Scheduling deliveries under uncertainty

Adriana F. Gabor, Rommert Dekker, Timon van Dijk, and Peter van Scheepstal

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2009-040-LIS
Publication	June 2009
Number of pages	23
Persistent paper URL	http://hdl.handle.net/1765/16236
Email address corresponding author	gabor@ese.eur.nl
Address	Erasmus Research Institute of Management (ERIM) RSM Erasmus University / Erasmus School of Economics Erasmus Universiteit Rotterdam P.O.Box 1738 3000 DR Rotterdam, The Netherlands Phone: + 31 10 408 1182 Fax: + 31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

Bibliographic data and classifications of all the ERIM reports are also available on the ERIM website:
www.erim.eur.nl

ERASMUS **R**ESEARCH **I**NSTITUTE OF **M**ANAGEMENT

REPORT SERIES
RESEARCH IN MANAGEMENT

ABSTRACT AND KEYWORDS	
Abstract	Quite often transportation companies face two types of jobs, ones which they can plan themselves and ones which have to be done on call. In this paper we study the scheduling of these jobs, while we assume that job durations are known beforehand as well as windows in which the jobs need to be done. We develop several heuristics to solve the problem at hand. The most successful are based on defining an appropriate buffer. The methods are assessed in extensive experiments on two aspects, viz. efficiency, in the sense that they carry out many jobs and certainty, in the sense that they provide information beforehand about which jobs they will execute.
Free Keywords	stochastic scheduling, distribution problems
Availability	The ERIM Report Series is distributed through the following platforms: Academic Repository at Erasmus University (DEAR), DEAR ERIM Series Portal Social Science Research Network (SSRN), SSRN ERIM Series Webpage Research Papers in Economics (REPEC), REPEC ERIM Series Webpage
Classifications	The electronic versions of the papers in the ERIM report Series contain bibliographic metadata by the following classification systems: Library of Congress Classification, (LCC) LCC Webpage Journal of Economic Literature, (JEL), JEL Webpage ACM Computing Classification System CCS Webpage Inspec Classification scheme (ICS), ICS Webpage

Scheduling deliveries under uncertainty

Adriana F. Gabor[1,*], Rommert Dekker[1] , Timon van Dijk[1], and Peter van Scheepstal[2]

[1] Econometrics Institute,

Erasmus University Rotterdam,

P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

[2]FEL-TNO

P.O. Box 96864, 2509 JG The Hague, The Netherlands

* *Corresponding author*: gabor@ese.eur.nl

June 2009

Quite often transportation companies face two types of jobs, ones which they can plan themselves and ones which have to be done on call. In this paper we study the scheduling of these jobs, while we assume that job durations are known beforehand as well as windows in which the jobs need to be done. We develop several heuristics to solve the problem at hand. The most successful are based on defining an appropriate buffer. The methods are assessed in extensive experiments on two aspects, viz. efficiency, in the sense that they carry out many jobs and certainty, in the sense that they provide information beforehand about which jobs they will execute.

Key words: stochastic scheduling, distribution problems

1. INTRODUCTION

In this paper we focus on a problem often encountered in transportation. There are two types of jobs (tasks) to be scheduled on a fleet of vehicles. One type of jobs, the so-called *plan jobs*, can be scheduled at any moment during a time window. The jobs of the second type, the *call jobs*, have to be executed at the moment the customer calls in. Only a time window in which the customer will call her/his job is known in advance, but not the exact moment of the call. The second type of jobs are considered more important, thus have a higher priority. The question the scheduler faces is how to maximize the expected weighted number of completed jobs while being able to give a certain guarantee to customers that their jobs will be executed.

We encountered this problem with barge transportation of chemicals, where call jobs are orders for transporting highly reactive substances, and supply transportation in the army, where call jobs are orders coming from the battle field. In both cases, plan jobs are regular tasks, for which the exact completion time is less important.

The variant of our problem with known call moments is related to interval scheduling problems, in particular to the class in which, given a set of parallel machines available, the weight of each job and the time window in which it should be executed, one has to maximize the number of (weighted) jobs that can be feasibly scheduled. Arkin and Silverberg(1997) showed that if the starting time and completion time for each job are given, the problem can be reformulated as a minimum cost network flow problem and thus solved in polynomial time. The interval scheduling problem with a fixed number of machines becomes NP-hard if each job can be carried out by a given subset of machines. Heuristics and exact algorithms for this variant are discussed in Kolen and Kroon (1993). The interval scheduling problem is also NP-hard if for each job only a time window is known and not the exact interval when the job should be completed. Approximation algorithms for time constraint scheduling problems, with the objective of maximizing the number of weighted jobs completed, can be found in Bar-Noy et al. (2001) and Berman and Dasgupta (2000). Rojanasoonthon and Bard (2005) describe an exact algorithm based on branch and price for a scheduling problem with time windows arising from a NASA application. Several other heuristics to tackle similar problems are described in Pinedo (2005).

If the call moments are random, the problem we focus on in this paper bares similarities with fleet management problems. In fleet management problems, one has to assign a set of vehicles to customer demands that arise randomly in time. Typically, each vehicle and customer is situated at a certain location, and moving vehicles from one location to another requires one or more time periods. The problem in this paper is related to the variant in which all vehicles, respectively customers are situated at one location. Fleet management problems can be formulated as multistage stochastic integer programs, which are notoriously difficult. As an alternative, several approximation algorithms have been proposed during the last two decades. Powell (1987), Frantzeskakis and Powell (1990), Cheung and Powell (1996) propose approximation algorithms for the case without time windows. For the variant with time windows, Powell and Carvalho (1998) propose an approximate dynamic algorithm based on a linear approximation of the value functions. In a sequel of articles, Godfrey and Powell (2002a), (2002b), develop nonlinear approximations of the value function which experimentally outperforms the linear approximation.

The problems studied in literature manly focus on maximizing the weighted number of jobs completed. In practice, however, a scheduler is often not only interested in the execution of as many jobs as possible, but also in being able to guarantee to customers the completion of their jobs beforehand. In such a context, a scheduler has to find a trade-off between reserving machines in advance and executing as many jobs as possible. In the problem studied here, call jobs are known to be more important than plan jobs and a time window is given in which jobs may be called. Therefore, we address the following questions:

Since call jobs are more important, should one reserve machines for them in advance? What would be a good reservation scheme? Should one reserve machines for the entire time window or only for a part of the time window? What kind of guarantees can be given that jobs will be executed? What is the tradeoff between the quality of the solution and computation time when reservation schemes are combined with methods based on priority lists, which are very fast, and when they are combined with methods based on integer programming?

In order to answer these questions, we consider three reservation policies: *no reservation* of machines for call jobs, *full reservation,* or reservation of machines for call jobs for their entire time window and a novel *partial* or *probabilistic reservation* which reserves machines based on the stochastic properties of the call moments. In all the experiments we use two basic methods to solve the problem at hand: methods based on executing jobs according to some priority lists and methods which create the schedule based on integer programs. For both types of methods, we analyze the effect of reservation on the weighted number of completed jobs. The experiments we have conducted show that the use of probabilistic reservation outperforms pure priority list heuristics and full reservation planning in environments with high number of call jobs. The best performing method proved to be probabilistic reservation combined with integer programming based planning for plan jobs. However, this may be time consuming. If running time is an issue, combining partial reservation of machines with priority lists gives comparable results with a very low computational effort.

The paper is organized as follows. In Section 2 we give a mathematical model of the problem in terms of machine scheduling and present the application from which it originated. In Section 3 we describe in detail the heuristics we have considered. Section 4 presents the design of the computational experiments and of the results obtained. We finalize the paper with concluding remarks about the advantages and disadvantages of the proposed heuristics.

2. PROBLEM DEFINITION AND STRUCTURING

2.1 General problem definition in job scheduling terms

We next give a description of the problem in terms of machine scheduling, that best relates to literature. There are two types of jobs, call jobs, denoted by J_c and plan jobs, denoted by J_p, which need to be executed on K machines within a finite time horizon [0,T]. We assume that all events (release of job, start of a job, end of a job and due date) can take place only at a set of discrete moments in time, denoted by Δ. Since the problems we encountered have usually long durations, and are based on agreements regarding the possible starting times of call jobs, this assumption is not restrictive.

Plan jobs can be scheduled by the decision maker (i.e. the production scheduler), while call orders must start when the customer decides, otherwise they are lost. For each plan job $n \in J_p$, the following data are given: release date r_n, due date f_n and duration d_n. For each call job $n \in J_c$, the customer indicates in advance a time window when he/she may call her orders and the probability p_{nt} of calling job n at time $t \in \Delta$. There is no interaction between the jobs or machines and each machine can execute one job at a time. Each job needs one machine for completion. The overall problem here is how to schedule the jobs such that the expected number of executed jobs is maximized, while preference is given to call jobs. In this paper we chose to model the fact that call jobs have priority upon plan jobs by assigning them higher weights and to maximize the total expected weight of the executed jobs

We will next illustrate this problem with an example from supply transportation in the army.

2.1.2 – Problem with the Royal Netherlands Army

In this section we describe the problem as it originated with the Royal Netherlands Army (RNLA) (see Scheepstal (1999) and Verduijn et al. (2000)). This was the main motivation to start the research. RNLA planned to go over to a different logistical concept in which flat racks are used as transportation units. There are different types of flat racks for different types of goods, such as flat racks for ammunition, refrigerator flat racks for food or medical goods and tank-flat racks for liquids like petrol and water. All these different flat racks can be carried by one single type of truck, a so called palletized loading system. Because trucks can now perform different kinds of distribution tasks, the RNLA is more flexible in achieving its logistical tasks. In order to reach the full potential of the flexibility the palletized loading system offers, all trucks are grouped in one logistic service provider (LSP). There are two types of customer-jobs, the earlier discussed call jobs and plan jobs. Call jobs generally come from units in the field that are in the front zone of the battle. When there is a brief moment in battle in which distribution is possible, the commander wants to use this immediately to reload it's stocks. Therefore the logistical units have to be on stand by to carry out the called jobs. The time interval in which the logistical units have to be on stand by is fixed several hours before on the basis of the military goals a battle unit-commander wants to achieve. On the other hand, there are units in the field, such as units responsible for telecommunication links, for which the exact moment of distribution is less important. They therefore indicate a time interval (the time window) in which the goods may be delivered. The LSP can plan the actual moment of delivery within the indicated interval.

The goal of the LSP is to optimize fleet planning within the uncertainty of the call jobs using the flexibility of the plan jobs.

The translation of the above transportation planning problem to the earlier discussed machine scheduling problem is easy. The jobs in the machine scheduling problem are the call and plan jobs of the transportation planning problem. Furthermore we assume that all jobs are full truck loads so only the duration of the jobs is of importance and not the actual route. This means that the trucks can be identified as the machines of the machine scheduling problem.

2. TYPES OF STRATEGIES

In this study we consider several types of strategies, which differ in the method with which jobs are scheduled and in the policy to reserve machines for call jobs. Jobs may be scheduled based on priority lists or based on the solution of an integer program, case in which we speak about a *planning*. If a strategy reserves machines for call jobs, it may do so for the entire duration of the call jobs (the call jobs are chosen such as to maximize the weighted number of planned call jobs), case in which we speak about *full reservation planning*, or for a part of the call jobs, not necessarily for their entire time window, case in which we speak about *partial reservation*. For partial reservation we propose a new method, based on the probabilistic characteristics of the data. Note that reservation schemes give the scheduler the possibility to guarantee the completion of the jobs for which machines are reserved. If the time windows of call jobs are large in comparison to their duration, full reservation schemes may perform badly with respect to the weighted number of completed jobs, since there are few machines available for plan jobs. In this case, partial reservation may be then a better solution. According to the above mentioned criteria, we distinguish the following strategies: strategies based on priority lists without/ with full/with partial reservation and strategies based on integer programming planning with full/ with partial reservation.

Before describing the heuristics in more detail, we study the deterministic version of our problem, on which the integer programs used in our methods are based.

3.1 Static planning – the deterministic case

In this section we consider the variant in which all call moments are known. Call jobs can then be seen as plan jobs with known start and end times. If for both plan and call jobs it holds that $r_n = f_n - d_n$, the problem of deciding which jobs to execute can be solved in polynomial time by reformulating the problem as a minimum cost flow problem (see e.g. Arkin and Silverberg (1987)). If plan jobs have time windows with slack ($r_n < f_n - d_n$), the problem of scheduling the jobs (with arbitrary priorities) reduces to a parallel machine scheduling problem with time windows, which can be proven to be NP complete in the strong sense by reducing it to the Multiprocessor Scheduling Problem, a known NP complete problem in the strong sense (see Garey and Johnson (2002)). For a detailed proof of this NP-completeness result we refer to Rojanasoonthon (2004). The existence of a pseudo-polynomial algorithm for solving the original scheduling problem of plan and call jobs is thus very unlikely.

Since in our context the machine scheduling problem with time windows is only a tool for finding a good feasible solution of the version with uncertain call moments, we will solve it by means of an integer program. There are several Integer programming formulations in the literature of scheduling problems with time windows (see Rojanasoonthon and Bard (2005), Pinedo (2005)). We choose here for a formulation dependent on the explicit time moments when a job may start, since it suits more the final goal of incorporating call jobs with stochastic starting times.

Denote by $T_n = \Delta \cap [r_{n,} f_n - d_n]$, the discrete time moments when job n may be started.

Let $x_{n,t}$ be a 0-1 variable indicating that job $n \in J_c \cup J_p$ starts being executed at time $t \in \Delta$ and $J_{t't}$ the set of jobs that might have started at some moment in time $t' \leq t$ and may still be in execution at time t.

The machine scheduling problem with time windows can be then formulated as

$$\text{Min} \quad \sum_{n \in J_p \cup J_c} \sum_{t \in T_n} p_n x_{n,t}$$

$$(IP_1) \qquad \sum_{t \in T_n} x_{n,t} \le 1, \text{ for each } n \in J_p \cup J_c, \tag{1}$$

$$\sum_{t' \le t} \sum_{n \in J_{t't}} x_{n,t} \le K, \text{ for each } t \in \bigcup_{n \in J_c \cup J_p} T_n, \tag{2}$$

$$x_{n,t} \in \{0,1\}, \text{ for each } n \in J_p \cup J_c \text{ and } t \in T_n.$$

Constraints (1) say that each job may be executed at most once. Constraints (2) indicate that at any moment t, no more than K machines can be used. Note that these constraints can be easily modified to model the situation in which the number of machines varies in time.

3.2 Stochastic planning methods

In this section we return to the initial problem, where the starting times of call jobs within the given time window are uncertain. Our goal is to design a schedule that maximizes the total expected weight of completed jobs. The presence of overlapping time windows and the dependence of the history of the process make this problem inherently difficult. At the beginning of this Section we have described the types of strategies one can employ to solve the problem at hand. Next we present the heuristics we considered in more detail.

Heuristics based on priority lists, no prior reservation for call jobs

MinSlack In this first heuristic, we execute at every moment the job with the least slack (the slack of a job n at time t is equal to $\max(f_n - d_n - t, 0)$). We assume that at any moment in time, among the jobs with 0 slack, call orders will be prioritized. Call jobs and plan jobs with the same slack are executed in decreasing order of their priority by duration ratio and in case of ties, in decreasing order of priorities. The idea behind this heuristic is to execute as many call jobs as possible (since they have higher weight) and in the same time to ensure that plan jobs are not postponed till a moment when their completion becomes impossible due to time window violations.

PBD (priority by duration) At any time moment execute the job with the highest priority by duration ratio and in case of ties the jobs with the highest priority. This heuristic favors short call jobs with highest weight. Note that in this heuristic the time windows are completely ignored.

By using priority lists based heuristics, the scheduler can't guarantee the execution of any jobs since the list of jobs changes with the call of every job. Therefore, if guaranteeing the execution of jobs is an important aspect, one may consider the following heuristic.

Heuristics based on priority lists with full reservation planning

CFMinSlack (Call orders First Min Slack) This method first reserves machines for call orders for the duration of their time window such that the maximum weighted number of call jobs for which machines are reserved is attained. At each moment in time when there are idle machines, we execute on them unscheduled call jobs, if any, and afterwards plan jobs, in increasing order of their slack. Clearly, the scheduler can guarantee the completion of all call jobs for which machines were reserved. The heuristic reserves machines for call jobs by means of the integer program (IP_1). An alternative would have been to implement the polynomial algorithm of Arkin and Silverbergh (1987).

Heuristics based on full reservation planning

The methods in this class are characterized by the fact that they reserves machines for call jobs for their entire time window and at certain moments in time they use optimal schedules for the remaining plan jobs.

FRPlan (Full Reservation Planning) FRPlan is the base case of this group. Beforehand a simultaneous planning of all call and plan orders is made, while reserving machines for call orders during their entire time window. For this, the integer program (IP_1) is used with the call moments equal to the release date and the duration of a call job equal to the duration of their time window. No changes are made to this schedule during the whole time horizon, so beforehand we know exactly which orders will be executed. This strategy is quite conservative, especially when time windows of call jobs are very large in comparison to their duration. The decision maker can, however, guarantee the realization of all the jobs scheduled by the integer program.

FRrePlan (Full Reservation rePlanning) This schedule improves FRPlan by using information about the realizations of calls to make a new (hopefully better) planning. Every time a call job is completed, we (re)plan the remaining jobs (like in *FRPlan*) given the actual state. Notice that the orders scheduled in the initial planning may not all be executed, due to possible changes in the planning while call jobs are revealed. Although *FRrePlan* and *FRPlan* start with the same solution, due to reoptimization, *FRrePlan* will obtain a solution with objective value at least equal with the objective value obtained by *FRPlan*. On the other hand, *FRrePlan* the computational effort required by executing *FRrePlan* will be much higher than for *FRPlan* . Moreover, with *FRrePlan*, the scheduler cannot offer any guarantee for the completion of plan jobs. One may do the rescheduling only after certain periods, but that is not investigated in this paper. In order to be able to give guarantees on the completion of a part of the jobs, while still making use of the freed capacity, one may make use of the following strategy.

FRPlan+inserts We use *FRPlan* to make an initial schedule. If at any moment capacity becomes available (due to call jobs completion), we consider whether any of the non planned orders can be inserted in the schedule without changing the original plan. Clearly, *FRPlan+inserts* will perform better than *FRPlan*, since it makes better use of the machines freed by call jobs, but will perform worse then *FRrePlan*, which uses at every moment an optimal schedule for the not completed jobs. However, the scheduler can guarantee the completion of the jobs initially planned with *FRPlan.*

CLFSL (Schedule Call orders and Long orders First, insert Short orders Later) The idea behind *CLFSL* is to reserve machines for call jobs for their entire time window and to schedule only long plan orders in advance, while short plan jobs are executed when capacity is freed by call jobs. We characterize a job as long if it has a larger duration then the average job duration. For the reservation of machines for call jobs and the schedule of long plan jobs, we use (IP_1). When capacity becomes available, we execute the highest priority call order which has not been scheduled so far (if any). Then we execute the short plan jobs and the remaining long call jobs in decreasing order of their duration and in case of ties we choose the one with the least slack.

Heuristics based on partial reservation (probabilistic planning)

The drawback of the methods based on full reservation planning is that they may reserve more capacity than necessary. In the methods we will propose next, we reserve machines such that (hopefully) a high percentage of call jobs can be satisfied, thus having more available machines for plan jobs. Intuitively, one expects such a method to work well when the time windows of call jobs are much larger than their duration. The machines reserved for future call jobs form a *buffer*. Deciding the optimal size of the *buffer* is a problem as difficult as the original problem. The main complicating factor is the restricted number of machines available at any moment in time and the fact that jobs last more time units. Due to these phenomena the decision of accepting a call job has repercussions on the number of available machines for the entire execution of the job, making the random variables indicating the number of available machines at different times be dependent on each other. Thus, calculating the buffer based on direct computation of the probability distribution of the number of call jobs in progress at a certain moment in time is a complicated task.

For a small number of machines, one could decide the size of the buffer by simulation. For a large number of machines, simulating the process for all possible values of the buffer may be very time consuming. For this situation, we propose approximations inspired from stochastic inventory control (see Silver et al. [1999]).

Denote by $X_n(t)$ the random variable indicating whether job n is called at time t and let p_{nt} be the probability that $X_n(t)$ takes value 1.

At each moment in time t we look ahead a period of time of length S. For each $0 \le s \le S$, we estimate the new probabilities $\tilde{p}_{n,t+s}$ of jobs to be called in future, given the information available at time t. Thus, for $0 \le s \le S$,

$$\tilde{p}_{n,t+s} = P(X_n(t+s) = 1 \mid \sum_{t'<t} X_n(t') = k) = \begin{cases} 0, & if \quad k = 1 \\ \dfrac{p_{n,t+s}}{\sum_{s=0}^{T} p_{n,t+s}}, & if \ k = 0. \end{cases}$$

We approximate the number of jobs in execution at time $t+s$ by the number of jobs in execution at time $t+s$ in a system with infinite capacity.

Let $Y_n(t+s)$ be random variables indicating whether job n is being processed at time $t+s$ in the infinite capacity system. The variables $Y_n(t+s)$ are independent Bernoulli variables, with

$$P(Y_n(t+s)=1) = \hat{p}_{n,t+s} = \sum_{t' \le t+s \le t'+d_n} \tilde{p}_{n,t'}.$$

Thus, in case of sufficient capacity, the average and the standard deviation of the number of jobs in progress at time $t+s$ are given by $\mu_{t+s} = \sum_{n \in J_c} \hat{p}_{n,t+s}$, respectively

$\sigma^2_{t+s} = \sum_{i \in J_c} \hat{p}_{n,t+s}(1-\hat{p}_{n,t+s})$.

For each $0 \le s \le S,$ we define the buffer at time $t+s$ as

$$b_{t+s}(k_s) = \min\{C_{t+s}, \lceil \mu_{t+s} + k_S \sigma_{t+s} \rceil\}, \qquad (3)$$

where $C_{t+s} = |\{n \in J_c : r_n \le t+s \le f_n - d_n\}|$ is the maximum number of call jobs that could be in execution at time $t+s$ and k_S is a parameter which is decided in advance via simulations. Note that formula (3) for calculating the buffer is very simple and it only uses basic parameters of the distribution of the number of calls in progress.

Depending on the choice of S and k_S, we distinguish two types of strategies:

-*Time dependent buffer (STDB) strategies* In these strategies we define S as the remaining planning horizon. For several choices of the parameter k_S, we simulate the whole process with reservations made according to (3) for the remaining time horizon. Note that within a simulation k_S is constant. Jobs are scheduled according to the heuristics with which STDB is combined with. Finally we select the value of k_S which gives the highest average of weighted completed jobs.

-*Current moment buffer* (SCMB) In these strategies we define S as the shortest job duration. Beforehand, we simulate the process for several values of k_S, where at each time a job is called, machines are reserved according to (3) for a time period equal to the shortest job duration. Jobs are scheduled according to the heuristic with which SCMB is combined with. Note that since we only look ahead a short period of time, the methods based on the current moment buffer are not suited to be combined with planning methods, which are based on a schedule for the whole period of time for plan jobs.

We have experimentally analyzed the following heuristics based on buffer reservation.

STDB (simulation based time dependent buffer) heuristics

Min Slack with STDB At each time moment a job is called, reserve capacity according to *STDB* for the remaining time horizon. On the non reserved machines, schedule the still to be executed jobs in increasing order of their slack, starting with call jobs. As in *MinSlack*, we resolve ties between jobs with the same slack by choosing the jobs with highest priority by duration ratio and among jobs with the same slack and PBD, the job with the highest priority.

Plan jobs first (PRPlan with STDB) Calculate at the beginning of the planning horizon how many machines should be reserved for call orders at every moment using partial reservation. The buffer thus calculated for each time moment *t* remains fixed for the entire planning horizon and it does not change when more information on call jobs arrives (when a job is called or when a call job is completed). Use the remaining capacity to schedule the plan orders, by solving the integer program (IP_1). Plan orders are executed according to the schedule. Plan orders which are not scheduled, are not executed. The capacity available for call orders is equal to the buffer plus the left over capacity from scheduling the plan orders. Call orders are executed in decreasing order of their priority and only if there is enough capacity, otherwise they are skipped. If two call orders have the same priority, choose the one of shortest duration. Note that when using *PRPlan+STDB,* the scheduler can guarantee the completion of the plan jobs initially scheduled .

Replan plan orders (PRRePlan with STDB) In this method capacity is reserved for call orders according to STDB and a new schedule for plan jobs is made by using the integer program (IP_1) every time new information about call orders becomes available. Call orders are executed in decreasing order of their priority. The rescheduling of plan jobs results in an increased weighted number of completed jobs with respect to *PRPlanPF.* The computational effort however, will be higher.

SCMB (simulation based current moment buffer) heuristics

MinSlack +SCMB Every time a job is called, we reserve machines for the duration of the shortest job according to formula (3). We execute on the remaining available machines jobs in increasing order of their slack. More precisely, at each moment in time we will execute as many called jobs as possible, using the buffer and the idle nonreserved machines. If after planning the call jobs some machines are still idle, we start executing plan jobs in increasing order of their slack.

3. NUMERICAL EXPERIMENTS

Design of experiments. We considered six sets of experiments. The basic time step used is one quarter of an hour. The first 4 sets have the following characteristics in common: there are 180 jobs; the release dates of plan/call jobs are uniformly distributed between 1 and 96; the order length of plan/call orders is uniformly distributed between 4 and 20; due dates plan/call orders are uniformly distributed between 3+release date + length of job and 11+release date+ length of job; the call moments are uniformly distributed between the release and due date of the job.

Dataset 1 consists of 60 call jobs and 120 plan jobs, whereas *dataset 2* is made up of 120 call jobs and 60 plan jobs. In both sets all call jobs have priority 10 and all plan jobs a priority of 1. These data sets were designed for studying the effect of increased uncertainty (higher number of call jobs) on heuristics which favour short jobs. Note that on these datasets, heuristics based on executing jobs according to priority by duration ratio, favour short jobs.

Dataset 3 and *4* consist of the same jobs as dataset 1 and 2 respectively, except that the priority is five times the job length for call jobs and equal to the job length for plan jobs. Note that in these datasets all jobs called at the same moment have the same slack and priority by duration ration. Therefore, here call jobs with the highest priority will be executed first.

Datasets 5 and *6* are similar to dataset 4, but they contain 30 additional jobs of length 8 with known call moments. The first 10 jobs are released at time 25, the next 10 jobs are released at time 50 and the last 10 jobs are released at time 75. The additional jobs in data set 5 have priority 40 and in data set 6 priority 200. The scope of datasets 5 and 6 is to analyze the effects of peaks in demand on the proposed heuristics.

To get an accurate evaluation of the performances of the different heuristics, we have generated all the scenarios beforehand, so that each heuristic has to handle the same jobs at the same moments in time. Each dataset contains 400 different scenarios, which are divided in 10 groups of 40 scenarios. Within each group, the only differences between the scenarios are the call moments of the call jobs. The release dates, due dates and durations of the jobs are the same within a group.

For the methods based on simulations, we perform at each moment in time 100 simulations per group of data with different call moments and all other parameters unchanged for all values of $k_{t,S}$ between 0 and 3 with a step size of 0.1. Next an average performance is determined over these 100 simulations. We choose the value of $k_{t,S}$ for which maximum average utility is achieved.

For all data sets we have analyzed three different capacity levels: 30, 25 and 20 machines. Whereas 30 machines would be able to handle most of the jobs, 20 machines won't be able to execute a significant proportion of them.

The experiments are carried out with a simulation program, built in Microsoft Visual Basic 6.0, on an Intel computer with 2.33GHz processor and 3.23 GB internal memory. . If an integer programming model has to be solved, the program uses ILOG CPLEX 10.

Experimental results

A summary of the experimental results is given in tables 1-6 in the appendix. Tables 1, 2 and 3 show the performance of the heuristics with respect to the average weight of the completed jobs. We decided to take the presumable best strategy, PrRePlan with STDB, as base case. Its performance is given at the bottom of the table, together with the maximum possible performance when the call moments are known (CMknown). In the column *average* of Tables 4-6 we show the performance of each method relative to that of PrRePlan with STDB, that is (the average utility of the method /average utility of PRRePlan) – 100%. Furthermore, for each dataset and method, we show in the column *St.dev.* the standard deviation of the utility obtained . We chose not to present the standard deviation relative to the one of PrRePlan with STDB since in several cases the last value is 0. In Tables 6,7 and 8 we present the average computation time of the heuristics in seconds together with the standard deviation of the computation time..

Based on our experimental results, we draw the following conclusions:

Effects of uncertainty on the average weighted number of completed jobs

A higher number of call jobs causes a decrease in the performance of all the heuristics based on priority lists without reservation and of the full reservation heuristics. The methods based on probabilistic reservation are less influenced by the increased number of call jobs. This phenomenon can be seen by comparing the results for data sets 1and 2 or 3 and 4.

Effects of reservation on the average weighted number of completed jobs.

For all three values of the number of available machines, the pure priority lists based heuristics *Min Slack* and *PBD* are outperformed by the methods which use reservation. The performance of *MinSlack* improves when combined with full reservation (*CFMinSlack*) or with probabilistic reservation (*MinSlack+STDB* or *MinSlack+SCMB*).

Full reservation versus probabilistic reservation (buffer) with respect to the average weighted number of completed jobs

With only one exception, *PrRePlan+STDB* was the best strategy. Only for Data set 3, with 30 machines, the average results of *FRrePlan* were slightly better than those of *PrRePlan+STDB*. Our experiments also showed that in case of fewer call jobs (dataset 1 and 3), the full reservation methods *FRrePlan* and *CFMinSlack* were the methods performing best after *PrRePlan+STDB* , or in one case even better. In case of high uncertainty, (Dataset 2,4,5,6) partial reservation schemes *PrPlan+STDB*, *MinSlack+STDB* and *MinSlack+SCMB* outperform all the deterministic heuristics.

Planning methods versus priority lists based heuristics with respect to the average weighted number of completed jobs

When there are few machines available, planning methods perform better then priority lists methods. This result is easily explained by the fact that when an integer program is used by a heuristic, it will give the optimal solution, whereas a priority list based schedule will in many cases return only a suboptimal solution. The performance of priority based heuristics increases with the number of machines available.

Behaviour of heuristics in case of a peak in the number of jobs

The methods which give the highest utility for datasets 5 and 6 are *PrRePlan+STDB* and *MinSlack+STDB* and *MinSlack+SCMB.* This suggests that look ahead methods are more appropriate for bursty data then methods based on present information. The experiments show that a full reservation scheme in case of known peaks is also not necessarily beneficial, as *FRrePlan* and *CFminSlack* performs worse than the methods based on reserving only a buffer.

STDB versus SCMB with respect to the average weight of completed jobs

In most of the cases, methods based on *STDB* perform slightly better than methods based on *SCMB*. The only exception is for data set 4, for 25 vehicles. Looking ahead for the whole time horizon proves to be beneficial when the capacity is scarce. When there are enough machines, the length of the look ahead period does not seem to be of high importance.

The choice of the buffer size

With respect to the buffer, the experiments only indicate that the higher the uncertainty and the priority of call jobs, the higher the buffer should be.

Computation Time

Not surprisingly, the methods based on replanning *FRrePlan* and *PrRePlan+STDB* are computationally the most extensive. For datasets 2,4,5,6 *PrRePlan+STDB*, is in average 1.8 times faster then *FRrePlan*. For dataset 3, *FRrePlan* runs faster, especially when the number of vehicles is equal to 25. The bad performance of *PrRePlan+STDB* could be explained by the fact that certain choices of the buffer lead to difficult instances for the integer programming solver. The priority lists based heuristics with a buffer reservation *MinSlack+STDB* and *MinSlack+SCMB* are in average 846.33 times faster than *PrRePlan+STDB.* The experimental results thus indicate that when capacity is scarce, probabilistic reservation and planning is a good method. If running time is an issue, one may opt for *CFminSlack* if the degree of uncertainty is low, or for a priority list heuristics combined with probabilistic reservation if the degree of uncertainty is high. The quality of the solution will only slightly decrease. If there are enough machines, probabilistic reservation with one time planning *PRPlan+STDB* and priority lists with probabilistic reservation seem to give the best results.

With respect to the guarantees that can be given for the completion of jobs, if the number of available machines is large, it seems the best to use *PRPlan+STDB* if the degree of uncertainty is high (see the results on datasets 2,4,5,6). When the proportion of call jobs is low, it seems the best to use *FRrePlan* if a guarantee is needed only for call jobs, or *FRPlan+inserts* if a guarantee is needed for both call and plan jobs.

5. CONCLUSIONS

In this paper we focused on a stochastic scheduling problem often met in transportation. The main characteristic of this problem is that there are two types of jobs: jobs which can be planned by the scheduler whenever it is convenient, as long as their time window is not violated (plan jobs), and call jobs, which have to be executed upon the call of the customer. For call jobs, a time window is also known. We assumed that call jobs are more important than plan jobs. Our goal was to develop a policy for accepting/ rejecting jobs and a schedule of the accepted jobs that maximizes the expected weight of the completed jobs.

Our experiments show that reservation is crucial when capacity is scarce. Moreover, partial reservation based on probabilistic attributes outperforms full reservation methods or simple heuristics based on pure priority lists. If capacity is little, probabilistic reservation works best combined with planning methods. These methods are however, computationally extensive. Thus, if running time is an issue, combining probabilistic reservation with priority lists gives good results with little computation time. When capacity is sufficient, combining probabilistic reservation with priority lists yields similar results to the combination of probabilistic reservation and integer programming based planning.

REFERENCES

Arkin, E.M. and Silverberg, E.L. (1987) Scheduling jobs with fixed starting and finishing times, *Discrete Applied Mathematics* 18, 1-8.

Bar-Noy, A. Bar-Yehuda, R., Freund A., Naor J.S. and Schieber B (2001). A unified approach to approximating resource allocation and scheduling, J.ACM, 48 (5), 1068-1090 .

Cheung, R. K-M. and Powell W.B. (1996), An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management, *Operations Research* 44, 951-963.

Frantzeskakis L. and Powell, W.B. (1990), A successive linear approximation procedure for stochastic dynamic vehicle allocation problems, *Transportation Science*, 24 (1), 40-57.

Garey , M.R. and Johnson D.S.(2002), *Computers and Intractability, A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company, New York

Godfrey G.A. and Powell W.B.(2002a), An adaptive, dynamic programming algorithm for dynamic fleet management I : Single period travel times, *Transportation Science*, 36,
21-39,

Godfrey G.A. and Powell W.B.(2002b), An adaptive, dynamic programming algorithm for dynamic fleet management I : Multiperiod travel times, *Transportation Science*, 36, 40-54

Kolen, A.W. and Kroon, L.G.(1993), On the computational complexity of maximum shift class scheduling, *European Journal of Operational Research*, 64,. 138-151

Kolen, A.W.J., Lenstra, J.K. and Papadimitriou, C.H. and Spieksma F. C.R.(2007), Interval scheduling: a survey, *Naval Research Logistics*, 54, 530-542.

Pinedo, M. (2005), *Operations Scheduling, with applications in manufacturing and services*, Springer Series in Operations Research,.

Powell, W.B. (1987), An operational planning model for the dynamic vehicle allocation problem with uncertain demands, *Transportation Research* 21B, 217-232, 1987.

Powell, W.B. en T.A. Carvalho (1998), Dynamic control of multicommodity fleet management problems, *Transportation Science* 32, 90-109, 1998.

Rojanasoonthon, S.(2004) *Parallel Machine Scheduling with Time Windows*, Dissertation, The University of Texas at Austin

Rojanasoonthon, S., Bard J. (2005), A GRASP for Machine Scheduling with Time Windows, *INFORMS Journal of Computing*, Vol. 1, No.1, 32-51.

Scheepstal, P.G.M. van, *Fysieke distributie ten behoeve van de operationele eenheden van de Koninklijke Landmacht. Een methode om het aantal wissellaadsystemen te bepalen,* TNO-FEL, Den Haag, 1999 (in Dutch).

Silver, E., Pyke, D. and Peterson, R. (1998), *Inventory management and production planning and scheduling*, Wiley and Sons, NY.

Verduijn, T.M., D.A. Henstra, E. Huysman, S.A. van Merriënboer, P.G.M. van Scheepstal, A. Kwaijtaal (2000), *Blauwdruk logistiek beheers- en besturingsconcept voor de KL*, TNO-Inro en TNO-FEL, Delft (in Dutch)

APPENDIX

Table 1:Keyperformance heuristics, Number of machines=20

method	Dataset 1 Average (relative)	st. dev (absolute).	Dataset 2 Average (relative)	st.dev. (absolute)	Dataset 3 Average (relative)	st. dev (absolute).	Dataset 4 Average (relative)	st.dev. (absolute)	Dataset 5 Average (relative)	st.dev. (absolute)	Dataset 6 Average (relative)	st.dev. (absolute)
MinSlack	0.850	(1.45)	0.853	(1.68)	0.890	(8.57)	0.881	(11.06)	0.858	(12.26)	0.806	(55.04)
PBD	0.877	(1.22)	0.873	(1.70)	0.887	(9.04)	0.894	(10.47)	0.869	(11.14)	0.820	(51.83)
CF minSlack	0.990	(0.23)	0.896	(1.30)	0.986	(6.79)	0.923	(8.83)	0.866	(9.51)	0.858	(52.49)
FRPlan	0.979	(0.21)	0.836	(1.25)	0.947	(5.54)	0.850	(7.37)	0.842	(6.81)	0.840	(38.15)
FRrePlan	0.995	(0.17)	0.961	(1.26)	0.990	(6.65)	0.974	(9.02)	0.979	(8.29)	0.976	(60.88)
FRPlan+ inserts	0.985	(0.19)	0.879	(1.35)	0.972	(6.04)	0.912	(8.13)	0.853	(7.62)	0.849	(48.78)
CLFSL	0.967	(0.24)	0.881	(1.23)	0.972	(6.12)	0.917	(8.35)	0.858	(8.71)	0.853	(47.28)
PRPlan+STDB	0.99	(0.23)	0.99	(1.29)	0.97	(5.80)	0.98	(9.75)	0.949	(8.71)	0.968	(71.84)
minSlack+SCMB	0.982	(0.57)	0.985	(1.43)	0.986	(7.41)	0.991	(10.30)	0.975	(11.47)	0.980	(6.35)
minSlack+STDB	0.984	(0.53)	0.996	(1.15)	0.988	(7.31)	0.994	(9.27)	0.993	(11.17)	0.999	(72.73)
CMknown	1.010	(0.15)	1.015	(1.10)	1.019	(7.32)	1.030	(9.66)	1.029	(9.86)	1.025	(65.82)
PRrePlan+STDB	705.47	(0.19)	1197.43	(1.19)	4751.96	(7.39)	7364.21	(9.99)	7546.50	(10.7)	37212.48	(72.59)

Table 2:Keyperformance heuristics , Number of machines =25

method	Dataset1 Average (relative)	st. dev (absolute) .	Dataset 2 Average (relative)	st.dev. (absolute)	Dataset 3 Average (relative)	st. dev (absolute) .	Dataset 4 Average (relative)	st.dev. (absolute)	Dataset 5 Average 9relative)	st.dev. (absolute)	Dataeset 6 Average 9relative)	st.dev. (absolute)
MinSlack	0.962	(0.79)	0.954	(1.23)	0.968	(8.15)	0.959	(11.25)	0.913	(14.16)	0.886	(67.33)
PBD	0.964	(0.74)	0.958	(1.16)	0.968	(8.36)	0.965	(11.15)	0.920	(13.7)	0.895	(65.71)
CFminSlack	1.000	(0.08)	0.960	(0.84)	0.998	(7.70)	0.969	(10.31)	0.868	(10.39)	0.865	(65.72)
FRPlan	0.994	(0.14)	0.917	(1.06)	0.981	(7.77)	0.914	(8.74)	0.878	(8.23)	0.882	(47.47)
FRrePlan	0.999	(0.10)	0.986	(0.75)	0.997	(7.58)	0.982	(10.78)	0.979	(8.53)	0.976	(61.82)
FRPlan+ inserts	0.996	(0.13)	0.945	(1.01)	0.990	(7.74)	0.963	(9.56)	0.862	(9.04)	0.861	(58.05)
CLFSL	0.989	(0.20)	0.951	(0.95)	0.987	(7.86)	0.965	(9.81)	0.863	(10.26)	0.862	(63.28)
PRPlanPF +STDB	1.00	(0.28)	0.99	(0.34)	0.99	(7.62)	0.99	(11.19)	0.899	(9.48)	0.913	(74.61)
minSlack+ SCMB	0.996	(0.37)	0.998	(0.57)	0.996	(7.99)	0.991	(11.57)	0.978	(11.62)	0.974	(6.54)
minSlack+ STDB	0.996	(0.36)	1.000	(0.40)	0.996	(7.78)	0.989	(10.92)	0.993	(11.55)	0.999	(75.52)
CMknown	1.001	(0.04)	1.007	(0.33)	1.003	(7.61)	1.008	(12.87)	1.019	(9.75)	1.016	(67.94)
PRRePlan+ STDB	719.11	(0.10)	1247.84	(0.39)	4967.42	(7.53)	7831.87	(12.09)	8382.14	(11.17)	40632.90	(75.53)

Table 3:Key performance heuristics, Nrf machines=30

	Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5		Dataset 6	
method	average	dev.	average	dev.	average	dev.	average	dev.	average	st.dev.	average	.dev.
MinSlack	0.996	(0.33)	0.996	(0.48)	0.998	(8.14)	0.994	(13.03)	0.950	(13.33)	0.930	(73.61)
PBD	0.996	(0.34)	0.996	(0.48)	0.998	(8.02)	0.995	(13.09)	0.955	(12.99)	0.935	(71.35)
CFminSlack	1.000	(0.00)	0.994	(0.53)	1.000	(7.57)	0.996	(13.12)	0.860	(12.06)	0.852	(75.78)
FRPlan	1.000	(0.02)	0.973	(0.98)	0.999	(7.75)	0.971	(10.44)	0.917	(9.45)	0.918	(54.2)
FRrePlan	1.000	(0.00)	0.997	(0.23)	1.000	(7.58)	0.996	(13.20)	0.984	(8.78)	0.983	(66.47)
FRPlan+ inserts	1.000	(0.01)	0.985	(0.75)	1.000	(7.61)	0.993	(12.82)	0.857	(11.53)	0.851	(75.18)
CLFSL	0.999	(0.07)	0.990	(0.59)	0.999	(7.55)	0.995	(12.95)	0.859	(11.47)	0.853	(74.29)
PRPlanPF +STDB	1.00	(0.02)	1.00	(0.15)	1.00	(7.58)	1.00	(13.63)	0.875	(11.43)	0.874	(79.1)
minSlack+ SCMB	1.000	(0.09)	1.000	(0.08)	0.999	(7.75)	0.999	(13.88)	0.985	(10.68)	0.974	(5.46)
minSlack+ STDB	1.000	(0.00)	1.000	(0.07)	0.999	(7.70)	0.999	(13.73)	0.994	(9.4)	0.999	(74.)
CMknown	1.000	(0.00)	1.000	(0.02)	1.000	(7.58)	1.001	(14.01)	1.007	(8.92)	1.006	(71.34)
PRRePlan+ STDB	720	(0.0)0	1259.45	(0.090	4981.9	(7.58)	7927.42	(13.74)	8888.06	(9.46)	42812.86	(74.28)

Table 4: Computation Time (sec.), Nr. of vehicles=20

	Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5		Dataset 6	
Nr. machines20	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.
MinSlack	0.004	0.00	0.005	0.00	0.004	0.00	0.004	0.00	0.006	0.000	0.005	0.000
PBD	0.004	0.00	0.005	0.00	0.004	0.00	0.004	0.00	0.006	0.000	0.006	0.000
CFminSlack	0.006	0.00	0.005	0.00	0.006	0.00	0.005	0.00	0.006	0.000	0.007	0.000
FRPlan	0.002	0.00	0.000	0.00	0.000	0.00	0.002	0.00	0.002	0.000	0.003	0.000
FRrePlan	64.878	0.94	3.399	0.16	5.269	0.08	2.016	0.01	2.110	0.010	2.139	0.007
FRPlan+inserts	0.004	0.00	0.004	0.00	0.004	0.00	0.005	0.00	0.006	0.000	0.006	0.000
CLFSL	0.005	0.00	0.005	0.00	0.005	0.00	0.005	0.00	0.006	0.000	0.006	0.000
PRPlan+STDB	0.002	0.000	0.006	0.000	0.002	0.000	0.006	0.000	0.008	0.000	0.008	0.000
minSlack+SCMB	0.010	0.00	0.015	0.00	0.009	0.00	0.015	0.00	0.017	0.000	0.018	0.000
MinSlack+STDB	0.040	0.00	0.078	0.00	0.039	0.00	0.076	0.00	0.073	0.000	0.083	0.000
PRrePlan+STDB	14.538	0.74	1.024	0.01	7.612	0.12	1.270	0.03	0.966	0.010	0.962	0.006
CMknown	0.002	0.00	0.002	0.00	0.002	0.00	0.002	0.00	0.002	0.000	0.000	0.016

Table 5: Computation Time (sec.), Nr.of vehicles=25

	Dataset1		Dataset 2		Dataset 3		Dataset 4		Dataset 5		Dataset 6	
Nr. machines 25	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.
MinSlack	0.005	0.00	0.005	0.00	0.005	0.00	0.076	0.07	0.006	0.000	0.007	0.001
PBD	0.005	0.00	0.006	0.00	0.005	0.00	0.006	0.00	0.007	0.000	0.007	0.000
CFminSlack	0.006	0.00	0.005	0.00	0.006	0.00	0.006	0.00	0.006	0.001	0.006	0.000
FRPlan	0.000	0.00	0.002	0.00	0.003	0.00	0.005	0.00	0.005	0.000	0.002	0.000
FRrePlan	8.591	0.43	2.857	0.09	3.850	0.13	1.959	0.01	2.064	0.009	2.142	0.008
FRPlan+ inserts	0.004	0.00	0.004	0.00	0.004	0.00	0.004	0.00	0.005	0.000	0.006	0.000
CLFSL	0.005	0.00	0.004	0.00	0.005	0.00	0.004	0.00	0.006	0.000	0.006	0.000
PRPlanPF+ STDB	0.003	0.000	0.007	0.000	0.002	0.000	0.005	0.000	0.008	0.000	0.008	0.000
minSlack+ SCMB	0.010	0.00	0.016	0.00	0.010	0.00	0.016	0.00	0.018	0.000	0.019	0.000
MinSlack+ STDB	0.040	0.00	0.081	0.00	0.041	0.00	0.077	0.00	0.080	0.001	0.086	0.000
PRrePlan+ STDB	4.443	0.12	1.579	0.01	205.594	100.89	1.947	0.04	1.156	0.012	1.188	0.009
CMknown	0.002	0.00	0.001	0.00	0.002	0.00	0.001	0.00	0.002	0.000	0.000	0.016

Table 6: Computation Time (sec.), Nr. of vehicles =30

	Dataset1		Dataset 2		Dataset3		Dataset4		Dataset 5		Dataset 6	
Nr. machines30	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.	Av.	Sd. dev.
MinSlack	0.005	0.00	0.006	0.00	0.005	0.00	0.006	0.00	0.007	0.000	0.007	0.000
PBD	0.005	0.00	0.006	0.00	0.005	0.00	0.006	0.00	0.007	0.000	0.007	0.000
CFmin Slack	0.005	0.00	0.005	0.00	0.006	0.00	0.006	0.00	0.006	0.000	0.006	0.000
FRPlan	0.003	0.00	0.003	0.00	0.002	0.00	0.002	0.00	0.000	0.000	0.003	0.000
FRrePlan	0.371	0.05	1.908	0.02	0.120	0.01	1.707	0.02	1.924	0.007	2.069	0.011
FRPlan+ inserts	0.004	0.00	0.004	0.00	0.004	0.00	0.004	0.00	0.005	0.000	0.006	0.001
CLFSL	0.004	0.00	0.003	0.00	0.003	0.00	0.004	0.00	0.005	0.000	0.006	0.000
PRPlanPF+STDB	0.003	0.000	0.007	0.000	0.002	0.000	0.005	0.000	0.008	0.000	0.008	0.000
minSlack+SCMB	0.011	0.00	0.016	0.00	0.011	0.00	0.017	0.00	0.019	0.000	0.019	0.000
MinSlack+STDB	0.042	0.00	0.081	0.00	0.042	0.00	0.078	0.00	0.084	0.001	0.092	0.000
PRrePlan+STDB	2.569	0.02	1.437	0.01	2.958	0.02	1.590	0.01	1.408	0.011	1.362	0.012
CMknown	0.001	0.00	0.001	0.00	0.002	0.00	0.002	0.00	1.408	0.002	0.000	0.000

Publications in the Report Series Research* in Management

ERIM Research Program: "Business Processes, Logistics and Information Systems"

2009

How to Normalize Co-Occurrence Data? An Analysis of Some Well-Known Similarity Measures
Nees Jan van Eck and Ludo Waltman
ERS-2009-001-LIS
http://hdl.handle.net/1765/14528

Spare Parts Logistics and Installed Base Information
Muhammad N. Jalil, Rob A. Zuidwijk, Moritz Fleischmann, and Jo A.E.E. van Nunen
ERS-2009-002-LIS
http://hdl.handle.net/1765/14529

Open Location Management in Automated Warehousing Systems
Yugang YU and René B.M. de Koster
ERS-2009-004-LIS
http://hdl.handle.net/1765/14615

VOSviewer: A Computer Program for Bibliometric Mapping
Nees Jan van Eck and Ludo Waltman
ERS-2009-005-LIS
http://hdl.handle.net/1765/14841

Nash Game Model for Optimizing Market Strategies, Configuration of Platform Products in a Vendor Managed Inventory (VMI) Supply Chain for a Product Family
Yugang Yu and George Q. Huang
ERS-2009-009-LIS
http://hdl.handle.net/1765/15029

A Mathematical Analysis of the Long-run Behavior of Genetic Algorithms for Social Modeling
Ludo Waltman and Nees Jan van Eck
ERS-2009-011-LIS
http://hdl.handle.net/1765/15181

A Taxonomy of Bibliometric Performance Indicators Based on the Property of Consistency
Ludo Waltman and Nees Jan van Eck
ERS-2009-014-LIS
http://hdl.handle.net/1765/15182

A Stochastic Dynamic Programming Approach to Revenue Management in a Make-to-Stock Production System
Rainer Quante, Moritz Fleischmann, and Herbert Meyr
ERS-2009-015-LIS
http://hdl.handle.net/1765/15183

Some Comments on Egghe's Derivation of the Impact Factor Distribution
Ludo Waltman and Nees Jan van Eck
ERS-2009-016-LIS
http://hdl.handle.net/1765/15184

The Value of RFID Technology Enabled Information to Manage Perishables
Michael Ketzenberg, and Jacqueline Bloemhof
ERS-2009-020-LIS
http://hdl.handle.net/1765/15412

The Environmental Gains of Remanufacturing: Evidence from the Computer and Mobile Industry
J. Quariguasi Frota Neto, and J.M. Bloemhof
ERS-2009-024-LIS
http://hdl.handle.net/1765/15912

Economic Modeling Using Evolutionary Algorithms: The Effect of a Binary Encoding of Strategies
Ludo Waltman, Nees Jan van Eck, Rommert Dekker, and Uzay Kaymak
ERS-2009-028-LIS
http://hdl.handle.net/1765/16014

Language Selection Policies in International Standardization – Perception of the IEC Member Countries
Hans Teichmann and Henk J. de Vries
ERS-2009-031-LIS
http://hdl.handle.net/1765/16038

Dominant Design or Multiple Designs: The Flash Memory Card Case
Henk J. de Vries, Joost P.M. de Ruijter and Najim Argam
ERS-2009-032-LIS
http://hdl.handle.net/1765/16039

Standards Education Policy Development: Observations based on APEC Research
Donggeun Choi, Henk J. de Vries and Danbee Kim
ERS-2009-033-LIS
http://hdl.handle.net/1765/16040

Scheduling deliveries under uncertainty
Adriana F. Gabor, Rommert Dekker, Timon van Dijk, and Peter van Scheepstal
ERS-2009-040-LIS
http://hdl.handle.net/1765/16236

---

* A complete overview of the ERIM Report Series Research in Management:
https://ep.eur.nl/handle/1765/1

ERIM Research Programs:
LIS Business Processes, Logistics and Information Systems
ORG Organizing for Performance
MKT Marketing
F&A Finance and Accounting
STR Strategy and Entrepreneurship