workers may be allotted time for briefings, wardrobe changes, etc. These are denoted *preparation* and *de-preparation* times and are fixed to the start or end of the shifts. These types of breaks are not uncommon in the airport or ground handling industry [11]. We say in general that a shift is *active* when it is capable of covering demand, and *inactive* otherwise.

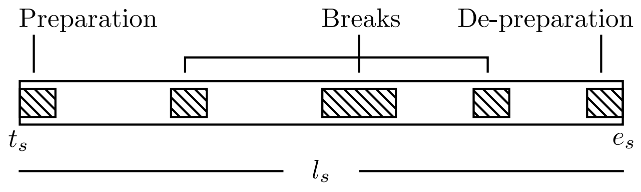


Figure 1: Illustration of shift composition and related variable names for a shift s. Inactive time from preparation time, de-preparation time and three breaks is shown in hatched boxes.

Depending on the flexibility of the shift parameters, a very large number of different shifts can be created. The number of possible shifts is a key factor in the complexity of shift design and related problems. In particular the added dimensions of breaks increase the number of shifts significantly.

The total number of possible shifts is

$$|\mathcal{S}| = \sum_{q \in Q} \frac{T}{g_q} \cdot \frac{L^{\max}_q - L^{\min}_q}{g_q} \cdot \prod_{b \in B_q} \frac{t^{\max}_{blq} - t^{\min}_{blq}}{g_q} \tag{1}$$

We may bound both $L^{\max}_q - L^{\min}_q$ and $t^{\max}_{blq} - t^{\min}_{blq}$ by the maximum shift length, which we may denote $L_{\max}$. This gives the following bound on the number of possible shifts

$$|\mathcal{S}| = O(|Q| \cdot T \cdot L_{\max} \cdot |B|). \tag{2}$$

To reduce the size of shifts that needs to be represented in the algorithm, we define a *shift template* s'. A shift template represents all shifts s that have different starting times t_s, but are otherwise identical. This means that any shift s can be obtained from the corresponding shift template s' by adding a starting time: $s = s'(t_s)$. Each shift template spans l'_s time slots that are either covered or uncovered, so we may consider s' as a binary vector of length l'_s. We denote the set of shift templates S', which by reduction from (2) has size

$$|S'| = O(|Q| \cdot L_{\max} \cdot |B|). \tag{3}$$

Although the size reduction by T is only linear in the size of the algorithm's input (as D has dimensions $|R| \times T$), the decrease may still be significant. For a planning period of one week with a granularity of 5 minutes, we get $T = 2016$.

3.2 Volume Constraints

To model restrictions of the workforce or organizational settings, the overall shift set can be limited by the use of volume constraints. Each volume constraint $v_i = (V_i, v_i^{\max})$ limits the number of shifts allowed within a (not necessarily continuous) time period. The set $V_i \in \mathcal{S}$ contains all shifts affected by the constraint. Thus, we may write each such constraint as

$$\sum_{s \in V_i} s \leq v_i^{\max}.$$

Adjusting the constraint set V_i or the volume limit $v_i^{\max}$ allow volume constraints to model a large number of different scenarios. If $v_i^{\max} = 0$, all shifts in V_i are prohibited, allowing a planner to model closing times or periods where shifts are not allowed to start or end. Setting other values for $v_i^{\max}$ allows the planner to limits varying types of shifts to match the workforce or labor regulations. For example, the total number of shifts, the number of shifts on a single day or the number of certain shift types (e.g. night shifts) may be limited using volume constraints.

The constraint set V_i can easily be adjusted to shift templates by considering the combination of shift template and start time, i.e. $V_i \in \{S' \times T\}$.

$$\sum_{(s',t) \in V_i} s'(t) \leq v_i^{\max}.$$

This adjustment allows us to use shift templates directly in volume constraints.

3.3 Workload Demand

The workload demand is represented as a two-dimensional integer matrix $D \in \mathbb{N}_0^{|R| \times T}$. Each entry D_{rt} specifies the required number of active shifts with suitable qualification to cover requirement r at time unit t.

The link between qualifications and requirements can be complex. A shift with qualification $q \in Q$ may be able to cover demand from several different requirements $r \in R$. Similarly, demand for requirement r can be covered by shifts with different qualifications. We say that q covers r, or $q \rightarrow r$, if shifts with qualification q can contribute to the coverage of demand of requirement r.

To describe the level of complexity between requirements and qualifications we introduce the *interaction level* of an operation as

$$\text{int} = \sum_{q \in Q} \frac{|\{q \mid q \to r\}|}{|Q|}, \tag{4}$$

the average number of requirements covered. Analogously, we may define the interaction of a single qualification as $\text{int}_q = |\{q \mid q \to r\}|$ and for a requirement as $\text{int}_r = |\{r \mid q \to r\}|$.

We assume that requirements are ordered by increasing interaction level, $r_1, \cdots, r_{|R|}$ such that $i < j \Rightarrow \text{int}_{r_i} \leq \text{int}_{r_j}$. We say that r_1 is the *most restrictive* requirement, as it can be covered by the fewest number of qualifications. We similarly assume that qualifications are ordered by decreasing interaction levels, such that q_1 can cover the largest number of requirements.

A shift with qualification q can cover a demand for at most one requirement r at each time t. It is allowed for a shift to cover different requirements at different times. This is because shift qualifications are considered "daily qualifications", which detail the requirement a worker is able to cover within a single day. For practical purposes, this means that given a shift set $\mathcal{S}^*$, the coverage can be calculated independently for each time unit.

4 Algorithm Overview

We wish to develop a solution method with fast running times, and which can be terminated prematurely due to time limitations and still produce a solution of reasonable quality. In this way, the method can be used as a stand-alone heuristic and as a construction heuristic for the initial stage of a local search metaheuristic, similar to the approach of DiGaspero et al. [5].

Due to the possible time limitation, we will prefer an algorithm that throughout the majority of its run is able to return a solution that satisfies the volume constraints and has a good coverage distribution across the planning period.

The H-SDP may be formally described as minimizing the weighted sum of understaffing, overstaffing and cost. To achieve the desired distribution of shifts in regard to the demand, understaffing and overstaffing is squared. As we consider cases where coverage takes precedence over cost, the weight of the cost term is set lower than the others.

The integer programming model of the H-SDP is

$$z^* = \min \phi_u \sum_{t=1}^{T} \sum_{r \in R} (u_{rt})^2 + \phi_o \sum_{t=1}^{T} \sum_{q \in Q} (o_{qt})^2 + \phi_c \sum_{i=1}^{n} c_i x_i \tag{5}$$

$$\text{s.t.} \sum_{i=1}^{n} a_{it} d_{ir} x_{it} + u_{rt} - \sum_{q \in Q} p_{iq} o_{qt} \geq D_{rt} \qquad \forall r, 1 \leq t \leq T \tag{6}$$

$$\sum_{i=1}^{n} a_{it} x_i + \sum_{r \in R} u_{rt} + \sum_{q \in Q} o_{qt} - \sum_{r \in R} D_{rt} = 0 \qquad 1 \leq t \leq T \tag{7}$$

$$\sum_{i=1}^{n} v_{ij} x_i \leq v_j^{\max} \qquad 1 \leq j \leq V \tag{8}$$

$$x_i, u_{rt}, o_{qt} \in \mathbb{N}_0 \tag{9}$$

Here, c_i is the cost of shift i, $1 \leq i \leq n$ and $a_{it} = 1$ if shift i is active at time t, $1 \leq t \leq T$ and $a_{it} = 0$ if shift i is not active at time t. $d_{ir} = 1$ if shift i can cover requirement r and $p_{iq} = 1$ if shift i has qualification q. D_{rt} is the demand for requirement r at time t. The decision variable x_i determines the quantity of each shift type i. Additionally, the decision variable u_{rt} and o_{qt} determines the understaffing of requirement $r \in R$ at time t and the overstaffing of qualification $q \in Q$ at time t, respectively.

Each volume constraint v_j is specified by the constraint limit $v_j^{\max}$ and the incidence matrix V, where $v_{ij} = 1$ if shift type i is affected by rule j, and $v_{ij} = 0$ otherwise.

The objective function (5) is a weighted sum of the total shift cost, total understaffing, and total overstaffing using the weight factors ϕ_c, ϕ_u and ϕ_o, respectively. Constraint (6) links shifts, understaffing and eligible overstaffing to the demand at each time unit. Constraint (7) enforces that the total amount of shifts, understaffing and overstaffing adds up, ensuring that a unit of understaffing or overstaffing is only used once. Constraint (8) ensures that the volume constraints are not violated. Finally, constraint (9) states that the shift, understaffing, and overstaffing variables are positive integers.

The formulation (5)–(9) defines an explicit formulation of the SDP, where any possible shift is represented by a column. This is in a sense an expanded model of the problem, since each shift is directly represented. This means that each shift template s' is represented a large number of times with different starting times.

This kind of model is typically used in the related shift scheduling problem, where it was originally proposed by Dantzig [4] as a set covering model. As the number of shift templates grows rapidly with the flexibility of the shift design, the model can become very large. For this reason, several implicit models have been proposed to reduce the size of the model. Most often the size is reduced by adding special *forward* and *backward* constraints to handle break placements, thus reducing the number of shifts considerably. See e.g. [1], [2], [8], and [13].

In this paper we use a modeling similar to the explicit model (5)–(9). Instead of reducing the problem size by implicit modeling, we use shift templates instead of shifts to obtain a lesser reduction in problem size. We then solve a relaxed version of the problem.

4.1 Iterated Relaxations

The main idea of the algorithm is to perform a two-step relaxation of the H-SDP to get a series of *inner* and *outer* subproblems. The outer subproblem considers one requirement at a time. For each requirement, a number of inner subproblems are solved, each producing a single non-overlapping sequence of shifts. Each outer subproblem terminates when no shifts are returned from the inner subproblem.

The outer subproblem splits the multi-requirement problem into a series of single-requirement problems. That is, we split the H-SDP with demand matrix D into separate subproblems SDP_r with demand vector $d = D_{r*}$. We iteratively solve the subproblems SDP_r, and for each restriction r we consider only qualifications that can cover r.

The relaxed subproblem is then essentially a single-skill shift design problem, although coverage is still evaluated across all subproblems. This means that any understaffing or overstaffing in the solution of one subproblem will be carried to the next. Also, the same qualification can occur in more than one subproblem, so the volume constraints will become more restrictive during the iterations. Each subproblem SDP_r is then still somewhat heterogeneous, although less than the original problem. The amount of information lost in the subdivision depends on the interaction level of the problem instance. To minimize the effect of previously considered subproblems, we consider the requirements in order of restrictiveness, starting with the most restrictive requirement, r_1.

The inner subproblem subdivides SDP_r even further. Instead of considering demand as an integer vector $d \in \mathbb{N}_0^T$, we consider a binary vector $d' \in \{0,1\}^T$ that for each time t denotes if uncovered workload remains, i.e. if $u_{rt} > 0$. Having reduced the dimension of the demand, we can analogously seek a "one-dimensional" solution to the problem as well. The relaxation of SDP_r is therefore to find a sequence of shifts $\sigma \in \mathcal{S}$ that covers d' as well as possible without violating any of the volume constraints v_i. We denote the binary subproblem 0-1-SDP_r. To avoid cases where 0-1-SDP_r repeatedly returns valid but poor shift sequences, we only return a sequence if it satisfies the termination criteria $\max_u$ and $\max_o$ that limits the maximum understaffing and overstaffing allowed for a sequence.

The conceptual idea of the algorithm is summarized as follows: For each requirement r_i, we solve the subproblem SDP_r by iteratively creating sequences of shifts as specified by the 1-dimensional inner subproblem 0-1-SDP_r. The conceptual algorithm is sketched in Algorithm 1.

Algorithm 1 Pseudocode of conceptual algorithm for solving the H-SDP

```
$\mathcal{S}^* \leftarrow \emptyset$
for $r \leftarrow r_1$ to $r_{|R|}$ do
    $Q_r \leftarrow \{q \mid q \rightarrow r\}$
    $d_r \leftarrow D_{r,*}$
    repeat
        $\sigma^* =$ Solve 0-1 SDP$_r$($Q_r, d_r$)
        $\mathcal{S}^* \leftarrow \sigma$
    until $\sigma^* = \emptyset$
end for
return
```

5 Solving the 0-1 Shift Design Problem

For the 0-1 shift design problem, a number of simplifications can be made which allow the problem to be solved more efficiently than the full H-SDP. As both the demand vector d' and the sequence of created shifts σ is one-dimensional, the volume constraints and coverage measures can be calculated efficiently. The sequence can be considered both as set of shifts: $\sigma = s_1 \cup s_2 \cup \cdots \cup s_n$; and as a set of shift templates: $\sigma = s'_1(t_1) \cup s'_2(t_2) \cup \cdots \cup s'_n(t_n)$. We will mainly use the latter representation in the following.

When considering a non-overlapping sequence of shifts σ, at most one shift can contribute to the demand at any time t. Therefore, we can consider σ as a binary vector $\sigma \in \{0,1\}^T$, which is obtained by concatenating the binary representations of the shift templates in σ.

When considering sequences of shifts for the 0-1-SDP$_r$, a lot of the complexities of the original H-SDP is removed. To compensate for this, the objective function is in some ways simpler, but require other terms to compensate for the simplifications. The quadratic coverage terms of the original problem u^2 and o^2 have been reduced to linear versions $u(\sigma)$ and $o(\sigma)$ since there is no effect of squaring the coverage of a $0-1$ demand. Instead, we rely on the iterated solution method to distribute the coverage. To ensure that the least capable shift is used whenever possible, we minimize the shift sequence's overall interaction level $\text{int}(\sigma)$. This will prioritize the least capable shifts for the currently considered requirement, and save the more capable shifts for less restrictive requirements considered in later iterations, where they are most likely to be usable.

As a technical term, we also introduce the *coverage* of a sequence on a 1-dimensional demand d' as the number of time units where $d'_t = 1$ and $\sigma_t = 1$.

For a shift sequence, the set of volume constraints can be combined into a single binary matrix $V' \in \{\textbf{true}, \textbf{false}\}^{T \times |S'|}$ that for each combination of shift template s' and start time s_t determines if the corresponding shift is legal. We write $V'(\sigma) = \textbf{true}$ if $V'[s_t, s'] = \textbf{true}$ for all shifts in σ. When computing a sequence, V' is considered static, so it is possible to construct a sequence where each individual shift is legal, but the entire sequence violates a volume constraint (since several shifts contribute to the same constraint). There are several ways to handle this, the simplest being to allow small violations to the value constraints. Another simple method is to arbitrarily remove a shift from the sequence, if it contributes to a violated volume constraint. After each sequence is created, V' is updated to reflect the new shifts.

The terms of ϕ and V' can be calculated efficiently by using simple operations.

$$\begin{aligned} u(\sigma) &= \|d'_t \wedge \neg\sigma\|_1 && (10)\\ o(\sigma) &= \|\neg d'_t \wedge \sigma\|_1 && (11)\\ cov(\sigma) &= \|d'_t \wedge \sigma\|_1. && (12)\\ c(\sigma) &= \sum_{s\in\sigma} c(s) && (13)\\ \mathrm{int}(\sigma) &= \sum_{s\in\sigma} \mathrm{int}_{q(s)} && (14)\\ V'(\sigma) &= \bigwedge_{s\in\sigma} V'(s) && (15) \end{aligned}$$

Here, $\|\cdot\|_1$ denotes the Manhattan norm, which is the sum of the vector elements. For the binary vectors used here, this corresponds to counting the number of ones in the vector. This problem is also known as the *population count* or *popcount* problem and can be solved efficiently by using e.g. the `HAKMEM` method [3].

We use a weighted sum of these measures as the objective function for the 0-1 SDP:

$$\phi(\sigma) = -\phi_{cov} cov(\sigma) + \phi_u u(\sigma) + \phi_o o(\sigma) + \phi_c c(\sigma) + \phi_{int} \mathrm{int}(\sigma).$$

The relaxed subproblem 0-1-SDP$_r$ may be formally described as an IP model by reducing the original model (5)–(9) in Section 4. The relaxed model is

$$\begin{aligned} z'^* = \min & \sum_{i=1}^{n} \phi(x_i) && && (16)\\ \text{s.t.} & \sum_{i=1}^{n} a_{it} x_i + u_t - o_t = d'_t && 1 \le t \le T && (17)\\ & \sum_{i=1}^{n} a_{it} x_i \le 1 && 1 \le t \le T && (18)\\ & \sum_{i=1}^{n} v_{ij} x_i \le v_j^{\max} - v'_j && 1 \le j \le V && (19)\\ & x_i, u_{rt}, o_{qt} \in \{0,1\} && && (20) \end{aligned}$$

With a slightly changed notation, $\phi(x_i)$ is the objective $\phi(s)$ for the shift s identified by x_i. The understaffing variables u_t and overstaffing variables o_t have been simplified, since for a one-dimensional demand, there can only be a single item of either understaffing or overstaffing. As in the H-SDP model (5)–(9), u_t and o_t are connected to the shifts by constraint (17). Constraint (18) ensures that there is no overlap in the generated shift sequence. Constraint (19) enforces the volume constraints, where v'_j is the shift contribution to volume constraint j obtained in previous iterations. In this way, the rules can be evaluated globally, across the different subproblems. Finally, constraint (20) states that the decision variables are binary.

5.1 The Dynamic Programming Recursion

We use dynamic programming to solve the 0-1-SDP$_r$. The dynamic programming table ν is two-dimensional and contains time as one dimension and the maximum number of time slots with uncovered demand as the other. A cell $\nu[u,t]$ on the table indexes a sequence ending at t and has maximum understaffing u.

A partial sequence can be any sequence $\sigma_{t,e}$ starting at time t and ending at time e. We may combine partial shift sequences by concatenation to create new partial sequences for which it holds

$$\sigma_{t,e} = \sigma_{t,t'} \oplus \sigma_{t',e} \tag{21}$$

$$\phi(\sigma_{t,e}) = \phi(\sigma_{t,t'}) + \phi(\sigma_{t',e}) \tag{22}$$

Some notable cases of partial sequences are the full sequence $\sigma = \sigma_{0,T}$ and a single shift template $s'(t) = \sigma_{t,d_{s'}}$.

Intuitively, the content of any cell $\nu[u,t]$ represents the best legal shift sequence σ of shifts that ends at t and leaves no more than u time slots of workload demand uncovered. To determine the best shift sequence for a table position, we use a restricted version of the objective function $\phi(\sigma)$. Since the understaffing $u(\sigma)$ is explicitly considered in the dynamic programming table, $\phi'(\sigma)$ consists of the remaining terms:

$$\phi'(\sigma) = \phi(\sigma) - u(\sigma) = -\phi_{cov}cov(\sigma) + \phi_o o(\sigma) + \phi_c c(\sigma) + \phi_{int}\text{int}(\sigma).$$

We use $\phi'(\sigma)$ as the dominance criterion of the dynamic programming recursion, so sequence σ_1 dominates σ_2 if $\phi'(\sigma_1) < \phi'(\sigma_2)$. For ease of notation, we use $\nu[u,t]$ to denote both the cell at (u,t) and the partial sequence indexed by cell.

We solve the dynamic programming table using a recursion that gradually builds sequences starting at cell $\nu[1,1]$. We set $\phi'(\nu[1,1]) = 0$ and $\phi'(\nu[1,1]) = \infty$ for all other t and u. From each cell $\nu[u,t]$, the partial shift sequence $\nu[u,t]$ is extended with all shift candidates to produce longer sequences. Thus from cell $\nu[u,t]$ we create sequence $\sigma' = \nu[u,t] \oplus s'$ and we set $\nu[u+u_s, t+l_s] = \sigma'$ if $\sigma' \in V'$ and $\phi'(\sigma') < \phi'(\nu[u+u_s, t+d_s])$. If no valid sequence exists for some understaffing/time pair (u,t) then $\phi(\nu[u,t]) = \infty$. Every cell should satisfy the invariant that the sequence indexed by the cell minimizes ϕ' over all partial sequences ending at t with understaffing less than u:

$$\nu[u,t] = \underset{\sigma_{0,t}}{\arg\min} \left\{ \phi'(\sigma) \mid u(\sigma) \leq u \cap \sigma \in V \right\}, \tag{23}$$

We may formally write the dynamic programming recursion as

$$\nu[u,t] = \min_{s' \in S'} \begin{cases} 0 & t = 0, \\ \phi'(s'(t - l_{s'}) + \nu[u - u(s'(t - l_{s'})), t - l_{s'}] & V'[s',t] = \mathbf{true}, \\ & t \geq l_{s'}, \\ & u \geq u(s'(t - l_{s'})) \end{cases} \tag{24}$$

From the recursion it can be seen directly that the invariant (23) will be satisfied for all cells.

After running the recursion, each cell $\nu[u,T]$ contains a full non-dominated valid shift sequence with understaffing u or less, if such a sequence exists. Each full sequence can then be selected as the solution σ^* to the 0-1-SDP$_r$. The approach we have chosen is to use the full objective function ϕ by taking

$$\sigma^* = \operatorname*{arg\,min}_{0 \leq u \leq u_{\max}} \{\phi'(\nu[u,T]) \mid o(\nu[u,T]) \leq o_{\max}\}. \tag{25}$$

If $\phi(\sigma^*) = \infty$ or no sequence with $o(\sigma) \leq \max_o$ can be found, we set $\sigma^* = \emptyset$. If this is the case, no shifts are created and the inner loop of the algorithm terminates. No further shifts are created for the current requirement r_i and the algorithm moves on to requirement r_{i+1}.

For the terms of ϕ' that are calculated using bitwise operations, the terms can also be calculated on partial sequences by adding a few extra bitwise shifts and bitmasks. This allows the calculation of ϕ' to be decomposed into separate partial sequences, as in (22).

5.2 Algorithm Complexity

The algorithm for solving the H-SDP may be described in detailed pseudocode in Algorithm 2. In the pseudo-code, `1d-Demand` produces the binary vector d' of one-dimensional demand. The function `UpdateRules` creates or updates the matrix V' of allowed shift templates based on the volume constraints. Function `CalculateSequence` uses dynamic programming to produce a sequence of shifts σ^*, which are then added to the solution set $\mathcal{S}^*$.

The complexity of the algorithm depends on the complexities of SDP$_r$ and 0-1-SDP$_r$. The inner problem 0-1-SDP$_r$ builds the dynamic programming table ν and runs the recursion. The size of ν is $T \times \|d'\|_1 \cdot u_{\max}$, where $u_{\max} \leq 1$ and $\|d'\|_1 \leq T$. For each cell in ν, the recursion loops over all shift templates $s' \in S'$. The evaluation of each shift template s' is done by calculating $\phi(s')$.

By using bitwise operations, $\phi(s')$ can be calculated in time $\log_b(|s'|)$, where b is the number of bits in a cpu register word (usually 32 or 64). For a word size of 32 bits, 5 words can represent a shift spanning 160 time units. With a granularity g of 5 minutes, 5 words can then represent a shift longer than 13 hours. We assume that this is always sufficient for representing shifts, i.e. that $\log_b(|s'|) \leq 5$. We consider $\phi(s')$ to be a constant time operation.

Algorithm 2 Algorithm pseudo-code

```
$\mathcal{S}^* \leftarrow \emptyset$
$u_{max} \leftarrow$ maximum understaffing ratio
$o_{max} \leftarrow$ maximum overstaffing ratio
for $r \leftarrow r_1$ to $r_{|R|}$ do
  repeat
    $d' \leftarrow$ 1d-Demand($r$)
    $u' \leftarrow u_{max} \cdot \|d'\|_1$
    $o' \leftarrow o_{max} \cdot \|d'\|_1$
    $V' \leftarrow$ UpdateRules()
    $\sigma \leftarrow$ CalculateSequence($d', V'$)
    $\mathcal{S}^* \leftarrow \mathcal{S}^* \cup \sigma$
  until $\sigma = \emptyset$ or $u_\sigma > u'$ or $o_\sigma > o'$
end for
return $\mathcal{S}^*$
```

The time complexity of 0-1-SDP$_r$ is then

$$O(T^2 \cdot |S'|).$$

The outer subproblem SDP$_r$ calls 0-1-SDP$_r$ until the termination criteria is met. The number of calls depends on the termination criteria and on the amount of demand. In the worst case, the sequence returned by each call will cover a single unit of demand, so the number of iterations may equal the number of units of demand for requirement r, $\|D_{r*}\|_1$.

The total time complexity of solving SDP$_r$ is then

$$O(\|D_{r*}\|_1 \cdot T^2 \cdot |S'|)$$

As we solve SDP$_r$ once for each $r \in R$, the overall time complexity of the algorithm is

$$O(\|D\|_* \cdot T^2 \cdot |S'|) \tag{26}$$

where $\|\cdot\|_*$ denotes the sum of all the elements in the matrix (corresponding to the Manhattan norm of the vectorization of the matrix),

$$\|D\|_* = \|vec(D)\|_1 = \sum_{r \in R} \sum_{t=1}^{T} D_{rt}.$$

The algorithm is polynomial in the factors $|R|$ and T, since the demand matrix D of size $|R| \times T$ is provided as input. The algorithm is pseudo-polynomial in the term $\|D\|_*$, as complexity depends directly on the values of the entries in D, which are given as integers in the input.

Note that even in the optimal case where perfect coverage is achieved, the number of iterations will still equal the highest point of demand in D_r, $\max_t\{D_{rt}\}$, which is also pseudo-polynomial in the size of the input.

The number of shift candidates is given by (3) in Section 3.1. By inserting in (26), we get

$$O(|R| \cdot \|D_r\|_* \cdot T^2 \cdot |Q| \cdot L_{\max} \cdot |B|).$$

From this we see that the number of shift templates is also pseudo-polynomial in the size of the input, as the complexity depends directly on the maximum shift length $L_{\max}$ which is given as an integer value. Combined with the other factors contributing to the number of shift templates, the complexity relating to $|S'|$ may increase rapidly.

6 Performance Considerations

The running time of the algorithm presented in the previous section is mainly dominated by the time complexity of the dynamic programming recursion. In this section we review several approaches to improving performance by reducing the time of the recursion.

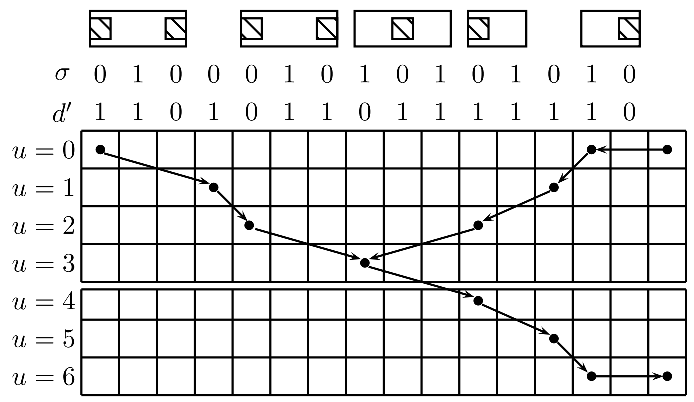
6.1 Bi-directional Recursion

The size of the dynamic programming table is determined by the maximum understaffing $\max_u$. Even with a restrictive choice of $\max_u$, the table may get large if there are many ones in the demand vector. However, the need for understaffing increases as the time t increases, so if we were to generate a sequence for only half of the planning period, the expected need for understaffing could be halved as well. This is the idea behind a bi-directional approach, where partial sequences are simultaneously generated from the beginning and end of the planning period and then merged into full sequences. The basic approach and expected reduction in table size is illustrated in Figure 2.

More formally, the bi-directional approach constructs partial sequences extending *forward* $\sigma^+ = \sigma_{0,t^+}$ and *backwards* $\sigma^- = \sigma_{t^-,T}$. The partial sequences can be combined into a full sequences if $t^- = t^+$, in which case the sequence score is $\phi(\sigma) = \phi(\sigma^+) + \phi(\sigma^-)$. We compute the forward sequences in the interval $[0; T^+]$ and the backward sequences in the interval $[T^-; T]$. To be able to merge the sequences, we must have $T^+ \geq T^-$. The distance between T^+ and T^- must be large enough to allow any full sequence to have at least one shift start/end point within the interval. To achieve this, we set $T^+ = \frac{T}{2} + \frac{L_{\max}}{2}$ and $T^- = \frac{T}{2} - \frac{L_{\max}}{2}$, where $L_{\max}$ is the maximum shift length.

The generation and merging of forward and backward sequences is illustrated in Figure 3.

When merging forward and backward sequences, all cells in the interval $[T^-; T^+]$ need to be considered. Since the dominance criterion $\phi'(\sigma)$ only considers a single sequence, each cell in $[T^-; T^+]$ can have a value for both the forward and backward recursion. We denote these $\nu^+[u, t]$ and $\nu^-[u, t]$, respectively.



σ 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0
d' 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0
$u = 0$
$u = 1$
$u = 2$
$u = 3$
$u = 4$
$u = 5$
$u = 6$

Figure 2: Illustration of a path through the table for a shift sequence σ. The shifts and associated binary vector are shown on top, above the binary demand vector. The tables shows the space requirements for the forward recursion (center + bottom) and the bi-directional approach (center only).

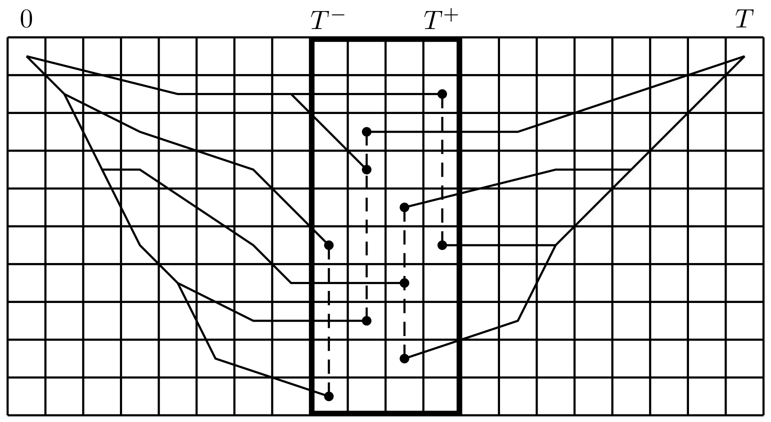


Figure 3: Merging forward and backward recursion paths. Paths can be merged if they share ending column (dashed).

Selecting the best sequence σ^* is similar to (25) for the single-directional recursion:

$$\sigma^* = \underset{\substack{0 \leq u \leq max_u \\ T^- \leq t \leq T^+}}{\arg\min} \{\phi(\nu^+[u,t]) + \phi(\nu^-[u,t]) \mid o(\nu^+[u,t]) + o(\nu^-[u,t]) \leq o_{\max}\}. \quad (27)$$

As for the single-directional recursion, the inner loop is terminated if $\phi(\sigma^*) = \infty$ or no sequence is found with $o(\sigma^*) \leq o_{\max}$.

6.2 Limiting Table Size

In order to limit the running time of the recursion, we limit the state space in several ways, some of which have already been described: The strict dominance of the objectives ensures that the dynamic programming table have only two dimensions, keeping the number of table cells relatively small.

Shift granularities and the volume constraints reduce the number of time slots that can be a start time or end time for a shift. Since the construction of the sequence is only concerned with the transitions between shifts, we remove cells where start or end cannot occur. We denote the set of remaining *active* time units T^*. For instances with restrictive constraints on the placements of shifts due to office hours or low shift granularity, T^* may be significantly smaller than T. The coverage objectives are still calculated on the full demand, taking into account that there may be an uneven distance between two adjacent time units in T^*.

Another determining factor in the running time of the recursion is the number of shift templates. The set of templates can be reduced by superimposing a more restrictive granularity g^* on the shift granularity g_q. Since g_q is related to both the flexibility of shifts and breaks, the effect of using g^* may be significant. Substituting g^* for g_q also affects the size of T^*.

Finally, the limit on understaffing $u_{\max}$ limits the table size directly, so setting a restrictive limit not only causes the algorithm to terminate sooner, but also reduces the running time of each recursion.

7 Computational Results

In this section we review the computational aspects of the dynamic programming algorithm on several instances from real life ground handling operations. We investigate the effect of different combinations of parameters on a subset of the instances, and present full results on all instances on the selected set of parameters. The experiments are focused on balancing overstaffing and understaffing. As a result, we will use a simplified model for cost, and not consider the effects of this parameter in detail.

We introduce the problem instances and their characteristics in Section 7.1. Parameter effects are presented in Section 7.2 and finally, large scale results are presented in Section 7.3.

Type	Abbre-viation	Description
Passenger Services	`pax`	Terminal work, such as check-in counter staffing and boarding.
Ramp	`ramp`	Aircraft-centric tasks, e.g. pushback or refueling.
Transportation	`trans`	Airport transportation tasks, such as baggage delivery.
Cargo	`cargo`	Loading and handling of cargo.
Operations	`ops`	General operations handling.

Table 1: Characterization of operation types

7.1 Problem Instances

The problem instances are taken from real-life instances modeling ground handling operations. There are several different types that varies in the *type* of operation, and in the size and complexity of the problem. The instance type reflects the ground handling tasks performed by the modeled operation. There are several types with different characteristics, as presented in Table 7.1 Another key characteristic of a problem instance is the *size*, which can be measured along the two axes of the workload demand curve. First, the number of distinct data points T, and secondly the total workload demand, measured as the area of the demand curves, $\|D\|_*$. The final characteristic is the complexity of the operation, which we measure as the interaction level between requirements and qualifications, int, as defined in (4). A list of problem instances and their characteristics is provided in Table 2.

7.2 Parameter Tuning Results

In this section we investigate the effects of different parameter values on the solution quality. We group the parameters into several groups, which we address separately in order to limit the overall complexity. To simplify the experiments presented here, we use pre-determined weights for the scoring function $\phi_w = (\phi_o, \phi_{cov}, \phi_u, \phi_{int}, \phi_c) = (100, -10, 1, 0.5, 0.1)$. These weights have been selected to maintain a lexicographic ordering between the individual terms. Overstaffing will always be weighted highest, followed by coverage, and so on. Setting the weights to these values corresponds to a planner having high priority on obtaining a good coverage and lowest priority on minimizing cost.

We review the performance of the algorithm with different sets of parameters using two sample scenarios: `pax.r.a` and `ramp.g.a`. These scenarios have been chosen because they represent each of the two major operation types, `pax` and `ramp` and are not too similar in terms of interaction level and size.

Scenario	$\|R\|$	$\|Q\|$	int	T	$\|D\|_*$
`cargo.m.a`	8	5	0.44	2016	15294
`ops.e.a`	3	5	0.5	8064	17887
`pax.e.a`	6	10	0.21	8064	59329
`pax.g.a`	1	2	0.5	1008	11981
`pax.g.b`	1	2	0.5	1008	14143
`pax.m.a`	10	9	0.52	2016	12796
`pax.r.a`	3	8	0.15	336	3721
`pax.r.b`	3	12	0.15	336	3721
`ramp.e.a`	9	15	0.08	1344	6677
`ramp.e.b`	7	10	0.58	8064	59383
`ramp.e.c`	7	10	0.58	8064	59383
`ramp.g.a`	1	2	0.5	1008	7275
`ramp.g.b`	1	2	0.5	1008	8710
`ramp.m.a`	13	9	0.19	2016	3448
`ramp.r.a`	1	4	0.25	336	1943
`trans.o.a`	6	12	0.5	1008	12925
`trans.o.b`	6	12	0.5	1008	19603

Table 2: Characteristics of problem instances

7.2.1 Termination Criteria

We first address the termination criteria $\max_u$ and $\max_o$. Table 3 presents the effects of running with different termination limits on `pax.r.a` and `ramp.g.a`.

Figure 4 graphically shows the connection between relative understaffing $u/|D|$, relative overstaffing $o/|D|$ and the running time of the experiments. As seen in the figure, the selected termination criteria each produce a non-dominated solution, where u cannot be decreased without increasing o. As $u/|D|$ approaches 0, $o/|D|$ becomes increasingly higher (and analogously for $o/|D|$ and $u/|D|$), which means that in many cases a balanced approach is preferable.

All non-dominated sets of parameters may be considered good, so choosing the best combination depends on the priorities of the planner.

The actual levels of coverage obtained by setting different termination criteria are illustrated again in Figure 5. The figure shows the coverage as contour on the workload for scenario `ramp.g.a` with the two termination criteria $(\max_u, \max_o) = (0.7, 0.4)$ and $(\max_u, \max_o) = (0.8, 0.6)$. The result for $(\max_u, \max_o) = (0.8, 0.6)$ covers more of the demand, but with a higher degree of overstaffing.

From the illustrations we identify $(\max_u, \max_o) = (0.7, 0.4)$ as the most promising combination for obtaining a balance between understaffing and overstaffing, while keeping the running time low. We choose these parameters for further study, as we wish to emphasize this aspect of the algorithm.

Scenario	$u_{\max}$	$o_{\max}$	Iters	o	u	$o/\|D\|_*$	$u/\|D\|_*$	$\|\mathcal{S}^*\|$	Time (s)
`pax.r.a`	0.5	0.2	29	288	4074	0.01	0.18	398	10.25
`pax.r.a`	0.5	0.4	33	666	3150	0.03	0.14	438	10.48
`pax.r.a`	0.5	0.6	37	1272	2478	0.06	0.11	465	10.96
`pax.r.a`	0.7	0.2	38	384	3570	0.02	0.16	448	14.66
`pax.r.a`	0.7	0.4	57	2472	1596	0.11	0.07	502	17.78
`pax.r.a`	0.7	0.6	84	4020	504	0.18	0.02	595	20.35
`pax.r.a`	0.8	0.2	49	576	3294	0.03	0.15	477	18.18
`pax.r.a`	0.8	0.4	119	4176	456	0.19	0.02	618	28.09
`pax.r.a`	0.8	0.6	143	4770	210	0.21	0.01	682	29.78
`pax.r.a`	0.9	0.2	179	3708	846	0.17	0.04	589	43.92
`pax.r.a`	0.9	0.4	287	5166	114	0.23	0.01	722	55.89
`pax.r.a`	0.9	0.6	290	5214	78	0.23	0	725	54.61
`ramp.g.a`	0.5	0.2	16	1036	4450	0.07	0.31	151	2.35
`ramp.g.a`	0.5	0.4	20	1926	2904	0.13	0.2	185	2.65
`ramp.g.a`	0.5	0.6	24	3120	1686	0.21	0.12	218	3.03
`ramp.g.a`	0.7	0.2	28	1590	3360	0.11	0.23	176	4.71
`ramp.g.a`	0.7	0.4	39	3506	1436	0.24	0.1	230	5.64
`ramp.g.a`	0.7	0.6	44	4454	932	0.31	0.06	250	6.47
`ramp.g.a`	0.8	0.2	44	2260	2554	0.16	0.18	197	7.91
`ramp.g.a`	0.8	0.4	61	4292	1010	0.29	0.07	248	9.22
`ramp.g.a`	0.8	0.6	68	5178	648	0.36	0.04	267	9.77
`ramp.g.a`	0.9	0.2	96	3812	1454	0.26	0.1	251	14.38
`ramp.g.a`	0.9	0.4	151	6694	412	0.46	0.03	311	17.06
`ramp.g.a`	0.9	0.6	170	7718	296	0.53	0.02	330	17.9

Table 3: Effects of altering termination parameters for the recursion. $\max_u$ and $\max_o$ are the termination limits, o and u are the resulting units of over- and understaffing and $o/|D|$ and $u/|D|$ presents the coverage ratio to the total demand. $|\mathcal{S}^*|$ is the number of created shifts.

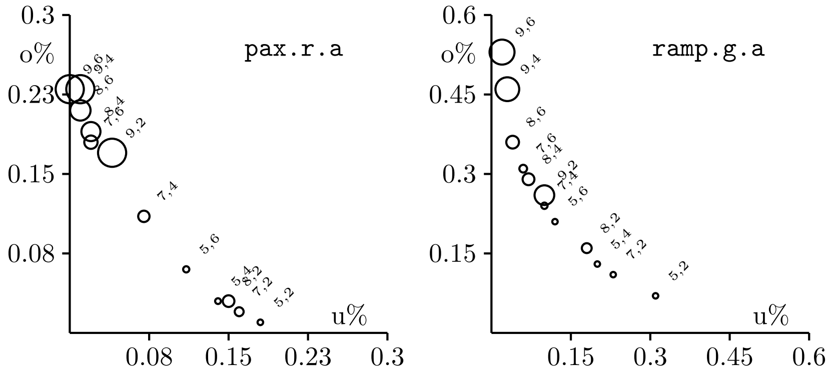


Figure 4: Illustration of results for different termination criteria for scenarios `pax.r.a` (left) and `ramp.g.a` (right). Each circle represents the results of a run. The position indicates the resulting mix of relative understaffing and overstaffing. The diameter of the circle represents the runtime. Used termination criteria $u_{\max}, o_{\max}$ ($\times 10$) is shown next to each run.

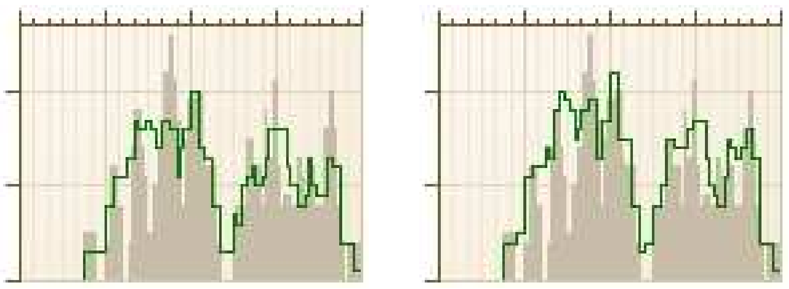


Figure 5: Coverage for a sample day using termination criteria $(\max_u, \max_o) = (0.7, 0.4)$ (left) and $(\max_u, \max_o) = (0.8, 0.6)$ (right) for scenario `ramp.g.a`.

Scenario	g^*	B-D	Iters	Cells	$\|S'\|$	$o/\|D\|_*$	$u/\|D\|_*$	$\|\mathcal{S}^*\|$	Time (s)
`pax.r.a`	60	N	57	50913.49	239.21	0.11	0.07	502	18.16
`pax.r.a`	60	Y	52	26357.06	242.46	0.07	0.11	481	10.58
`pax.r.a`	30	N	54	82778.50	1284.33	0.02	0.02	492	148.16
`pax.r.a`	30	Y	54	41149.57	1298.59	0.02	0.02	481	85.86
`ramp.g.a`	60	N	39	60447.15	31	0.24	0.1	230	5.83
`ramp.g.a`	60	Y	40	29840.85	31	0.24	0.11	232	4.46
`ramp.g.a`	30	N	37	124157.95	91	0.22	0.09	225	28.05
`ramp.g.a`	30	Y	39	61801.72	91	0.22	0.09	231	18.03

Table 4: Effects of altering shift granularity and recursion direction for the sample scenarios. B-D indicates whether the bi-directional recursion is used. Cells and $|S^*|$ is the average number of visited cells and shift templates per iteration. $o/|D|$ and $u/|D|$ is the relative amount of over- and understaffing compared to the total workload demand. $|\mathcal{S}^*|$ is the resulting number of shifts.

7.2.2 Performance Parameters

In this section, we review the effect of two parameters that directly influences the size of the problem: The bi-directional recursion, and the granularity of the used shift templates. The finer the granularity, the more different shift templates are allowed. We run experiments with a modest setting of 60 minutes and a more fine-grained setting of 30 minutes. Results of the experiments are presented in Table 4.

As seen, the bi-directional recursion is consistently faster than the single-direction version, as the average number of cells visited per iteration can be roughly halved. The time savings are in the range 25–50%, depending on the size of the problem.

Increasing the granularity greatly increases the number of shift templates S' and consequently, the running time is also increased. As an example, consider the single- and bi-directional run on pax.r.a. The exact same number of shifts was created in both cases, but with significantly better results using the 30 minute granularity. However, the running time was also increased significantly from 10 to 90 seconds.

We observe from the results that the bi-directional recursion provides a significant improvement in running times. We identify the bi-directional recursion with 60 minute granularity as the most promising candidate for the declared goal of quickly creating solutions of reasonable quality. If the focus instead was on higher solution quality, the 30 minute granularity is clearly preferred.

7.3 Dynamic Programming Results

This section presents the results for the bi-directional dynamic programming heuristic with 60 minute granularity. Results of running the heuristic on all scenarios are presented in Table 5. Table 5 shows that 10 of the 17 instances are solved within 15 seconds and 15 are solved within one minute, with the final two instances solved in 94 and 191 seconds. Table 5 also shows a large differences in

Scenario	Iters	$\lvert T^*\rvert$	$\lvert S'\rvert$	$o/\lVert D\rVert_*$	$u/\lVert D\rVert_*$	z^*	$\lvert\mathcal{S}^*\rvert$	Time (s)
`cargo.m.a`	28	59	7.86	0.03	0.42	1690.32	90	1.17
`ops.e.a`	11	587	38	0.11	0.24	2806.56	208	27.95
`pax.e.a`	34	587	84	0.1	0.13	11011.25	791	178.82
`pax.g.a`	64	560	31	0.23	0.14	51827.83	399	33.15
`pax.g.b`	74	560	31	0.23	0.13	67266.31	469	38.48
`pax.m.a`	37	59	6.68	0.17	0.56	8513.59	88	1.87
`pax.r.a`	52	153	242.46	0.07	0.11	5666.14	481	10.2
`pax.r.b`	63	153	145.92	0.12	0.06	2414.79	521	5.98
`ramp.e.a`	35	504	66.46	0.07	0.48	3442.19	606	14.03
`ramp.e.b`	45	517.8	30.47	0.22	0.26	34674.76	767	85.52
`ramp.e.c`	45	227.33	16.64	0.2	0.22	43681.81	780	26.71
`ramp.g.a`	40	140	31	0.24	0.11	23628.24	232	4.32
`ramp.g.b`	47	140	31	0.23	0.1	31522.56	278	5.22
`ramp.m.a`	17	59	13.88	0.06	0.67	607.73	15	1.63
`ramp.r.a`	23	153	111	0.09	0.1	974.59	276	2.29
`trans.o.a`	110	147	22	0.29	0.2	50372.61	462	10.05
`trans.o.b`	111	224	22	0.26	0.19	57942.08	664	25.01

Table 5: Results of running with the determined parameters on all scenarios. Iters shows the number of iterations. $|T^*|$ and $|S'|$ shows the average value of T^* and the average number of shift templates for the iterations. $o/\|D\|_*$ and $u/\|D\|_*$ shiws the relative staffing levels. z^* shows the total objective function value and $|\mathcal{S}^*|$ shows the number of created shifts.

the size reductions possible for the instances. The largest scenario, pax.e.a has an average of 587 active time units (out of 8064 total) and 84 shift candidates. In comparison, the scenario `ramp.e.c` is of comparable size, but was reduced much further to 227 active time units and 16 shift candidates. This instance was solved in 26 seconds, which illustrates the effectiveness of the size reductions.

8 Conclusions

We have presented an algorithm for the heterogeneous shift design problem that emphasizes an even distribution of shifts throughout the planning period. Furthermore, the algorithm is highly customizable in the amount of working rules satisfied and the trade-off available between running time and solution quality.

Experimental results are presented for a number of real-life problem instances taken from a variety of airport ground handling operations. The experiments show that the algorithm is able to generate good quality solutions for all problem instances quickly, and that the algorithm provides efficient parameters for balancing understaffing and overstaffing.

References

[1] I. Addou and F. Soumis. Bechtold-jacobs generalized model for shift scheduling with extraordinary overlap. *Annals of Operations Research*, 155(1):177–205, 2007.

[2] S. E. Bechtold and L. W. Jacobs. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351, 1990.

[3] M. Beeler, R. W. Gosper, and R. Schroeppel. Hakmem. Memo 239, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., 1972. Item 169.

[4] G. B. Dantzig. A comment on edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.

[5] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, and W. Slany. The minimum shift design problem. *Annals of Operations Research*, 155(1):79–105, 2007.

[6] D. Dowling, M. Krishnamoorthy, H. Mackenzie, and D. Sier. Staff rostering at a large international airport. *Annals of Operations Research*, 72(0):125–147, 1997.

[7] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[8] J. Herbers. *Models and Algorithms for Ground Staff Scheduling On Airports*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, Faculty of Mathematics, Computer Science and Natural Sciences, 2005.

[9] N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine. Airline disruption management-perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.

[10] H. C. Lau. On the complexity of manpower shift scheduling. *Computers & Operations Research*, 23(1):93–102, 1996.

[11] A. J. Mason, D. M. Ryan, and D. M. Panton. Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Operations Research*, 46(2):161–175, 1998.

[12] N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.

[13] M. Rekik, J.-F. Cordeau, and F. Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13(1):49–75, 2009.

[14] J. M. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24(3):275–287, 1982.

We consider the heterogeneous shift design problem for a workforce with multiple skills, where work shifts are created to cover a given demand as well as possible while minimizing cost and satisfying a flexible set of constraints.

We focus mainly on applications within airport ground handling where the demand can be highly irregular and specified on time intervals as short as five minutes. Ground handling operations are subject to a high degree of cooperation and specialization that require workers with different qualifications to be planned together. Different labor regulations or organizational rules can apply to different ground handling operations, so the rules and restrictions can be numerous and vary significantly. This is modeled using flexible volume constraints that limit the creation of certain shifts.

We present a fast heuristic for the heterogeneous shift design problem based on dynamic programming that allows flexibility in modeling the workforce. Parameters allow a planner to determine the level of demand coverage that best fulfills the requirements of the organization. Results are presented from several diverse real-life ground handling instances.

ISBN 978-87-90855-76-5

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk