SINGLE FIXED CRANE OPTIMISATION WITHIN A DISTRIBUTION CENTRE

J. Matthews[1] & **S.E. Visagie**[2*]

[1,2] Department of Logistics
University of Stellenbosch, South Africa
[1]14855054@sun.ac.za, [2]svisagie@sun.ac.za

ABSTRACT

This paper considersthe optimisation of the movement of a fixed crane operating in a single aisle of a distribution centre. The crane must move pallets in inventory between docking bays, storage locations, and picking lines. Both a static and a dynamic approach to the problem are presented. The optimisation is performed by means of tabu search, ant colony metaheuristics,and hybrids of these two methods. All these solution approaches were tested on real life data obtained from an operational distribution centre. Results indicate that the hybrid methods outperform the other approaches.

OPSOMMING

Die optimisering van die beweging van 'n vaste hyskraan in 'n enkele gang van 'n distribusiesentrum word in hierdie artikel beskou. Die hyskraan moet pallette vervoer tussen dokhokke, stoorposisies, en opmaaklyne. Beide 'n statiese en 'n dinamiese benadering tot die probleem word aangebied. Die optimisering word gedoen met behulp van tabu-soektogte, mierkolonieoptimisering,en hibriede van hierdie twee metodes. Al die oplossingsbenaderings is getoets met werklike data wat van 'n operasionele distribusiesentrum verkry is. Die resultate toon aan dat die hibriedmetodes die beste oplossings lewer.

* Corresponding author

1. INTRODUCTION

Distribution centres (DCs) play an important role in the supply chain. DCs may be viewed as a connecting hub between manufacturers, suppliers, and retailers, and may come in varying forms intended to deal with different types of and areas in supply chains. Because DCs playa vital role in the supply chain, it is important to analyse and optimise their internal processes. In this paper the operation of moving goods within a retail DC is considered.

The operations in this DC mainly deal with orders that may be viewed as requests by retail stores for a selection of distinct goods inspecific quantities. A central operation within the DC is order picking,ensuring that goods received are reorganised into batches destined for retail stores. Order picking is the most cost-intensive and time-consuming operation within the DC [7]. The order-pick operation usually requires the manual unpacking and re-packing of individual stock keeping units (SKUs) from larger cartons into smaller cartons to meet the demand of the retail stores. This operation may rely on the building of a picking line –a set of SKU types arranged on a designated floor space to be picked and re-packed. Other main operations in the DC include receiving, storage, and distribution of goods. The physical movement of pallets and/or cartons is central to all these operations.

The equipment currentlyused to move pallets in the DC under consideration is pump trollies, fork lifts, and fixed cranes. The cranes are fixed,mounted to the roof and floor of the DC. The main focus of this paper is on the moving of pallets (also called jobs) by these cranes. The objective is to determine a job sequence of a crane that minimises the total completion time of all these jobs. The integration of jobs from different operations,i.e. order picking and storage, are also considered and compared with the DC'scurrent method.

The DC has a static layout.A schematic representation of the floor plan is shown in Figure 1. Order picking is handled by picking lines contained at certain levels or floors within the storage racks (see Figure2).

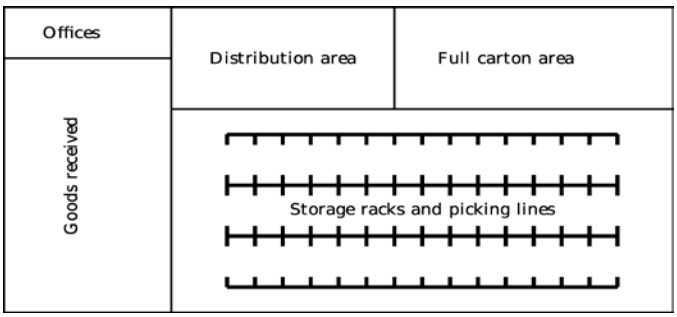

Figure 1: A schematic representation of the layout (floor plan) of the DC.

Most of the internal deadlines and time constraints relate to setting up and completingthe picking lines. Picking lines are active for 8 hours a day.However, the DC runs on a 24-hour basis,usingnight shifts to build and dismantle picking lines and to storenew stock.

The picking lines are positioned below the storage racks (as shown in Figure 2). Cranes are able to move vertically and horizontally. Each half of the picking line can be servedby a single crane. In order to get pallets from the floor into storage (levels 5-10), pallets need to be placed in a docking bay at the base of the storage aisles. Manual pump trolleys are required to load pallets onto the docking racks. Pallets can also be moved between aisles by means of these pump trolleys.

The DC'scrane system is responsible for all the movement of the pallets above the floor level. These crane movements are mainly to store goods and to build and dismantlepicking lines.

The current method of the DC is to focus all the efforts of a crane on one of three activities: building a picking line (moving pallets from storage racks to picking lines), dismantling a picking line (moving pallets from picking lines to storage racks), or storing a batch of received pallets (moving pallets from docking bays to storage racks). The DC's current method requires a crane to complete a small set of jobs – typically 30 – conforming to a single activity until that set is complete.The crane driver uses his own discretion to determine the order in which the jobs within the set are completed. This implies that there is limited or no integration between the jobs from different activities, and that the optimisation within each set is limited to the choices of the crane driver. For example, if 20 jobs on a picking line need to be completed before the picking can begin, and 20 pallets need to be stored from the goods received floor,the crane would either first complete all 20 jobs for the picking line and then complete the storage jobs, or *vice versa*– not a combination of both.

The rest of this paper is structured as follows. Two modelling approaches are introduced to optimise crane movement in the DC. In Section 2 we introduce a static modelling approach, where all the jobs that need to be performed are known in advance. This approach is generalised in Section 3 to add new jobsdynamically. Results from real life instances of the problem are presented in Section 4. The paper end in Section5 with a discussion of the results and with recommendations flowing from the results.

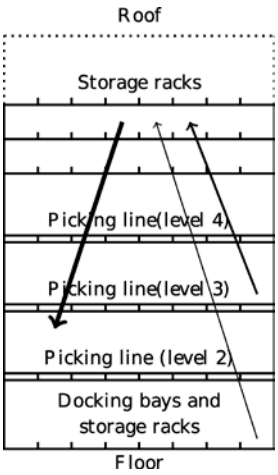

Figure 2: A schematic representation of the crane movement associated with building a picking line, dismantling a picking line, and storing received pallets, resulting in a building job (thick), a breaking job (medium), and a storage job (thin) respectively.

2. STATIC MODELLING APPROACH

The problem under consideration may be viewed as a dynamic version of an ordering problem with side constraints [5], or as a sequential ordering problem with deadlines (SOPD) [1]. The term 'dynamic' implies that the available information changes over time. To solve the dynamic model, a static version of the problem is considered each time a new batch of jobs enters the system.

2.1 Assumptions

For modelling purposes the following assumptions were made in consultation with the DC'smanagement [3]:

1. The availability and speed of the pump trolleys allow pallets to be loaded onto docking bays at such a high rate that cranes do not have to wait for pump trolleys to deliver a pallet to the docking bay. It is thus assumed that any job has starting and ending positions only within a storage rack, picking line, or docking bay.
2. Four types of jobs are considered:

- *storage job*: Pallets that are moved from a docking bay to a storage rack.
- *building job*: Pallets that are moved from a storage rack into a picking line.
- *breakdown job*: Pallets that are removed from a picking line (after all the picking has been completed) and placed back into a storage rack.
- *replenishment job*: Pallets that are moved from a storage rack into a picking line with picking still in progress.

Figure 2 illustrates the movements needed to perform these jobs. A replenishment job requires the same crane movements as a building job.

3. The travelling times of cranes are deterministic. Travelling times were measured and a formula was developed for estimating the expected time needed for a crane to move between loading bays. The physical loading and off-loading times are added to the travel times.
4. A job may have at most two direct prerequisites. The first possible prerequisite is the job that creates the open storage bay at the destination, and the second is the job that first places the required pallet at the starting destination. These prerequisites occur most often when a picking line is built or broken. For testing purposes these prerequisites are determined before optimisation starts.
5. Every job has a strict deadline. For the metaheuristic approachesa job may be completed after the deadline, although this is heavily penalised. This exception is included because, in the dynamic problem, deadlines may be broken through the inclusion of new jobs and delays that occur in real time – for example, when a crane breaks down.

2.2 A mixed-integer programming formulation

A static sequential ordering problem (SOP) may be defined as a set of n jobs that need to be processed by a machine. Each job requires a specific amount of machine time, and a release and completion time are associated with it. Certain precedences occur between different jobs, as well as a set-up cost when changing between two jobs. The objective is to find a sequence of the jobs such that all the constraints are satisfied and the completion time of the last job is minimised [1].

The release times for the jobs will not be considered in the static problem; however, release times will affect the dynamic problem. The following mixed-integer programming model for this revised SOPD without release times is based on a formulation by Maffioli & Sciomachen[11]. Let

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ is followed by job } j \\ 0 & \text{otherwise} \end{cases}$$

w_i be the time at which job i begins and

p_i be the position of job i within the cycle.

The following parameters are used in the model. Let

c_{ij} be the time it takes for the crane to move from job i to job j,

s_i be the time it takes for job i to be completed,

u_i be the latest starting time of job i,

n be the total number of jobs, and

M^* be any large number (greater than the total completion time).

In terms of the defined symbols, the objective is to

$$\text{minimise} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,2,\ldots,n \tag{3}$$

$$w_i + s_i + c_{ij} - w_j \leq (1 - x_{ij})M^* \qquad i = 1,2,\ldots,n \text{ and } j = 2,3,\ldots,n \tag{4}$$

$$w_i \leq u_i \qquad i = 1,2,\ldots,n \tag{5}$$

$$p_1 = 1 \tag{6}$$

$$p_i - p_j + nx_{ij} \leq n - 1 \qquad i = 1,2,\ldots,n \text{ and } j = 2,3,\ldots,n \tag{7}$$

$$p_i < p_j \qquad \text{whe n job } i \text{ is a prerequisite of job } j \tag{8}$$

$$x_{ij} \in \{0,1\} \qquad i = 1,2,\ldots,n \text{ and } j = 1,2,\ldots,n \tag{9}$$

$$w_i, p_i \geq 0 \qquad i = 1,2,\ldots,n. \tag{10}$$

The objective function in (1) minimises the total time travelled between all the jobs. Constraint sets (2) and (3) ensure that each job is completed only once. Constraint set (4) allocates a start time to each job, ensuring that a job starts only once its predecessor has finished. Constraint set (5) ensures that all the deadlines are met. Job 1is considered a dummy job in that all distances to and from it are 0, and it is forced to be the first job in the cycle by constraint (6). Each job is assigned a position in the cycle, and subtours are eliminated by constraint set (7). The prerequisite relationships are handled by constraint set (8), ensuring that if job i is a prerequisite of job j then job i would be executed before job j.

2.3 Solution approaches to the SOPD in the literature

The SOPD may be described as a combination of two variants of the travelling salesman problem (TSP):TSP with precedence (TSPP), and the asymmetrical TSP with time windows (ATSPTW). A study of exact solution approaches to ATSPTWs by Ascheuer et al. [2] suggests that ATSPTWs of order $n > 40$ require computation times that are too long to use in a dynamic context. The computation time involved when using an exact solution re-optimisation approach is too large for the given problem, as the rate at which the cranes move requires a solution within a few minutes. So metaheuristics need to be considered to reduce the computation time.Landrieu et al.[9] suggest using tabu search as a solution approach. Their results shows considerably lower computation times for problems of order $n < 90$, with the best solutions obtained having values at most 1% above the optimal. A second type of meatheuristics suggested by Cheng & Mao [3] is that of an ant colony algorithm. The data gathered by Cheng & Mao [3] showed, however, that there is a chance that the algorithm might not find feasible solutions for the given test cases.

A threshold accepting heuristic was developed by Nikolakopoulos & Sarimveis [16] for ASTPTWs and SOPs. The heuristic is based on local search techniques with some variational forms of simulated annealing. This algorithm was tested using test cases by Reinelt [17] with good results for both SOP and ASTPTW. This algorithm is out-performed with respect to SOPS by a heuristic algorithm specifically designed for SOPs by Gambradella & Dorigo [6] – a hybridisation of an ant colony algorithm and local search heuristics. The algorithm was also tested using test cases from Reinelt [17], with most of the known upper bounds being improved in reasonable time. A genetic algorithm was applied by Moon et al. [13] to a travelling salesman problem with precedence constraints (TSPP), which is similar to a SOP. The proposed method showed good results for small- to medium-sized problems; however, larger problems (i.e. $n > 50$), which will be required for the crane system, were not handled effectively.

Similar types of TSP variants – such as single vehicle routing problems (SVRP) and its variants, machine scheduling (MS),and more complex multiple vehicle routing problems (VRP) and variants – have been handled with metaheuristic methods. The most flexible of the heuristics is tabu search, with tabu implementations for most TSP variants [8, 9, 11]. Two metaheuristic methods – tabu search and ant colony algorithms – were developed and implemented on the basis of the results found in the literature.

2.4 Tabu search

The specific SOPD considered here has two main types of constraints: those for precedence and those for deadlines. The algorithm must search for possible solutions or job sequences that conformfirst to precedence constraints, secondto deadlines or to minimising the penalty for missing deadlines, and finally to minimising the total run time of the crane.

The solution is represented as an ordered set J of jobs, with job J_i performed before job J_k if $i < k$. The neighbourhood for this algorithm consists of all solutions that can be attained by a switch of two jobs (switch move) or the insertion of one job from its original position into a new one (insertion move). The inverse move of a switch move is the switch move, while the inverse of an insertion is the insertion of the same job, but with the position being the original position before the insertion. The tabu search makes use of an active tabu list as discussed in Józefowska et al. [8], where the number of iterations for which a move remains tabu is equal to the tabu list size or tenure. The tabu tenure is a percentage (10%) of the number of jobs, and is changed dynamically as the problem changes (for example, when new jobs are added to the problem), allowing for easier use in the dynamic problem.

The initial solution conforms to the precedence constraints. A solution may be found by starting with a job that has no precedence (such a job should always exists for the problem to be feasible), and then selecting jobs that have no unselected precedence nodes. In order to ensure precedence feasibility, neighbours will only be considered that conform to these constraints.

The deadline constraints do not necessarily hold in the initial solution. The tabu search will seek out solutions with a lower deadline penalty until a solution is found that fulfils the constraints. The penalty of breaking deadlines is calculated as the sum of the total delay for each job that does not meet its deadline. Once a solution is found that meets all the deadlines, the algorithm attempts to minimise the starting time of the last job, and hence total run time, without breaking the precedence and deadline constraints. The aspiration level consists of multiple goals – namely, deadlines and run times.This allows the algorithm to move towards infeasible solutions with respect to deadlines if a local optimum is reached. A similar structure using a relaxation of goal functions was used by Cordeau & Laporte [4] for a multi-vehicle dial-a-ride problem in order to allow the algorithm to move through infeasible neighbourhoods. It is also important that the tabu search is able to handle infeasible solutions, because they may be created when new jobs are added or when real time delays (for example, a breakdown) occur.

2.5 Ant colony algorithms

SOPs may be presented graph theoretically by a weighted connected digraph (V, E), with a set of vertices V and a set of edges E between the nodes, where the vertices may be seen as a job and the weight of an edge as the distance between two jobs. In the algorithm each ant k traverses the graph and constructs a cycle of n stages during each iteration t. The path between vertices i and j for each ant depends on several elements. The first is a list of destinations not yet visited when ant k is currently at destination i, represented as set D_i^k. This defines the possible movements in each step. The visibility of an edge, which is the reciprocal of the distance of that edge, is used to direct ants towards close destinations. Pheromone quantities on each edge of a trail, called the intensity of the trail, direct ants to

previously attractive paths. Each vertex V_i has a set R_i of predecessors associated with it. The random proportional transition rule used in the model may be stated as

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in D_i^k} (\tau_{il}(t))^\alpha \cdot (\eta_{ij})^\beta} & \text{if} \quad j \in D_i^k \text{ and } R_i \notin D_i^k \\ 0 & \text{if} \quad j \notin D_i^k, \end{cases} \tag{11}$$

where $p_{ij}^k(t)$ is the probability that ant k traverses edge (i, j) at iteration t, α and β are two parameters controlling the relative importance of the trail intensity, $\tau_{ij}(t)$, and the visibility, η_{ij} [2]. When $\alpha = 0$, only visibility is taken into consideration, and when $\beta = 0$ only pheromone trails are considered.

After a cycle, an ant leaves a quantity $\Delta\tau_{ij}^k(t)$ of pheromone on its route. This quantity of pheromone is calculated by means of the formula

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & \text{if } (i, j) \in T^k(t) \\ 0 & \text{if } (i, j) \notin T^k(t), \end{cases} \tag{12}$$

where $T^k(t)$ is the path traversed by ant k during iteration t, $L^k(t)$ is the length of the path, and Q is a parameter [2].

A form of 'evaporation' for the sub-optimal solutions when updating the trails is

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}, \tag{13}$$

where $\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$ [2].

A new variant on the ant colony system was implemented in order to handle deadline constraints. The first method adjusted the visibility parameters, taking deadlines into account, as well as penalising any route that breaks deadlines. The new visibility is recalculated as

$$\eta_{ij} = \frac{1}{d_{ij}} \cdot \frac{\lambda}{u_j}, \tag{14}$$

where d_{ij} is the distance between job i and j, u_j is the deadline associated with job j and λ is a parameter controlling the influence of deadlines. An increase in λ decreases the effect of the deadlines. A route is penalised by allowing for the value of $\tau_{ij}(t)$ to become negative. If a route does not conform to a deadline, each edge (i, j) on the route will assume a negative $\tau_{ij}(t)$.

A second variant based on the method developed by Cheng & Mao [3] for TSPTW proposes to use an adjusted random proportional transition rule. For an ant at job i, let w_j be the time at which the ant arrives at job i; then $G_{ij} = u_j - w_i$ is the slack associated with the

deadline of job j. A nonlinear relationship

$$g_{ij} = \frac{1}{1+\exp(\delta(G_{ij}-\mu))}, \tag{15}$$

holds between g_{ij} and G_{ij} where μ is the average of all G_{ij} and δ is a scaling parameter to adjust the limits of g_{ij}. The new random proportional transition rule is given as

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}\cdot(\eta_{ij})^{\beta}\cdot(g_{ij})^{\gamma}}{\sum_{l\in D_i^k}(\tau_{il}(t))^{\alpha}\cdot(\eta_{ij})^{\beta}\cdot(g_{ij})^{\gamma}} & \text{if } j\in D_i^k \text{ and } R_i \notin D_i^k \\ 0 & \text{if } j\notin D_i^k, \end{cases} \tag{16}$$

where γ is a parameter controlling the relative importance of the slack associated with the deadlines.

All the parameters were configured by means of experimentation. Combinations of values for all the parameters were tested individually, using a number of representative data based on historical data as well as benchmark instances for SOPs [17]. The best combination of parameters was used.

2.6 Hybrid algorithms

A study done by Gambradella & Dorigo [6] suggested using ant colony algorithms in conjunction with local search heuristics. The ant colony algorithm can find good solutions faster thanthe tabu search, and the tabu search is better suited for smaller local improvements. Therefore two hybrid methods, HB1 and HB2, were developed in order to use the properties of tabu search methods and ant colony algorithms. The first method attempts to assist the tabu search method by generating a relatively good initial solution quickly:first it runs an ant colony algorithm, and then passes the resulting solution to a tabu search method as the current solution.

The second method calls the tabu and ant methodssequentially. An ant colony algorithm is run, and the solution is passed to a tabu search. Once the tabu search is complete, the initial pheromone levels of the ant colony algorithm are adjusted, and the ant colony algorithm is called again. This process continues until a stopping criterion is met. The initial pheromones are adjusted by means of the formula

$$\tau_{ij}(0) = (1-\rho)\cdot\tau_{ij}(0) + M\frac{Q}{Z}, \tag{17}$$

where M is a scaling parameter effecting the change in the initial pheromone levels for the new instance of the ant colony algorithm, and Z is the length of the route generated by the previous tabu search. As M increases, the effect of the previous iteration on the initial pheromones is increased.

3. DYNAMIC MODELLING APPROACH

During a typical working day in the DC, jobs become available at different times as trucks arrive, picking lines are completed or planned, and functioning picking lines are replenished. This continuous inclusion and completion of jobs adds a dynamic element to the problem that needs to be handled.

A common approach to solving dynamic SOPs and their variants is to make use of existing algorithms designed for static problems. According to Ichoua et al. [7], these adaptations of static algorithms may be divided into three categories: fast local update procedures, re-optimisation procedures, and hybrid procedures.

A re-optimisation procedure is selected to model the dynamic system, and the tabu search is selected for the re-optimisation, as the algorithms using ant colony methods require parameter changes depending on the number of jobs and length of deadlines. Although the two hybrid methods marginally outperform the tabu search when applied to the DC scenarios, the computational effort of reconfiguring parameters does not justify the insignificant gain. In order for a re-optimisation procedure to be used, a system needs to be developed that can collect the changing information and generate a new static problem. A decision support system,the 'dynamic real time job scheduler', was developed and coded in order tohandle the dynamic elementsin real time.

The first process that the job scheduler must handle is the inclusion of additional jobs in the system. The job insertion process attempts to insert a new job into the best position, such that the new job does not break deadlines. This is achieved by searching for the latest feasible positionthat meets the prerequisites, where the new job does not break its deadline constraints. If such a position does not exist, the job is placed in the position that minimises the penalty of the broken deadline. This will ensure that emergency jobs receive high priority even if the deadline has passed. If a new job has a low priority, it will be placed at the end of the current schedule. Upon completion of a job the job release process is induced. The first job in the current schedule is removed, and the information is sent to the crane.

In order to test the performance of the dynamic realtime job scheduler, a simulator was coded with a virtual systems operator and a virtual crane. The virtual systems operator sends new orders in real time to the job scheduler with information about prerequisites and deadlines. The virtual crane requests a job from the scheduler when it is empty, and while it has a job, it will process the job until the required job completion time has elapsed.

A small example to illustrate what happens during the dynamic process is given. It is based on the sample of data given in Table 1.

Job	Prerequisites	Deadline	Release time
J_1	–	11:00	10:00
J_2	–	11:00	10:00
J_3	–	11:00	10:00
J_4	–	10:25	10:20
J_5	J_3	11:30	10:30
J_6	J_2	11:30	10:30

Table 1: Data used to illustrate the process of dynamic optimisation. The release time (the time at which a job is entered into the system) and the list of future jobs are in reality not known in advance.

Assume that the crane is busy with job J_0, the time is 10:15, and the list of pending jobs is J_1 - J_2 - J_3. Let the crane finish job J_0 at 10:18. The next job (J_1) is then given to the crane, and the list of pending jobs changes to J_2 - J_3. At 10:20 job J_4 enters the system, and because of its deadline it is placed at the top of the list, resulting in a pending job list of J_4 - J_2 - J_3. Once job J_1 is completed, job J_4 is sent to the crane. The crane is still busy with job J_4 when jobs J_5 and J_6 enter the system at 10:30. The pending job list is now J_2 - J_3 - J_4 - J_5 - J_6. Re-optimisation then takes place while job J_4 is performed, resulting in a pending job list of J_3 - J_5 - J_2 - J_6 at 11:32 when job J_4 is completed.

4. RESULTS

Different test instances were run for each of the four algorithms (i.e. tabu search, ant colony, HB1, and HB2) and individual solution qualities were compared in order to determine the best solution method. A set of benchmark problems, developed by Stecco et al. [17] of size $n > 100$, was used to estimate parameters. All testing was performed on an Intel Pentium Core 2 Duo 2.4 Ghz processor with 1Gb RAMrunning Microsoft Windows XP with Service Pack 3, and the code was programmed in JAVA 1.6 [6, 10].

4.1 Static modelling results

Test data sets making use of historical data from the DC were used to compare the algorithms with the DC'scurrent method. These data sets were generated using real data captured by the DC over several days. Four types of scenarios were generated after consultation with the DC manager, incorporating only two different activities [3]. The combinations of these activities are: building and breaking two consecutive picking lines on the same level (BBS), building and breaking two different picking lines (BBD), building a picking line with storage of goods (BS), and building picking lines where the picking line is situated in another aisle and the pallets need to be moved to the floor (FS). For the BBS scenario,the old leftover stock from a previous picking line must be replaced by new stock for the next picking line. A pallet of new stock may only be placed in the picking line once the old stock (if any) has been removed from the required bay.

Scenarios were generated by pairing two sets of jobs, each associated with different activities (see assumption 2) from the same historical data set to create a single set of jobs[1]. These initial scenarios were run with the exclusion of deadlines (generating SOP instances) as well as with binding deadlines (generating SOPD instances). The exact solutions to the scenarios without deadlines may thus be viewed as lower bounds for the scenarios with binding deadlines. Both ant colony algorithms for SOPD presented in Section2.5 were tested, with similar results. The maximum run time for each algorithm is 90 seconds, as this is the duration of a typical job. In an attempt to compare the results with an exact solution, the SOP scenarios were all solved to optimality using Lingo 11 [4]. The scenarios with binding deadlines were not solved, but the exact solutions for the scenarios without deadlines were used as lower bounds. The results were also compared with those calculated based on the DC'scurrent method discussed in Section 1. These individual sets were, however, optimised independently using the formulation (1) – (10), resulting in a best case scenario of that used in practice.

From the results in Figure 3, it can be seen that the ant colony algorithm has the worst performance for all instances, with especially poor performance for the instances with deadlines. The DC's method is shown to be inferior to the tabu search and hybrid methods. The ant colony at times performs worse than the DC's method. The percentage gain throughcombining the two types of jobs involved in building a picking line and breaking down another one is given in Table 2.

Figure 4 illustrates the performance of all the algorithms, as well as the method used by the DC for instances where a picking line is broken and a new one is built. The results are similar to Figure 3, in that the ant colony algorithm performs poorly and the DC's method is improved by the tabu search and hybrid methods. Table 3 shows large improvements on the DC's method, suggesting that this particular combination of jobs should be grouped together in practice.

From the results based on instances where a picking line is built and a set of pallets isstored (Figure 5),there is an improvement in the ant colony performance; however, the improvements on the DC's method and the exact solution are minimal for all algorithms. Table 4 illustrates the low improvements on the DC's method, suggesting that improvements may be less, compared with other combinations, if these job types are combined in practice.

[1]BBS1_68_D indicates data set 1 of type BBS and size 68 with binding deadlines.

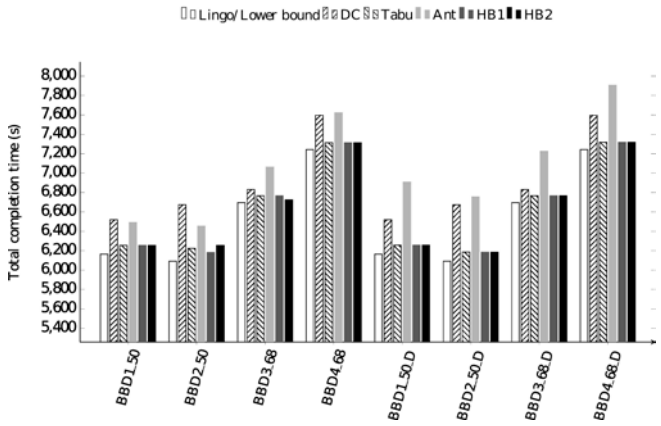

Figure 3: A plot of the total completion times by each algorithm for the DC data sets, where a picking line is built and a different picking line is broken down.

Data set	Tabu	Ant	HB1	HB2
BBD1_50	4.24%	0.43%	4.24%	4.24%
BBD2_50	7.20%	3.40%	7.93%	6.68%
BBD3_68	0.97%	-3.27%	0.97%	1.60%
BBD4_68	3.88%	-0.35%	3.88%	12.3%
BBD1_50_D	4.20%	-5.59%	4.2%	4.20%
BBD2_50_D	7.87%	-1.21%	7.90%	7.90%
BBD3_68_D	0.97%	-5.46%	0.97%	0.97%
BBD4_68_D	3.83%	-3.90%	3.83%	3.83%

Table 2: Percentage improvement of total completion times with respect to the DC'scurrent method, using scenarios with building and breaking down different picking lines.

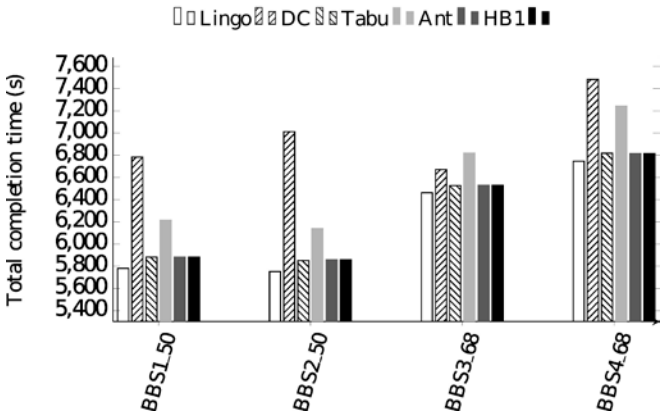

Figure 4: A plot of the total completion times by each algorithm for the DC data sets, where two succeeding picking lines are broken down and built.

Data set	Tabu	Ant	HB1	HB2
BBS1_50	15.29%	9.15%	15.29%	15.29%
BBS2_50	19.82%	14.24%	19.68%	19.68%
BBS3_68	2.21%	-2.18%	2.19%	2.19%
BBS4_68	9.74%	3.31%	9.79%	9.79%

Table 3: Percentage improvement of total completion times forthe DC'smethod,using scenarios with building and breaking down the same picking line.

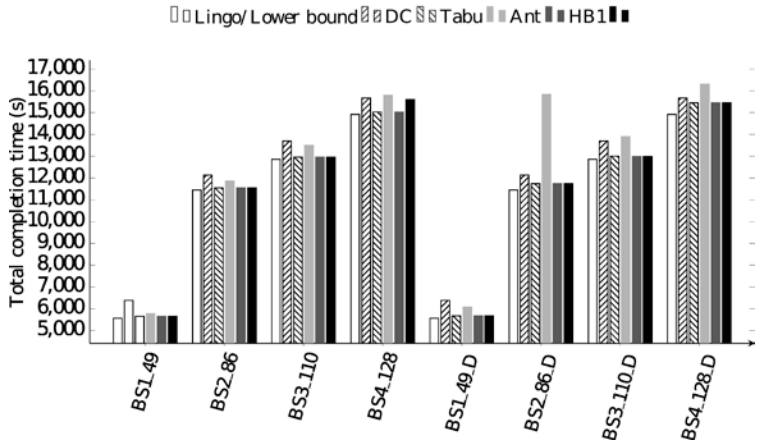

Figure 5: A plot of the total completion times by each algorithm for the DC data sets, where a picking line is built and a set of pallets isstored.

Data set	Tabu	Ant	HB1	HB2
BS1_49	12.97%	10.51%	12.97%	12.97%
BS2_86	5.09%	2.30%	5.09%	5.09%
BS3_110	5.66%	1.44%	5.66%	5.66%
BS4_128	4.25%	-0.85%	4.25%	0.40%
BS1_49_D	12.49%	4.99%	12.49%	12.49%
BS2_86_D	3.41%	-23.38%	3.41%	3.41%
BS3_110_D	5.33%	-1.52%	5.40%	5.40%
BS4_128_D	1.40%	-3.98%	1.40%	1.40%

Table 4: Percentage improvement of total completion times for the DC's method,using scenarios with building a picking line as well as the storage of goods.

The results shown in Figure 6 illustrate the improvements made on the DC's method, as well as the poor performance of the ant colony algorithm when a set of storage jobs iscombined with building a picking line in another aisle, where all the pallets are required to go to the floor first. The significant improvements presented in Table 5 suggest the combination of these job types in practice.

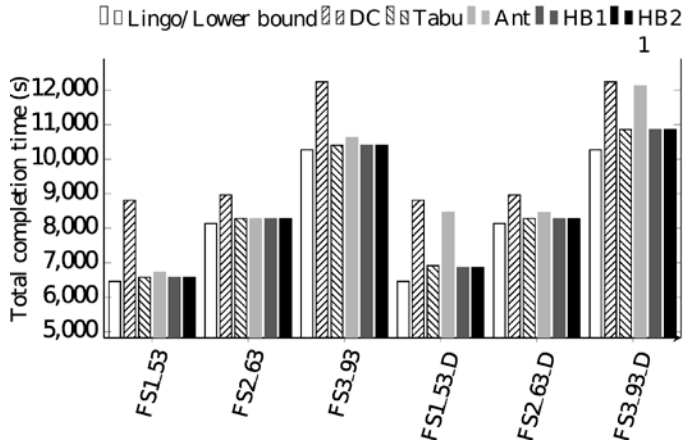

Figure 6: A plot of the total completion times by each algorithm for the DC data sets, where a picking line is built in another aisle and a set of pallets isstored.

Data set	Tabu	Ant	HB1	HB2
FS1_53	33.96%	31.00%	33.96%	33.96%
FS2_63	8.25%	8.25%	8.25%	8.25%
FS3_93	17.66%	15.20%	17.66%	17.66%
FS1_53_D	27.45%	4.16%	28.44%	28.44%
FS2_63_D	8.25%	6.05%	8.25%	8.25%
FS3_93_D	12.69%	0.94%	12.69%	12.69%

Table 5: Percentage improvement of total completion times with respect to the DC'smethod,where a picking line is built in another aisle and a set of pallets is stored.

The DC's method,where no job integration is used, was tested against the optimisation with full integration. The data suggests that improvements may be achieved by combining different job types, with gains of up to 20% for certain combinations. The scenarios that combine storage jobs with building a picking line in another aisle, as well as simultaneously building and breaking two succeeding picking lines, show the largest improvements.

4.2 Dynamic modelling results

Real-time instances were generated using real data from the DC over several days. Similar instanceswith varying dynamic levels were compared with the DC's method. All testing was again performed on an Intel Pentium Core 2 Duo 2.4 Ghz processor with 1Gb RAM running Microsoft Windows XP with Service Pack 3, and the code was programmed in JAVA [10].

Scenarios were generated in a similar fashion to the static problem. However, the inclusion of release times was now taken into account. A job was assigned a release time and a deadline in accordance with the management philosophy.

A scenario may be classified according to a level of job completion percentage, which indicates the required percentage of the previously entered jobs that need to be completed before a new set of jobs may enter the system. If this percentage approaches 0%, the problem becomes a static problem; and if the percentage approaches 100%, the problem becomes one where the DC's current method is used. Therefore a total of five cases are considered:those of 0%, 30%, 50%, and 70% relative to the DC's current method (100%).

From the results in Table 6, it can be seen that the dynamic system used to schedule the jobs of the crane yields faster overall completion times. On average an improvement of 7% is realised, which may be translated to an improvement of about 50 minutes in a 12-hour shift. The low improvement for data set F4 may be attributed to the combination of jobs.

	Job completion percentage			
Data set	0%	30%	50%	70%
F1	10.81%	7.98%	7.46%	5.57%
F2	18.83%	14.02%	12.31%	9.61%
F3	10.57%	7.73%	7.02%	4.86%
F4	4.51%	1.75%	1.75%	0.95%

Table 6: Percentage improvement of total completion times using dynamic reoptimisation in comparison with the case where the job completion rate is 100%, which represents the DC's method.

5. CONCLUSION AND RECOMMENDATIONS

In Section 2 the SOPD was used to model a static scenario, and an exact formulation for the problem was supplied. The solution time needed for an exact solution is too long, and so the development of metaheuristic methods for solving this problem was introduced. Tabu search and ant colony metaheuristic methods, as well as hybrids of these methods, were considered. A dynamic model was developed, based on the solution methods previously obtained, where the release times of the jobs are taken into account. As soon as new jobs enter the system or

a job is completed, the system refreshes the current information and resolves the problem. Both static and dynamic scenarios were generated in order to compare solutions that use job integration with those of the DC's current method.

The results shown in Section4 indicate that ant colony algorithms were not as effective as tabu search, but hybrids of the two methods yield better results when considering scenarios based on the real-life data supplied by the DC. The results obtained from the static model were conclusive: the DC's method is not optimal, with integration producing gains as large as 20%. The results based on realistic dynamic scenarios show an average of 7% improvement on the DC's method.

It is recommended that a dynamic optimisation system be introduced into the operational software currently in use by the DC. The results indicate that the use of the presented algorithms could save up to 1 hour per 12-hour shift.

6. REFERENCES

[1] **Alcaide, D., Rodriguez-Gonzalez, A. & Sicilia, J.** 2003. An approach to solve a hierarchicalstochastic sequential ordering problem, *Omega*, 31(3), 169-187.

[2] **Ascheuer, N., Fischetti, M. & Grötschel, M.** 2001. Solving the asymmetrical travelling salesman problem with time windows by branch-and-cut, *Mathematical Programming*, 90(3), 475-506.

[3] **Cheng, C. & Mao, C.** 2007.**A** modified ant colony system for solving the travelling sales-man problem with time windows, *Mathematical and Computer Modelling,* 46(9-10), 1225-1235.

[4] **Cordeau, J. & Laporte, G.** 2003. A tabu search heuristic for the static multi-vehicle dial a ride problem, *Transportation Research* Part B, 37(6), 579-594.

[5] **Domingo, E.** 2009. Manager at PEP distribution centre, Kuilsriver, South Africa [personal communication], contactable at <krvmgr@pepstores.com>.

[6] **Gambradella, L. & Dorigo, M.** 2000. An ant system hybridized with a new local search for the sequential ordering problem, *INFORMS Journal on Computing*, 12(3), 237-255.

[7] **Ichoua, S., Gendreau, M. & Potvin, J.** 2007.Planned route optimization for real-time vehicle routing. In *Dynamic fleet management, concepts, systems, algorithms and case studies*, chapter 1, 1-18. Springer, New York.

[8] **Józefowska, J., Waligóra, G. & Weglarz, J.** 2002. Tabu list management methods for a discrete-continuous scheduling problem, *European Journal of Operational Research*, 137,288-302.

[9] **Landrieu, A., Mati, Y. & Binder, Z.** 2001. A tabu search heuristic for the single vehicle pickup and delivery problem with time windows, *Journal of Intelligent Manufacturing*, 12(5-6), 497-508.

[10] **Lindo Systems.** Lingo 11 [online, cited May 1st, 2009],http://www.lindo.com/.

[11] **Mafoli, F. & Sciomachen, A.** 1997. A mixed-integer model for solving ordering problems with side constraints, *Annals of Operations Research*, 69(0), 277-297.

[12] **Microsoft Windows XP** [online, cited May 1st, 2009], available from http://www.microsoft.com/.

[13] **Moon, C., Kim, J., Choi, C. & Seo, Y.** 2002. *An* efficient genetic algorithm for the travelling salesman problem with precedence constraints, *European Journal of Operational Research*, 140(3), 606-617.

[14] **Mulcahy, D.E.** 1994. *Warehouse distribution and operations handbook*. McGrawHill, U.S.A.

[15] **Nanry, W. & Barnes, J.** 2000. Solving the pickup and delivery problem with time windows using reactive tabu search, *Transportation Research* Part B, 34(2), 107-121.

[16] **Nikolakopoulos, A. & Sarimveis, H.** 2007. A threshold accepting heuristic with intense local search for the solution of special instances of the travelling salesman problem, *European Journal of Operations Research*, 177(3), 1911-1929.

[17] **Reinelt, G.** 1995. *Tsplib* [online, cited May 1st, 2009], available from: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

[18] **Stecco, G., Cordeau, J.F. & Moretti, E.** 2009. A tabu search heuristic for a sequence-dependent and time-dependent scheduling problem on a single machine, *Journal of Scheduling*,112(1), 3-16.

[19] **Sun Microsystems.** JAVA [online, cited May 1st, 2009], available from http://java.sun.com/.

[20] **Valls, V., Perez, M. & Quintanilla, M.** 1998. A tabu search approach to machine scheduling, *European Journal of Operational Research*, 106(2-3), 277-300.