The Checkpoint Ordering Problem[*]

Philipp Hungerländer[†]

July 15, 2017

Abstract

We suggest a new variant of a row layout problem: Find an ordering of n departments with given lengths such that the total weighted sum of their distances to a given checkpoint is minimized. The Checkpoint Ordering Problem (COP) is both of theoretical and practical interest. It has several applications and is conceptually related to some well-studied combinatorial optimization problems, namely the Single-Row Facility Layout Problem, the Linear Ordering Problem and a variant of parallel machine scheduling.

In this paper we study the complexity of the (COP) and its special cases. The general version of the (COP) is NP-hard in the weak sense. We propose both a dynamic programming algorithm and an integer linear programming approach for the (COP). Our computational experiments indicate that the (COP) is hard to solve in practice. While the run time of the dynamic programming algorithm strongly depends on the length of the departments, the integer linear programming approach is able to solve instances with up to 25 departments to optimality.

Keywords: Combinatorial optimization; dynamic programming; integer linear programming; global optimization; facilities planning and design.

1 Introduction

In this paper we introduce and analyze a new variant of a row layout problem. An instance of the Checkpoint Ordering Problem (COP) consists of n one-dimensional departments, with given positive lengths $\ell_1, \ldots, \ell_n$ and weights $w_1, \ldots, w_n$, and a checkpoint on a fixed position, e.g. left-aligned or at the center position. The optimization problem can be written down as

$$\min_{\pi \in \Pi_n} \sum_{i \in [n]} w_i z_i^\pi, \tag{1}$$

where Π_n is the set of permutations of the indices $[n] := \{1, 2, \ldots, n\}$ and z_i^π is the distance between the center of department i and the checkpoint with respect to a particular permutation $\pi \in \Pi_n$.

Let us start with elaborating on the connections of the (COP) to the Linear Ordering Problem (LOP), the Single-Row Facility Layout Problem (SRFLP) and scheduling on identical parallel machines with the objective of minimizing the sum of weighted completion times.

The Single-Row Facility Layout Problem (SRFLP). The simplest known layout type is single-row layout. It arises as the problem of ordering stations on a production line, where the material flow is handled by an automated guided vehicle (AGV) in both directions on a straight-line path [18]. An instance of the

[*]This work was partially supported by the Austrian Science Fund (FWF) Project J3793-N95.

[†]Laboratory for Information & Decision Systems, MIT, USA, philipp.hungerlaender@aau.at

`(SRFLP)` consists of n one-dimensional departments, with given positive lengths $\ell_1, \dots, \ell_n$, and pairwise weights w_{ij}. The optimization problem can be written down as

$$\min_{\pi \in \Pi_n} \sum_{\substack{i,j \in [n] \\ i<j}} w_{ij} z_{ij}^{\pi}, \tag{2}$$

where Π_n is the set of permutations of the indices $[n]$ and z_{ij}^{π} is the center-to-center distance between departments i and j with respect to a particular permutation $\pi \in \Pi_n$. Note that the weights are assumed to be non-negative for all row layout problems to ensure boundedness of the objective value of the optimal layout. For the `(SRFLP)` $w_{ij} \geq 0$ further guarantees that all departments are placed next to each other without spacing. For the `(COP)` we do not need this restriction on the weights as space between the departments is not allowed.

Several practical applications of the `(SRFLP)` have been identified in the literature, such as the arrangement of rooms on a corridor in hospitals, supermarkets, or offices [33], the assignment of airplanes to gates in an airport terminal [35], the arrangement of machines in flexible manufacturing systems [18], the arrangement of books on a shelf and the assignment of disk cylinders to files [27].

Similar applications are conceivable for the `(COP)`, e.g. the rooms on a corridor could be arranged such that the weighted sum of their distances with the office of the head is minimized or planes could be assigned to gates such that the weighted sum of their distances from the entrance of the airport terminal is minimized. When comparing the `(SRFLP)` with the `(COP)`, we observe that the problems are quite similar. One difference is that an `(SRFLP)` instance has $\binom{n}{2}$ weights while a `(COP)` instance has only n weights.

The Linear Ordering Problem `(LOP)`. Ordering problems associate to each ordering (or permutation) of the set $[n]$ a profit and the goal is to find an ordering of maximum profit. In the simplest case of the Linear Ordering Problem `(LOP)`, this profit is determined by those pairs $(u, v) \in [n] \times [n]$, where u comes before v in the ordering. Thus in its matrix version the `(LOP)` can be defined as follows. Given an $n \times n$ matrix $A = (a_{ij})$ of integers, find a simultaneous permutation π of the rows and columns of A such that

$$\sum_{\substack{i,j \in [n] \\ i<j}} a_{\pi(i),\pi(j)},$$

is maximized. Equivalently, we can interpret a_{ij} as weights of a complete directed graph G with vertex set $V = [n]$. A tournament consists of a subset of the arcs of G containing for every pair of nodes i and j either arc (i, j) or arc (j, i), but not both. Then the `(LOP)` consists of finding an acyclic tournament, i.e. a tournament without directed cycles, of G of maximum total edge weight.

Although the `(LOP)` and the `(COP)` have apparently a similar structure, it is harder to directly relate these two problems. We will show in Section 2 that the `(COP)` with left-aligned or right-aligned checkpoint is in fact a `(LOP)` with some additional structure and can be solved efficiently by a greedy algorithm.

Scheduling on identical parallel machines. Furthermore we can relate the `(COP)` to the NP-hard [25] scheduling on identical parallel machines with the objective of minimizing the total weighted completion time that is defined as follows: We are given a set of jobs $\mathcal{J}$ that have to be scheduled on m identical parallel machines. Each job $j \in \mathcal{J}$ is specified by its processing time $p_j \geq 0$ and by its weight $w_j \geq 0$. Every machine can process at most one job at a time, and every job has to be be processed on one machine in an uninterrupted fashion. The completion time of job j is denoted by C_j. The goal is to minimize the total weighted completion time $\sum_{i \in \mathcal{J}} w_i C_i$. In the standard classification scheme of Graham et al. [13], this scheduling problem is denoted by $P||\sum w_j C_j$ for m part of the input, and by $Pm||\sum w_j C_j$ for constant m.

For the special case of only one machine the problem can be solved in polynomial time by Smith's Rule [34] that suggests to process the jobs in the order of non-increasing ratios w_j/p_j. For a constant number $m \geq 2$ of machines, the problem is weakly NP-hard as it can be solved in pseudopolynomial time by dynamic programming approaches [22, 24, 31]. For m part of the input, the problem is NP-hard in the strong sense by transformation from 3-PARTITION [11]. For further details on the complexity of various related variants of this scheduling problem we refer to problem SS13 in [11]. For a more general overview on machine scheduling we refer to the survey article by Lawler et al. [23].

From a computational point of view $P||\sum w_jC_j$ has been tackled by various branch-and-bound methods [2, 3, 6, 9, 32], for which determining the optimal solution of instances with 30 or more jobs and two or more identical machines is typically difficult [38].

When comparing $P||\sum w_jC_j$ and the (COP), we observe that there two important differences between the two problems:

1. The checkpoint must not lie exactly at a splitting point of two departments but it can also be covered by a department. I.e. the checkpoint does not necessarily define a partition of the departments. When considering a scheduling set-up the (COP) can be described as follows: It is allowed to split one arbitrary job into two parts at any point and then the two parts have to be scheduled first on the two machines.

2. For the (COP) the sum of the lengths of the departments that are placed to the left and to the right of the checkpoint are predetermined through the position of the checkpoint. E.g. for a centered checkpoint the sum of the lengths of the departments to the left and to the right of the checkpoint has to be equal. For $P||\sum w_jC_j$ the identical machines typically have no capacity restrictions.

Due to these differences it is not possible to directly carry over polyhedral results, dynamic programming algorithms and (mixed) integer linear programming models and their corresponding approximation results [28] from scheduling on identical parallel machines to the (COP).

Toy Examples. Now let us further clarify the similarities and differences of the (SRFLP) and the (COP) with the help of a toy example: We consider 4 departments with lengths $\ell_1 = 1,\ \ell_2 = 2,\ \ell_3 = 3,\ \ell_4 = 4$. Additionally we are given the pairwise weights $w_{12} = w_{14} = w_{34} = 1,\ w_{13} = w_{24} = 2$. For the (COP) we assign department 1 to row 2 and all other departments to row 1 and hence disregard the weights $w_{24} = 2$ and $w_{34} = 1$. Figure 1 illustrates the optimal layouts and the associated costs for both problems.

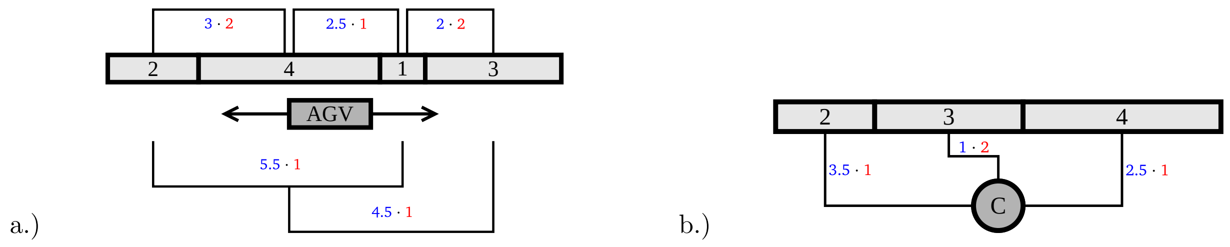

Figure 1: We are given the following data: $\ell_1 = 1,\ \ell_2 = 2,\ \ell_3 = 3,\ \ell_4 = 4,\ w_{12} = w_{14} = w_{34} = 1,\ w_{13} = w_{24} = 2$. In a.) we display the optimal layout for the (SRFLP) with corresponding costs of $3 \cdot 2 + 2.5 \cdot 1 + 2 \cdot 2 + 5.5 \cdot 1 + 4.5 \cdot 1 = 22.5$. And in b.) we depict the optimal layout for the (COP) with department 1 assigned to row 2 and all other departments assigned to row 1, disregarding the weights $w_{24} = 2$ and $w_{34} = 1$. Further we assume that the checkpoint lies at the center. Then the costs of the optimal (COP) layout are $3.5 \cdot 1 + 1 \cdot 2 + 2.5 \cdot 1 = 8$.

Finally we also want to clarify the workings of the (LOP) with the help of a toy example. We consider 4 objects and the weights $w_{12} = w_{41} = w_{34} = 1,\ w_{31} = w_{24} = 2$. Figure 2 illustrates the optimal ordering of the objects and the corresponding benefit.

Outline. The main contributions of this paper are the following:

- We propose a new combinatorial optimization problem that is both of theoretical and practical interest.

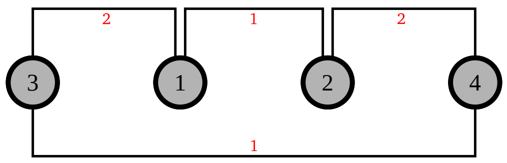

Figure 2: We are given 4 objects and the weights $w_{12} = w_{41} = w_{34} = 1,\ w_{31} = w_{24} = 2$. We display the optimal (LOP) solution with the corresponding benefit of $1 + 1 + 2 + 2 = 6$.

- We study the complexity of the (COP) and its special cases, pointing out several connections to related problems.
- We propose two exact approaches for the (COP), namely a dynamic programming algorithm and an integer linear programming approach.
- We demonstrate the practical difficulty of the (COP) in a computational study. In this context let us also refer to our companion paper [19] for a comparison of the empirical difficulty of several row facility layout problems (including the (COP)) on a variety of well-known benchmark instances.

The paper is structured as follows. In Section 2 we study the complexity of the (COP) and its special cases. In Sections 3 and 4 we suggest a dynamic programming algorithm and an integer linear programming approach for the (COP) respectively. Finally in Section 5 we conduct computational experiments, indicating the practical applicability and limitations of the approaches suggested. Section 6 concludes the paper.

2 Complexity of the Checkpoint Ordering Problem

Consider the decision variant of the (COP): given some value M we ask whether there exists a permutation of the departments such that the obtained costs are at most M:

Decision Checkpoint Ordering (DCO):
Instance: n departments with given lengths $\ell_i \in \mathbb{N},\ i \in [n]$, and integer weights $w_i,\ i \in [n]$, and a checkpoint on a fixed position.
Question: Is there an ordering π of the departments such that the total costs $\sum_{i\in[n]} w_i z_i^\pi$ are $\leq M$?

In the following proof we assume w.l.o.g. that the checkpoint is located at the center in order to simplify the presentation.

Theorem 1. *DCO is NP-complete.*

Proof. It is clear that DCO $\in$ NP since a nondeterministic algorithm needs only to guess an ordering π and then can check in polynomial time if the corresponding costs are $\leq M$.

To prove that DCO is NP-complete, we give an NP-complete problem and a polynomial-time transformation to DCO. The following problem is NP-complete (see Subsection 3.1.5 of [11], originally proven by [20]):

PARTITION:
Instance: A finite set A and a "size" $s(a) \in \mathbb{N}$ for each $a \in A$ such that $\sum_{a\in A} s(a) = 2B$ is even.
Question: Is there a subset $A' \subseteq A$ such that $\sum_{a\in A'} s(a) = B$?

Now we transform an instance of PARTITION to an instance of DCO as follows. We replace each element $a \in A$ with given size $s(a)$ by a department a with length $\ell_a = s(a)$ and weight $w_a = s(a)$. Additionally to this local replacement we use an enforcer[1] by introducing a further department t with length $\ell_t = 1$ and

[1]A picturesque term suggested by [36].

weight $w_t = 2(B+1)^2$. Clearly this DCO instance can be constructed from the PARTITION instance in polynomial time.

If the ordering π is optimal then the center of department t is located exactly above the checkpoint because of the large weight of department t. Due to the definition of the weights of the departments $a \in A$, it does not change the objective value if we switch the positions of two departments both located left or right of t. Hence the only way to influence the objective value is to decide whether the departments should be located left or right of t. If we can find a subset $A' \subseteq A$ of the departments such that $\sum_{a \in A'} \ell_a = B$ and place them left of t and all other departments right of t, then the corresponding ordering π is for sure optimal. If the sum of lengths of the departments left of t in the optimal ordering π does not give B, then there exists no subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = B$.

In summary we have shown that there exists an ordering π of the departments such that the total costs are $\leq M := B(B+1)$ if and only if there exists a subset $A' \subseteq A$ of the corresponding PARTITION instance such that $\sum_{a \in A'} s(a) = B$. □

In the layout context the above result can be interpreted as follows: The minimization of the inter-row costs that occur in all multi-row layout problems is NP-hard, even in its simplest version. This is not only an interesting theoretical insight on its own but also supports our definition of the Single-Row bound in our companion paper [19]. Next let us consider some more specialized versions of the `(COP)` that turn out to be solvable by a greedy algorithm and hence in particular in polynomial time.

First we assume that all departments have the same length. In this case the optimal permutation of the departments can be obtained by a simple greedy selection. We choose the permutation π with the following property: The higher the weight of a department the smaller is its distance from the checkpoint. Next we give a short formal argument for the above claim and refer to Figure 3 for a toy example of this variant of the `(COP)`.

Fact 2. *A permutation π is optimal for the* `(COP)` *with identical department lengths iff it ensures the inequalities*

$$(z_i - z_j)(w_i - w_j) \leq 0, \quad i, j \in [n], i < j, \tag{3}$$

where z_i, $i \in [n]$, denotes the distance of the center of department i from the checkpoint.

Proof. The change of the objective function caused by swapping two departments i and j is $-(z_i - z_j)(w_i - w_j)$. Hence in particular the change in the objective function caused by this swap is independent of the length and weights of all other departments $k \in [n]$, $k \neq i$, $k \neq j$. Now assume that there exists an optimal permutation that does not ensure one inequality in (3), i.e. $-(z_i - z_j)(w_i - w_j) < 0$ for some departments i and j. Now swapping the two departments improves the objective value. As all other departments are not affected by pairwise swaps, it is not possible to improve the objective value by an arbitrary number of pairwise changes if (3) holds. Hence (3) is not only a necessary condition but also a sufficient one. □

Note that the special case of the `(SRFLP)` where all department lengths are equal and the weights are binary is still NP-hard [12]. This problem is called Minimum Linear Arrangement `(LA)`, belongs to the class of graph layout problems and is NP-hard even if the underlying graph G is bipartite [11]. `(LA)` was originally proposed by Harper [16, 17] to develop error-correcting codes with minimal average absolute errors and was since then applied to VLSI design [37], single machine job scheduling [1, 30] and computational biology [21, 26]. There exist approximation algorithms for `(LA)` with performance guarantee $O(\log n)$ [4, 29] and $O(\sqrt{\log n} \log \log n)$ [5, 10]. For further details on graph layout problems we refer to the survey paper of Díaz et al. [7].

Next we assume that the checkpoint is left-aligned or right-aligned. Also in this case the optimal permutation of the departments can be obtained by a simple greedy selection. We choose the permutation π with the following property: The higher the relative weight w_i/ℓ_i of a department i, the smaller is its distance from the checkpoint. Next we give a short formal argument for the above claim and refer to Figure 3 for a toy example of this variant of the `(COP)`. To facilitate the presentation of the proof, we assume w.l.o.g. that

1. the checkpoint is left-aligned and
2. the relative weights $w_i/\ell_i,\ i \in [n]$, are all distinct.

The following result is in fact known as Smith's Rule [34] in the scheduling context, where it describes a greedy algorithm to solve single-machine scheduling with the objective of minimizing the sum of completion times. For convenience we restate the proof in our notation.

Fact 3. *[34] The permutation π is optimal for the* (COP) *with a left-aligned checkpoint iff it satisfies the conditions*

$$\frac{w_i}{\ell_i} > \frac{w_j}{\ell_j},\ i, j \in [n],\ \pi(i) < \pi(j). \tag{4}$$

Proof. Assume that there exists an optimal permutation that does not satisfy condition (4) for two departments i and j: $\frac{w_i}{\ell_i} < \frac{w_j}{\ell_j}$ and $\pi(i) < \pi(j)$. Then there are also two neighboring departments k and l $(\pi(i) \leq \pi(k) < \pi(k) + 1 = \pi(l) \leq \pi(j))$ that do not satisfy condition (4): $\frac{w_k}{\ell_k} < \frac{w_l}{\ell_l}$ and $\pi(k) + 1 = \pi(l)$. Now if we swap k and l, the value of the objective function changes by the term $w_k\ell_l - w_l\ell_k$. But this term is negative as $\frac{w_k}{\ell_k} < \frac{w_l}{\ell_l}$ holds which yields a contradiction to the assumption that the permutation was optimal. Finally note that condition (4) defines a unique permutation, hence it is not only necessary but also sufficient. □

We can interpret the (COP) with left-aligned checkpoint also as a (LOP) with special structure. We collect the lengths of the departments in a column vector ℓ and the weights of the departments in a column vector c. Now we aim to find a simultaneous permutation π of the rows and columns of $A = \ell c^\top$ such that

$$\sum_{\substack{i,j\in[n]\\ i<j}} a_{\pi(i),\pi(j)},$$

is minimized. Contrary to the general (LOP), which is NP-hard [12], the matrix entries of W are not independent but determined by an outer product of two vectors. We have just seen in Fact 3 that the (COP) with left-aligned checkpoint can be solved by a simple greedy heuristic. Hence this special structure of the (LOP) cost matrix W as an outer product of two vectors is the reason why this (LOP) version can be solved in polynomial time.

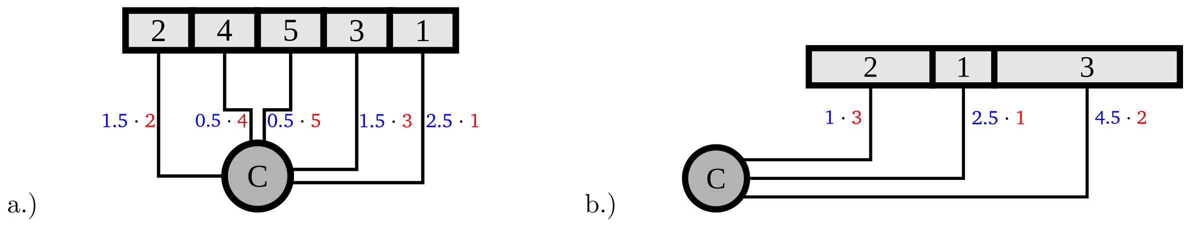

Figure 3: In a.) we display the optimal layout for the (COP) with identical department lengths equal to 1 and weights $w_1 = 1,\ w_2 = 2,\ w_3 = 3,\ w_4 = 4,\ w_5 = 5$. The corresponding layout costs are $0.5\cdot5+0.5\cdot4+1.5\cdot3+1.5\cdot2+2.5\cdot1 = 14.5$. In b.) we show the optimal layout for the (COP) with left-aligned checkpoint on the following instance with 3 departments: $\ell_1 = 1, \ell_2 = 2,\ \ell_3 = 3,\ w_1 = 1,\ w_2 = 3,\ w_3 = 2$. The associated layout costs are $1 \cdot 3 + 2.5 \cdot 1 + 4.5 \cdot 2 = 14.5$.

We summarize the above results as follows: The (COP) is NP-hard and its "hard" part is to determine where the centers of the departments are located with respect to the checkpoint in the optimal solution. In the following section we show that the (COP) with one checkpoint that is neither left- nor right-aligned can be solved by a dynamic programming algorithm and hence is NP-hard in the weak sense. If the number of checkpoints is not part of the input, then the (COP) is NP-hard in the strong sense.

3 A Dynamic Programming Algorithm for the (COP)

We exploit the following two simple properties of (COP) layouts for designing a dynamic programming algorithm:

- In every (COP) layout there is at most one department covering the checkpoint. All other departments lie completely to the left or to the right of the checkpoint.
- In an optimal (COP) layout all departments, except for the department covering the checkpoint, have to be arranged in non-increasing order from the checkpoint to the border of the layout with respect to their relative weights $w_i/\ell_i,\ i \in [n]$. This structural characteristic is denoted as V-shaped property [8, 15].

In order to simplify the description of our dynamic programming algorithm we assume w.l.o.g. that the checkpoint is centered and hence that the sum of the lengths of the departments to the left and to right of the checkpoints has to be equal. Now let us outline the workings of our dynamic programming algorithm in detail.

The input for our algorithm is a (COP) instance, where the list of departments is given in non-decreasing order with respect to their relative weights $w_i/\ell_i,\ i \in [n]$. First we choose one of the given departments as center department that covers the checkpoint, where ℓ_c is the length of the center department, ℓ_c^1 is the length of the part of the center department left to the checkpoint and ℓ_c^2 is the length of the part of the center department right to the checkpoint. Hence $\ell_c = \ell_c^1 + \ell_c^2$ and additionally due to symmetry we can always assume w.l.o.g. that $\ell_c^1 \geq \ell_c^2$ holds.

Next we choose an alignment of the center department above the checkpoint: We start with the right end of the center department above the checkpoint and then shift the center department 0.5 to the right in each iteration until the center of the center department is placed directly above the checkpoint. Clearly it suffices to consider 0.5 shifts of the center department as the checkpoint is centered and $\ell_i \in \mathbb{N},\ i \in [n]$.

In the inner loops we determine with the help of the following recursive relation whether to place the remaining departments to the left or to the right of the checkpoint

$$F_j(s) = \frac{\ell_j w_j}{2} + \min\left\{F_{j-1}(s+\ell_j) + \left(s+\ell_c^1\right) w_j \ ;\ F_{j-1}(s) + \left(M - s + \ell_c^2\right) w_j\right\}, \tag{5}$$

where s indicates the remaining free space to the left of the checkpoint and M gives the overall remaining free space either to the left or to the right of the checkpoint, which is equal to the sum of the lengths of the departments not yet assigned. As we arrange the departments in non-increasing order with respect to their relative weights $w_i/\ell_i,\ i \in [n]$, from the checkpoint to the border, the V-shaped property of the resulting layout is ensured.

We refer to Algorithm 1 for a detailed description of our dynamic programming algorithm that we implemented in C. In Section 5 we computationally compare the dynamic programming algorithm to an integer linear programming approach for the (COP) that we suggest in the following section. In summary our dynamic programming algorithm for solving the (COP) runs in pseudo-polynomial time, to be precise in $O(n^2 \cdot S \cdot \max_{i\in[n]}\{\ell_i\})$ with

$$S = \sum_{i=1}^{n} \ell_i. \tag{6}$$

Hence the (COP) with one checkpoint is NP-hard in the weak sense. If the number of checkpoints is not part of the input, then the (COP) is NP-hard in the strong sense, which can be proven by a deduction from 3-PARTITION.

To further clarify the workings of our dynamic programming algorithm let us consider a toy example with centered checkpoint and the following input data: $\ell_1 = \ell_2 = 2,\ \ell_3 = \ell_4 = 1,\ w_1 = 1,\ w_2 = w_3 = 2,\ w_4 = 4$. The optimal objective value is 9 and the optimal layouts are $(1,3,4,2)$, $(1,4,3,2)$, $(2,3,4,1)$ and $(2,4,3,1)$. All $F_j(\cdot,\cdot)$-values for $j \in [n-1]$ that were determined by our algorithm for this example are stated in Figure 4.

Algorithm 1 `Dynamic Programming Algorithm for the (COP)`

```
1: Input: A (COP) instance $\mathcal{I}$, i.e. a list of departments $\tilde{D} = \{\tilde{d}_i\},\ i \in [n]$, with corresponding lengths $\tilde{L} = \{\tilde{\ell}_i\},\ i \in [n]$, and weights $\tilde{W} = \{\tilde{w}_i\},\ i \in [n]$, where the departments are given in non-decreasing order regarding their relative weights $\tilde{w}_i/\tilde{\ell}_i,\ i \in [n]$.
2: Output: Optimal solution for the given (COP) instance.

3: Determine the gcd $t$ of the elements in $\tilde{L}$ and divide all elements in $\tilde{L}$ by $t$: $\tilde{\ell}_i = \tilde{\ell}_i/t,\ i \in [n]$.
4: Set $S = \sum_{i\in[n]} \tilde{\ell}_i$ and $z^* = \infty$.
5: for $h \in [n]$ do
6:     Set $d_c = \tilde{d}_h$, $\ell_c = \tilde{\ell}_h$, $w_c = \tilde{w}_h,\ D = \tilde{D} \setminus d_c = \{d_i\}$, $L = \tilde{L} \setminus \ell_c = \{\ell_i\}$, $W = \tilde{W} \setminus w_c = \{w_i\},\ i \in [n-1]$.
7:                         ▷ Choose a center department and remove it from the list of departments.
8:     for $i \in [\ell_c + 1]$ do
9:         Set $\ell_c^1 = \frac{2\ell_c - i + 1}{2}$, $\ell_c^2 = \ell_c - \ell_c^1$.
10:        if $\left(S/2 - \ell_c^1\right) \in \mathbb{N}$ then
11:            Set $M = S - \ell_c,\ s = S/2 - \ell_c^1$.
12:            Initialize $F_0\,(s) = (\ell_c^1 - \ell_c^2)w_c/2$ and $F_0\,(a) = \infty$ for all $a \in [M] \cup \{0\}$.
13:            for $j \in [n-1]$ do
14:                Set $M = M - \ell_j$.
15:                for $s$ from 0 to $M$ do
16:                    $F_j(s) = \ell_j w_j/2 + \min\left\{F_{j-1}(s + \ell_j) + \left(s + \ell_c^1\right) w_j\ ;\ F_{j-1}(s) + \left(M - s + \ell_c^2\right) w_j\right\}$.
17: ▷ For a given placement of the departments $\ell_c$ and $[j-1]$, decide with the help of the recursive dynamic programming relation whether Department $j$ should be placed left or right of the checkpoint.
18:                end for
19:            end for
20:            if $F_{n-1}(0) < z^*$ then
21:                Set $z^* = F_{n-1}(0)$ and update the corresponding best layout found.
22:            end if
23:        else there exists no (COP) solution for this alignment of $d_c$.
24:        end if
25:    end for
26: end for
27: return the objective value $t \cdot z^*$ (and the associated layout) of the optimal solution for the instance $\mathcal{I}$.
```

4 An Integer Linear Programming Formulation for the (COP)

In the section we propose an integer linear programming (ILP) approach for solving the (COP). To simplify the notation, we consider the (COP) on $n-1$ departments. First we introduce binary ordering variables $x_{ij},\ i,j \in [n],\ i < j$, with the interpretation

$$x_{ij} = \begin{cases} 1, & \text{if department } i \text{ lies to the left of department } j, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

in order to relate the positions of the $n-1$ departments to each other and to the checkpoint n that is again w.l.o.g. assumed to be centered. To ensure transitivity on the ordering variables we use the 3-cycle inequalities

$$0 \le x_{ij} + x_{jk} - x_{ik} \le 1, \quad i,j,k \in [n],\ i < j < k, \tag{8}$$

that rule out the existence of directed 3-cycles and are sufficient for guaranteeing that there is no directed cycle.

Now the distances of the departments from the checkpoint can be expressed as quadratic terms in ordering variables: For department $i \in [n-1]$, we sum up the lengths of the departments left of i plus $\ell_i/2$ and

s	Departments placed			
	$\{1\}$	$\{1,2\}$	$\{1,2,3\}$	$\{2,3,4\}$
0	∞	∞	10	12
1	1	5	6	
2	∞	∞		
3	∞			
4	∞			

Center department is Department 1 with $\ell_c^1 = 2$.

s	Departments placed			
	$\{1\}$	$\{1,2\}$	$\{1,2,3\}$	$\{1,2,3,4\}$
0	∞	4	9	11
1	∞	∞	9	
2	0	4		
3	∞			
4	∞			

Center department is Department 1 with $\ell_c^1 = 1$.

s	Departments placed			
	$\{2\}$	$\{1,2\}$	$\{1,2,3\}$	$\{1,2,3,4\}$
0	∞	∞	9	11
1	2	4	5	
2	∞	∞		
3	∞			
4	∞			

Center department is Department 2 with $\ell_c^1 = 2$.

s	Departments placed			
	$\{2\}$	$\{1,2\}$	$\{1,2,3\}$	$\{1,2,3,4\}$
0	∞	2	7	13
1	∞	∞	7	
2	0	2		
3	∞			
4	∞			

Center department is Department 2 with $\ell_c^1 = 1$.

s	Departments placed			
	$\{3\}$	$\{1,3\}$	$\{1,2,3\}$	$\{1,2,3,4\}$
0	∞	3	7	9
1	∞	∞	∞	
2	1	3		
3	∞	∞		
4	∞			
5	∞			

Center department is Department 3 with $\ell_c^1 = 1$.

s	Departments placed			
	$\{4\}$	$\{1,4\}$	$\{1,2,4\}$	$\{1,2,3,4\}$
0	∞	4	8	9
1	∞	∞	∞	
2	2	4		
3	∞	∞		
4	∞			
5	∞			

Center department is Department 4 with $\ell_c^1 = 1$.

Figure 4: All $F_j(\cdot,\cdot)$-values for $j \in [n-1]$ that were determined by our dynamic programming algorithm for a toy example with centered checkpoint and input data $\ell_1 = \ell_2 = 2,\ \ell_3 = \ell_4 = 1,\ w_1 = 1,\ w_2 = w_3 = 2,\ w_4 = 4$. The optimal objective value is 9 and the optimal layouts are $(1,3,4,2)$, $(1,4,3,2)$, $(2,3,4,1)$ and $(2,4,3,1)$.

denote it by d_i. Furthermore we compute the position of the checkpoint d_c as $S/2$, i.e. the total length of the departments divided by 2. Then we subtract d_i from d_c. This difference gives the distance of the center of department i from the checkpoint, if department i is located to the left of the checkpoint. If department i is located to the right of the checkpoint, this difference is minus the distance of the center of department i from the checkpoint. Therefore we multiply this difference by the term $(2x_{in} - 1)$ that is 1, if the center of department i lies to the left of the checkpoint and -1 if the center of department i lies to the right of the checkpoint:

$$z_{in} = (2x_{in} - 1)\,(d_c - d_i)\,, \quad i \in [n-1], \tag{9}$$

with

$$d_c = \frac{S}{2}, \qquad d_i = \frac{\ell_i}{2} + \sum_{\substack{j\in[n-1]\\ j<i}} \ell_j x_{ji} + \sum_{\substack{j\in[n-1]\\ j>i}} \ell_j (1 - x_{ij}), \quad i \in [n-1].$$

The additional multiplication with $(2x_{in} - 1)$ ensures a correct calculation of all distances through the following constraints:

$$z_{in} \geq 0, \quad i \in [n-1]. \tag{10}$$

Expanding and simplifying (9) yields

$$z_{in} = (2x_{in} - 1)\left(\frac{S - \ell_i}{2} - \sum_{\substack{j\in[n-1]\\ j<i}} \ell_j x_{ji} - \sum_{\substack{j\in[n-1]\\ j>i}} \ell_j (1 - x_{ij})\right), \quad i \in [n-1]. \tag{11}$$

To model the (COP) as an ILP we apply standard linearization and introduce new variables for all products of ordering variables in (11):

$$y_{inji} = x_{in}x_{ji},\ i,j \in [n-1],\ j < i, \quad y_{inji} = x_{in}(1 - x_{ij}),\ i,j \in [n-1],\ j > i. \tag{12}$$

Note that we have to introduce y_{inji} for all $i \neq j$ as the variable i appears twice in the indices. Now (11) can be further rewritten as:

$$z_{in} = (2x_{in} - 1)\frac{S - \ell_i}{2} + \sum_{\substack{j\in[n-1]\\ j<i}} \ell_j x_{ji} + \sum_{\substack{j\in[n-1]\\ j>i}} \ell_j (1 - x_{ij}) - 2\sum_{\substack{j\in[n-1]\\ j\neq i}} \ell_j y_{inji}, \quad i \in [n-1]. \tag{13}$$

Moreover we use the following standard constraints to relate the orderings variables and their products:

$$\begin{aligned} y_{inji} \le x_{in},\ i \in [n], \quad y_{inji} \le 1 - x_{ij},\ i,j \in [n],\ j > i, \quad y_{inji} \le x_{ji},\ i,j \in [n],\ j < i, \\ y_{inji} \ge x_{in} - x_{ij},\ i,j \in [n],\ j > i, \quad y_{inji} \ge x_{in} + x_{ji} - 1,\ i,j \in [n],\ j < i. \end{aligned} \tag{14}$$

In summary we obtain the following ILP model for the (COP):

$$\begin{aligned} \min \quad & \sum_{i\in[n]} w_i z_{in} \\ \text{s.t.} \quad & (6), (8), (13), (14), \\ & x_{ij} \in \{0,1\}, \quad i,j \in [n],\ i < j, \\ & y_{inji} \in \{0,1\}, \quad i,j \in [n],\ i \neq j. \end{aligned}$$

In the following section we computationally compare an ILP approach based on this model with our dynamic programming algorithm introduced in the previous section.

5 Computational Experiments

We report the results of computational experiments with our dynamic programming algorithm and integer linear programming (ILP) approach respectively. All computations were conducted on an Intel Xeon E5160 (Dual-Core) with 2 GB RAM, running Debian 5.0 in 64-bit mode. The algorithms were implemented in C (dynamic programming algorithm) and Gurobi 6.5 (ILP) respectively. To generate (COP) instances we use layout benchmark instances from the literature by simply choosing one department i as checkpoint and deducing the (COP) weights from the pairwise weights as follows: $w_j = w_{ij},\ j \in [n],\ j \neq i$. In order to test the effect of varying department lengths on our approaches, we generated additional instances by using the same weights but substituting the original department lengths lying in the range of 1 to 20 with random department lengths between 1 and 10000. We add an L to the instance name for indicating the instances with the new random department lengths. All instances considered can be downloaded from `http://tinyurl.com/layoutlib`.

In Tables 1 and 2 we state the results of our ILP approach. For instances with up to 25 departments the ILP succeeds in determining the optimal solution. For instances with a higher number of departments (except from "ste36.5") we observe quite large gaps even after 24h. The performance of the ILP approach is hardly influenced by the department lengths as can be seen by comparing the results in Tables 1 and 2.

In Tables 3 and 4 we state the results of our dynamic programming algorithm that determines the optimal solution for all benchmark instances with short department lengths within seconds. As our dynamic programming algorithm runs in $O(n^2 \cdot S \cdot \max_{i \in [n]}\{\ell_i\})$, it is much slower on the benchmark instances with large department lengths, where it is able to solve instances with up to 33 departments. If the dynamic programming algorithm does not finish within the time limit, then it does not provide a lower bound and hence also no global gap for the optimal solution.

Comparing our two approaches we observe that the dynamic programming algorithm is clearly superior for all benchmark instances from the facility layout literature that only contain departments with short lengths $\ell \leq 100$. For instances with department lengths up to 10000 the dynamic programming algorithm is still preferable although there already exist instances (see e.g. "ste36.5") on which the ILP yields clearly better results. Finally for instances with even larger department lengths the ILP gradually outperforms the dynamic programming algorithm.

6 Conclusion

In this paper we proposed a new variant of a row facility layout problem and two exact algorithms for solving it. The Checkpoint Ordering Problem `(COP)` is weakly NP-hard. It is both of theoretical and practical interest and has several important relations to other well-studied combinatorial optimization problems. In our computational study we showed that the `(COP)` is hard to solve in practice for both dynamic programming and integer linear programming approaches.

It would be interesting to examine if the models, results and algorithms for scheduling on two parallel machines, which is a very well-studied problem, can be used to obtain stronger approximation results and/or to design stronger exact approaches for the `(COP)` in particular and layout problems in general.

Acknowledgments: I gratefully acknowledge constructive comments by two anonymous referees that led to the current version of the paper.

Instance	Lower bound	Upper bound	Gap in %	Time	B&B nodes
P15	189	189	0.0	30	15360
P17	675.5	675.5	0.0	1:59:00	2368737
P18	679.5	679.5	0.0	3:05:00	3241655
H_20	710	710	0.0	5:30:00	4767725
N25_05	368	368	0.0	17:00:00	3885648
H_30	645	1441	55.2	24:00:00	1593134
N30_05	973.5	3224.5	68.8	24:00:00	1514491
Am33_03	600.5	1898.5	86.4	24:00:00	106547
Am35_03	185.5	2132.5	91.3	24:00:00	210111
ste36.5	1444	1444	0.0	1:24:00	38781
N40_5	81	2772	97.1	24:00:00	42135
sko42-5	84	4059	97.9	24:00:00	27352

Table 1: Results obtained by our ILP approach on instances with regular department lengths using Gurobi 6.5 [14] restricted to one thread on our machine with a time limit of 24h. The running times are given in sec or in h:min:sec respectively.

Instance	Lower bound	Upper bound	Gap in %	Time	B&B nodes
P15L	148059.5	148059.5	0.0	1:08	24589
P17L	438935.5	438935.5	0.0	28:25	504935
P18L	589182.5	589182.5	0.0	24:51	225180
H_20L	245666	245666	0.0	2:43:41	1109189
N25_05L	401212.5	555504.5	27.7	24:00:00	3050435
H_30L	389374	754755	48.4	24:00:00	4860433
N30_05L	239436	935762.5	74.4	24:00:00	483913
Am33_03L	138929.5	1094984.5	87.3	24:00:00	322711
Am35_03L	24.5	1645539.5	100	24:00:00	44175
ste36.5L	813222.5	814001.5	0.1	24:00:00	662589
N40_5L	192379	2518555	92.4	24:00:00	535358
sko42-5L	67631	2363015	97.1	24:00:00	411416

Table 2: Results obtained by our ILP approach on instances with large department lengths using Gurobi 6.5 [14] restricted to one thread on our machine with a time limit of 24h. The running times are given in min:sec or in h:min:sec respectively.

Instance	Optimum	Time
P15	189	0.01
P17	675.5	0.02
P18	679.5	0.02
H_20	710	0.15
N25_05	368	0.03
H_30	1439	0.46
N30_05	3191.5	3.70
Am33_03	1879.5	0.55
Am35_03	2116.5	0.32
ste36.5	1444	1.73
N40_5	2747	0.64
sko42-5	3694	6.42

Table 3: Results obtained by our dynamic programming algorithm on instances with regular department lenghts. The running times are given in sec.

Instance	Optimum	Time
P15L	148059.5	47:45
P17L	438935.5	1:54:44
P18L	589182.5	3:58:42
H_20L	245666	1:32:45
N25_05L	554740.5	10:31:00
H_30L	752966	9:58:20
N30_05L	920622.5	16:10:33
Am33_03L	1078710.5	20:51:16
Am35_03L	*	*
ste36.5L	*	*
N40_5L	*	*
sko42-5L	*	*

Table 4: Results obtained by our dynamic programming algorithm on instances with large department lengths. The running times are given in min:sec or in h:min:sec respectively. The entries * indicate that the algorithm did not finish within the time limit of 24h.

References

[1] D. Adolphson. Singlemachine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6:40–54, 1977.

[2] M. Azizoglu and O. Kirca. On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research*, 113(1):91–100, 1999.

[3] H. Belouadah and C. N. Potts. Scheduling identical parallel machines to minimize total weighted completion time. *Discrete Applied Mathematics*, 48(3):201 – 218, 1994.

[4] C. F. Bornstein and S. Vempala. Flow metrics. *Theoretical Computer Science*, 321:13–24, 2004.

[5] M. Charikar, M. T. Hajiaghayi, H. Karloff, and S. Rao. ℓ_2^2 spreading metrics for vertex ordering problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 1018–1027, New York, USA, 2006.

[6] Z.-L. Chen and W. B. Powell. Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, 11(1):78–94, 1999.

[7] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34: 313–356, 2002.

[8] S. Eilon and I. G. Chowdhury. Minimising waiting time variance in the single machine problem. *Management Science*, 23(6):567–575, 1977.

[9] S. E. Elmaghraby and S. H. Park. Scheduling jobs on a number of identical machines. *AIIE Transactions*, 6(1):1–13, 1974.

[10] U. Feige and J. R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Information Processing Letters*, 101:26–29, 2007.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[12] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63. ACM, 1974.

[13] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.

[14] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2016. URL `http://www.gurobi.com`.

[15] N. G. Hall and M. E. Posner. Earliness-tardiness scheduling problems, I: Weighted deviation of completion times about a common due date. *Operations Research*, 39(5):836–846, 1991.

[16] L. H. Harper. Optimal assignments of numbers to vertices. *SIAM Journal on Applied Mathematics*, 12: 131–135, 1964.

[17] L. H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1:385–393, 1966.

[18] S. S. Heragu and A. Kusiak. Machine Layout Problem in Flexible Manufacturing Systems. *Operations Research*, 36(2):258–268, 1988.

[19] P. Hungerländer. A comparison of global optimization approaches for row facility layout problems. Technical report, Alpen-Adria Universität Klagenfurt, Mathematics, Optimization Group, TR-AAUK-M-O-16-07b, 2017.

[20] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[21] R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 278–285, New York, USA, 1993. ACM.

[22] E. L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16(1):77 – 84, 1969.

[23] E. L. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*, pages 445–522. Elsevier, 1993.

[24] C.-Y. Lee and R. Uzsoy. A new dynamic programming algorithm for the parallel machines total weighted completion time problem. *Operations Research Letters*, 11(2):73–75, 1992.

[25] J. K. Lenstra, A. H. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[26] G. Mitchison and R. Durbin. Optimal numberings of an N N array. *SIAM Journal on Algebraic and Discrete Methods*, 7:571–582, 1986.

[27] J.-C. Picard and M. Queyranne. On the one-dimensional space allocation problem. *Operations Research*, 29(2):371–391, 1981.

[28] M. Queyranne and A. S. Schulz. Polyhedral approaches to machine scheduling. Technical report, Working paper, 2004.

[29] S. Rao and A. W. Richa. New approximation techniques for some linear ordering problems. *SIAM Journal on Computing*, 34:388–404, 2005.

[30] R. Ravi, A. Agrawal, and P. Klein. Ordering problems approximated: Single-processor scheduling and interval graphs connection. In J. L. Albert, B. R. Artalejo, and B. Monien, editors, *18th International Colloquium on Automata, Languages and Programming*, volume 150 of *Lecture Notes in Computer Science*, pages 751–762. Springer-Verlag New York, 1991.

[31] M. H. Rothkopf. Scheduling independent tasks on parallel processors. *Management Science*, 12(5): 437–447, 1966.

[32] S. C. Sarin, S. Ahn, and A. B. Bishop. An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research*, 26(7):1183–1191, 1988.

[33] D. M. Simmons. One-dimensional space allocation: An ordering algorithm. *Operations Research*, 17: 812–826, 1969.

[34] W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3 (1-2):59–66, 1956.

[35] J. Suryanarayanan, B. Golden, and Q. Wang. A new heuristic for the linear placement problem. *Computers & Operations Research*, 18(3):255–262, 1991.

[36] T. G. Szymanski. Assembling code for machines with span-dependent instructions. *Communications of the ACM*, 21:300–308, 1978.

[37] A. Vanelli and G. S. Rowan. An eigenvector based approach for multistack VLSI layout. In *Proceedings of the Midwest Symposium on Circuits and Systems*, volume 29, pages 135–139, 1986.

[38] J. Xu and R. Nagi. Identical parallel machine scheduling to minimise makespan and total weighted completion time: a column generation approach. *International Journal of Production Research*, 51 (23–24):7091–7104, 2013.