Competitive solutions for a dock assignment problem by means of Lagrangian relaxation

Lotte Berghman and Roel Leus

Department of Decision Sciences and Information Management
KULeuven, Leuven, Belgium
{Lotte.Berghman ; Roel.Leus}@econ.kuleuven.be

1 Introduction

Toyota is one of the world's largest automobile manufacturers, selling over 7.5 million models (including Hino and Daihatsu) annually on all five continents, and generating almost 130 billion euro in net revenues. Since 1999, the total warehouse space floor of the European Distribution Centre TPCE (Toyota Parts Centre Europe) located in Diest (Belgium) has been expanded to 67.700 m^2. Toyota's Distribution Centre delivers to 28 European distributors on a daily basis.

At TPCE, the warehouse has some 50 gates, each with a capacity of one trailer, where goods can either be loaded on an empty trailer or be unloaded from a loaded trailer. Besides the warehouse with the gates, the site also contains two parking lots, which can be seen as a buffer where trailers can be temporarily parked. All transportation activities of uncoupled trailers between these parking lots and the gates are done by terminal tractors, which are tractors designed for use in ports, terminals and heavy industry. Because of the numerous loading, unloading and transportation activities, TPCE needs a schedule specifying the starting time and the assigned gate or terminal tractor for each activity.

Each arriving trailer, for which the planned arrival time is known, is dropped off by the trucker at a parking lot and afterwards transported to a gate by the terminal tractor. After unloading or loading at the gate, the trailer is transported back to the parking lot by the terminal tractor, where it will be picked up by a trucker later on. The planning is currently done manually. The goal of this paper is to examine opportunities for automation of this procedure.

We model the situation at TPCE as a three-stage flexible flow shop. The facilities are disjunctive, in the sense that each machine may process at most one task at a time. Preemption of a task is not allowed and none of the machines has buffer storage for work-in-process. The first stage consists in the movement of the trailer by a terminal tractor from the parking lot to a gate, the second stage are the loading and unloading tasks and the third stage is the transportation by a tractor back to the parking lot. Identical terminal tractors execute both the first and the third stage. In these stages, the processing times are modeled as being independent of the driving distance because the actual driving time of the tractor is small compared to the time it takes the driver to follow the safety instructions and attach the trailer to the tractor. Therefore, the processing times in the first and the third stage are non-zero constants. The second stage is executed by a batch of identical machines (representing the gates), so the processing time depends only on the job and is independent of the machine. Each task of stage two has to be scheduled on exactly one gate. Each task of stage one and three is to be scheduled on exactly one terminal tractor.

The gate assigned to a trailer is considered to be occupied also during the transportation stages one and three, mainly for safety reasons. Consequently, unlike a standard flexible flow shop, the 'gate'-resources are not exclusively tied to only one stage. After loading or unloading, a trailer cannot immediately be transported to the parking lot if both tractors are busy. The trailer remains at the gate until a terminal tractor becomes available, which may prevent other trailers from being loaded or unloaded there. We refer to this phenomenon as *blocking*.

A detailed problem statement is given in the next section. A mathematical formulation of the problem is provided in Section 3. The representation and the generation of a schedule is discussed in Section 4. The algorithm presented in Section 5 is based on Lagrangian relaxation and makes use of the formulation of Section 3.

2 Definitions and detailed problem statement

T is the set of all tasks (also referred to as activities); J is the set of jobs (or trailers), with $|J| = n$. Each job $j \in J$ is a vector (t_1, t_2, t_3) of three tasks, one at each stage (the first component is the task in the first stage, etc.). T can be partitioned as follows: $T = T^1 \cup T^2 \cup T^3$ with T^i the tasks of stage i $(i = 1, 2, 3)$. A second partition is $T = T_U \cup T_L$, where the set T_U contains all tasks related to a trailer that has to be unloaded, while T_L gathers all the tasks pertaining to a trailer to be loaded.

Each task $t \in T^1$ has a ready time r_t. For the unloading tasks $t \in T_U$, this ready time equals the planned arrival time of the trailer. For the loading tasks $t \in T_L$, $r_t = 0$ because it is assumed that all the goods to be loaded on the trailers are in the warehouse and that the empty trailer is waiting on the parking lot. Each stage-two unloading task $t \in T_U \cap T^2$ has a due date d_t equal to the ready time and each three-stage loading task $t \in T_L \cap T^3$ has a deadline $\overline{d}_t$ based on the driving time to the customer and the agreed arrival time at the customer. Each of the tasks in these two sets also has a weight w_t representing the importance of early processing of the trailer. All transportation activities between the parking lot and the gates have a constant duration of 1, independent of the distance. There are τ identical terminal tractors available for both the first and the third stage, and $m < n$ identical gates constitute the resources in the second stage. Each machine (either a gate or a tractor) can process at most one task at a time. The processing time p_t of a task $t \in T^2$ is the time needed to load or unload the trailer at the gate.

Our problem consists in scheduling the tasks in such a way that the total weighted lateness (or tardiness, since $d_t = r_t$) of unloading the trailers is minimized and the total weighted earliness of the transportation activities of a loaded trailer back to the parking space, is maximized. In this way, all incoming shipments are in the warehouse as early as possible and all export trailers are ready for transport by a trucker as quickly as possible.

3 Mathematical formulation

We describe a time-indexed formulation, with H the planning horizon with length $H_{\max}$. For all tasks $t \in T$ and time periods $u \in H_t$, the binary variable $x_{tu} = 1$ if task t starts in time period u, $= 0$ otherwise, with H_t the time window of task t.

$$\min \sum_{t \in T_U \cap T^2} w_t \left(\left(\sum_{u \in H_t} u x_{tu} \right) + p_t - d_t \right) + \sum_{t \in T_L \cap T^3} w_t \left(\left(\sum_{u \in H_t} u x_{tu} \right) + 1 - \overline{d}_t \right)$$

subject to

$$\sum\nolimits_{u \in H_t} x_{tu} = 1 \qquad \forall t \in T \tag{1}$$

$$\sum\nolimits_{(t_1,t_2,t_3) \in J} \left(x_{t_1 u} + x_{t_3 u} + \sum\nolimits_{v \le u} (x_{t_2 v} - x_{t_3 v}) \right) \le m \qquad \forall u \in H \tag{2}$$

$$\sum\nolimits_{(t_1,t_2,t_3) \in J} (x_{t_1 u} + x_{t_3 u}) \le \tau \qquad \forall u \in H \tag{3}$$

$$x_{t_2,u+1} = x_{t_1u} \qquad \forall(t_1,t_2,t_3) \in J; \forall u \in H \quad (4)$$
$$\sum_{v=u+p_{t_2}}^{H_{\max}} x_{t_3v} = x_{t_2u} \qquad \forall(t_1,t_2,t_3) \in J; \forall u \in H \quad (5)$$

The objective function minimizes the weighted tardiness of the stage-two unloading tasks and maximizes the weighted earliness of the stage-three loading tasks. The tardiness is the time between the completion of the unloading and the due date, while the earliness is the time between the arrival at the parking lot and the deadline. The first constraint set in the formulation requires each task to be processed exactly once, either on a gate or by a terminal tractor. The constraints (2) ensure that in each time interval at most m activities are executed. A gate is considered to be occupied during transportation, loading or unloading, and blocking, where the latter refers to the time period between the end of stage two and the start of stage three. Constraints (3) enforce the capacity of the terminal tractors. Finally, constraints (4) and (5) implement the precedence constraints between the three stages.

4 Schedule representation and schedule generation schemes

In line with most improvement heuristics for scheduling problems, we will not operate directly on a schedule but rather on some representation of a schedule that is efficient and effective for the functioning of the algorithm. After an operation on a solution (a schedule's representation), the new solution is transformed into a schedule by means of a *schedule generation scheme*. Our schedule representation is an *activity list* (a permutation of the tasks). As a task of stage two starts immediately after the corresponding task of stage one (see constraints (4)), the tasks of stage two are not included in the list. We opt for a so-called *serial* schedule generation scheme (see Hartmann and Kolisch (2000) for details). This type of scheme can always generate an optimal schedule for a resource-constrained scheduling problem when a regular measure of performance is considered (which is also the case in this paper).

There are two particularities to our setting that hamper a straightforward scan of the set of activity lists. The first is the difficulty of producing a *feasible* schedule; this is further discussed in the next section. The second inconvenience is the blocking: as long as the stage-three activity for a trailer is not executed, the assigned gate is occupied, although stage two may already be completed. We call a permutation *valid* if it respects the inter-stage precedences and if the generation scheme is able to find a free gate at each iteration in which a stage-two task has to be planned, which means (informally) that the stage-two and stage-three activities of each job should not be too far apart in the list.

5 The algorithm

When the capacity constraints (2) and (3) are relaxed using Lagrange multipliers (see, e.g., Fisher (1981)), easily solvable independent job-level subproblems are obtained. The multipliers act as prices that regulate the use of the machines. For each task, the optimal starting time strikes a balance between these machine prices and the tardiness or earliness of the job. The remaining scheduling problem is solved in a running time that is linear in the number of jobs and the length of the planning horizon. The relaxed problem is solved multiple times, and at each iteration the multipliers are updated by means of either subgradient optimization or a dedicated multiplier adjustment procedure. After a predetermined number of iterations or when the gap reaches a threshold, the optimization is halted and we obtain a relaxed solution, which is usually not a feasible schedule. This solution should subsequently be made feasible.

Verifying the existence of a feasible schedule for a set of tasks with release times and deadlines is NP-complete in the strong sense, even on a single processor (see Garey and Johnson (1977)). Consequently, finding a feasible schedule for the considered flexible flow-shop problem is also NP-hard.

The two-phase heuristic approach based on list scheduling and pairwise exchange that is presented by Tang and Xuan (2006) allows to "derive a feasible solution in most cases". We have adapted their procedure to our setting by rendering each permutation valid before schedule generation, and it turns out that frequently, a feasible solution is not obtained – in a number of instances, the algorithm enters into an infinite loop by a stepwise positioning earlier in the list of tasks that do not meet their deadline.

We have therefore decided to proceed with the development of enumeration schemes for activity lists, in search for a feasible solution. We construct a search tree in which each node is equated with a permutation of the activities (only those of stages one and three). The root node of the tree is based on the Lagrangian solution. We propose to use the number of adjacent swaps that are needed to transform an activity list into another one as a distance measure between the two lists. The distance between a parent node and its child nodes in the search tree is always one, so there is only one adjacent swap needed to transform a parent into each of its children, and the search procedure gradually moves away from the starting solution as it explores higher-indexed levels of the tree. The quality of each node is measured by the weighted number of jobs that is not finished before its deadline and the total time over all jobs by which the deadlines are violated. This value is used to assess the 'degree' of infeasibility of each solution. Our tree traversal strategy is best-first. The search procedure is interrupted once a feasible schedule is found, although it might in principle be continued in search for higher-quality feasible solutions.

We implement two enumeration schemes. In the first scheme, we only work with valid permutations (the nodes containing invalid combinations are not generated), while both valid and invalid permutations are examined in the second enumeration scheme. Appropriate pruning rules are incorporated in order to avoid redundancy. Multiple implementation variants are tested: the starting solution can also be a random permutation or a permutation that is derived from the formulation's LP relaxation, and (especially for random initial solutions), a depth-first search strategy can also be expected to exhibit a good performance.

References

Fisher, M.L., 1981, "The Lagrangian relaxation method for solving integer programming problems", *Management Science*, Vol. 27, pp. 1–18.

Garey, M.R. and D.S. Johnson, 1977, "Two-processor scheduling with start-times and deadlines", *SIAM Journal on Computing*, Vol. 6, pp. 416–426.

Hartmann, R. and S. Kolisch, 2000, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 127, pp. 394-407.

Tang, L. and H. Xuan, 2006, "Lagrangian relaxation algorithms for real-time hybrid flowshop scheduling with finite intermediate buffers", *Journal of the Operational Research Society*, Vol. 57, pp. 316–324.