Minimization of Passenger Takeoff and Landing Risk in Offshore Helicopter Transportation: Models, Approaches and Analysis

Fubin Qian *[1], Vitaly Strusevich[2], Irina Gribkovskaia[1], and Øyvind Halskau[1]

[1]Molde University College, Postboks 2110, N-6402 Molde, Norway
[2]School of Computing and Mathematical Science, University of Greenwich, Park Row, Greenwich, London SE10 9LS U.K.

Nov. 2013

Abstract

Offshore petroleum industry uses helicopters to transport the employees to and from installations. Takeoff and landing represent a substantial part of the flight risks for passengers. In this paper, we propose and analyze approaches to create a safe flight schedule to perform pickup of employees by several independent flights. Two scenarios are considered. Under the non-split scenario, exactly one visit is allowed to each installation. Under the split scenario, the pickup demand of an installation can be split between several flights. Interesting links between our problem and other problems of combinatorial optimization, e.g., parallel machine scheduling and bin-packing are established. We provide worst-case analysis of the performance of some of our algorithms and report the results of computational experiments conducted on randomly generated instances based on the real sets of installations in the oil fields on the Norwegian continental shelf. This paper is the first attempt to handle takeoff and landing risk in a flight schedule that consists of several flights and lays ground for the study on more advanced and practically relevant models.

Keywords: helicopter transportation, takeoff and landing risk, machine scheduling, bin-packing, worst-case analysis, heuristics.

*Corresponding author.
E-mail addresses: qianfubin@hotmail.com (F. Qian), V.Strusevich@greenwich.ac.uk (V. Strusevich), irina.gribkovskaia@himolde.no (I. Gribkovskaia), oyvind.halskau@himolde.no (Ø. Halskau)

1 Introduction

In offshore petroleum industry, employees are transported by helicopters to and from offshore installations in the Norwegian Sea and the North Sea areas. Travel by a helicopter is more comfortable and less harmful in terms of travel sickness and tiredness as compared to travel by ship with a considerably longer duration. However, helicopter transportation is perceived by many offshore employees to be a risky part of their work. They experience heaviness and weightlessness during takeoff and landings, heavy noise, strong vibrations, and even sometimes incidents or accidents. Vinnem et al.[1] claim that the hazards associated with helicopter transportation of personnel are among the main risks experienced by offshore employees.

Helicopter accidents are reported fairly frequently. European offshore helicopter data reveal that there have been 23 fatal and major injury accidents in the offshore oil industry from 1968 to 2000 [2]. A recent summary report from the Helicopter Safety Study 3 (HSS-3), undertaken by SINTEF Trondheim, indicates that among 28 OGP[1] offshore accidents from 2000 to 2005, 22 occurred during the takeoff or landing phase. Since the first version of this paper was written, three accidents have occurred in the North Sea, two in 2012 and one very recently, in August 2013; the latter accident claimed four lives.

Helicopter planning to offshore installations may be done by a helicopter operating company (as for Statoil) or by the oil company itself (as in Petrobras). Safety is always considered together with traditional measures while planning. But in all known publications on helicopter planning to offshore installations the problem is defined as a vehicle routing problem (VRP) with traditional cost or distance objectives. The vehicle routing problem is well-studied, and numerous computational techniques are available for its solution, see for example [3], [4] and [5] for the recent work.

Among the few published research papers on the helicopter routing to offshore installations, Moreno et al.[6] and Menezes et al.[7] seek to minimize the flight costs, the number of flights, and the total number of offshore landings in order to improve helicopter flight safety. Qian et al.[8] introduced a risk measure for passenger transportation by helicopter and defined a safe passenger helicopter transportation problem as a vehicle routing problem with pickups and deliveries with a risk objective in terms of the expected number of fatalities. Two types of accidents, i.e. takeoff and landing accidents and cruise accidents, are considered as possible during a flight. The problem is formulated as an integer linear program, which generates Hamiltonian solutions. The term 'Hamiltonian' refers to the fact that each installation is visited exactly once for the combined pickup and delivery within a flight. Qian et al.[9] extend the previous work and consider a general routing policy, under which each installation is allowed to be visited twice if necessary, once for delivery and once for pickup. A tabu search heuristic was implemented for real-life instances based on data taken from the Norwegian Continental Shelf. A comprehensive study was conducted to gain insights into passenger transportation safety by comparing the solutions obtained from optimizing risk or cost objectives under different helicopter routing policies. In these two papers ([8], [9]), the helicopter routing problem was defined as VRP, with risk objective which is partly distance dependent (in cruise risk part) and partly sequence dependent (in takeoff and landing risk part). One of the interesting findings of these two papers is that takeoff and landing risks are the major part of the transportation risk for passengers.

In this study we, while fully appreciating the importance of the traditional measures of the schedule quality, would like to focus an another important factor, the takeoff and landing risk. The fact that we want to create a schedule that minimizes such a risk does not mean

[1]The International Association of Oil & Gas Producers

that we want to dismiss flight schedules that perform well against traditional cost functions. The minimum risk flight schedules will complement those schedules that can be found by the vehicle routing techniques. Even if the minimum risk flight schedules might not appear to be the most cost-effective, we believe that having these schedules at hand will extend the number of options for the decision-maker to choose from. Focusing on minimizing risk allows us not to rely on the VRP techniques, but develop methods that are related to the problems of machine scheduling and bin packing.

As one of first attempts to handle the minimum risk scheduling for several flights, in our model we make several simplifying assumptions:

- the risk is measured as the total number of people exposed to takeoffs and landings in all flights;
- we focus on minimization of passenger takeoff and landing risk in helicopter transportation, provided that only pickup demands are considered.

In reality, measuring the total risk should additionally include the assessment of the cruise risk factors; however, the latter aspect again can be modeled by the VRP, and in this paper we only address the takeoff and landing risk, as the most substantial risk component which should be handled by a different approach.

Also in practice, a typical flight schedule performs both pickups and deliveries, e.g., changing the staff of a shift on a platform, totally or partly. The problem of minimizing the takeoff and landing risk with simultaneous pickups and deliveries for a single flight has been successfully handled by [9]. In the case of several flights, we are still not aware of a possible approach to tackling simultaneous pickup and deliveries by several flights, even uncapacitated. The model that allows pickup only (or, in the symmetric case, delivery only) is not totally irrelevant and can be useful in the case of removing staff from installations in the case of their conservation or emergency. Delivery only and pickup only take place in the case of transporting teams of visitors (inspectors, journalists, researchers, etc.) to and then from the installations.

We do not expect that our main model, currently being stripped off most difficulties that arise in practice, will lead to an immediate practical implementation. Still, we believe that it captures many features that distinguish this direction of research. We see this paper as a necessary stage which is aimed at (i) initiating research on safe helicopter transportation by several flights; (ii) establishing links between the safe helicopter transportation problems and traditional problems of combinatorial optimization, including bin-packing and scheduling problems on parallel machines; (iii) laying grounds for the study on more advanced, enhanced and practically relevant models.

From the point of view of the theory of Operational Research, we think that the established similarities and differences between our problems and the related problems of combinatorial optimization are especially attractive; in particular one of our models leads to a scheduling problem on parallel machines with the processing conditions that have not been studied before in full form.

In our models of safe helicopter pickup, two scenarios of transportation are considered: (i) the non-split scenario under which all people to be picked up from an offshore installation are collected by a single flight, and (ii) the split scenario under which multiple visits to installations are allowed, each flight collecting a part of people to be picked up. For each scenario, we present several algorithms accompanied by worst-case analysis of their performance and/or computational experiments.

The remainder of this paper is organized as follows. In Section 2, the connections between the safe helicopter transportation problem and scheduling problems on parallel machines are presented and the related work is reviewed. The non-split scenario is studied in Section 3. The split scenario is considered in Section 4, followed by conclusions in Section 5.

2 Problem Formulation and Links with Machine Scheduling and Bin-Packing

In all models considered in this paper, a *flight* is a route of a helicopter that starts at an onshore heliport with no passengers on board, visits the selected offshore installations exactly once in a certain order and ends at the same heliport. In principle, during a visit both pickup and deliveries may take place, however in this paper we focus on the pickup operations only. See Qian et al.[9] for a study of the pickup and delivery operations in a single flight.

Computational results in [9] show that for the considered instances the most essential component of the risk associated with offshore transportation is the takeoff and landing risk. Let

- f_{TL} denote the probability of an accident during a combined takeoff/landing operation;
- z denote the probability of a fatal outcome for an individual involved in a takeoff/landing accident;
- PTL denote the total number of people exposed to takeoffs and landings.

For example in [8] the frequency f_{TL} of takeoff and landing accidents is 0.65 per one million of takeoff and landing pairs, while the probability z is assumed to be equal to 1. Thus, the takeoff and landing risk can be measured as the product $PTL \cdot f_{\mathrm{TL}} \cdot z$. The probabilities f_{TL} and z can be seen as constants that do not depend on the routing decisions. Thus, in order to minimize the takeoff and landing risk, we need to minimize the variable component of the risk function, i.e., the PTL. In turn, the problem of minimizing the PTL is closely associated with certain scheduling and sequencing problems, as explained below.

Assume that a helicopter of capacity Q is available, i.e., no more than Q passengers can be present on board in any stage of a flight. Let $N = \{1, 2, \ldots, n\}$ be a set of installations to be visited, and an installation $j \in N$ is associated with a pickup demand p_j. Throughout this paper for a non-empty subset $N' \subseteq N$ of installations we denote

$$p(N') = \sum_{j \in N'} p_j,$$

i.e., $p(N)$ denotes the total pick-up demand from all installations.

Define $m = \lceil p(N)/Q \rceil \geq 2$, i.e., m is the smallest integer such that the inequality

$$p(N) \leq mQ \tag{1}$$

holds. Notice that (1) implies that at least m flights will be needed in order to pick up all $p(N)$ people.

The main object of our study is the following problem of Safe Helicopter Pickup (SHP):

Problem SHP: Given a set $N = \{1, 2, \ldots, n\}$ of installations, the pickup demands $p_j, j \in N$, and the helicopter capacity Q that satisfy (1), find a capacity-feasible flight schedule S that satisfies total pickup demand and minimizes the *total risk* $R(S)$ measured as PTL,

the total number of people exposed to takeoffs and landings. The actual risk in terms of the expected number of fatalities can be obtained by applying the formula $R(S) \cdot f_{\mathrm{TL}} \cdot z$.

In a flight schedule S, a flight is defined by the following decisions: (i) a subset of the installations to be visited by the helicopter, and (ii) the order in which the helicopter should visit the assigned installations. In what follows, we consider two possible pickup scenarios:

- *Non-split* scenario, under which no installation is visited more that once and the helicopter picks up all p_j passengers from an installation j;
- *Split* scenario, under which an installation can be visited by several flights, and a flight is allowed to pick up a part of the overall pickup demand p_j.

Notice that for the non-split scenario, we additionally assume that $p_j \leq Q$ for all $j \in N$. On the other hand, for the split scenario, an additional third decision has to be made: how many people to pickup from an installation during a given flight. As explained later, for the split scenario a flight schedule with m flights always exists, while for the non-split scenario that need not be the case and further discussion is required to define what should be seen as an acceptable solution to the problem.

The models that arise in helicopter transportation to minimize total risk have interesting links to various scheduling models on a single machine or on identical parallel machines to minimize the sum of the completion times. This allows us to transfer various algorithmic ideas known in machine scheduling to the safe transportation application under consideration.

To illustrate the mentioned link, let us start with a non-split scenario and a single flight that has to perform pickup from $h \leq n$ installations, assuming that the total pickup demand from the chosen installations does not exceed the helicopter's capacity Q. Suppose that these installations are visited in accordance with a permutation $(\pi(1), \pi(2), \ldots, \pi(h))$, where $\pi(j)$ refers to the installation that is the j-th in the route. The flight starts from a heliport 0, then visits installation $\pi(1)$, followed by installation $\pi(2)$, and so on, the last visited installation is $\pi(h)$, after which the helicopter returns to the heliport. The helicopter starts with no passengers, lands at installation $\pi(1)$, picks $p_{\pi(1)}$ passengers, flies to installation $\pi(2)$, lands there with $p_{\pi(1)}$ passengers on board, picks additional $p_{\pi(2)}$ passengers, and so on. It can be observed that the flight takes off from installation $\pi(k)$, $1 \leq k \leq h-1$, and lands at the next installation $\pi(k+1)$ with $\sum_{j=1}^{k} p_{\pi(j)}$ passengers on board. Besides the flight takes off from the last installation $\pi(h)$ and lands at the heliport with $\sum_{j=1}^{h} p_{\pi(j)}$ passengers on board. Thus, the total risk associated with the flight and measured as the total number of passengers exposed to takeoffs and landings is equal to $\sum_{k=1}^{h} \sum_{j=1}^{k} p_{\pi(j)}$, which also can be written as

$$R_1(\pi) = \sum_{j=1}^{h} p_{\pi(j)} (h - j + 1) = \sum_{j=1}^{h} j p_{\pi(h-j+1)}. \tag{2}$$

As proved by [10], a permutation that minimizes (2) can be found by matching small values of p_j to large multipliers, or, alternatively, by matching large values of p_j to small multipliers. This implies that a permutation that satisfies

$$p_{\pi(1)} \leq p_{\pi(2)} \leq \cdots \leq p_{\pi(h)} \tag{3}$$

minimizes $R(S)$. Interpreting this well-known result in terms of helicopter transportation, Qian et al.[9] demonstrate that an optimal sequence of installations to be visited by a single helicopter in order to minimize the total risk can be found by visiting the installations in non-decreasing order of the pickup demands.

Notice that the problem of finding a permutation π that delivers the minimum to the function $\sum_{k=1}^{h}\sum_{j=1}^{k} p_{\pi(j)}$ can be seen as a scheduling problem of processing h jobs on a single machine to minimize the total completion time, provided that p_j is the processing time of job j. This scheduling problem in accordance with the three-field classification scheme developed by Graham[11] is denoted by $1\,||\sum C_j$, where the first field is responsible for the machine environment ("1" means that we have a single machine), the middle field is reserved for specific processing conditions (not needed in this case) and the third field presents the objective function to be minimized. Indeed, the completion time $C_{\pi(k)}$ of the job in position $\pi(k)$ is equal to the total processing time of all previously scheduled jobs, i.e., to $\sum_{j=1}^{k} p_{\pi(j)}$. Notice that in scheduling terms, the total pickup demand $\sum_{j=1}^{h} p_{\pi(j)}$ for the flight defines the maximum completion time, typically called the *makespan* and denoted by $C_{\max}$. As proved by Smith [12], the permutation of jobs that solves problem $1\,||\sum C_j$ can be found by the so-called the *Shortest Processing Time* rule (*SPT rule*), i.e., by sorting the jobs in non-decreasing order of their processing times, as given by (3).

Consider now the general Problem SHP in which the pickup operations should be performed by m independent flights. We will denote the flights by M_1, M_2, $\cdots$, M_m. Our purpose is to find a way of performing the pickup by m flights so that the total risk is minimized. We call the largest number of passengers on board a flight, i.e., the number of passengers brought to the heliport, the *load* of that flight. Temporarily, we ignore the capacity of the helicopter and assume that in principle each flight can have an arbitrary load.

As stated in the introduction, we consider neither the actual time-tabling of the flights, nor the distance to travel, which are typical issues in vehicle routing, see for example [3], [4] and [5].

Let N_i denote the set of installations visited by flight M_i, $1 \leq i \leq m$. For an individual flight that visits $h_i = |N_i|$ selected installations given by the sequence $(0, \pi_i(1), \pi_i(2), \ldots, \pi_i(h_i), 0)$, the total risk is still determined by a formula similar to (2). In particular, the last installation visited by the flight will contribute its pickup demand $p_{\pi_i(h_i)}$, the previous installation will contribute $2p_{\pi_i(h_i-1)}$, and so on: the installation in the position $h_i - k + 1$, i.e., in the $k-$th position from behind, will contribute $kp_{\pi_i(h_i-k+1)}$. The overall risk R of performing m flights can be written as

$$R = \sum_{i=1}^{m}\sum_{j=1}^{h_i} p_{\pi_i(j)}(h_i - j + 1) = \sum_{i=1}^{m}\sum_{j=1}^{h_i} jp_{\pi_i(h-j+1)}. \tag{4}$$

Exactly the same expression defines the sum of the completion times of processing jobs of set N on m parallel identical machines $M_1, M_2, \ldots, M_m$, provided that machine M_i processes the jobs of set N_i in the sequence $\pi_i(1), \pi_i(2), \ldots, \pi_i(h_i)$, where $h_i = |N_i|$. This implies Problem SHP with m flights (ignoring the capacity of the helicopter) reduces to the well-known scheduling problem of minimizing sum of the completion times on m parallel machines, traditionally denoted by $Pm\,||\sum C_j$, where we write "Pm" in the first field to refer to m parallel identical machines; see [11]. The latter scheduling problem is known to be solvable in polynomial time. The solution algorithm belongs to the family of list scheduling algorithms introduced by Graham [13]. A general List Scheduling (LS) algorithm applied to parallel identical machines can be described as follows.

Algorithm LS

Step 1. Form an arbitrary list L of the jobs.

Step 2. While list L is not empty do:

(a) At any time that a machine becomes available, take the first job in the list and assign it to the machine. Remove the assigned job from the list.

Step 3. Stop.

As proved by Conway et al. [14], Algorithm LS solves problem $Pm \,||\, \sum C_j$, provided that the list L in Step 1 is formed by the SPT rule, i.e., the jobs are renumbered so that

$$p_1 \leq p_2 \leq \cdots \leq p_n. \tag{5}$$

In terms of safe helicopter transportation, Algorithm SPT admits a natural interpretation: scan the installations in the SPT order given by (5) and assign the next installation to a flight that currently has the smallest load. Below we present a version of the algorithm that requires $O(n \log n)$ time.

Algorithm SPT

Step 1. Renumber the installations in accordance with (5). Let $n = qm + r$, where $q \geq 1$ and $0 \leq r \leq m - 1$. Create m empty flights M_1, M_2,..., M_m.

Step 2. For k from 1 to q do

(a) For i from 1 to m do

(a1) Assign installation $(k-1)\,m + i$ to the $k-$th position in flight M_i.

Step 3. If $r > 0$ then for i from 1 to r do

(a) Assign installation $qm + i$ to the $(q+1)\,-$th position in flight M_i.

Step 4. Call the resulting flight schedule $S_{SPT}(m)$ and stop.

Notice that only finding the SPT sequence of the installations requires $O(n \log n)$ time, while the rest of Algorithm SPT takes only $O(n)$ time. The algorithm determines the flight schedule $S_{SPT}(m)$ which associates each flight M_i with a set N_i of installations and specifies in which order those installations have to be visited, $1 \leq i \leq m$.

Example 1. Consider Problem SHP of finding the minimum risk schedule of picking up people from 8 installations by 3 flights M_1, M_2 and M_3. The pickup demand values are given in Table 1. Notice that $n = 8$, $m = 3$, $q = 2$ and $r = 2$.

j	1	2	3	4	5	6	7	8
p_j	8	10	7	6	5	4	4	3

Table 1: Numerical example for Algorithm SPT for $m = 3$

The SPT sequence of installations is $(8, 6, 7, 5, 4, 3, 1, 2)$. In Step 2 for $k = 1$ Algorithm SPT will create the flights $M_1 = (8)$, $M_2 = (6)$ and $M_3 = (7)$. After running Step 2 for $k = 2$, we obtain the flights $M_1 = (8, 5)$, $M_2 = (6, 4)$ and $M_3 = (7, 3)$. Finally, Step 3 forms the flights in their final form: $M_1 = (8, 5, 1)$, $M_2 = (6, 4, 2)$ and $M_3 = (7, 3)$. Thus, the loads of the flights M_1, M_2 and M_3 are $3 + 5 + 8 = 16$, $4 + 6 + 10 = 20$ and $4 + 7 = 11$, respectively. The total risk is equal to $(3 \times 3 + 2 \times 5 + 8) + (3 \times 4 + 2 \times 6 + 10) + (2 \times 4 + 7) = 76$.

Given an instance of Problem SHP, Algorithm SPT outputs a flight schedule $S_{SPT}(m)$. However, Algorithm SPT ignores the capacity Q of a helicopter. If the flight schedule $S_{SPT}(m)$ is capacity-feasible, i.e., if $\max\{p(N_i)|1 \leq i \leq m\} \leq Q$, then this schedule is optimal and the original Problem SHP is solved.

From now on, we consider the capacitated version of Problem SHP, i.e., the load of each flight cannot exceed Q. Assume that schedule $S_{SPT}(m)$ is not capacity-feasible.

A scheduling problem associated with Problem SHP with m flights is problem $Pm|C_j \leq Q|\sum C_j$, where in the middle field we write "$C_j \leq Q$" to stress that all jobs have a common deadline Q. In terms of helicopter transportation C_j is understood as the number of people on board of the flight at the moment of the takeoff at installation j. A related interpretation of the helicopter capacity in scheduling terms is that we only consider flight schedules S for which $C_{\max}(S) = \max\{C_j|j \in N\} \leq Q$, i.e., the makespan of such a schedule is bounded by Q. The later scheduling problem is unlikely to be solvable in polynomial time, since the problem of checking the existence of a schedule for identical parallel machines with a bounded makespan is NP-complete; in fact, for $m = 2$ such a problem is equivalent to the famous NP-complete *Partition* problem.

The total risk and the helicopter capacity are related to the sum of the completion times and the makespan of the corresponding schedule, respectively. We may put Problem SHP into correspondence to a non-preemptive scheduling problem on parallel machines with two criteria, $\sum C_j$ and $C_{\max}$. Stein and Wein [15] and Aslam et al. [16] prove that for a general scheduling problem there exists an algorithm that delivers a schedule with the values of these objectives a constant factor away from the optimal values. The problems on parallel machines to minimize the makespan subject to the minimum total flow time have received a certain attention. Gupta and Ho [17] consider the two-machine problem and Gupta and Ruiz-Torres [18] address the problem with m machines. Eck and Pinedo [19] propose heuristics for both two-machine and m-machine problems. These results, although relevant to Problem SHP under the non-split scenario, are not directly applicable, since in our case we have got a hard upper bound Q on the value of the makespan.

Thus, given an instance of Problem SHP to decide whether there exists a capacity-feasible flight schedule with m flights under the non-split scenario is NP-complete. On the other hand, if we increase the number of flights, we may easily find a feasible flight schedule. In fact, a flight schedule with n flights is always available, in which each installation j is served by the flight $(0, j, 0)$ that visits no other installations. However, such a schedule, although the safest possible, may be impractical due to a high cost and/or long time to implement.

The problem of finding the smallest number of capacity-feasible flights that can satisfy the total pickup demand is essentially a *bin-packing* problem. In our notation, the latter problem can be formulated as follows: given a set N of items with item j being of size p_j, pack all items in the smallest number of bins of size Q each. The bin-packing problem is NP-hard and is one of the most studied problems in combinatorial optimization, especially from the approximability point of view. We refer to the classical survey by Coffman et al. [20].

Strictly speaking, Problem SHP under the non-split scenario should be seen as a bicriteria problem in which the number of flights and the total risk should be minimized simultaneously and a solution is delivered in the form of Pareto-optimal flight schedules. However, this problem is intractable, having the bin-packing as a part of it. On the other hand, in practice, there is no need to know the full efficiency frontier of the bicriteria problem. For the decision-maker it will be sufficient to be provided with several feasible flight schedules to choose from. We pursue this approach in Section 3, where we present and analyze several algorithms for finding feasible schedules with various numbers of flights. Our algorithms explore the links of

Problem SHP with both parallel machine scheduling and bin-packing.

Now we pass to discussing Problem SHP under the split scenario. Recall that under that scenario not all people from an installation are picked up by a single flight; instead, an installation can be visited by several flights, each taking a part of the initial pickup demand. The first analogy, that one may think of, is to associate Problem SHP under the split scenario with *preemptive* parallel machine scheduling. Recall that in parallel machine scheduling allowing preemption means that the processing of a job on any machine can be interrupted at any time and resumed later on, possibly on another machine. In the final schedule, all jobs must be completed, and a job is not allowed to be processed on more than one machine at a time. Leung and Pinedo [21] give polynomial algorithms for problem $Pm|pmtn, C_j \leq \bar{d}_j| \sum C_j$; here in the middle field we (i) write "*pmtn*" to stress that preemption is allowed, and (ii) indicate that job j has a deadline $\bar{d}_j$ by which it must be completed. Extensions of this result to the machines that may have individual speeds are given by McCormick and Pinedo [22] and Gonzalez et al. [23].

Since the helicopter capacity Q can be understood as a deadline, common to all jobs, there is a certain analogy between Problem SHP under the split scenario and problem $Pm|pmtn, C_j \leq \bar{d}_j = Q| \sum C_j$. Unfortunately, this analogy is not fully applicable. Consider a flight schedule S_{flight} for Problem SHP in which the pickup demand from an installation j is served by two flights: flight M' that takes p'_j people and and flight M'' that takes p''_j people, where $p_j = p'_j + p''_j$. Compare this with a schedule S_{pmtn} on parallel machines in which job j is processed with preemption for p'_j time units on machine M' and for p''_j time units on machine M''. There are two points of difference between these two schedules:

(i) in schedule S_{pmtn} the two pieces of job j cannot be processed simultaneously, but there is no such restriction in schedule S_{flight}; schedule S_{flight} allows splitting and allocating the installation demand to different flights, but it does not mean that the multiple visits to the installation by these flights will happen at the same point in time.

(ii) in schedule S_{pmtn} only that piece of job j that completes last defines the completion time of that job and therefore contributes to the objective function, but in schedule S_{flight} p'_j people contribute toward the risk of flight M' and p''_j people contribute toward the risk of flight M''.

In the scheduling literature, there is another type of preemption, known as *split*, which allows different pieces of the same job to be processed on several machines at a time. Serafini [24] has studied a scheduling problem that arises in the production of different types of fabric in a textile industry. Each job can be split arbitrarily and processed independently on machines and the objective is to minimize the maximum tardiness and the maximum weighted tardiness, provided that the jobs are assigned different weights. Xing and Zhang [25] give a systematic study of split scheduling on parallel machines. In particular, they show that for many objective functions, including $\sum C_j$, there exists an optimal split schedule in which all machines complete simultaneously, and such a schedule can be found in polynomial time. Split scheduling on parallel machines still does not address the point of difference (ii) mentioned above, and this approach cannot be used directly for solving Problem SHP.

Schedule $S_{SPT}(m)$ found by Algorithm SPT provides the smallest risk if m flights are used, but it is not necessarily feasible, so that the value $R(S_{SPT}(m))$ can serve as a lower bound on the total risk for a flight schedule under the split scenario. We present some algorithms and their worst-case and computational analysis in Section 4.

We conclude this section with a summary of the links between parallel machine scheduling and Problem SHP; see Table 2.

Table 2: The links between Problem SHP and parallel machine scheduling

Problem SHP	**Parallel machine scheduling**
flights	identical parallel machines
installations	jobs
pickup demands	processing times
total risk	sum of completion times
flight load	makespan
helicopter capacity	common deadline
	or upper bound on makespan
non-split scenario	nonpreemptive scheduling
split scenario	preemptive or split scheduling
	(close but not exact analogy)

3 Non-Split Scenario

In this section, we consider Problem SHP of minimizing the total risk under the non-split scenario. No installation is visited twice, i.e., a helicopter that visits an installation j must pick up all p_j passengers, $j = 1, 2, \ldots, n$. The capacity of the helicopter is denoted by Q, and in order to be able to apply the non-split scenario we assume that $p_j \leq Q$ for each installation j. Each flight is defined by (i) a subset of the installations to be visited by each flight, and (ii) the order in which the assigned installations are visited by a flight. The total pickup demand and the helicopter capacity satisfy the inequality (1), in which m is the smallest possible integer, $m \geq 2$. We describe and analyze several techniques that create feasible flight schedules. The decision-maker chooses a schedule from the provided list of several solutions characterized by the number of flights and the total risk value.

Consider a flight schedule $S_{SPT}(m)$ in which all installations are assigned to m routes by Algorithm SPT described in Section 2. In what follows, we assume that this schedule is not capacity-feasible.

As mentioned in Section 2, to check whether there exists a feasible non-split flight schedule with m flights is NP-hard. We start with establishing an upper bound on the number of flights that would guarantee the existence of a capacity-feasible non-split flight schedule.

Lemma 1 *For Problem SHP that satisfies (1) there exists a capacity-feasible non-split flight schedule with at most* $2m - 1$ *flights, and this bound is tight.*

Proof: Suppose that m' is the smallest number of flights for which a required feasible flight schedule $S(m')$ exists, and $m' \geq 2m$. In schedule $S(m')$, the installations are split into groups N_i, $1 \leq i \leq m'$, each associated with the corresponding flight M_i. There exists a group N' for which $p(N') \leq \frac{1}{2}Q$; otherwise,

$$p(N) = \sum_{i=1}^{m'} p(N_i) > (2m)\,\frac{1}{2}Q = mQ.$$

If there exists another group N'' with $p(N'') \leq \frac{1}{2}Q$, then $p(N') + p(N'') \leq Q$, and groups N' and N'' can be merged into one group, reducing the number of flights, which leads to a contradiction. Below we show that such a group always exists, so that $m' \geq 2m$ cannot be the smallest number of flights.

Let group $\overline{N}$ be an arbitrary group other than N'. If $p(N') + p(\overline{N}) \leq Q$, then we can take group $\overline{N}$ as group N''. Otherwise, the total pickup demands of the remaining $2m-2$ groups is less than $p(N) - \left(p(N') + p(\overline{N})\right) \leq (m-1)\,Q$, and at least one of them must have the total pickup demand of at most $\frac{1}{2}Q$, and that group can be taken as group N''.

To see that in the worst case the value of $2m-1$ cannot be reduced, take an instance of the problem with $2m-1$ installations, each of pickup demand m, so that $p(N) = m\,(2m-1)$ and $Q = 2m-1$. There exists a feasible flight schedule $S(2m-1)$ with $2m-1$ flights, each visiting one installation and having a load of at most $m < Q$ passengers. In any flight schedule with less than $2m-1$ flights at least two installations will be assigned to the same flight, but then a load of such a flight is at least $2m > Q$. This proves the lemma. ∎

We know that a flight schedule found by Algorithm SPT provides the smallest total risk for a given number of flights. Among the schedules that will be offered to the decision-maker will be such that uses m' flights, $m < m'$, and assigns installations to flights according to the SPT rule, as described below.

Algorithm SPT-NonSplit

Step 1. Define $m' := m$.

Step 2 Repeat

- **(a)** Define $m' := m' + 1$.
- **(b)** Run Algorithm SPT for m' flights and find schedule $S_{SPT}(m')$.

until schedule $S_{SPT}(m')$ is capacity feasible.

Step 3. Output schedule $S_{SPT}(m')$ and stop.

Since the SPT sequence of jobs need to be created only once, the running time of Algorithm SPT-NonSplit is at most $O\,(mn)$. Since schedule $S_{SPT}(m')$ found by the algorithm is an SPT flight schedule and m' is the smallest number of flights for which such a schedule exists, it follows that under the non-split scenario the total risk reaches its minimum for schedule $S_{SPT}(m')$.

For Example 1 with $Q = 19$, we see that schedule $S_{SPT}(3)$ is infeasible, since flight M_2 has a load of 20. Algorithm SPT-NonSplit will output schedule $S_{SPT}(4)$ in which flight M_i visits installations i and $i+4$ in this order, $1 \leq i \leq 4$. The total risk of schedule $S_{SPT}(4)$ is equal to 63.

Still, the number of flights m' in schedule $S_{SPT}(m')$ can be too large for a practical implementation. In fact, if there exists an installation with the pickup demand equal to Q, then there is no feasible SPT schedule with less than n flights, i.e., an SPT schedule will consists of $m' = n$ flights, exactly one to each installation. The same is true if the sum of the smallest and the largest pickup demands exceeds Q. For schedule $S_{SPT}(m')$ to exist, the number of flights m' in terms of m and Q that satisfy (1) can be as large as $Q\,(m-1)+1$. The latter value is achieved for an instance with $n = Q\,(m-1)+1$ installations, such that $p_1 = p_2 = \cdots = p_{n-1} = 1$ and $p_n = Q$.

Below we provide several approaches that might find feasible non-split flight schedules with a smaller number of flights.

For a flight schedule $S_{SPT}(m)$, at least one flight is capacity-feasible due to (1). Denote the number of infeasible flights by u, where $1 \leq u \leq m-1$. Algorithm SPT assigns any

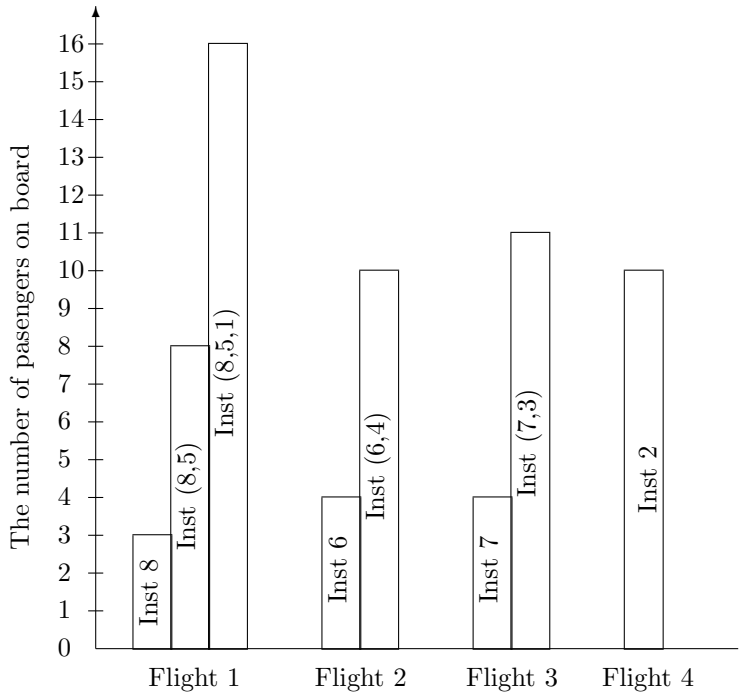

Figure 1: Flight schedule created by Algorithm SPTu

new installation to the flight that currently has the smallest load, and is therefore feasible; otherwise all partial flights are infeasible and (1) does not hold. This implies that the u flights become infeasible because each of them visits one of the u installations $n-u+1,\ldots,n$, i.e., the last u installations in the SPT sequence (5). The corresponding pickup demands $p_{n-u+1},\ldots,p_n$ are the u largest in the whole set N. We can transform schedule $S_{SPT}(m)$ into a capacity-feasible schedule $S(m+u)$ with $m+u$ flights by assigning each of the installations $n-u+1,\ldots,n$ to a new individual flight. Formally, the algorithm for finding such a schedule is described below.

Algorithm SPTu

Step 1. Run Algorithm SPT and find schedule $S_{SPT}(m)$. Identify the capacity-infeasible flights and compute their number u.

Step 2. For i from 1 to u do

(a) Move installation $n-u+i$ to a new flight M_{m+i}.

Step 3. Call the resulting schedule $S(m+u)$ and stop.

The running time of Algorithm SPTu is $O(n\log n)$.

For Example 1 with $Q=19$, Algorithm SPTu will transform schedule $S_{SPT}(3)$ into schedule $S(4)$ by moving installation 2 to a new flight M_4. The total risk of schedule $S(4)$ is equal to 66; see Figure 1. The figure shows how the number of people on board changes dynamically for each flight as the installations are visited in the specified order.

Below we compare the values of total risk in schedules $S_{SPT}(m)$ and $S(m+u)$.

Theorem 1 *For Problem SHP under the non-split scenario, let $S_{SPT}(m)$ and $S(m+u)$ be the flight schedules found by Algorithm SPT and Algorithm SPTu, respectively. Then*

$$\frac{R(S_{SPT}(m))}{R\left(S(m+u)\right)} \leq 1 + \frac{u}{m}. \tag{6}$$

Proof: If required, renumber the flights in schedule $S_{SPT}(m)$ in such a way that the first u flights are infeasible, with installation $n-u+i$ being the last for flight M_i, $1 \leq i \leq u$. Let S' be a (partial) flight schedule obtained from schedule $S_{SPT}(m)$ by removing the installations $n-u+1, \ldots, n$. Let $C^{(i)}$ denote the total load of flight M_i in schedule S'. It follows that

$$C^{(1)} \leq C^{(2)} \leq \cdots \leq C^{(m)} \leq Q. \tag{7}$$

Additionally let $F^{(i)}$ denote the total risk of flight M_i in schedule S', so that $R(S') = \sum_{i=1}^{m} F^{(i)}$. The value $R(S')$ contributes into each $R(S_{SPT}(m))$ and $R(S\left(m+u\right))$, so that

$$\begin{aligned} R(S_{SPT}(m)) &= \sum_{i=1}^{m} F^{(i)} + \sum_{i=1}^{u} \left(C^{(i)} + p_{n-u+i}\right); \\ R(S\left(m+u\right)) &= \sum_{i=1}^{m} F^{(i)} + \sum_{i=1}^{u} p_{n-u+i}. \end{aligned} \tag{8}$$

It follows that

$$\frac{R(S_{SPT}(m))}{R(S\left(m+u\right))} = \frac{\sum_{i=1}^{m} F^{(i)} + \sum_{i=1}^{u} \left(C^{(i)} + p_{n-u+i}\right)}{\sum_{i=1}^{m} F^{(i)} + \sum_{i=1}^{u} p_{n-u+i}} = 1 + \frac{\sum_{i=1}^{u} C^{(i)}}{R(S\left(m+u\right))}. \tag{9}$$

Recall that in schedule $S_{SPT}(m)$ each flight $M_1, \ldots, M_u$ is infeasible, i.e.,

$$\sum_{i=1}^{u} \left(C^{(i)} + p_{n-u+i}\right) > uQ \geq \frac{u}{m} p(N)$$

where the last inequality is due to (1). Since

$$\sum_{i=1}^{m} C^{(i)} + \sum_{i=1}^{u} p_{n-u+i} = p(N),$$

we deduce that

$$\sum_{i=u+1}^{m} C^{(i)} \leq \frac{m-u}{m} p(N).$$

Due to (7), it is clear that $C^{(u+1)}$ is no larger than the average of $C^{(u+1)}, \ldots, C^{(m)}$, i.e.,

$$C^{(u+1)} \leq \frac{1}{m-u} \sum_{i=u+1}^{m} C^{(i)} \leq \frac{p(N)}{m},$$

and we use $C^{(1)} \leq C^{(2)} \leq \cdots \leq C^{(u)} \leq C^{(u+1)}$ to obtain

$$\sum_{i=1}^{u} C^{(i)} \leq \frac{u}{m} p(N).$$

Notice that the value of $p(N)$ can be understood as the total risk of a flight schedule in which each installation is served by an individual flight. Since this is the least-risk option, for any feasible flight schedule S the lower bound

$$R(S) \geq p(N) \tag{10}$$

holds. Coming back to (9), we derive

$$\frac{R(S_{SPT}(m))}{R(S\,(m+u))} = 1 + \frac{\sum_{i=1}^{u} C^{(i)}}{R(S\,(m+u))} \leq 1 + \frac{\frac{u}{m}p(N)}{p(N)} = 1 + \frac{u}{m},$$

which proves the required inequality (6). ∎

Theorem 1 shows that a feasible schedule with $m+u$ flights exists. It also gives the range of values of the total risk for such a schedule: due to (6), we have that

$$R(S\,(m+u)) \in \left[\frac{m}{m+u}R(S_{SPT}(m)), R(S_{SPT}(m))\right].$$

In the worst case, $u = m-1$, and we need $2m-1$ routes, which complies well with Lemma 1, and the bound (6) becomes

$$\frac{R(S_{SPT}(m))}{R(S\,(m+u))} \leq 2 - \frac{1}{m}.$$

It is possible to find a feasible schedule with less than u new flights, but with a larger total risk than $R(S\,(m+u))$. This can be done by employing bin-packing algorithms. The decision-maker will be provided with a number of alternatives to choose from.

As pointed out in Section 2, the problem of finding the smallest number of capacity-feasible flights is essentially a *bin-packing* problem. We associate an installation j with an item, pickup demand p_j with a size of item j, and the helicopter capacity Q with a bin capacity. Since in our setting all items are available in advance and we are allowed to consider them in any order, we are dealing with the *off-line* model of bin-packing. Let B^* denote the smallest number of bins into which the items can be packed. Then, in terms of Problem SHP this implies that there exists a capacity-feasible schedule with B^* flights.

Bin-packing is an NP-hard problem, and the main direction of research of this problem since the 1970s has been the design and analysis of approximation algorithms; see the survey by Coffman et al. [20]. Let H be a bin-packing heuristic which packs the items into B_H bins. Suppose the items are organized in a list L. The ratio ρ_H defined by

$$\frac{B_H}{B^*} \leq \rho_H$$

is called the *absolute worst-case bound* of an Algorithm H if the above inequality holds for all lists L. Among most popular approximation algorithms for the off-line bin-packing are *First Fit Decreasing (FFD)* and *Best Fit Decreasing (BFD)* heuristics. Both heuristics scan the items in the order opposite to SPT, i.e., the larger an item is the earlier the packing decision is taken; hence "decreasing" in their names. Both heuristics start with one empty bin and try to place the next item into one of the existing bins; if that is impossible, a new bin is opened and the item is placed there. The FFD method considers the bins in the order they have been opened and places the item into the first found bin which can accommodate that item; hence "First Fit". The BFD method among all bins into which the item fits, selects the one that has the smallest room. Both algorithms require $O(n \log n)$ time.

Simchi-Levi [26] shows that for any polynomial-time bin-packing algorithm H the absolute worst-case ratio ρ_H cannot be smaller than 3/2 unless P=NP, and proves that the FFD and BFD algorithms are the best possible from an accuracy point view, since $\rho_{\mathrm{FFD}} = \rho_{\mathrm{BFD}} = 3/2$. An algorithm by Zhang et al. [27] also delivers the absolute worst-case ratio of 3/2 but requires $O(n)$ time.

Notice that our Lemma 1 and Theorem 1 can be interpreted in terms of bin-packing for the instances, in which the total size of items $p(N)$ is bounded by m times the size Q of a bin, and no packing into less than m bins is possible. It should be noted that in our analysis we compare the found number of bins with m, not with the optimal number of bins as is normally done in bin-packing analysis.

Another bin-packing method that we use is the *Subset Sum (SS)* heuristic due to Caprara and Pferschy [28]. The SS heuristic packs one bin at a time, by finding a set of unpacked items that fill as close as possible the bin capacity. Finding such a set is the classical subset sum problem, which can be solved by a dynamic programming algorithm. In our case, the running time of the SS heuristic is $O(n^2 \max p_j)$. This time, although pseudopolynomial, is still computationally acceptable due to the range of pickup demand values we deal with in Problem SHP. In the worst case the SS heuristic may perform worse than, e.g., the FFD method, but it has a feature that the bins are filled close to their capacity.

We now describe a two-step procedure of finding a solution to Problem SHP based on finding a suitable number of flights in the first step by bin-packing algorithm.

Algorithm BP

Step 1. Run Algorithm BPH, where BPH is one of the algorithms FFD, BFD, and SS. Let B_H be the found number of flights (bins), and let N_i be the set of installations assigned to flight M_i, $1 \leq i \leq B_H$.

Step 2. Generate a flight schedule S_H by sorting the installations of each set N_i in the SPT order. Stop.

Let Algorithm BP with the bin-packing rules FFD and BFD be applied to Example 1 with $Q = 19$. Then both algorithms will output the same schedule S_H with 3 flights, in which $M_1 = (1, 2)$, $M_2 = (5, 4, 3)$ and $M_3 = (8, 6, 7)$ with the total risk equal to 81. If the SS rule is used then schedule S_H can be defined by the flights $M_1 = (6, 5, 2)$, $M_2 = (7, 3, 1)$ and $M_3 = (8, 4)$ with the total risk equal to 78.

We have also designed Algorithm TS, which is a tabu search method that finds a safe flight schedule with a suitable number of flights. Unlike the pure bin-packing heuristics FFD, BFD and SS, Algorithm TS not only searches for the smallest number of flights, but also tries to find the assignment of installations to flights which may be good from the point of view of the total risk.

The tabu search method has been originated by Glover [29] and has been successfully applied to a wide range of combinatorial optimization problems. We refrain from delivering a detailed formal description of Algorithm TS. Instead, below we outline how the tabu search can be adapted for our purposes. We assume that the reader is familiar with the general scheme of tabu search and with the concepts of the tabu list, the aspiration criterion, etc.

The terminology of Problem SHP is used. We start with a description of a procedure that generates an initial solution. Create m empty flights M_1, $M_2, \ldots, M_m$. Let j be a randomly chosen installation. Scan the installations in accordance with the sequence $(j, j+1, ..., n, 1, ..., j-1)$. Checking the existing flights in the order of their numbering starting from M_1, assign the next installation to the first flight which after this assignment remains

capacity-feasible. If no such flight is found, assign the current installation to flight M_m anyway. In the resulting flight schedule, flight M_m may be infeasible, while all other flights are feasible.

Starting from an initial solution, the algorithm works with a penalized objective function $f(S) = R(S) + \alpha q(S)$, where $R(S)$ is the total risk for a flight schedule S and $q(S)$ is the total capacity violation across all flights in S. In each iteration a positive parameter α is updated by either being divided by $1 + \delta$ (if the current solution is feasible) or being multiplied by $1 + \delta$ (if the current solution is infeasible). This mechanism of updating parameter α guides the search to feasible solutions; see [30].

Each solution S is associated with an attribute set

$$B(S) = \{(j, M_i) | \text{installation } j \text{ is visited by flight } M_i\}.$$

The neighborhood $N(S)$ of solution S is defined by all solutions that can be reached from S by removing installation j from route M_i and inserting it into another route $M_{i'}$. A transition from a solution S to a new solution $S' \in N(S)$ is called a *move*, which can also be interpreted by the removal of the attribute (j, M_i) from the set $B(S)$ and the inclusion of the attribute $(j, M_{i'})$ into the set $B(S')$.

Continuous diversification is achieved by penalizing the attributes most frequently added to the solution when evaluating a move. Let $H(S) \subseteq N(S)$ be an admissible subset of neighbor solutions which can be reached from S by a non-tabu move, or which are feasible and meet the aspiration criterion. Let $\rho_{ji'}$ be the frequency at which the attribute $(j, M_{i'})$ has been present in the solution. Any solution $S' \in H(S)$ such that $f(S') \geq f(S)$ is penalized by a term $\rho(S') = \gamma R(S)\sqrt{nm}\rho_{ji'}/\lambda$, where γ is used to control the intensity of diversification and λ is the iteration counter. The selection of the best solution $S' \in H(S)$ is based on the penalized objective $f(S') + \rho(S')$. The algorithm terminates when a fixed number η of iterations is reached, where η is a user-controlled parameter.

The algorithm starts with m defined by (1), and if the tabu search procedure does not yield a feasible solution after η iterations, the number is increased by 1.

The tabu search algorithm works with four user-controlled parameters, i.e., δ, γ, θ, η. The two parameters δ and γ are selected from the uniform distribution over $(0, 1)$. The parameter θ is an integer selected from the uniform distribution over $\{1, 2, \cdots, \lceil\sqrt{mn}\rceil\}$. In our experiments, Algorithm TS was executed five times with $\eta = 10^5$ for each instance, and the best results out of the five runs were recorded.

We have conducted computational experiments with the described algorithms that find feasible non-split flight schedules. Our test-bed contains four sets of instances with 9, 14, 34, or 37 installations; these are actual numbers of installations in the oil fields on Norwegian continental shelf. The small A-instances and B-instances have 9 and 14 installations, respectively; see [8]. The large Flesland and Sola instances have 34 and 37 installations, respectively; see [9]. Each set contains 10 instances with pickup demands uniformly distributed in [1, 18], i.e., 40 instances in total. The helicopter capacity Q is set equal to 19, which corresponds to a real aircraft.

Tables 3 and 4 summarize our computational results for A, B instances and Flesland, Sola instances, respectively. For each instance, we include the m value, and for each found flight schedule S we present its risk value $R(S)$ and the number of flights (in brackets). Also, for each set of instances we report their respective average values (line Avg.).

The SPT-NonSplit and SPTu solutions have lower takeoff and landing risk values, which can be explained by the fact that the corresponding solution is either an SPT schedule or obtained from an SPT schedule. The price for a lower risk is a fairly large number of flights,

especially for schedules found by Algorithm SPT-NonSplit. For example, even for smaller A-instances with 9 installations only, Algorithm SPT-NonSplit outputs schedules in which each installation is visited by an individual flight (instances A2, A4-A6, and A8-A10). The same phenomenon is observed for the B-instances (instances B1, B4, B5, B8 and B9) and for the Flesland instances (instances F2, F4 and F5). Recall that a feasible SPT schedule will have as many flights as installations if the sum of the smallest and the largest pickup demand exceed Q. For the generated Sola instances with 39 installations the latter condition does not hold, and always less than 39 flights are used. Observe that the number of flights in schedules found by Algorithm SPT-NonSplit could be as large as $2m$ (see instances F5 and S7). Algorithm SPTu typically delivers flight schedules with moderate numbers of flights, only for one instance A2 the number of flights is equal to the number of installations. The risk values are no less that those in schedules found by Algorithm SPT-NonSplit, but never larger than those found by the remaining four algorithms, that are aimed at reducing the number of flights.

Algorithms FFD, BFD and SS are pure bin-packing procedures that do not take into account the risk issues while searching for the number of feasible flights. The TS algorithm uses the risk component as part of its dynamically updated objective; therefore, it is not surprising that the average risk value delivered by TS algorithm is better than those provided by the bin-packing methods. Only on several occasions (e.g., instances B2, F2, F3, S4) the TS algorithm produces a flight schedule with a larger risk than that found by one of the bin-packing algorithms; but then the TS schedule consists of less flights.

For 16 out of the 20 A- and B-instances and on 9 occasions for larger instances, the smallest possible number of flights matching the lower bound m is determined. The schedules with m flights, if found by any algorithm, are always also found by the TS algorithm and have either the same of smaller risk values; this is why we mark with * the corresponding positions in the TS column of our tables.

Algorithm SS on average creates more flights than Algorithms FFD and BFD, which is especially noticeable for the large Flesland and Sola instances and is consistent with the absolute worst-case ratios of these algorithms (larger than 3/2 for Algorithm SS).

Thus, a decision-maker who is looking for a non-split schedule with the smallest risk and does not mind a fairly large number of flights, should make a choice between the solutions provided by Algorithms SPT-NonSplit and SPTu. A decision-maker who is after a non-split schedule with a small number of flights should rely on the TS algorithm which is superior to the bin-packing algorithms in terms of both the number of flights and the minimal takeoff and landing risk.

4 Split Scenario

In this section, we consider the split scenario for Problem SHP. Recall that under this scenario we want to create a schedule with exactly m flights, where m is the smallest value that satisfies (1). We present two heuristic algorithms, both based on transformation of an infeasible schedule $S_{SPT}(m)$ found by Algorithm SPT. Recall that Algorithm SPT minimizes the sum of the completion times on identical parallel machines, and for the latter problem preemption if allowed does not reduce the value of the function, as shown by Rothcopf [31].

For schedule $S_{SPT}(m)$, let a flight M_i visits the installations in accordance with the sequence π_i, $1 \leq i \leq m$. Our first heuristic takes schedule $S_{SPT}(m)$ as an input and creates a single sequence π of n installations as the merger of the sequences $(\pi_1, \pi_2, \ldots, \pi_m)$. Our heuristic scans the installations in the order π, and splits the sequence into m flights, closing

Table 3: Computational results for A-instances and B-instances from Qian et al.[8]

		SPT-NonSplit	SPTu	FFD	BFD	SS	TS
Ins.	m	$R(S)$ (flights)	$R(S)$ (flights)	$R(S)$ (flights)	$R(S)$ (flights)	$R(S)$ (flights)	$R(S)$ (flights)
A1	5	85 (8)	88 (7)	104 (5)	105 (5)	105 (5)	104 (5*)
A2	6	113 (9)	113 (9)	129 (7)	129 (7)	129 (7)	129 (7)
A3	5	82 (8)	94 (6)	102 (5)	102 (5)	102 (5)	101 (5*)
A4	5	91 (9)	93 (8)	110 (5)	110 (5)	110 (5)	110 (5*)
A5	5	81 (9)	87 (7)	102 (5)	102 (5)	103 (5)	98 (5*)
A6	5	92 (9)	102 (7)	110 (6)	110 (6)	110 (6)	110 (6)
A7	5	89 (7)	89 (7)	104 (5)	108 (5)	108 (5)	104 (5*)
A8	5	94 (9)	101 (7)	112 (6)	112 (6)	112 (6)	109 (6)
A9	5	86 (9)	93 (7)	108 (5)	108 (5)	108 (5)	108 (5*)
A10	6	100 (9)	104 (8)	118 (6)	118 (6)	118 (6)	118 (6*)
Avg.	5.2	91.3 (8.6)	96.4 (7.3)	109.9 (5.5)	110.4 (5.5)	110.5 (5.5)	109.1 (5.5)
B1	8	136 (14)	146 (11)	172 (8)	172 (8)	172 (8)	170 (8*)
B2	8	142 (13)	152 (11)	177 (8)	177 (8)	173 (9)	177 (8*)
B3	7	123 (11)	133 (9)	146 (7)	152 (7)	149 (7)	145 (7*)
B4	8	136 (14)	149 (11)	172 (8)	172 (8)	173 (8)	170 (8*)
B5	7	125 (14)	135 (10)	160 (7)	163 (7)	163 (7)	160 (7*)
B6	8	140 (13)	146 (11)	170 (8)	170 (8)	174 (8)	169 (8*)
B7	6	126 (8)	126 (8)	145 (6)	145 (6)	148 (6)	143 (6*)
B8	8	135 (14)	144 (11)	167 (8)	167 (8)	167 (8)	161 (8*)
B9	7	132 (14)	143 (11)	167 (8)	167 (8)	169 (8)	159 (8)
B10	6	112 (10)	116 (9)	136 (6)	136 (6)	138 (6)	135 (6*)
Avg.	7.3	130.7 (12.5)	139 (10.2)	161.2 (7.4)	162.1 (7.4)	162.6 (7.5)	158.9 (7.4)

Table 4: Computational results for Flesland and Sola instances from Qian et al.[9]

		SPT-NonSplit	SPTu	FFD	BFD	SS	TS
Ins.	m	$R(S)$(flights)	$R(S)$(flights)	$R(S)$(flights)	$R(S)$(flights)	$R(S)$(flights)	$R(S)$(flights)
F1	21	398 (33)	402 (31)	456 (24)	456 (24)	456 (24)	456 (24)
F2	20	363 (34)	377 (30)	445 (21)	445 (21)	442 (22)	445 (21)
F3	16	300 (33)	341 (23)	401 (17)	401 (17)	403 (17)	405 (16*)
F4	20	373 (34)	387 (29)	442 (21)	442 (21)	438 (22)	442 (21)
F5	17	307 (34)	342 (24)	412 (17)	412 (17)	405 (17)	396 (17*)
F6	19	344 (33)	361 (27)	418 (19)	418 (19)	423 (19)	415 (19*)
F7	18	336 (30)	350 (26)	400 (19)	400 (19)	404 (19)	396 (19)
F8	16	302 (31)	328 (23)	374 (16)	374 (16)	377 (17)	374 (16*)
F9	17	323 (30)	336 (25)	384 (19)	384 (19)	386 (19)	382 (19)
F10	18	334 (33)	364 (25)	416 (18)	416 (18)	421 (18)	416 (18*)
Avg.	18.2	338 (32.5)	358.8 (26.3)	414.8 (19.1)	414.8 (19.1)	415.5 (19.4)	412.7 (19)
S1	18	343 (35)	381 (27)	458 (19)	459 (19)	453 (20)	450 (19)
S2	20	368 (36)	394 (28)	446 (21)	446 (21)	448 (22)	442 (21)
S3	22	401 (36)	412 (33)	472 (24)	472 (24)	472 (24)	472 (24)
S4	19	356 (36)	389 (27)	448 (20)	448 (20)	446 (20)	452 (19*)
S5	18	329 (36)	366 (25)	445 (18)	445 (18)	432 (19)	425 (18*)
S6	17	326 (32)	359 (25)	419 (18)	419 (18)	423 (18)	413 (18)
S7	18	338 (36)	377 (26)	436 (19)	436 (19)	439 (19)	425 (19)
S8	19	345 (36)	379 (26)	438 (19)	440 (19)	444 (19)	433 (19*)
S9	18	339 (32)	360 (25)	412 (19)	414 (19)	414 (19)	408 (19)
S10	18	334 (35)	358 (27)	425 (18)	427 (18)	427 (18)	423 (18*)
Avg.	18.7	347.9 (35)	377.5 (26.9)	439.9 (19.5)	440.6 (19.5)	439.8 (19.8)	434.3 (19.4)

the current flight each time when the capacity Q is reached. Its formal description is given below.

Algorithm Split1

Step 1. Run Algorithm SPT and find schedule $S_{SPT}(m)$. Create the sequence π by merging the sequences $\pi_1, \pi_2, \ldots, \pi_m$. Renumber the installations in the order they appear in sequence π. Define $u_0 = v_0 = 0$.

Step 2. For i from 1 to $m-1$ do

(a) Find an installation u_i such that

$$v_{i-1} + \sum_{j=u_{i-1}+1}^{u_i-1} p_j < Q, \; v_{i-1} + \sum_{j=u_{i-1}+1}^{u_i} p_j \geq Q.$$

Compute $v_i := v_{i-1} + \sum_{j=u_{i-1}+1}^{u_i} p_j - Q$.

(b) Form flight M_i that picks up v_{i-1} people from installation u_{i-1} (provided that $v_{i-1} > 0$, otherwise the installation is not visited), p_j people from each installation j, $u_{i-1} + 1 \leq j \leq u_i$, and $p_{u_i} - v_i$ people from installation u_i.

Step 3 Form the last flight M_m that picks up v_{m-1} people from installation u_{m-1} (provided that $v_{m-1} > 0$, otherwise the installation is not visited) and p_j people from all remaining installations.

Step 4. If required, for each flight $M_i, 1 \leq i \leq m$, rearrange the assigned installations in non-decreasing order of the numbers of passengers to be picked up.

Step 5. Call the resulting schedule $S_{\text{Split1}}(m)$ and stop.

The running time of the algorithm is $O(n \log n + nm)$. The way it performs the splitting is very similar to the classical "wrap around" algorithm by McNaughton [32] that finds a makespan-optimal preemptive schedule for the scheduling problem on m identical parallel machines.

Algorithm Split1 applied to Example 1 with $Q = 19$ works as follows. The initial permutation π is $(8, 5, 1, 6, 4, 2, 7, 3)$. In the course of running the algorithm, the pickup demands of installations 6 and 7 are split. In our description of the flights in schedule $S_{\text{Split1}}(3)$ obtained after the sorting in Step 4, all passengers are collected from an installation, unless stated otherwise. Flight M_1 visits installations 8, 6 (to pick up 3 passengers), then 5 and 1 with the risk $3 + 6 + 11 + 19 = 39$. Flight M_2 takes 1 person from installation 6, then visits installations 4 and 2, and then picks up 2 people from installation 7; the total risk of that flight is $1 + 7 + 17 + 19 = 44$. Finally, flight M_3 picks up 2 people from installation 7 and then visits installation 3; the total risk of that flight is $2 + 9 = 11$. The risk of the schedule is 94.

Our second algorithm also explores the structure of schedule $S_{SPT}(m)$ and splits only those demands that make the corresponding flights in that schedule infeasible. Assume that there are u infeasible flights in schedule $S_{SPT}(m)$, where $1 \leq u \leq m-1$. Similarly to the proof of Theorem 1, we renumber the flights in schedule $S_{SPT}(m)$ in such a way that the first u flights are infeasible, with installation $n - u + i$ being the last for flight M_i, $1 \leq i \leq u$. Removing the installations $n-u+1, \ldots, n$ from schedule $S_{SPT}(m)$ we obtain a (partial) flight schedule S' in which the values of total load $C^{(i)}, 1 \leq i \leq m$, satisfy (7).

We show that there exists a split flight schedule with m flights with the total risk that is less than $\left(2-\frac{1}{m}\right) R(S_{SPT}(m))$.

In schedule $S_{SPT}(m)$, for an infeasible flight M_i, $1 \le i \le u$, denote

$$x_i := C^{(i)} + p_{n-u+i} - Q$$

and call this value the excess of demand p_{n-u+i}. Informally, x_i is the number of passengers that cannot be taken from installation $n-u+i$ and has to be redistributed between other flights. Let $X = \sum_{i=1}^{u} x_i$ be the total excess of u infeasible flights due to the pickup demands $p_{n-u+1}, \cdots, p_n$. To create a feasible solution with m flights, we redistribute the total excess over the $m-u$ flights that are feasible in schedule $S_{SPT}(m)$.

Starting from $i=1$, reassign x_1 passengers from installation $n-u+1$ to the first available feasible flight M_{u+1}. If the flight remains feasible, i.e., its total load is still at most Q, move x_2 passengers from installation $n-u+2$ to the first available feasible flight, etc.; if not, find by how much the load of flight M_1 exceeds Q and assign the excess to the next available flight. Repeating this process, the whole excess X will be eventually redistributed due to (1). The formal description of the algorithm is given below.

Algorithm Split2

Step 1. Run Algorithm SPT. Compute the load L_i of each flight M_i, $1 \le i \le m$. If required, modify the found schedule appropriately as described above, so that the first u flights are infeasible. Determine excess values x_i, $1 \le i \le u$. Initialize $i := j := 1$.

Step 2 While $i \le u$ do

(a) Make flight M_i pick up x_i passengers less from installation $n-u+i$. Close flight M_i by redefining its load $L_i := Q$. Include installation $n-u+i$ to the first feasible flight M_{u+j} to be currently the last visited and make the flight pick up x_i passengers of from that installation. Increase the load of flight M_{u+j} by $L_{u+j} := L_{u+j} + x_i$. If $L_{u+j} \le Q$, then go to Step 2(c); otherwise go to Step 2(b).

(b) While $L_{u+j} > Q$ do

(b1) Update the excess value $x_i := L_{u+j} - Q$. Make flight M_{u+j} pick up x_i passengers less from installation $n-u+i$. Close flight M_{u+j} by redefining its load $L_{u+j} := Q$. Include installation $n-u+i$ to the first feasible flight M_{u+j+1} to be currently the last visited and make the flight pick up x_i passengers of from that installation. Increase the load of flight M_{u+j+1} by $L_{u+j+1} := L_{u+j+1} + x_i$. Set $j := j+1$.

(b2) End while.

(c) Set: $i = i+1$.

(d) End while.

Step 3. Call the resulting schedule $S_{\text{Split2}}(m)$ and stop.

If Algorithm Split2 is applied to Example 1 with $Q = 19$ and in the final schedule the original numbering of the flights is restored, then flight M_1 is exactly as in schedule $S_{SPT}(3)$, and flight M_2 is almost as in $S_{SPT}(3)$, except not 10 but 9 passengers are picked up from installation 2. The remaining passenger is picked up from installation 2 by flight M_3. If the additional visit to installation 2 by flight M_3 is sequenced to be the last in the

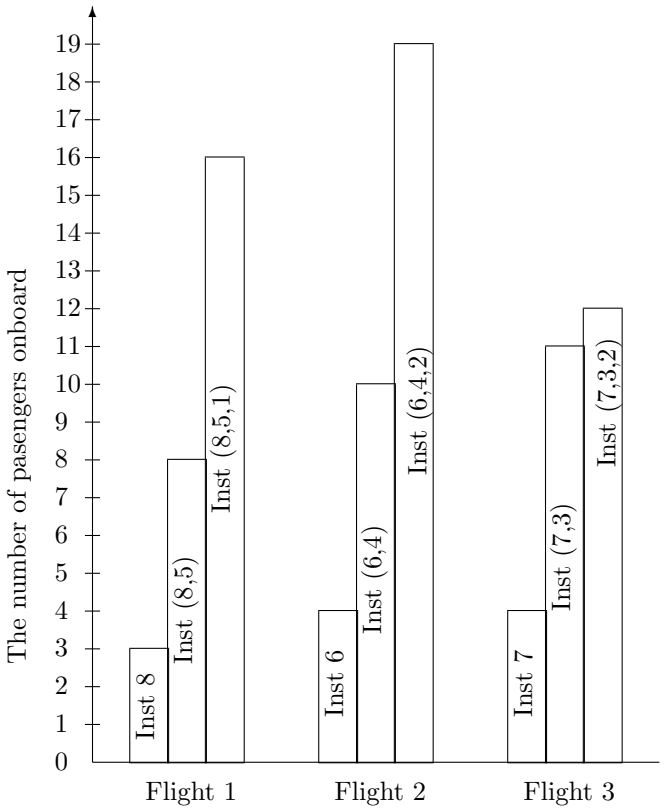

Figure 2: Flight schedule created by Algorithm Split2

route (as described in Algorithm Split2), then the total risk of schedule $S_{\text{Split2}}(3)$ is equal to $(3+8+16)+(4+10+19)+(4+11+12)=87$; see Figure 2. If for flight M_3 the installations are visited in non-decreasing order of the numbers of passengers to be picked up, the total risk can be reduced to become $(3+8+16)+(4+10+19)+(1+5+12)=78$.

The following statement analyzes the worst-case performance of Algorithm Split2.

Theorem 2 *Let $S_{SPT}(m)$ and $S_{\text{Split2}}(m)$ be the flight schedules found by Algorithm SPT and Algorithm Split2, respectively. Then the bound*

$$\frac{R(S_{\text{Split2}}(m))}{R(S_{SPT}(m))} \leq 2 - \frac{1}{m} \tag{11}$$

holds and this bound is tight.

Proof: Suppose that Algorithm Split2 redistributes the excess x_i of installation $n-u+i$ to z_i flights, $1 \leq i \leq u$.

Since for this redistribution we may only use the flights $M_{u+1}, \ldots, M_m$, it follows that $z_1 \leq m-u$. If $z_1 > 1$ then $z_1 - 1$ flights starting from M_{u+1} are loaded to their full capacity of Q. Then for installation $n-u+2$ we have $z_2 \leq m-u-(z_1-1)$.

Similarly, for $2 \leq v \leq u$, to redistribute excess x_v of installation $n-u+v$ we have $m-u-\sum_{i=1}^{v-1}(z_i-1)$ available flights. This means that

$$z_u \leq m-u-\sum_{q=1}^{u-1}(z_i-1)\,,$$

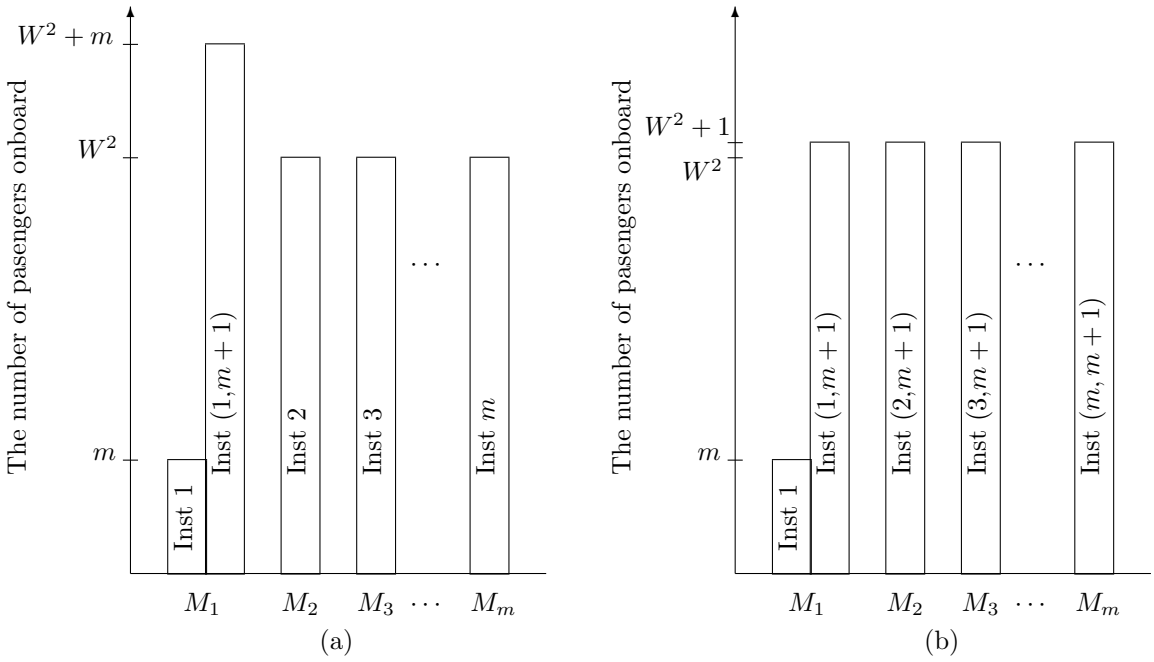

Figure 3: (a) Schedule $S_{SPT}(m)$; (b) Schedule $S_{\text{Split2}}(m)$

i.e.,

$$\sum_{i=1}^{u} z_i \leq m - u + (u-1) = m - 1.$$

In the worst case, in schedule $S_{\text{Split2}}(m)$ up to $m-1$ flights will have a load of Q and one flight will have a load of $p(N) - (m-1)\,Q$. In schedule $S_{\text{Split2}}(m)$ compared to schedule $S_{SPT}(m)$ there will be at most $m-1$ additional visits to installations, and in the worst case at most $m-2$ of these visits will lead to a load of Q each and one visit will lead to a load of $p(N) - (m-1)\,Q$. This implies

$$R(S_{\text{Split2}}(m)) \leq R(S_{SPT}(m)) + (m-2)\,Q + p(N) - (m-1)\,Q.$$

Due to (1) and the lower bound (10) we deduce that

$$\begin{aligned} R(S_{\text{Split2}}(m)) &\leq R(S_{SPT}(m)) + p(N) - Q \\ &\leq R(S_{SPT}(m)) + p(N) - \frac{p(N)}{m} \\ &= R(S_{SPT}(m)) + \frac{(m-1)\,p(N)}{m} \leq \left(2 - \frac{1}{m}\right) R(S_{SPT}(m)), \end{aligned}$$

which proves the required bound (11).

To see that this bound is tight, take an instance of Problem SHP with $m+1$ installations such that $p_1 = m$, $p_2 = \cdots = p_{m+1} = W^2$, and $Q = W^2 + 1$, where W is a sufficiently large number. We have that $p(N) = m\left(W^2+1\right)$, so that (1) holds as equality. It follows that $R(S_{SPT}(m)) = m + (m-1)\,W^2 + \left(W^2+m\right) = m\left(W^2+2\right)$; see Figure 3(a). In schedule $S_{\text{Split2}}(m)$ flight M_1 picks up m people from installation 1 and $W^2 - m + 1$ people from

installation $m+1$. Each flight M_i, $2 \leq i \leq m$, visits installation i to pickup W^2 people and installation $m+1$ to pick up one passenger; see Figure 3(b). We have that $R(S_{\text{Split2}}(m)) = m + (m-1)W^2 + mQ = m + (m-1)W^2 + m\left(W^2+1\right) = 2m + W^2(2m-1)$.

As W grows, the ratio

$$\frac{R(S_{\text{Split2}}(m))}{R(S_{SPT}(m))} = \frac{W^2(2m-1)+2m}{m\left(W^2+2\right)}$$

goes to the ratio of the coefficients for W^2, i.e., to $2 - \frac{1}{m}$. ∎

Since Algorithm Split2 creates at most $m-1$ additional visits, its running time is $O(n \log n + nm)$. The algorithm delivers a split schedule with m flights with a total risk that is less that factor 2 away from the lower bound $R(S_{SPT}(m))$.

Provided that all pickup demand values do not exceed Q, in schedules found by Algorithm Split1 at most $m-1$ installations will be served by two flights, thereby creating at most $m-1$ additional visits (as in the case of Algorithm Split2). While the total number of these additional visits is bounded by the same expression $m-1$ for both algorithms, Algorithm Split1 guarantees that no installation is visited more than twice. For Algorithm Split2 this is not the case, and for an installation its pickup demand can be split by several flights. In practice, this is not a welcomed features, as this increases the total number of landings and takeoffs at some installations which might be difficult to schedule.

We have conducted a series of computational experiments with Algorithms Split1 and Split 2. Please notice that in our implementation of Algorithm Split2 we additionally rearrange the installations of each flight in non-decreasing order of the numbers of passengers to be picked up. The same test-bed as in Section 3 is used. The m value and the risk values $R(S)$ for the schedules found by Algorithm Split1 and Algorithm Split2 are presented for each instance in Table 5. Additionally, in order to validate the performance of Algorithm Split 2 in practice, the table includes the lower bound on the total risk $LB = R(S_{SPT}(m))$ and the ratio of $R(S)$ for the schedules found by Algorithm Split2 to that lower bound.

Algorithm Split2 outperforms Algorithm Split1 on average, and in fact for almost all instances, except A2, A8 and F1. The risk values for the schedules generated by Algorithm Split2 are less than $1.13 \cdot LB$ on average for small instances and $1.15 \cdot LB$ for large instances, which is less than the theoretically proved bound $2 - \frac{1}{m}$. Recall that for 25 of the considered instances a non-split schedule with m flights can be found; see Tables 3 and 4. The risk values of the non-split schedules are less that those for split schedules; this is because in a split schedule some of the installations will contribute several times to the overall risk value.

The following practical recommendations can be deduced from this study. The decision-maker should first try the non-split scenario, and only turn to the split scenario if the smallest number of flights is imperative and non-split schedules with such number of flights cannot be found. For finding split schedules, Algorithm Split2 should be given preference, and only if the number of landings/takeoffs for some installation(s) is not acceptable, Algorithm Split1 should be employed as an alternative.

5 Conclusions and Future Research

This paper analyzes the passenger takeoff and landing risk for helicopter pickup in offshore petroleum industry under two scenarios. Under the non-split scenario each installation is visited only once and all people from an installation are picked up. Under the split scenario, the

Table 5: Computational results for the splitting algorithms

Ins.	m	Split1	Split2	LB	Split2/LB	Ins.	m	Split1	Split2	LB	Split2/LB
A1	5	127	120	104	1.15	B1	8	220	192	170	1.13
A2	6	156	160	138	1.16	B2	8	207	198	177	1.12
A3	5	122	107	101	1.06	B3	7	171	152	145	1.05
A4	5	118	118	106	1.11	B4	8	204	191	170	1.12
A5	5	128	105	98	1.07	B5	7	189	183	158	1.16
A6	5	134	132	118	1.12	B6	8	211	192	169	1.14
A7	5	116	109	104	1.05	B7	6	183	163	143	1.14
A8	5	132	140	119	1.18	B8	8	196	177	161	1.10
A9	5	128	126	108	1.17	B9	7	218	193	166	1.16
A10	6	141	132	118	1.12	B10	6	179	156	135	1.16
Avg.	5.2	130.2	124.9	111.4	1.12		7.3	197.8	179.7	159.4	1.13
F1	21	553	564	485	1.16	S1	18	549	509	449	1.13
F2	20	552	527	455	1.16	S2	20	554	515	451	1.14
F3	16	465	454	391	1.16	S3	22	597	546	492	1.11
F4	20	547	516	452	1.14	S4	19	562	524	447	1.17
F5	17	459	442	387	1.14	S5	18	495	481	415	1.16
F6	19	490	476	415	1.15	S6	17	518	501	424	1.18
F7	18	483	466	405	1.15	S7	18	520	509	434	1.17
F8	16	479	425	372	1.14	S8	19	511	501	431	1.16
F9	17	474	473	401	1.18	S9	18	490	463	418	1.11
F10	18	484	458	414	1.11	S10	18	530	482	416	1.16
Avg.	18.2	498.6	480.1	417.7	1.15		18.7	532.6	503.1	437.7	1.15

pickup demand of an installation can be split between different flights. The risk is measured as the total number of passengers exposed to takeoffs and landings.

We demonstrate the links between our safe transportation problem and problems of scheduling and bin-packing. Under the non-split scenario, we offer several algorithms that find flight schedules with various numbers of flights and risk values for the decision-maker to choose the one to be implemented. In the case of the split scenario, we give two algorithms to transform an infeasible low risk schedule into a feasible split schedule. Transformation done by one of our algorithms increases the risk less than twice, compared to the global, not attainable, lower bound.

Computational results are presented on randomly generated instances based on the real numbers of installations in the oil fields on the Norwegian continental shelf.

The pickup only model studied in this paper is somewhat dual to that in which delivery only is performed by several flights. In the latter case, a flight will start at the heliport with all passengers to be delivered on board, visits the installations in some established order and leaves the required number of passengers in each installation, and returns to the heliport with no passengers on board. The delivery model also accepts two scenarios, non-split and split. The problem is closely related to that of pickup only, and does not need to be studied separately. To handle the delivery model, we can do the following:

1. For each installation, treat the delivery demands as pickup demands.
2. Applying the corresponding scenario, find flight schedules for the resulting artificial pickup problem.
3. Convert found flight schedules into schedules for the original delivery problem by reversing the order in which the installations are visited by each flight.

Notice that the value of the total risk for a flight schedule created for the artificial pickup problem is equal to that for the original delivery problem, since the same values need to be

added for each flight, but in the opposite order. For example, if for Example 1 the values in Table 1 are the delivery demands, then a possible flight schedule for the non-split scenario can be converted from that found by Algorithm SPTu: flight M_1 visits the installations in the order $(1, 5, 8)$, for flights M_2 and M_3 the sequences are $(4, 6)$ and $(3, 7)$, respectively, while flight M_4 visits installation 2 alone. To visualize such a schedule, reverse the order of the bars for each flight in Figure 1.

Recall that Qian et al. [9] show how to perform simultaneous pickup and delivery by a single flight of sufficient capacity. We are not aware how to extend their result for more than one flight, even if the flight capacity is ignored. Counterexamples show that using a version of Algorithm LS with a list of jobs organized by some simple priority rules does not solve the problem optimally.

Among further research goals of studying the range of safe helicopter problems are the following:

- perform a clear classification of various problems and determine their computational complexity status, including those of simultaneous pickup and delivery;
- consider weighted versions of the problems, in which different risk values for landing and takeoffs are taken into account;
- consider problems that allow penalties for excluding installations from a flight schedule;
- develop approximation algorithms and metaheuristics to handle enhanced safe helicopter transportation problems.

Acknowledgements

We are grateful to the anonymous referees for their constructive critics of our paper that has helped us to outline clearly the scope of this study.

References

[1] J. E. Vinnem, T. Aven, T. Husebø, J. Seljelid, O. J. Tveit, Major hazard risk indicators for monitoring of trends in the Norwegian offshore petroleum sector, Reliability Engineering and System Safety 91 (2006) 778–791.

[2] G. Morrison, Helicopter safety offshore, Health and Safety Executive, Aberdeen, 2001.

[3] M. Caramia, F. Guerriero, A heuristic approach to long-haul freight transportation with multiple objective functions, Omega 37 (2009) 600–614.

[4] H. Ma, B. Cheang, A. Lim, L. Zhang, Y. Zhu, An investigation into the vehicle routing problem with time windows and link capacity constraints, Omega 40 (2012) 336347.

[5] A. Subramanian, E. Uchoa, L.S. Ochi, A hybrid algorithm for a class of vehicle routing problems, Computers & Operations Research 40 (2013) 2519-2531.

[6] L. Moreno, M. Poggi de Aragão, E. Uchoa, Column generation based heuristic for a helicopter routing problem, Lecture Notes in Computer Science, Springer, Berlin, 2006, pp. 219–230.

[7] F. Menezes, O. Porto, M. L. Reis, L. Moreno, M. Poggi de Aragão, E. Uchoa, H. Abeledo, N. Carvalho do Nascimento, Optimizing helicopter transport of oil rig crews at Petrobras, Interfaces 40 (2010) 408–416.

[8] F. Qian, I. Gribkovskaia, Ø Halskau, Helicopter routing in the Norwegian oil industry: Including safety concern for passenger transport, International Journal of Physical Distribution & Logistics Management 41 (2011) 401–415.

[9] F. Qian, I. Gribkovskaia, G. Laporte, Ø Halskau, Passenger and pilot risk minimization in offshore helicopter transportation, Omega 40 (2012) 584–593.

[10] G. H. Hardy, J. E. Littlewood, G. Polya, Inequalities, Cambridge University Press, London, 1934.

[11] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics 5 (1979) 287–326.

[12] W. E. Smith, Various optimizers for single state production, Naval Research Logistics Quarterly 3 (1956) 56–66.

[13] R. L. Graham, Bounds for certain multiprocessing anomalies, Bell System Technical Journal 45 (1966) 1563-1581.

[14] R. W. Conway, W. L. Maxwell, L. W. Miller, Theory of scheduling, Addison Wesley, Reading, 1967.

[15] C. Stein, J. Wein, On the existence of schedules that are near-optimal for both makespan and total weighted completion time, Operations Research Letters 21 (1997) 115–122.

[16] J. Aslam, A. Rasala, C. Stein, N. Young, Improved bicriteria existence theorems for scheduling, Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (1999) 846–847.

[17] J. N. D. Gupta, J. C. Ho, Minimizing makespan subject to minimum flowtime on two identical machines, Computers & Operations Research 28 (2001) 705–717.

[18] J. N. D. Gupta, A. J. Ruiz-Torres, Minimizing makespan subject to minimum total flow-time on identical parallel machines, European Journal of Operational Research 125 (2000) 370–380.

[19] B. T. Eck, M. Pinedo, On the minimization of the makespan subject to flowtime optimality, Operations Research 41 (1993) 797–801.

[20] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, Approximation algorithms for bin packing: A survey, in: D. S. Hochbaum (Eds.), Approximation algorithms for NP-hard problems, PWS Publishing Company, Boston, 1997, pp.46–93.

[21] J. Y.-T. Leung, M. Pinedo, Minimizing total completion time with parallel machines with deadline constraints, SIAM Journal on Computing 32 (2003) 1370–1388.

[22] S. T. McCormic, M. Pinedo, Scheduling n independent jobs on m uniform machines with both flowtime and makespan objectives: a parametric analysis, ORSA Journal on Computing 7 (1995) 63–77.

[23] T. F. Gonzalez, J. Y.-T. Leung, M. Pinedo, Minimizing total completion time on uniform machines with deadline constraints, ACM Transactions on Algorithms (TALG) 2 (2006) 95–115.

[24] P. Serafini, Scheduling jobs on several machines with job splitting property, Operations Research 44 (1996) 617–628.

[25] W. Xing, J. Zhang, Parallel machine scheduling with splitting jobs, Discrete Applied Mathematics 103 (2000) 259–269.

[26] D. Simchi-Levi, New worst-case results for the bin packing problem, Naval Research Logistics 41 (1994) 579–585.

[27] G. Zhang, X. Cai, C. K. Wong, Linear time-approximation algorithms for bin packing, Operations Research Letters 26 (2000) 217–222.

[28] A. Caprara, U. Pferschy, Worst-case analysis of the subset sum algorithm for bin packing, Operations Research Letters 32 (2004) 159–166.

[29] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 13 (1986) 533–549.

[30] J.-F. Cordeau, M. Gendreau, G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks 30 (1997) 105–119.

[31] M. H. Rothkopf, Scheduling independent tasks on independent processors, Management Science 12 (1966) 437–447.

[32] R. McNaughton, Scheduling with deadlines and loss functions, Management Science 6 (1959) 1–12.