Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-13 (2019) 511–516

Scheduling for last-mile meal-delivery processes

Matteo Cosmi [*] Gaia Nicosia [*] Andrea Pacifici [**]

[*] *Dipartimento di Ingegneria, Università degli Studi "Roma Tre", via della Vasca Navale 79, 00146 Rome, Italy (e-mail: {matteo.cosmi, gaia.nicosia}@uniroma3.it)*
[**] *Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università degli Studi di Roma "Tor Vergata", Via del Politecnico 1, 00133 Rome, Italy (e-mail: andrea.pacifici@uniroma2.it)*

Abstract: We address a single machine scheduling problem arising in a last mile delivery setting for a food company. The same problem finds obvious applications also in the context of internal manufacturing logistics. A set of food orders are placed by the customers and are to be fulfilled by the company. Each order comprises a delivery point and an ideal delivery time An order is considered on time if it is delivered within a certain given time interval around the ideal delivery time. All food is prepared in a single production facility (restaurant) and immediately carried to the customers by a single courier, who may dispatch one or two different orders in a single trip. Since late deliveries correspond to canceled orders and an economic loss for the company, it is of interest to schedule orders so that the number of late orders is minimized. We model the resulting decision problem as a special single machine scheduling problem and propose different mixed integer programs to solve it. Their performance is assessed through a computational study on a set of test instances derived by our real-world application.

Keywords: Scheduling, Integer Programming, Optimization Problems, Combinatorial Mathematics, Internal Logistics, Food Delivery.

1. INTRODUCTION

The logistic sector has always been a source of demanding problems and applications for automation and optimization. In the last few years this sector has had an incredible transformation due to the recent spreading of e-commerce, which in turn has also been reflected in the exponential growth of the food delivery sector as shown by the spreading of companies like GrubHub, Just-Eat or Deliveroo. This increasing interest for the food delivery logistic is also witnessed by the large number of works published and/or submitted in the last year: See, e.g., Cosmi et al. (2018); Ozbaygin and Savelsbergh (2018); Reyes et al. (2018); Steever et al. (2018); Yildiz and Savelsbergh (2018).

In this paper, we address a scheduling problem motivated by an application for food pick-up and delivery in an urban area. The company provides food transportation services from restaurants to final customers who specify the restaurant to order at, their location (i.e., the food destination), and the ideal delivery time, besides, of course, the type and amount of required dishes. Orders are shipped by a number of couriers who may dispatch multiple orders within the same working shift but can only deliver a limited number of orders (usually, one) in a single restaurant-to-client trip. In this context, typical decisions concern the (*i*) assignment to couriers and (*ii*) scheduling of orders so that the maximum delay with respect to the desired delivery time, or the number of late—and, possibly, withdrawn—orders are minimized. Cosmi et al. (2018) present a novel approach, based on a time-expanded network, capable to solve instances with several restaurants and customers in an amount of time compatible with an online application.

Here, we focus on the *single-courier single-restaurant* case, which gives rise to a special single-machine scheduling problem. Besides being an interesting problem in itself, solving the latter one can also be exploited as a component routine in more general procedures for the original (multi-courier multi-restaurant) problem, or integrating the corresponding mathematical program within a simulation framework (see, e.g., Alfieri et al. (2015)).

The paper is organized as follows. In Sections 2 and 3 we rigorously describe the addressed problem and propose an model that considers possible aggregation of orders in pairs served in a single trip of the courier. We then present four different Integer Linear Programming models (Section 4) and compare their performance through the results of an extended computational campaign illustrated in Section 5. Finally, in Section 6 some conclusions are drawn.

2. PROBLEM DESCRIPTION

In the addressed problem, there is a single restaurant, where the food is prepared, and a given set of meal-orders $J = \{1, \dots, n\}$, with their ideal delivery times d'_j and restaurant-destination travel times t_j, for all $j \in J$.

The restaurant may be regarded as a depot from which the courier starts when he is shipping, say, the j-th order, and where he will go back immediately after the order has been delivered to the customer. It is assumed that the selected courier, once he has gathered the food at the restaurant, will immediately deliver the food without any idle waiting time. Therefore, the actual delivery time is given by the pick-up time at the restaurant plus the travel time between the restaurant and the destination. Moreover, an order is considered on time if it is delivered in an interval of δ minutes centered around the ideal time chosen by the customer. (In the considered application $\delta = 30$ minutes which means that an early or late delivery is accepted if it is not earlier or later than 15 minutes.) The courier is considered a single moving resource and he is not available for processing any other order $i \neq j$ until he will be back again to the restaurant.

In conclusion, we may view the orders as jobs/tasks and the single courier as a processing resource in a scheduling system where each task $j \in J$ is associated to the following data.

The *due date* d_j is the latest possible time for the courier to be back to the restaurant after delivering the j-th order on time to the customer and it is therefore $d_j = d'_j + \frac{1}{2}\delta + t_j$. The *release date* r_j is the earliest possible time for the courier to start from the restaurant and deliver the j-th order on time to the customer. It can be written as $r_j = d'_j - \frac{1}{2}\delta - t_j$. The *processing time* p_j represents the amount of time the courier is busy with order j and, neglecting possible waiting times at the restaurantand at the customer, it is given by the total travel time $p_j = 2t_j$ of the courier from the restaurant to the corresponding customer destination and back to the restaurant. (We assume the restaurant schedules the required preparations so that the food is ready for immediate pick-up by the courier, for any order $j \in J$.)

In this framework, an order j is on time if and only if the corresponding task starts (i.e., the courier picks the food at the restaurant) not before r_j and completes (i.e., the courier returns back to the restaurant and he is available for another trip) not later than d_j. Finding a sequence of orders to minimize the number of late deliveries is a special case of the single machine scheduling problem with (arbitrary) release dates and due dates $1|r_j|\sum U_j$. The latter problem is, in general, strongly $\mathcal{NP}$-hard (see, e.g., Garey and Johnson (1979)). Algorithms to address this problem (or its relaxations) have been presented in, e.g., Dauzère-Pérès (1995); Baptiste et al. (2003); Dauzère-Pérès and Sevaux (2004). A relevant application is presented in, e.g., Adacher and Flamini (2014).

In our problem, as a consequence of the above definitions, we have the relation:

$$d_j = r_j + p_j + \delta \tag{1}$$

which makes our scheduling problem a special case of $1|r_j|\sum U_j$, since due dates and release dates are interdependent. We denote our problem as $1|r_j, \delta|\sum_j U_j$. In the literature, δ is the so-called *slack* and the problems where $d_j \leq r_j + p_j + \delta$ (a relaxation of Equation (1)) are often referred to as *slack-interval* scheduling problems. In this context, Cieliebak et al. (2004) seek a schedule that minimizes the number of machines necessary to complete all jobs on time. The authors show that, recurring to a suitable model on interval graphs, their problem can be solved in polynomial time if $\delta \in \{0, 1\}$. Otherwise, they propose a solution algorithm that runs in $O(n(\delta + 1)^H H \log H)$, where n and H are the number of jobs and overlapping intervals, respectively. A refined algorithm for the *feasibility* version of the same problem is proposed in van Bevern et al. (2017). Its cost is $O(n\delta m \log(\delta m)(\delta + 1)^{m(2\delta+1)} + n \log n)$, where m is the number of available machines. As shown in the above papers, the problem is already $\mathcal{NP}$-hard when $\delta = 2$ for arbitrary m, while it is tractable for fixed parameter $m + \delta$. The complexity for the special case $m = 1$ and $\delta > 1$ is still an open question.

3. ORDER AGGREGATION

We consider the problem in which the courier may not only pick one order at a time, but he is also allowed to serve two orders together, in order to save time. In this case, the deliveries to the customers are performed in a single trip from the restaurant to two different locations and back. We look again at this composite service as a special task (in the following equivalently referred to as a *twin order* or a *twin task* or, simply, a *twin*). Hereafter, we suppose that after the first delivery the courier immediately proceeds to consign the second order of a twin:

No-wait assumption: When processing a twin, the courier is not allowed to introduce idle time between the two deliveries (and possibly wait for the second order not to arrive too early).

Clearly, a courier may fulfil two orders i and j in a single trip (and hence, tasks i and j are allowed to be in the same twin) only if it is possible to meet the corresponding delivery-time constraints under the no-wait assumption. In this case, the twin composed by the ordered pair of tasks (i, j) (hereafter, for notation simplicity, indicated by ij) is called a *feasible twin*, so that it may be regarded as a single aggregated task. If t_{ij} denotes the travel time between clients i and j, the following proposition holds.

Proposition 1. Given two orders $i, j \in J \times J$, the twin ij is feasible if and only if:

$$\delta_{ij} \overset{\text{def}}{=} \delta - |t_{ij} - d_j + d_i + 1/2(p_j - p_i)| \geq 0 \tag{2}$$

Proof. If a courier starts the trip associated to twin order ij at time θ, then, in order to have an on-time twin, the following must hold:

$$\begin{aligned}
&\theta \geq r_i \\
&\theta + 2t_i \leq d_i \\
&\theta + t_i + t_{ij} \geq r_j - t_j \\
&\theta + t_i + t_{ij} + t_j \leq d_j
\end{aligned}$$

where the first two inequalities guarantee that the first order i is on time, while the last two take into account that the delivery point of order j, due to the no-wait assumption, is reached after $t_i + t_{ij}$ time units.

Recalling that, for a task $j \in J$, the processing time p_j equals twice the travel time from the restaurant to the delivery point of order j and that Equation (1) holds, the thesis follows. □

We denote the set of feasible twins by

$$D = \{ij \ : \ (i, j) \in J \times J, \ \delta_{ij} \geq 0\}. \tag{3}$$

For notational convenience, we include in D, the special symbol jj (for any $j \in J$) in order to represent a single task j as a twin (but not aggregated to another task). Clearly, $jj \in D$ for all $j \in J$.

Naturally extending the concept of on-time order, we say that a twin order $ij \in D$ is on time if both its component are so. Obviously, feasibility condition (2) is only necessary for a twin to be on time.

We are now in the position to define a release date r_{ij} and a due date d_{ij} associated to any feasible twin task $ij \in D$ so that, if the starting time of twin ij belongs to the interval $[r_{ij}, r_{ij} + \delta_{ij}]$, then ij is on time.

The above mentioned parameters can be computed as shown below, depending on the sign of the quantity:

$$\tau_{ij} \overset{\text{def}}{=} t_{ij} - (d_j - d_i) - \frac{1}{2}(p_i - p_j).$$

With some algebra, it is not hard to show that the release date of a twin ij can be expressed as

$$r_{ij} \overset{\text{def}}{=} d_i - \delta - p_i \; (= r_i) \qquad \text{if } \tau_{ij} > 0 \tag{4}$$

$$r_{ij} \overset{\text{def}}{=} d_j - \delta - \frac{1}{2}(p_j + p_i) - t_{ij} \qquad \text{if } \tau_{ij} \leq 0. \tag{5}$$

Due to the no-wait assumption, the amount of time in which the courier is busy with a twin ij is given by the the overall travel time:

$$p_{ij} \overset{\text{def}}{=} t_{ij} + t_i + t_j = t_{ij} + \frac{1}{2}(p_i + p_j). \tag{6}$$

As a consequence of Equations (4) and (6), the definition of the due date of a twin ij follows:

$$d_{ij} \overset{\text{def}}{=} r_{ij} + \delta_{ij} + p_{ij}. \tag{7}$$

Observe that, if $\tau_{ij} > 0$, then $d_{ij} = d_j$.

4. OPTIMIZATION MODELS

Hereafter, we propose four different ILP models that we have developed to solve the above described scheduling problem. The first two programs fall in the class of Natural-date/Sequencing Models, while the third model is a positional type of ILP and the fourth is a time-indexed formulation based on "pulse" start variables (Artigues et al. (2015)).

Our objective function aims at minimizing the number of late tasks, so in the objective in the programs below we use binary variables y_{ij} (representing whether a twin order is late or not) multiplied by a weight w_{ij} that takes into account if the task corresponds to a twin order or not. More precisely, for each $ij \in D$, we set $w_{ij} = 2$ if $i \neq j$, i.e. if ij is a twin order, and $w_{ij} = 1$ if $i = j$. Hence, the definition of w_{ij} represents the fact when a twin order is late, we consider as late both of its orders. Note that this assumption can be made since, whenever there exists a schedule σ containing a twin order with only one order on time (while the other one is late), it is possible to prove that such schedule is always dominated by another schedule $\bar{\sigma}$ where the twin order is split into two single orders, one on time and one late, and the value of the objective function is such that $f(\bar{\sigma}) \leq f(\sigma)$.

4.1 ILP1: Twin Sequencing Model

In our first model, ILP1, besides the y_{ij} variables introduced above, we use as decision variables:

- $s_{ij} \in \mathbb{R}_+$, representing the starting time of the pair of tasks ij;
- $z_{ij}^{hk} \in \{0, 1\}$, encoding whether hk immediately precedes ij.

Moreover, we introduce two additional "dummy" orders, that correspond to the first and the last scheduled tasks. We call them α and ω, respectively, and we consider also the twin orders (α, α) and (ω, ω).

ILP1 is as follows:

$$\min \sum_{ij \in D} w_{ij} y_{ij} \tag{8}$$

$$\text{s.t.} \quad s_{ij} \geq r_{ij} \qquad ij \in D \tag{9}$$

$$s_{ij} \geq s_{hk} + p_{hk} - (1 - z_{ij}^{hk})M \qquad hk, ij \in D \tag{10}$$

$$s_{ij} + p_{ij} \leq d_{ij} + My_{ij} \qquad ij \in D \tag{11}$$

$$\sum_{hk \in D} (z_{ii}^{hk} + \sum_{j: ij \in D} z_{ij}^{hk} + \sum_{j: ji \in D} z_{ji}^{hk}) = 1 \qquad i \in J \cup \{\omega\} \tag{12}$$

$$\sum_{hk \in D} (z_{hk}^{ii} + \sum_{j: ij \in D} z_{hk}^{ij} + \sum_{j: ji \in D} z_{hk}^{ji}) \leq 1 \qquad i \in J \cup \{\alpha\} \tag{13}$$

$$\sum_{hk \in D} (z_{ij}^{hk} - z_{hk}^{ij}) = 0 \qquad ij \in D \tag{14}$$

$$\sum_{hk \in D} z_{\alpha\alpha}^{h,k} = 0 \tag{15}$$

$$y_{ij}, s_{ij} \in \{0, 1\} \qquad ij \in D \tag{16}$$

$$z_{ij}^{hk} \in \{0, 1\} \qquad ij, hk \in D \tag{17}$$

Constraints (9)-(10) impose that a twin ij starts after its predecessor has been completed and after its release time r_{ij}. If it is not possible to complete a twin ij before its delivery time, constraints (11) set $y_{ij} = 1$ counting the delay in the objective function. Constraints (12)-(13) force each order $j \in J$ to have exactly one predecessor and successor (they also impose that α has to have one successor and ω one predecessor). In addition, if hk is the predecessor of ij, then ij is the successor for hk, so if $z_{ij}^{hk} = 1$ then we must have $z_{hk}^{ij} = 1$ and this is imposed by constraints (14). Finally, constraint (15) prevents the fake task α from having a predecessor.

4.2 ILP2: Task Sequencing Model

The model below is quite similar to the one described above. The main difference lies in the starting-time and precedence variables that, in this model, always refer to single tasks. Moreover, here we consider task i as a predecessor of j if the former comes before the latter in the schedule.

We define the following decision variables:

- $s_i \in \mathbb{R}_+$, representing the starting time of task i;
- $x_{ij} \in \{0, 1\}$, encoding whether task i precedes task j;
- $\beta_{ij} \in \{0, 1\}$, indicating whether tasks i and j are scheduled as a twin order ij.

Formulation ILP2 is illustrated below.

$$\min \sum_{ij\in D} w_{ij}y_{ij} \tag{18}$$

$$\begin{aligned}
s.t.\quad & s_i \geq \sum_{ij\in D} r_{ij}\beta_{ij} && i\in J && (19)\\
& s_j \geq c_i - M(\beta_{ij} - x_{ij} + 1) && i,j\in J : i\neq j && (20)\\
& s_i \geq c_j - M(\beta_{ij} + x_{ij}) && i,j\in J : i\neq j && (21)\\
& s_j \geq s_i - M(1-\beta_{ij}) && ij\in D && (22)\\
& s_i + p_{ij} \leq d_{ij} + M(y_{ij} + 1 - \beta_{i,j}) && ij\in D && (23)\\
& s_j + p_{ij} \leq d_{ij} + M(y_{ij} + 1 - \beta_{i,j}) && ij\in D && (24)\\
& c_j = s_j + \sum_{i:ij\in D} p_{ij}\beta_{ij} && j\in J && (25)\\
& x_{ij} + x_{ji} = 1 && ij : i\neq j\in D && (26)\\
& x_{jh} \geq x_{ih} - M(1-\beta_{ij}) && \begin{array}{c} ij\in D\\ h\in J : h\neq i,j\end{array} && (27)\\
& \sum_{j:ij\in D} \beta_{ij} \leq 1 && i\in J && (28)\\
& \sum_{i:ij\in D} \beta_{ij} \leq 1 && j\in J && (29)\\
& \sum_{j:ij\in D} \beta_{ij} + \sum_{h:hi\in D, h\neq i} \beta_{hi} = 1 && i\in J && (30)\\
& s_i \in \mathbb{R}_+ && i\in J && (31)\\
& \beta_{ij}, x_{ij}, y_{ij} \in \{0,1\} && ij\in D && (32)
\end{aligned}$$

Constraints (28)-(30) impose that each task i must always be associated and hence scheduled within a twin order (comprising order (i,i)). Constraints (19) force each task i to always start after the release time of the twin order which it is associated to. Constraints (20)-(21) are standard disjunctive constraints imposing precedences between each pair of tasks i and j not belonging to the same twin order. These constraints do not hold if i is scheduled as single task, whereas, in the latter case, constraint (22) holds. Each single task has to be completed before its twin order delivery time otherwise the twin order is considered late (23)-(24). Constraints (26) impose that if i precedes j then it is not possible that j precedes i. Constraint (27) sets that if i and j are scheduled in the twin order ij then if i precedes h also j has to precede h.

4.3 ILP3: Positional Model

The third ILP model that we propose is a Positional Model where the main variables are binary variables x_{jh} that indicate whether task j is scheduled in position h and a variable stating if in posistion h there is as single task or a twin order. By using these variables it is possible to derive, for each task scheduled in position h, its completion time and, hence, whether it is late or not.

For brevity we avoid describing the whole formulation and limit ourselves to present the experimental results concerning ILP3 in Section 5.

4.4 ILP4: Time-indexed Model

Hereafter, we illustrate a time-indexed formulation for our problem which is based on binary variables x^t_{ij}, such that $x^t_{ij} = 1$ if and only if the pair ij starts at time t (this type of variables are often referred to as pulse variables). We also define a set of tasks $D^\theta = \{ij \in D \text{ s.t. } r_{ij} \leq \theta\}$ containing only tasks having a release time not greater than θ. As in the above models, we also use y_{ij} binary variables to keep track of the number of the late tasks.

$$\min \sum_{ij\in D} w_{ij}y_{ij} \tag{33}$$

$$\begin{aligned}
s.t.\quad & \sum_{t=r_{ij}}^{T} t x^t_{ij} + p_{ij} \leq d_{ij} + My_{ij} && ij\in D && (34)\\
& \sum_{t=1}^{T} \sum_{hk\in D^t : i\in\{h,k\}} x^t_{hk} = 1 && i\in J && (35)\\
& \sum_{\substack{hk\in D\\ hk\neq ij}} \sum_{t=\theta}^{\min\{T,\theta+p_{ij}-1\}} x^t_{hk} \leq n(1-x^\theta_{ij}) && \begin{array}{c} ij\in D\\ \theta = 1,2,\dots T\end{array} && (36)\\
& x^t_{ij}, y_{ij} \in \{0,1\} && \begin{array}{c} ij\in D,\\ t = 1,2,\dots T\end{array} && (37)
\end{aligned}$$

Constraints (34) impose that each twin order which is completed after its due date is considered late. Constraints (35) guarantee that each task i is scheduled in a twin order ik (or hi) starting after its release time r_{ik} (or r_{hi}). Constraints (36) imply that if an order ij starts at time θ, then no other task can start in the succeeding p_{ij} time periods.

5. COMPUTATIONAL RESULTS

Hereafter we describe the results of a computational campaign aimed at comparing the quality of the four proposed ILPs. The tests were run on instances derived from real world scenarios in which the number of orders n varies from a minimum of 5 to a maximum of 31. The considered time horizon is roughly a courier shift, 4 hours, and corresponds, in all instances, to the busiest hours of the day, namely those spanning over dinner time. Instances refer to a number of different restaurants and relate to different days of the year, therefore they may vary in geographical and temporal distribution of orders.

Models were built using the JuMP package (Dunning et al. (2017)) for Julia and solved by Gurobi (ver. 8.0.0). All tests were performed on a computer equipped with an Intel Xeon E5-2643v3 3.40 GhZ CPU and 32 GB RAM.

In all computational experiments we set a time limit of 300 seconds (an instance not solved within this limit is declared unsolved). In Table 1 we report on the number of instances solved within the time limit for each ILP (columns 3-6). Instances are grouped so that in each class the instances have the same number n of orders (reported in column 1). Note that, since all the instances are derived from a real world application, we do not have the same numbers of instances in each class (see column 2).

These results show that the Task Sequencing model, ILP2, performs better than the other ILPs solving more than 92% of the considered instances. Regarding computation times, ILP2 appears to be a quite efficient model: it is able to solve each instance in less than 5 seconds on the average, see Tables 2 and 3. Table 2 also shows that ILP2 is the fastest model in terms of average number of explored

Table 1. Number of optimally solved instances

n	# Inst	Twin Seq	Task Seq	Posit	Time-Ind
5	1	1	1	1	1
7	2	2	2	2	2
9	2	2	2	2	2
10	113	113	113	113	108
11	91	91	91	91	88
12	63	63	63	63	57
13	42	40	42	38	39
14	27	22	27	17	23
15	25	17	25	8	22
16	15	7	14	5	12
17	20	12	20	8	15
18	18	5	16	3	7
19	12	4	11	3	4
20	12	1	8	1	0
21	5	1	3	0	0
22	8	1	4	1	1
23	3	0	1	0	0
24	2	0	0	0	0
25	2	0	0	0	0
26	4	0	1	0	0
27	5	0	0	0	0
29	2	0	0	0	0
30	2	0	0	0	0
31	2	0	0	0	0
Overall	478	382	444	356	381

nodes. Moreover, ILP2 takes less than 10% of the time needed by the Time-Indexed formulation, less than 37.5% of the Positional model and 64% of ILP1. Looking at Table 3, we may also observe that for instances with less than 17 orders, ILP2 solves 99.73% of the available instances in an average time lower than 3 seconds. Even when the number of orders is between 17 and 20 this model is able to solve 88.71% of the instances on an average time of 53 seconds. When the size of the problem increases to over 21 orders, ILP2 becomes less effective finding the optimal solution in less than 5 minutes only in the 25.71% of the tests. Table 3 reports also maximum, minimum and average time spent by ILP1 to find the optimal solution. (The acronym TL means that no optimal solution was found within the time limit.)

Table 2. Computation times and number of explored nodes (averages)

Formulation	Time (s)	Explored Nodes [1]
Twin sequencing	7.704	2331
Task sequencing	4.923	482
Positional	13.128	16126
Time-Indexed	56.755	3576

In order to find out how effective is order aggregation, we have compared the solutions obtained in two settings: the first one permitting order aggregation and the others in which twin orders are not allowed (i.e. when there are only single tasks). As illustrated by Figure 1, the experiments show that:

- In 224 out of 444 (i.e., 50.45% of the) optimally solved instances, order aggregation reduces the number of late tasks (compared to the single-task setting) and the average reduction in the objective function value is around 61%.
- The reduction in the objective caused by order aggregation becomes more relevant together with the size of the instances (i.e. number of tasks).

[1] This column reports the average number of explored nodes for those instances optimally solved by all models.

Table 3. Computation times (sec) of Twin (ILP1) and Task Sequencing (ILP2) models

	Twin Sequencing			Task Sequencing		
n	T_{max}	T_{min}	T_{mean}	T_{max}	T_{min}	T_{mean}
5	0.004	0.004	0.004	0.004	0.004	0.004
7	0.069	0.022	0.045	0.062	0.004	0.033
9	0.308	0.161	0.234	0.053	0.023	0.038
10	225.843	0.0170	2.686	1.388	0.014	0.119
11	91.533	0.0250	3.540	1.398	0.018	0.158
12	66.241	0.099	2.650	2.169	0.020	0.284
13	TL	0.138	5.881	4.526	0.025	0.581
14	TL	0.319	7.004	17.773	0.035	1.564
15	TL	1.350	39.046	8.943	0.044	2.139
16	TL	1.136	31.335	TL	0.078	1.774
17	TL	1.319	18.462	183.590	0.079	20.302
18	TL	29.085	60.099	TL	0.081	25.716
19	TL	9.043	31.231	TL	0.145	22.467
20	TL	81.851	81.851	TL	0.277	52.894
21	TL	64.300	64.300	TL	0.938	113.741
22	TL	84.064	84.064	TL	0.409	27.074
23	TL	-	-	TL	15.130	15.130
24	TL	-	-	TL	-	-
25	TL	-	-	TL	-	-
26	TL	-	-	TL	42.434	42.434
27	TL	-	-	TL	-	-
29	TL	-	-	TL	-	-
30	TL	-	-	TL	-	-
31	TL	-	-	TL	-	-

Finally, it is also worth to mention that in 28 over 34 (i.e., 96.55% of the) instances the best (sub-optimal) solution obtained in the order aggregation case, within the 300 seconds time limit, improves upon the *optimal* solutions found with the standard single-task model. This proves that order aggregation is effective compared to the single delivery model, even when the system imposes short computation times for finding feasible solutions.

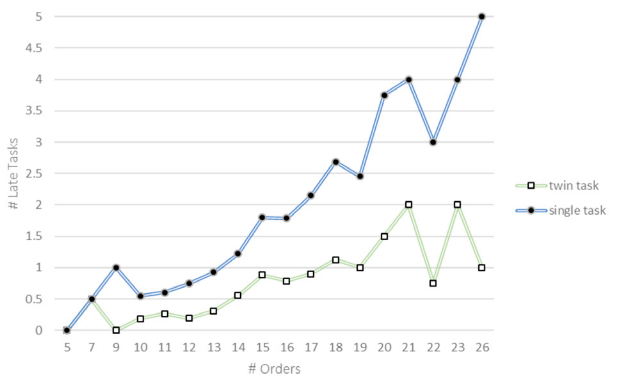

Fig. 1. Order aggregation vs. Single-task model: Objective values comparison

6. CONCLUSIONS

We address a single-restaurant single-courier last-mile-delivery scheduling problem in which the objective is the maximization of the orders on time. To improve the quality of the solutions we consider the possibility of merging two orders, so that a single courier can deliver one or two meals in each trip.

The results are promising. In particular, they show that serving more than one customer in a single trip considerably reduces the number of late tasks (and this is also true for the instances not solved to optimality). This means that solving the problem with order aggregation is a matter of interest to improve the quality of service for a restaurant or a food delivery company.

The experiments also suggest that the classical sequencing models (ILP1 and ILP2) appear to be the most effective ones, immediately followed by the Time-Indexed formulation. It would be interesting to compare the performance of the proposed ILPs to other integer programs, such as, for instance packing formulation as in Agnetis et al. (2009) and/or a combinatorial branch and bound able to exploit some non-trivial lower bounds which may be derived with arguments similar to those illustrated in Nicosia and Pacifici (2017).

Additional promising directions of research may include (but are not limited to): (*i*) Investigating polynomially solvable sub-cases possibly induced by bounding the number of possible travel lengths (like, e.g., in Detti (2008)). (*ii*) The design of a branch and price ad-hoc algorithm using either the time indexed formulation or a packing formulation. (*iii*) Studying a last mile delivery problem in which the courier works for more than one delivery company. In this case, the problem becomes a multi-agent scheduling problem where different agents compete for the usage of a single machine (see e.g. Nicosia et al. (2018); Agnetis et al. (2013, 2015); Marini et al. (2013)).

REFERENCES

Adacher, L. and Flamini, M. (2014). Aircraft ground routing and scheduling optimization. In *UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, UKSim 2014*, 45–350.

Agnetis, A., Alfieri, A., and Nicosia, G. (2009). Single-machine scheduling problems with generalized preemption. *INFORMS Journal on Computing*, 21(1), 1–12.

Agnetis, A., Nicosia, G., Pacifici, A., and Pferschy, U. (2013). Two agents competing for a shared machine. In P. Perny, M. Pirlot, and A. Tsoukiàs (eds.), *Algorithmic Decision Theory*, 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg.

Agnetis, A., Nicosia, G., Pacifici, A., and Pferschy, U. (2015). Scheduling two agent task chains with a central selection mechanism. *Journal of Scheduling*, 18(3), 243–261.

Alfieri, A., Matta, A., and Pedrielli, G. (2015). Mathematical programming models for joint simulation–optimization applied to closed queueing networks. *Annals of Operations Research*, 231(1), 105–127.

Artigues, C., Koné, O., Lopez, P., and Mongeau, M. (2015). Mixed-integer linear programming formulations. In C. Schwindt and J. Zimmermann (eds.), *Handbook on Project Management and Scheduling Vol.1*, 17–41. Springer International Publishing.

Baptiste, P., Peridy, L., and Pinson, E. (2003). A branch and bound to minimize the number of late jobs on a single machine with release time constraints. *European Journal of Operational Research*, 144(1), 1 – 11.

Cieliebak, M., Erlebach, T., Hennecke, F., Weber, B., and Widmayer, P. (2004). Scheduling with release times and deadlines on a minimum number of machines. In J.J. Levy, E.W. Mayr, and J.C. Mitchell (eds.), *Exploring New Frontiers of Theoretical Informatics*, 209–222. Springer US, Boston, MA.

Cosmi, M., Oriolo, G., Piccialli, V., Terranova, S., and Ventura, P. (2018). Last-mile delivery for a food company. In *EURO/ALIO 2018, Bologna, Italy, June 2018.*

Dauzère-Pérès, S. (1995). Minimizing late jobs in the general one machine scheduling problem. *European Journal of Operational Research*, 81, 134–142.

Dauzère-Pérès, S. and Sevaux, M. (2004). An exact method to minimize the number of tardy jobs in single machine scheduling. *Journal of Scheduling*, 7(6), 405–420.

Detti, P. (2008). Algorithms for multiprocessor scheduling with two job lengths and allocation restrictions. *Journal of Scheduling*, 11(3), 205–212.

Dunning, I., Huchette, J., and Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2), 295–320.

Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

Marini, C., Nicosia, G., Pacifici, A., and Pferschy, U. (2013). Strategies in competing subset selection. *Annals of Operations Research*, 207(1), 181–200.

Nicosia, G. and Pacifici, A. (2017). Scheduling assembly tasks with caterpillar precedence constraints on dedicated machines. *International Journal of Production Research*, 55(6), 1680–1691.

Nicosia, G., Pacifici, A., and Pferschy, U. (2018). Competitive multi-agent scheduling with an iterative selection rule. *4OR*, 16(1), 15–29.

Ozbaygin, G. and Savelsbergh, M. (2018). An iterative re-optimizing framework for the dynamic vehicle routing problem with roaming delivery locations. *Optimization Online*. URL `http://www.optimization-online.org/DB_HTML/2018/08/6784.html`.

Reyes, D., Erera, A., Savelsbergh, M., Sahasrabudhe, S., and O'Neil, R. (2018). The meal delivery routing problem. *Optimization-Online*. URL `http://www.optimization-online.org/DB_HTML/2018/04/6571.html`.

Steever, Z., Karwan, M.H., and Murray, C.C. (2018). Dynamic Courier Routing for a Food Delivery Service. *Optimization Online*. URL `http://www.optimization-online.org/DB_HTML/2018/10/6864.html`.

van Bevern, R., Niedermeier, R., and Suchý, O. (2017). A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack. *Journal of Scheduling*, 20(3), 255–265.

Yildiz, B. and Savelsbergh, M. (2018). Provably high-quality solutions for the meal delivery routing problem. *Optimization Online*. URL `http://www.optimization-online.org/DB_HTML/2018/05/6624.html`.