The Multiple Checkpoint Ordering Problem

Philipp Hungerländer[*] Kerstin Maier[†]

November 19, 2017

Abstract

The multiple Checkpoint Ordering Problem (`mCOP`) aims to find an optimal arrangement of n one-dimensional departments with given lengths such that the total weighted sum of their distances to m given checkpoints is minimized. In this paper we suggest an integer linear programming (`ILP`) approach and a dynamic programming (`DP`) algorithm, which is only exact for one checkpoint, for solving the `mCOP`. Our computational experiments show that there is no clear winner between the two methods. While the `ILP` approach is hardly influenced by increasing the number of checkpoints or the length of the departments, the performance of our `DP` algorithm deteriorates in both cases.

Keywords: Combinatorial optimization, dynamic programming, integer linear programming, row layout problems.

1 Introduction

In this paper we introduce and analyze the multiple Checkpoint Ordering Problem (`mCOP`), which is a new variant of a row layout problem. An instance of the `mCOP` consists of n one-dimensional departments $D := \{1, 2, \dots, n\}$ with given positive lengths $\ell_1, \dots, \ell_n$, m checkpoints $C := \{n+1, \dots, n+m\}$ with given positions and pairwise weights $w_{ij},\ i \in D,\ j \in C$. We are looking for a non-overlapping placement of the departments without gaps between them, where the weighted sum of the distances of all departments to all checkpoints is minimal. The corresponding optimization problem can be written down as

$$\min_{\pi \in \Pi_n} \sum_{i \in D, j \in C} w_{ij} z_{ij}^{\pi},$$

where Π_n is the set of permutations of the departments D and z_{ij}^{π} is the distance between the center of Department i and Checkpoint j with respect to a particular permutation $\pi \in \Pi_n$.

[*]Alpen-Adria Universität Klagenfurt, Austria, philipp.hungerlaender@aau.at
[†]Alpen-Adria Universität Klagenfurt, Austria, kerstin.maier@aau.at

Hungerländer [3] introduced and analyzed the Checkpoint Ordering Problem (`COP`), which is the special case of the `mCOP` with $m = 1$. The `COP` is weakly NP-hard, while the complexity of the `mCOP` is still open. The `mCOP` has connections to other combinatorial optimization problems like the Single-Row Facility Layout Problem (`SRFLP`) and scheduling on m identical parallel machines with the objective of minimizing the sum of weighted completion times 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 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 $\sum_{j \in \mathcal{J}} w_j C_j$. In the standard classification scheme of Graham et al. [2] 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. The key differences between the `mCOP` and the weakly NP-hard [5] $Pm||\sum w_j C_j$ are the following ones:

1. For the `mCOP` the sum of the lengths of the departments that are placed to the left and to the right of the checkpoints are predetermined through the positions of the checkpoints as there are no spaces allowed between the departments. E.g. for the `COP` with a centered checkpoint the sums of the lengths of the departments to the left and to the right of the checkpoint have to be equal. Contrary to that for the $Pm||\sum w_j C_j$ there are typically no capacity restrictions imposed on the machines.
2. The checkpoints must not lie exactly at a splitting point of two departments, but they can also be covered by departments. I.e. the checkpoints do not necessarily define a partition of the departments. When considering a scheduling set-up, the `COP` can be described as follows: We are given two machines and it is allowed to split an arbitrary job into two parts at any point and then the two parts have to be scheduled first on the two machines. The `mCOP` with $m \geq 2$ cannot be formulated in this scheduling set-up anymore because the distances of the departments to all checkpoints are relevant in the objective.

Due to these differences it is not possible to directly carry over complexity and polyhedral results, dynamic programming (`DP`) algorithms and integer linear programming (`ILP`) models and their corresponding approximation results from scheduling on identical parallel machines [4] to the `mCOP`.

In this paper we propose two solution approaches for the `mCOP` and compare them in a computational study, where we observe that the `mCOP` seems much harder to solve in practice than the related `SRFLP` and $Pm||\sum w_j C_j$. There is no clear winner between our two methods. While the `ILP` approach is hardly influenced by the department lengths and number of checkpoints considered, the performance of the `DP` algorithm, which is only exact for one checkpoint, deteriorates for increasing department lengths and an increasing number of checkpoints.

The paper is structured as follows. In Sections 2 and 3 we suggest an `ILP` approach and a `DP` algorithm for solving the `mCOP` and in Section 4 we conduct computational experiments, indicating the practical applicability and limitations of the approaches suggested. For future research it would be interesting to design more sophisticated exact approaches and heuristics for the mCOP. Furthermore it is still an open question if the mCOP is weakly or strongly NP-hard.

2 An ILP Formulation for the mCOP

In this section we propose an integer linear programming (ILP) approach for solving the mCOP with an arbitrary but fixed number of checkpoints that is a generalization of the ILP for the COP suggested in [3]. First we define S as the sum of the lengths of all departments

$$S = \sum_{i \in D} \ell_i. \tag{1}$$

The locations of the m checkpoints are defined by $p_j \in [0,1],\ j \in C$, where $S \cdot p_j$ gives the position of Checkpoint j.

Next we introduce binary ordering variables $x_{ij},\ i \in D,\ j \in D \cup C,\ i < j$,

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

to relate the positions of the n departments to each other and to the m checkpoints. To ensure transitivity on these variables, we use the 3-cycle inequalities

$$0 \le x_{ij} + x_{jk} - x_{ik} \le 1, \quad i,j \in D,\ k \in D \cup C,\ i < j < k, \tag{2}$$

which are sufficient for guaranteeing that there is no directed cycle.

Now we are able to express the distances of the departments from the m checkpoints as quadratic terms in ordering variables. The position d_i of the center of Department $i \in D$ is given as the sum of the lengths of the departments left of i plus $\ell_i/2$. The difference $d_j - d_i,\ i \in D,\ j \in C$, gives the distance of the center of Department i to Checkpoint j, 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 Checkpoint j. Therefore we multiply $d_j - d_i,\ i \in D,\ j \in C$, by the term $(2x_{ij} - 1)$ that is 1, if the center of Department i lies to the left of Checkpoint j and -1 otherwise:

$$z_{ij} = (2x_{ij} - 1)\left(d_j - d_i\right),\ i \in D,\ j \in C, \tag{3}$$
$$d_i = \frac{\ell_i}{2} + \sum_{k \in D,\ k<i} \ell_k x_{ki} + \sum_{k \in D,\ k>i} \ell_k (1 - x_{ik}),\ i \in D, \quad d_j = S \cdot p_j,\ j \in C,$$

Expanding and simplifying (3) yields

$$z_{ij} = (2x_{ij} - 1)\Big(S \cdot p_j - \frac{\ell_i}{2} - \sum_{\substack{k \in D \\ k<i}} \ell_k x_{ki} - \sum_{\substack{k \in D \\ k>i}} \ell_k (1 - x_{ik})\Big),\ i \in D,\ j \in C. \tag{4}$$

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

$$z_{ij} \geq 0, \quad i \in D,\ j \in C. \tag{5}$$

To model the `mCOP` as an `ILP`, we apply standard linearization and introduce new variables for all products of ordering variables in (4):

$$y_{ijki} = x_{ij}(1 - x_{ik}),\ i < k, \qquad y_{ijki} = x_{ij}x_{ki},\ i > k,$$

where $i, k \in D,\ j \in C$. Now (4) can be further rewritten as:

$$\begin{aligned} z_{ij} =& (2x_{ij} - 1)\left(S \cdot p_j - \frac{\ell_i}{2}\right) + \sum_{\substack{k \in D \\ k<i}} \ell_k x_{ki} \\ &+ \sum_{\substack{k \in D \\ k>i}} \ell_k (1 - x_{ik}) - 2 \sum_{\substack{k \in D \\ k \neq i}} \ell_k y_{ijki}, \quad i \in D,\ j \in C. \end{aligned} \tag{6}$$

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

$$\begin{aligned} &y_{ijki} \leq x_{ij}, \qquad y_{ijki} \leq 1 - x_{ik},\ i < k, \qquad y_{ijki} \leq x_{ki},\ i > k, \\ &y_{ijki} \geq x_{ij} - x_{ik},\ i < k, \qquad\qquad y_{ijki} \geq x_{ij} + x_{ki} - 1,\ i > k, \end{aligned} \tag{7}$$

where $i, k \in D,\ j \in C$. Overall we obtain the following `ILP` model for the `mCOP`:

$$\begin{aligned} \min \quad & \sum_{i \in D,\ j \in C} w_{ij} z_{ij} \\ \text{s.t.} \quad & (1),\ (2),\ (5) - (7), \\ & x_{ij} \in \{0,1\}, \quad i \in D,\ j \in D \cup C,\ i < j, \\ & y_{ijki} \in \{0,1\}, \quad i, k \in D,\ j \in C,\ i \neq j. \end{aligned}$$

3 A Dynamic Programming Algorithm for the `mCOP`

In [3] an exact dynamic programming (`DP`) algorithm for solving the `COP` was proposed. In this section we suggest how to extend this algorithm to the `mCOP`. As our extension is not exact for $m \geq 2$, it is still an open question if the `mCOP` with $m \geq 2$ is weakly or strongly NP-hard. Note that $Pm||\sum w_j C_j$ is weakly NP-hard as it can be solved in pseudopolynomial time by a `DP` approach [5].

Now let us give a brief outline of our `DP` algorithm. In an optimal layout departments that are positioned to the left or to the right of *all* checkpoints adhere to the well-known V-shaped property [1], i.e. they are arranged in non-increasing order from the leftmost or rightmost checkpoint to the border of the layout with respect to their relative weights $\left(\sum_{j \in C} w_{ij}\right)/\ell_i,\ i \in D$. Contrary to that departments with a high relative weight that are located between two checkpoints should not necessarily be positioned close to a checkpoint. This is why we arrange departments between two checkpoints k and $k+1$ in non-increasing order regarding the ratio $\left(\sum_{j=n+1}^{k} w_{ij}\right) / \left(\sum_{j=k+1}^{n+m} w_{ij}\right),\ i \in D$. Unfortunately the obtained arrangements may not be optimal for $m \geq 2$. In fact the approach is not even guaranteed to find a feasible solution. Nonetheless our `DP` algorithm proves to be a good heuristic for the `mCOP`, in particular if m is small.

At the heart of our approach lies a recursive relation that is used to decide where Department j should be placed with respect to the checkpoints. For one checkpoint the recursion tells us whether to assign Department j to the left or to the right of the checkpoint:

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

where s indicates the remaining free space to the left of the checkpoint, M gives the overall remaining free space either to the left or to the right of the checkpoint, c is the center department covering the checkpoint and ℓ_c^1 (ℓ_c^2) is the length of the part of the center department left (right) to the checkpoint.

A detailed description of our `DP` algorithm, including in particular a discussion of the general recursive relation for $m \geq 2$ checkpoints, is omitted in this short paper due to space limitations and will be provided in a forthcoming paper.

4 Computational Experiments

All experiments were performed on a Linux 64-bit machine equipped with Intel(R) Xeon(R) CPU e5-2630 v3@2.40GHz and 128 GB RAM. The algorithms were implemented in C (`DP`) and Gurobi 6.5 (`ILP`) respectively. To generate `mCOP` instances, we utilized benchmark instances from row layout literature by simply randomly choosing $m+n$ departments from these instances and using them as our n departments and m checkpoints. Accordingly we took the corresponding pairwise connectivities in these instances as our `mCOP` weights $w_{ij},\ i \in D,\ j \in C$.

In our computational study we consider two different instance sets from the literature. `AnKeVa80` consists of 80 departments with department lengths between 1 and 60. `HuRe40` contains 40 departments with department lengths between 1 and 10. Each of our `mCOP` instances consists of 10-30 departments and has 4 checkpoints. We choose the checkpoint positions dependent on the number of checkpoints considered, but independent from the number of departments. All instances can be downloaded from http://tinyurl.com/layoutlib.

In Tables 1 and 2 we respectively state the results of our `ILP` approach and our `DP` algorithm. We observe that the `mCOP` with $m \geq 2$ is already very hard to solve to optimality for instances of moderate size. In particular the `mCOP` seems much harder to solve in practice than the closely related strongly NP-hard Single-Row Facility Layout Problem and the weakly NP-hard $Pm||\sum w_j C_j$.

The performance of the ILP approach is hardly influenced by increasing the number of checkpoints or the length of the departments. Contrary to that the performance of our DP algorithm deteriorates both for an increasing number of checkpoints and for larger department lengths. Note that on the AnKeVa80 instances the DP algorithm does not provide any *feasible* solution when considering all 4 checkpoints. Nonetheless for each number of checkpoints there are instances for which the DP algorithm provides better solutions than the ILP approach.

References

[1] S. Eilon and I. G. Chowdhury: Minimising Waiting Time Variance in the Single Machine Problem. *Management Science*, 23(6):567–575, 1977.

[2] 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 (eds.) *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier 1979.

[3] P. Hungerländer: The checkpoint ordering problem. *Optimization*, 1–14, 2017.

[4] M. Queyranne and A.S. Schulz: *Polyhedral approaches to machine scheduling.* Technical report, 2004.

[5] M.H. Rothkopf: Scheduling Independent Tasks on Parallel Processors. *Management Science*, 12(5):437–447, 1966.

# Checkpoints	1			2		
Instance	Best solution	Gap [%]	Time	Best solution	Gap [%]	Time
AnKeVa80set4dep15	2042.00	0.0	00:10:21	7280.50	0.0	00:53:25
AnKeVa80set4dep20	2280.50	20.3	12:00:00	10802.00	34.6	12:00:00
AnKeVa80set4dep25	3663.00	60.9	12:00:00	19273.00	65.6	12:00:00
AnKeVa80set4dep30	5594.00	96.2	12:00:00	25929.00	96.3	12:00:00
HuRe40set4dep15	259.50	0.0	00:01:31	894.75	0.0	00:06:22
HuRe40set4dep20	303.00	0.0	00:02:56	1169.50	0.0	01:53:40
HuRe40set4dep25	324.50	0.0	00:05:15	2084.50	15.9	12:00:00
HuRe40set4dep30	1116.00	37.9	12:00:00	3529.75	64.2	12:00:00

# Checkpoints	3			4		
Instance	Best solution	Gap [%]	Time	Best solution	Gap [%]	Time
AnKeVa80set4dep15	7662.25	0.0	05:01:40	976.20	0.0	07:20:14
AnKeVa80set4dep20	14017.00	61.7	12:00:00	22402.50	73.2	12:00:00
AnKeVa80set4dep25	25523.50	78.1	12:00:00	34161.00	86.7	12:00:00
AnKeVa80set4dep30	34290.00	99.9	12:00:00	44552.50	97.4	12:00:00
HuRe40set4dep15	1176.50	0.0	00:35:10	1718.50	0.0	02:17:54
HuRe40set4dep20	1486.50	27.3	12:00:00	3757.30	53.4	12:00:00
HuRe40set4dep25	2685.50	44.1	12:00:00	4655.80	81.2	12:00:00
HuRe40set4dep30	3952.50	97.3	12:00:00	9037.90	94.5	12:00:00

Table 1: Results obtained by our `ILP` approach using Gurobi 6.5 restricted to one thread with a time limit of 12h. The running times are given in hh:mm:ss.

# Checkpoints	1			2		
Instance	Best solution	DP vs. ILP	Time	Best solution	DP vs. ILP	Time
AnKeVa80set4dep15	2280.50	0.00	00:00:02	7280.50	0.00	12:00:00
AnKeVa80set4dep20	2042.00	-10.46	00:00:01	10899.00	0.89	12:00:00
AnKeVa80set4dep25	3663.50	0.00	00:00:06	20482.00	6.27	12:00:00
AnKeVa80set4dep30	5584.00	-0.17	00:00:08	25885.00	-0.17	12:00:00
HuRe40set4dep15	259.50	0.00	00:00:01	894.75	0.00	00:00:14
HuRe40set4dep20	304.00	0.00	00:00:01	1187.50	1.54	00:02:22
HuRe40set4dep25	324.50	0.00	00:00:01	2118.50	1.66	00:07:26
HuRe40set4dep30	1115.00	-0.09	00:00:01	3544.25	0.41	00:20:44

# Checkpoints	3			4		
Instance	Best solution	DP vs. ILP	Time	Best solution	DP vs. ILP	Time
AnKeVa80set4dep15	8651.75	12.91	12:00:00	-	-	12:00:00
AnKeVa80set4dep20	15419.00	10.00	12:00:00	-	-	12:00:00
AnKeVa80set4dep25	-	-	12:00:00	-	-	12:00:00
AnKeVa80set4dep30	-	-	12:00:00	-	-	12:00:00
HuRe40set4dep15	1176.50	0.00	01:08:56	1719.50	0.06	12:00:00
HuRe40set4dep20	1517.50	2.09	12:00:00	4074.90	8.45	12:00:00
HuRe40set4dep25	2681.50	-0.01	12:00:00	5226.60	12.26	12:00:00
HuRe40set4dep30	4022.50	1.77	12:00:00	8859.30	-1.98	12:00:00

Table 2: Results obtained by our `DP` algorithm with a time limit of 12h. The running times are given in hh:mm:ss. In column `DP` *vs.* `ILP` we compute $\frac{\text{Solution of DP - Solution of ILP}}{\text{Solution of DP}}$, hence in case of a negative entry the `DP` gave a better solution than the `ILP`.