Data-driven warehouse optimization: deploying skills of order pickers

Marek Matusiak*[1], René de Koster†[2] and Jari Saarinen‡[1]

[1]Finnish Centre of Excellence in Generic Intelligent Machines Research, Aalto University, P.O. Box 15500, 00076 Aalto, Finland

[2]Department of Management of Technology and Innovation, Rotterdam School of Management, Erasmus University, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

June 29, 2015

Abstract

Batching orders and routing order pickers is a commonly studied problem in many picker-to-parts warehouses. The impact of individual differences in picking skills on performance has received little attention. In this paper, we show that taking into account differences in the skills of individual pickers when assigning work has a substantial effect on total batch execution time and picker productivity. We demonstrate this for the case of a Finnish retailer. First, using time-stamped picking data, multilevel modeling is used to forecast batch execution times for individual pickers by modeling individual skills of pickers. Next, these forecasts are used to minimize total batch execution time, by assigning the right picker to the right order batch. We formulate the problem as a joint order batching and generalized assignment model, and solve it with an Adaptive Large Neighborhood Search algorithm. For the sample company, we are able to improve state-of-the-art batching and routing methods by almost 10% taking skill differences among pickers into account and minimizing the sum of total order processing time. Compared to assigning order batches to pickers only based on individual picker productivity, savings of 6% in total time are achieved.

***Keywords*—** Logistics, Order picking, Analytics, Combinatorial optimization, Data driven modeling

1 Introduction

Order picking is the most important, and the most expensive process in distribution centers. It is estimated that order picking operations are responsible for 55% (Drury, 1988) to 65% (Coyle et al., 1996) of the total cost to operate a distribution center. Advances in technology and in knowledge of the picking process have most probably reduced the actual costs of the process, but they still form a substantial part of a distribution center's costs. In spite of the advent of highly mechanized warehouses, orders are still picked manually in most warehouses, with pickers traveling the warehouse to retrieve the

*mmatusia@gmail.com

†rkoster@rsm.nl

‡jari.p.saarinen@gmail.com

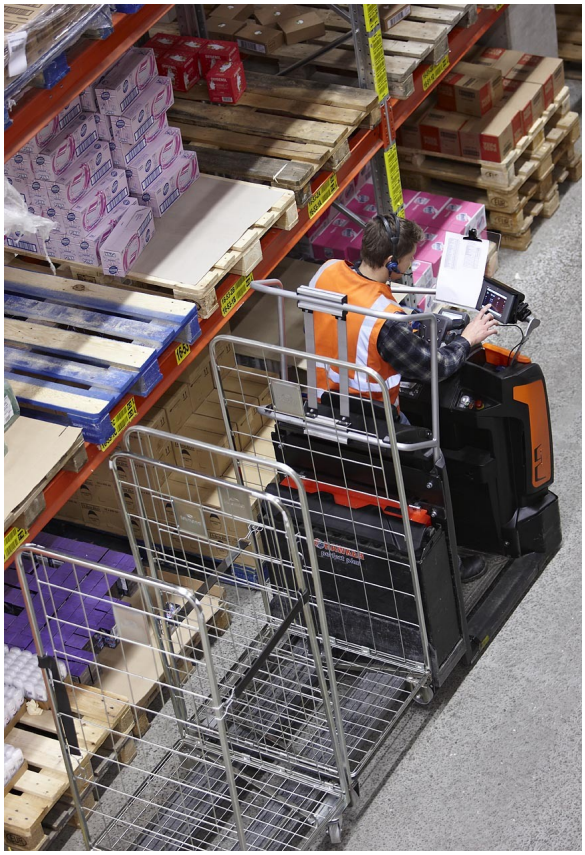

Figure 1: An order picker towing two empty bins. Image courtesy of Rocla.

items for an order (picker-to-parts processes) (De Koster et al., 2007). These manual picking processes are increasingly supported by advanced technologies, such as pick-by-light systems, supporting mobile terminals, and voice picking systems, which have made the order picking process more reliable and efficient (Berger and Ludwig, 2007, Weaver et al., 2010, Baumann, 2013). These computer-assisted picking processes generate extensive logs, including who performed which pick at which location and at which point in time. Warehouses often pick many items (products) on a daily basis, leading to large amounts of log data.

Data on individual order pickers, readily available in many warehouses, can be used to improve the efficiency of the order-picking process significantly. We illustrate this for the case of a Finnish retailer operating a single large order picking warehouse supplying all its stores and e-commerce customers. Time-stamped picking logs are used to build models that forecast the batch execution time of individual order pickers. Due to differences in picking skills, significant differences in performance will exist among pickers. In the case of the retailer, we will show that 13% of the variance in batch execution time is due to such differences. All pickers employed by the retailer have been trained, but they have different physical and mental skills. Some are better at handling heavy masses, or better at driving the pick trucks, or can stack items faster in a roll cage (see Figure 1). The effect of all these factors is extracted from the data, and is used to calculate the time forecasts using a multilevel model. Cross-validation and analysis of variance are used to verify the significance of the added factors and to show differences exist among the pickers. Hierarchical clustering is used to find groups of pickers that have similar skills and to illustrate the results of the regression.

The forecasts are used to determine which orders should be combined in a pick tour (i.e., a batch) and executed by which picker. The problem is formulated as a joint order batching and generalized assignment (BatchGAP) model. In this model, batches are assigned to pickers with the objective to minimize the sum of total batch execution times. Our results can be used by warehouse managers to analyze different skills of pickers and either to dispatch the right order to the best from the available pickers, or in the case of hiring flex workers, to consult past log data and hire those pickers matching the current orders.

The BatchGAP problem is complex. We therefore solve it heuristically with an Adaptive Large Neighborhood Search Algorithm (ALNS). The ALNS exploits differences in pickers using a model that integrates routing, batching, and generalized assignment. By assigning batches and orders to those pickers who are best qualified to execute them, overall picking productivity can be increased and batch execution time can be reduced significantly.

The ALNS algorithm is wrapped in a simulated annealing framework. At each iteration, the ALNS chooses a neighborhood search heuristic to improve a current solution, partly based on past performance of each heuristic. The heuristics focus on moving orders between batches and moving batches between pickers. ALNS-type methods have been shown to work very effectively on a wide variety of computationally hard problems such as container scheduling (Gharehgozli et al., 2013), vehicle routing (Ropke and Pisinger, 2006), scheduling technicians and tasks with varying skills (Cordeau et al., 2010), and order batching and picker routing (Matusiak et al., 2014).

We test the model and algorithm for the case of a Finnish retailer and compare results with the real execution times of the batches assigned to the pickers who actually executed them. The warehouse uses the *first-free* method and assigns the first available picker to the batch to be executed next. We also compare the results with a state-of-the-art travel-distance based order batching method: Variable Neighborhood Search of Albareda-Sambola et al. (2009), combined with the *first-free* heuristic. Using this method, improvements of almost 10% are achieved. Compared to assigning work to the fastest pickers first (based on average picking speed), taking more detailed picking skills into account still saves 6% of the total time. Our method is, in principle, applicable to any modern warehouse using computer assisted picking tools storing time-stamped operational data. However, large real instances require a trade-off between solution time and quality. This trade-off can be addressed, e.g., by choosing a faster batching algorithm to form an initial solution.

We proceed as follows. Section 2 relates this paper to other work in the literature, particularly on order batching, worker modeling, and worker to work assignment. In Section 3, the batch execution time forecasting model is presented. Section 4 introduces the batching and assignment model, and Section 5 describes the routing method and ALNS algorithm. In Section 6, we report our results and we conclude in Section 7.

2 Related Work

In this section, we present research related to our study. We distinguish three topics: research on order batching and picker routing in picker-to-parts order picking processes, assigning jobs to pickers based on their skills and capabilities to complete the jobs, and modeling order pickers in a warehouse context using data logs.

Research on optimizing picker-to-parts order-picking processes has focused on routing pickers (Ratliff and Rosenthal, 1983, Randolph, 1993, Theys et al., 2010), combining orders into batches to minimize travel distance and time (De Koster et al., 1999, Gademann and Van de Velde, 2005, Albareda-Sambola et al., 2009, Henn and Wäscher, 2012, Matusiak et al., 2014), and more recently, scheduling pickers and avoiding congestion (Hong et al., 2012). A recent overview can be found in De Koster et al. (2007) describing different features of the models used. The combined order batching and picker routing problem is complex and several heuristics have been proposed to solve it. Gademann and Van de Velde (2005) use a column generation approach for batching. Routing is done with the polynomial time optimal traveling salesman problem (TSP) algorithm presented in Ratliff and Rosenthal (1983), which is restricted to rectangular warehouses without a middle cross-aisle and with a single depot. The TSP algorithm by Ratliff and Rosenthal (1983) was further extended to include multiple drop-offs in De Koster and Van der Poort (1998), and a middle cross-aisle in Roodbergen and De Koster (2001). Albareda-Sambola et al. (2009) and Henn and Wäscher (2012) use heuristic routing strategies (e.g., S-shape and Largest Gap) combined with Variable Neighborhood Search and Attribute Based Hill-climber based batching algorithms, respectively. As the combinatorial space of possible batches is explored with a search algorithm, routing has to be done for each new, previously unrouted batch. This limits using computationally heavy, general purpose TSP algorithms, such as the Lin-Kernighan-Helsgaun algorithm (Helsgaun, 2000, Theys et al., 2010), in conjunction with a batching algorithm. Matusiak et al. (2014) deal with a precedence constrained routing case, where the savings gained when batching multiple customer orders can be approximated without the need to explicitly route all batches.

Assigning jobs to workers, based on their *capabilities* has been studied relatively well. We distinguish between *capability* (the degree to which a person is able to carry out a certain task, regardless of the time it takes) and *skills* (the speed at which such a task can be carried out). In our case, all pickers have the capability to pick the batches, albeit at different speeds. Differences in skills may be caused by, for example, the ability to lift heavy objects, motivation, ability to handle high volume batches, and knowledge of the warehouse — many of which are not directly quantifiable.

Taking advantage of workers' capabilities has been studied and modeled as a generalized assignment problem (GAP). For extensive surveys on the subject, see Cattrysse and Van Wassenhove (1992) and Pentico (2007). Campbell and Diaby (2002) solve a GAP for cross-trained workers with an assignment heuristic. Each worker has a predetermined value from zero to one characterizing his/her ability to work in a department (to do a job). Another stream of research investigates how worker speed depends on endogenous and exogenous factors (Bendoly et al., 2006). Powell and Schultz (2004) show that workers increase speed in the presence of a visible backlog. Doerr and Arreola-Risa (2000) investigate how the variability of task completion times is affected by different tasks and workers. They find that the most significant factors affecting the variability are: (1) the worker who did the work and (2) the interaction effect between the worker and the task, while the task type in question had no significant influence on its own. Juran and Schruben (2004) use personality and demographic data to predict task execution times of a collaborative two-worker task. They find that including information of individual workers has a significant impact on simulation accuracy compared to assuming no differences among workers. Bartholdi and Eisenstein (1996) show that if workers are sequenced from slowest to fastest in a production line and if they have the possibility of changing stations, but not of passing another worker,

a stable partition of work will spontaneously emerge. Items can be passed to subsequent workers. This leads to the production rate converging to the maximum possible value. These studies reinforce the hypothesis of picker skills playing a significant role in total batch execution time, which is further studied in sections 3 and 5.

We have not found any studies that measure the impact of skills and exploit the differences in skills to assign workers to jobs, i.e., where workers can do the work, but have different execution speeds, depending on job parameters. The next section shows how we use log data to forecast batch execution times.

3 Forecasting Expected Batch Execution Time

Most warehouse management systems (WMS) store order picking logs, which are captured at a very detailed level by advanced picking tools. In this section, such log data are used to construct models to forecast the batch execution time for individual pickers. Batch execution time may depend on many factors, such as the details of the batch to be picked, but also on behavioral factors, such as intrinsic and extrinsic motivation and ability (Larco Martinelli, 2010). However, rather than explicitly including behavioral factors, we implicitly include these by only considering the past performance of each picker, as this is what can be found in the WMS data. WMS data of a picked order in a batch typically include: picker ID, roll cage IDs in which the items are picked, drop-off locations of the roll cages, time stamp of each order line, slot address per line, item IDs, and number of units picked. Multilevel modeling is naturally suited for distinguishing between-group, i.e., between-picker, differences. As potential independent variables, we select the number of pick lines in a batch, total batch travel distance, total pick item mass and volume in a batch, as well as the mean pick height level at which items are picked during the picking tour. We use the data to extract and test the significance of the following skills on the total time using multilevel modeling, where the pickers form the "groups":

- agility: ability to stop on and off the pick truck (stressing for the knees, see Figure 1); modeled by the number of lines;
- driving skill; modeled by the batch travel distance;
- skill in picking heavy items (strength); modeled by the of total batch mass;
- skill in picking at low or high level; which may be influenced by picker height; modeled by the mean pick level of a batch;
- skill in picking large volume batches; modeled by the total batch volume.

We continue by describing the dataset, the data cleaning, the regression method, and then present the regression and finally illustrate the regression results by clustering picker models.

3.1 Warehouse and Dataset Description

Our sample warehouse is a large picker-to-parts warehouse in Finland. We obtained three months of extensive pick data from the warehouse's pick-by-voice system, in total nearly two million time-stamped logs. The warehouse has three cross-aisles, 57 aisles, seven drop-off locations, unidirectional travel in

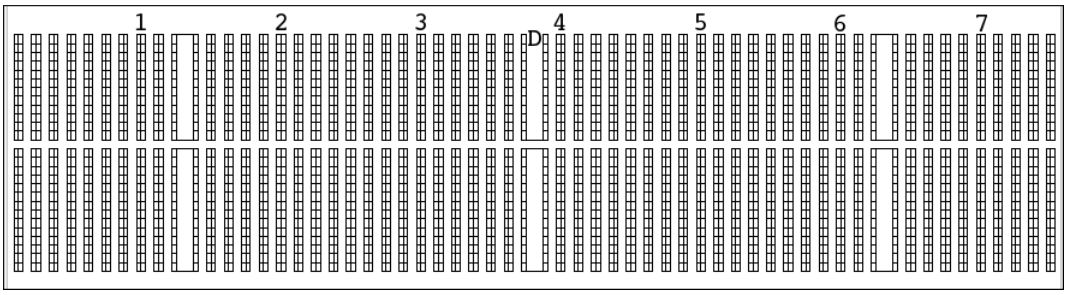

Figure 2: Warehouse layout. The drop-off locations are marked 1-7. D stands for depot.

the aisles, and a single depot storing empty roll cages (see Figure 2 for the layout). Each customer order contains a maximum of 50 order lines, and is picked to a single roll cage using a motorized truck that can transport a maximum of three roll cages. Thus each batch consists of a maximum of three customer orders.

Our data set allows us to extract original customer orders and batches, and to identify the picker who did the work. The batch execution time depends on: (1) picker ID; (2) total number of lines; (3) total travel distance calculated from the distance between all sequentially visited locations in a tour $[m]$; (4) total mass $[kg]$; (5) mean pick level — where 1 is low and 2 is high; (6) total volume $[m^3]$. Picker height can also play a part, but these data were unavailable from the retailer.

The models are then used in an ALNS algorithm as explained in sections 4, 5, and 6 to assign orders and batches to pickers. The whole dataset consists of 37,841 batches worked by 229 pickers during a three-month period.

3.2 Data Cleaning

To find possible outliers, the data were preprocessed. The results are summarized in Table 1.

Three different ways of picking were found: (i) multiple orders in a batch (pick tour), (ii) individual orders, and (iii) multiple orders with no apparent routing, and all lines with the same timestamp. The second and third methods can occur in exceptional situations and represent non-standard ways of working. The second occurs when picking heavy items to pallets instead of roll containers (*Pallet*), and the third occurs when the warehouse manager signs off previously picked orders by a single press of a button (*Timestamp*). The large majority of the batches (83%), represented by (i), is used in our regression modeling; (ii) and (iii) were not considered because they do not represent a standard way of working.

Some items could not be mixed with other products, e.g., because they were highly fragile. The weight of such line items is artificially inflated to reach the threshold weight of 600 kg per order. Orders containing such lines were discarded (*Mass*). In addition, batches that took too long to execute were also discarded. This can be caused by breaks, shift changes in the picking process, or task interruptions with other causes. Based on experience in warehouses with similar picking processes, the threshold was set at a maximum of two hours net picking time per batch (*Long time*). Batches that were picked by more than one picker were also excluded (*Many pickers*).

Neither the output nor the inputs were allowed to contain zero values — all such batches were omitted from consideration (*Zeros*). Finally, to construct a reasonably large set of cross-validation data, for each picker included in the model was required to have at least 75 batches of input data (80% of

Table 1: Summary of data cleaning. The cleaning categories are in the order they are applied — thus *Speed* is first and *Picker out* is last.

	Total	Pallet	Timestamp	Speed	Mass	Long time	Many pickers	Zeros	Picker out	Remaining
Batches	37841	3257	3148	852	553	146	146	34	4070	24669
Percentage	100%	8.6 %	8.3 %	2.3%	0.1%	<0.1%	<0.1%	<0.1%	10.8%	65.2%

Table 2: Mean values and coefficients of variation per batch before and after data cleaning.

	Time $[min]$	Lines	Travel $[m]$	Mass $[kg]$	Level	Vol $[m^3]$
Before clean-up						
μ	28.1	45.7	568.4	204.0	1.1	1.0
CV	0.74	0.70	0.72	0.67	0.11	0.43
After clean-up						
μ	27.1	45.7	571.4	203.1	1.1	1.0
CV	0.74	0.70	0.72	0.66	0.11	0.42

which are used for tuning the model) (*Picker out*). After the data cleaning and the requirement on the number of data lines, 99 pickers out of 229 qualified. Table 1 shows a summary of the results of the data cleaning. After completing the data cleaning, 24,669 batches remained to be used for modeling the pickers.

Table 2 lists the means and coefficients of variation (*CV*) for the inputs and outputs. The independent variables are: *Lines* (x_1), *Travel* (x_2), *Mass* (x_3) *Level* (x_4), and Volume (x_5). We see that the data cleaning process hardly affects the mean and cv of the explanatory variables.

3.3 Multilevel Modeling

Let $w \in \mathcal{W}$ denote the groups (the pickers) and $r \in \mathcal{R}$ are indices to the data (the batches). Furthermore, let vector $\boldsymbol{\beta}_w$ contain the model parameters for group w and let $t_{r,w}$ be the output from a model, which forecasts total batch processing time, for some r and w. Using the notation of Bryk and Raudenbush (1992), a linear multilevel forecasting model can be formulated as in equations (1).

$$t_{r,w} = \beta_{0,w} + \sum_{i=1}^{n} \beta_{i,w} x_{i,r} + \kappa_{w,r} \tag{1a}$$

$$\beta_{i,w} = \gamma_i + u_{i,w} \quad \forall i \in \{0, ..., n\}, \tag{1b}$$

where $x_{i,r}$ is the *i*th element of input vector $\boldsymbol{x}_r$, $\kappa_{w,r}$ is the within-group error term, and γ_i is the slope effect of the dependent variable $i \in \{1, ..., n\}$. For group w, $u_{0,w}$ is the intercept error and $u_{i,w}$ are slope errors $\forall i \in \{1, ..., n\}$. A two-level multilevel model is used to forecast each picker's pick time per batch. Following Bliese (2002), three important sources of variation are found: within group (σ^2), between-group variation in intercepts (τ_0), and between-group variation in slopes (τ_1). The models are built using R software and its packages, *multilevel* and *nlme*.

The data is divided in two parts. For each picker, a random 80% of the batches are used for regression, and the rest (20%) for cross-validation purposes only. As the forecasting models are directly used to calculate estimates of batch execution times in Section 6, we want the forecasts to be as accurate as possible. In regressing, the batches are ordered by timestamp to accommodate possible

learning-based autocorrelations in the data. We have chosen to solely use goodness of fit criteria in model selection as they are more generally applicable than F-tests and equally powerful (Fan and Huang, 2001). Plots of batch execution times and independent variables show that variance increases with total batch execution time, implying the data are heteroscedastic. This appears to be largely remedied by applying a logarithmic transform of the output data (batch execution times).

First, a level-1 model is constructed, i.e., a model with between-picker variability only in the intercept. Second, the level-1 model is extended to level-2 by allowing between-picker variability in the slopes.

Step 1: intercept variation. In this section, we test for for significance of between-picker variation in the intercept. If the variation is not significant, it will not matter which picker gets a job — all pickers will perform in a similar manner. Hence, solving any assignment problem will be pointless. However, if there are between-picker differences, one should strive for a better assignment of work.

Independent variables are added to the model stepwise and tested whether there is significant variation between pickers. The variability of the intercept term is examined with a level-1 *Null Model* (Bryk and Raudenbush, 1992):

$$t_{r,w} = e^{(\beta_{0,w}+\kappa_{w,r})} E(e^{\kappa_{w,r}}) \tag{2a}$$

$$\beta_{0,w} = \gamma_0 + u_{0,w}. \tag{2b}$$

where $t_{r,w}$ is batch execution time of batch r for picker (group) w, γ_0 is the common intercept, $\kappa_{w,r}$ the within-group error term and $u_{0,w}$ the between-group error term, and $E(e^{\kappa_{w,r}})$ is the Smearing Estimate (Duan, 1983) of picker w (i.e., the expected value of the retransformed residuals) used to correct the model bias resulting from the non-linearity of retransforming the logarithmic dependent $t_{r,w}$. In combined form, the model is $t_{r,w} = e^{(\gamma_0+u_{0,w}+\kappa_{w,r})} E(e^{\kappa_{w,r}})$.

The Null Model has two possible sources of variance, τ_0 for how much the groups' intercept varies from the overall intercept (γ_0), and σ^2 for the within-group variance. The Intraclass Correlation Coefficient $ICC(1) = \tau_0/(\tau_0 + \sigma^2)$ (Bryk and Raudenbush, 1992, Kreft and De Leeuw, 1998) equals 0.103, implying 10.3% of the total variance in the natural logarithm of time (13.1% of the non-transformed time) is due to or related to differences between pickers. This suggests that it is beneficial to assign the right batches to the right pickers.

Analysis of variance (ANOVA) is used to test for the significant difference in -2 log-likelihood ratios between a model with and a model without a random error term in the intercept (Bliese, 2002). A model with a high -2 log-likelihood is better than one with a low one, and the statistical significance of the difference is tested. To achieve this, we compare the Null Model and a generalized least squares (GLS) fit of a similar intercept-only model which does not contain the between-group error term. The null hypothesis that no significant difference in the -2 log-likelihood scores exists is rejected ($p < 0.0001$), so individual aspects explain up to 13.1% of the total variance in non-transformed batch execution time.

Step 2: slope variation and model selection. The level-1 multilevel model is now extended into a level-2 model by including group level errors for slope. For n independent variables, the model can be formulated as in equations (3) with the similar notation as in (1).

$$t_{r,w} = e^{(\beta_{0,w} + \sum_{i=1}^{n} \beta_{i,w} x_{i,r} + \kappa_{w,r})} E(e^{\kappa_{w,r}}) \tag{3a}$$

$$\beta_{i,w} = \gamma_i + u_{i,w} \quad \forall i \in \{0, ..., n\} \tag{3b}$$

A model is selected using a stepwise procedure. The selection procedure is summarized in Table 3.

At each iteration, a new independent variable is added to the current model. ANOVA is used to test whether a random effect should be added to the new model. This is done by comparing the -2 log-likelihoods of two versions of the new model: the first with only a fixed effect for the new term, the second with both fixed and random effects. Models are selected based first on the AIC and second on the sum of squared-error residuals (SS) of the set of batches used for cross-validation.

The procedure begins by adding the first term, number of lines (*Lines*) to the Null Model (see Table 3). To better fit the data to a normal distribution, we compare two models: one with log-transformed input data (Model 1L) and one with non-transformed data (Model 1X). Comparing AIC and SS for both models shows that Model 1L is better. A random effect cannot be excluded ($p < 0.0001$). Next, the second independent variable (*Travel*) is added to Model 1L, resulting in models 2LX (non-transformed variable) and 2LL (transformed variable). Both models are also tested for inclusion of random effects. The transformed input is better in terms AIC and SS for Model 2LL than for Model 2LX; therefore Model 2LL is chosen. The process continues by adding each of the log-transformed independent variables and the non-transformed ones, always testing for the significance of the random effect. The tested models are shown after Model 1L in Table 3. This results in Model 5LLLLX, which has logarithmic inputs for all other terms apart from *Volume*.

The addition of interaction terms between the independent variables to Model 5LLLLX did not significantly improve any fit or error measure, or the regression did not converge. The final model in the notation of Bryk and Raudenbush (1992) is

$$\begin{aligned} f(\boldsymbol{\beta}_w, \boldsymbol{x}_r) =& \beta_{0,w} + \beta_{1,w} \ln(x_{1,r}) + \beta_{2,w} \ln(x_{2,r}) + \beta_{3,w} \ln(x_{3,r}) \\ &+ \beta_{4,w} \ln(x_{4,r}) + \beta_{5,w} x_{5,r} + \kappa_{w,r} \end{aligned} \tag{4a}$$

$$t_{r,w} = e^{f(\boldsymbol{\beta}_w, \boldsymbol{x}_r)} E(e^{\kappa_{w,r}}) \tag{4b}$$

$$\beta_{i,w} = \gamma_i + u_{i,w} \quad \forall i \in \{0, ..., 5\} \tag{4c}$$

To test for multicollinearity, a single multiple linear regression model was built with the variables of Model 5LLLLX and run over all data. The maximum VIF (variance inflation factor) value was 4.4 (for $\ln(Lines)$) suggesting that multicollinearity is not a major issue. Visual verification of the scatter plots of the picker models' residuals against batch time showed that heteroscedasticity does play a role, but its effect is mitigated by the log-transformations.

Multivariate OLS (ordinary least squares) and GLS models are fitted using the all independent variables, some log-transformed as in Model 5LLLLX, over the whole teaching data set. Between-picker differences are not taken into account. The model coefficients in the fitted OLS and GLS models coincide. ANOVA is used to test the null hypothesis of Model 5LLLLX not having a significantly better

Table 3: Multilevel model selection procedure

Model[1]	Int(0)	Lines(1)	Travel(2)	Mass(3)	Level(4)	Volume(5)	AIC[4]	SS [5]	R^2_m [6]	R^2_c [6]	p(rand)[7]
0	***[2]	NA[3]	NA	NA	NA	NA	46006	5.6719e+09	0	0.10	<0.0001
1X	***	***	NA	NA	NA	NA	26631	4.6986e+09	0.59	0.68	<0.0001
1L	***	***	NA	NA	NA	NA	16834	2.2053e+09	0.68	0.82	<0.0001
2LX	***	***	***	NA	NA	NA	13922	2.0428e+09	0.71	0.84	0.0498
2LL	***	***	***	NA	NA	NA	13225	1.8963e+09	0.64	0.86	<0.0001
3LLX	***	***	***	***	NA	NA	12675	1.9143e+09	0.65	0.85	<0.0001
3LLL	***	***	***	***	NA	NA	12345	1.8727e+09	0.61	0.87	<0.0001
4LLLX	***	***	***	***	***	NA	11996	1.8517e+09	0.55	0.89	<0.0001
4LLLL	***	***	***	***	***	NA	11992	1.8510e+09	0.66	0.87	<0.0001
5LLLLX	***	***	***	***	***	***	11688	1.8290e+09	0.66	0.87	<0.0001
5LLLLL[8]											
OLS/GLS[9]	***	***	***	***	***	***	17780	2.5847e+09	0.79	-	-

[1] The model number. Zero indicates the intercept-only model. The characters following the model number indicate the modification done to a previous model. If an independent variable is added, the notation is: not included (N); in a non-transformed form with a random effect (X); or log-transformed form with a random effect (L). The addition of an autocorrelation model is denoted by (A). The last addition is the rightmost character.

[2] Confidence in the fixed-effect independent variable, "***" indicates $p \leq 0.001$, "-" that the independent variable is not significant.

[3] Independent variable not included in the model.

[4] The Akaike information criterion of the model, a measure of information lost if the model is used to represent the data. Lower is better.

[5] Sum of squared residuals resulting from the cross-validation of the models.

[6] R^2_m is used to describe the proportion of variance explained only by fixed factors, while R^2_c is used to describe the variance explained by both fixed and random (between-picker) factors (Nakagawa and Schielzeth, 2013).

[7] ANOVA test result on whether a between-picker error term (random effect) should not be included in the model.

[8] The regression did not converge for this model.

[9] An OLS/GLS multivariate regression over the whole data, with some of the variables log-transformed as with the best model 5LLLLX (the coefficients of both the OLS and GLS models are the same).

Table 4: Model betas for 5LLLLX with scaled data, effects on $ln(Time)$.

	Intercept	ln(Lines)	ln(Travel)	ln(Mass)	ln(Level)	Vol
β	3.2	0.66	0.16	0.17	-0.44	-0.16
Random effect stdev	0.40	0.09	0.04	0.08	0.38	0.12

-2 log-likelihood value compared to the GLS model. The null hypothesis is rejected with $p < 0.0001$.

Using the multilevel regression results, it is possible to forecast the batch execution time of each picker. The model coefficients allow the pickers to be characterized pickers in the following way:

- a picker with a low $ln(Lines)$ represents an agile worker;
- a picker with a low $ln(Travel)$ represents a quick driver;
- a picker with a low $ln(Mass)$ represents a strong person;
- a picker with a low $ln(Level)$ can easily pick items from the high level (i.e., he or she is tall);
- a picker with a low Vol represents a picker who can handle large volumes better.

The resulting elasticities and standard deviations of Model 5LLLLX and the corresponding random effects can be found in Table 4. The betas (elasticities) show that the number of lines in a batch is the most important factor in the model. The second row shows the between-picker standard deviation of the model. The standard deviations of $ln(Lines)$, $ln(Travel)$ and $ln(Mass)$ are quite similar. The effect of mean picking level seems to vary the most among the picker models.

The effect of increasing any of the log-transformed inputs is proportional to the (re-transformed) output in this model. The scaled model beta can directly be used to calculate the effect of an independent variable on the dependent variable. For example, for the fixed effect model, a 1% increase in the number of lines in a batch (*Lines*) results in an 0.66% increase in total batch execution time. For the non-transformed independent variable, *Volume*, a unit increase ($1m^3$) will reduce the batch execution time by 0.16%. For Model 5LLLLX, the standard deviation of the residual is 0.32, which means that if the forecasted time for a picker to pick a batch is $t_{w,r}$, there is an approximate 68% chance that it is accurate by a factor of $e^{0.32} = 1.37$, i.e., it is in the range $[t_{w,r}/1.37 \quad 1.37t_{w,r}]$ (Gelman and Hill, 2007, p. 62). Variance of batch execution time thus increases proportionally to the value of time.

Table 4 shows that the batch execution time of the average picker grows with the number of order lines, the distance traveled, and total batch mass, all of which seem reasonable: as the amount of work, travel, or mass increases, it is natural that time increases as well. A higher pick level seems to make picking quicker, which suggests that product to storage location allocation can be improved. The negative coefficient for volume might result from the effect that some of the batches that contain relatively few items of high volume that can be stacked fast.

Picker clustering To group and illustrate similarities among pickers, hierarchical clustering is performed on the forecasting models. To show the effect each of the coefficients has on the batch execution time, we scale them by the corresponding standard deviations of the data (log-transformed where appropriate). Note, that the input data were not scaled before the regression as we were interested in keeping the output in the original scale of seconds.

Table 5: Picker clusters, cluster sizes, mean scaled coefficients and their standard deviations (in brackets). A bolded value for a cluster means the pickers in the cluster are better in a skill than most of the other pickers; conversely it is underlined if they are worse.

Cluster type	size	ln(Lines)	ln(Travel)	ln(Mass)	ln(Level)	Vol
Average, better with heavy items	34	0.56 (0.03)	0.17 (0.02)	**0.13** (0.02)	-0.04 (0.03)	-0.08 (0.02)
Average, poorer with large items	29	0.49 (0.02)	0.19 (0.04)	0.14 (0.02)	-0.06 (0.02)	-0.03 (0.02)
Agile, weak	14	**0.47** (0.02)	0.17 (0.03)	0.20 (0.02)	-0.04 (0.03)	-0.06 (0.02)
Fast, good stacker	12	0.57 (0.04)	**0.13** (0.03)	0.14 (0.04)	**-0.07** (0.03)	**-0.11** (0.04)
Strong, not agile	9	0.61 (0.05)	0.20 (0.02)	**0.07** (0.04)	-0.03 (0.04)	-0.05 (0.03)
Outlier	1	**0.38** (0.00)	0.04 (0.00)	0.27 (0.00)	-0.05 (0.00)	-0.04 (0.00)
Overall	99	0.53 (0.06)	0.17 (0.04)	0.14 (0.04)	-0.05 (0.03)	-0.06 (0.04)

The clustering is performed using R software and the function $hclust$ using the complete linkage method, which defines the cluster distance between two picker clusters to be the maximum Euclidean distance between their individual components. By visual inspection from a resulting dendogram, the number of clusters is chosen to be six. Table 5 shows the mean scaled picker coefficients and the standard deviations of each cluster and the number of pickers present in each cluster. The final row, labeled "Overall", shows the mean scaled coefficients of all the pickers. From this, we can see that the number of lines has the largest effect on the batch execution time. We have characterized each cluster with an appropriate label to show how it differs from the average picker. The evaluation of cluster differences from the mean is done by checking the interval formed by each cluster's mean scaled coefficients and their standard deviations against those in "Overall". Differences among pickers can be seen, e.g., from the clusters "Agile, weak" and "Strong, not agile". The pickers in the "Agile, weak"-cluster cannot handle high-mass batches well (high $ln(Mass)$ coefficient), but are agile pickers otherwise (low $ln(Lines)$). Conversely, pickers in the "Strong, not agile" cluster (low $ln(Mass)$) are good at handling heavy batches, but are otherwise not agile pickers (high $ln(Lines)$).

From the example and analysis above, we can see differences in the factors between picker clusters. The pickers have different strengths and weaknesses that become apparent when comparing to the mean scaled picker coefficients and their standard deviations. In conclusion, we have shown that a significant part, i.e., 13.1%, of the total batch execution time is due to differences in pickers. These differences can be modeled using process data, and can be used to analyze the strengths and weaknesses of individual pickers. The differences in picker skills imply it makes sense to allocate the right picker for each batch.

4 Joint Batching and Generalized Assignment Problem

As the objective of the optimization presented in this section, customer orders and batches are assigned to those order pickers who have the best skills to execute them, thus minimizing the total batch execution time (including pick, travel, and setup time).

Order batching for batch sizes of three or more orders has been shown to be NP-hard (Gademann and Van de Velde, 2005). To further assign batches to pickers, the order batching model from Gademann and Van de Velde (2005) is extended to include a generalized assignment problem, which is also NP-hard (Fisher et al., 1986). This results in a joint batching and generalized assignment problem (BatchGAP), which is sufficiently complex to justify the use of a heuristic.

The problem is defined as follows. Let $\mathcal{R}$ be the set of all possible batch combinations from the set of orders $\mathcal{O}$, each batch consisting of a maximum of N orders. The orders that are to be picked during

a day arrive early each morning. Let $\mathcal{R}_s \subseteq \mathcal{R}$ be the set of batches present in solution $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of feasible and complete solutions to the BatchGAP. Furthermore, let $\mathcal{W}$ be the set of modeled pickers. The mapping of each customer order in $\mathcal{O}$ to a batch $r \in \mathcal{R}$ is characterized with a zero-one vector $\boldsymbol{a_r}$ of length $|\mathcal{O}|$. If $a_{or} = 1$, order o is included in batch r, otherwise it is not. Each batch $r \in \mathcal{R}$ has multiple order lines that need to be picked in no particular order during a picking tour $y \in \mathcal{Y}_r$, where $\mathcal{Y}_r$ is the set of all possible tours for batch r.

After all the lines of an order have been picked, the order must be left at an order-specific drop-off location in the warehouse. All pick and drop-off locations must be visited at least once. Thus a tour is a solution to a TSP, where the drop-off locations have to be visited after the last pick in the batch (see Figure 2). All picking tours start and end at a depot. Each batch $r \in \mathcal{R}$ has a travel distance of d_r associated with the tour-construction method used. The forecast total batch execution time, $t_{w,r}$, depends on the parameters of batch r and the forecasting model of picker w. When calculating the forecast $t_{w,r}$, a TSP corresponding to the batch r to get the travel distance d_r needs to be solved. The TSP is solved separately with a heuristic (see Section 5.1). The TSP constraints are not included in the model below, as the model is mainly illustrative of the problem, and the additional constraints would unnecessarily complicate it. A heuristic is used to solve the TSP as no fast optimal solution to a precedence-constrained TSP (note that the last location of each order is fixed) exists for a warehouse with a middle cross-aisle and unidirectional travel. The tour length of a TSP related to a particular batch is used as one of the inputs when forecasting $t_{w,r}$ using equations (4). Each picker $w \in \mathcal{W}$ has a maximum working time during a shift, M_w.

The following optimization model can be formulated for the BatchGAP:

$$\min \sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}} t_{w,r} X_{w,r} \tag{5}$$

subject to

$$\sum_{w \in \mathcal{W}} X_{w,r} \leq 1 \quad \forall r \in \mathcal{R} \tag{6}$$

$$\sum_{r \in \mathcal{R}} t_{w,r} X_{w,r} \leq M_w \quad \forall w \in \mathcal{W} \tag{7}$$

$$\sum_{o \in \mathcal{O}} a_{o,r} \leq N \quad \forall r \in \mathcal{R} \tag{8}$$

$$\sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}} a_{o,r} X_{w,r} = 1 \quad \forall o \in \mathcal{O} \tag{9}$$

$$X_{w,r} \in \{0, 1\} \tag{10}$$

The goal is to find a solution s^* such that the set of batches $\mathcal{R}_{s*} \subseteq \mathcal{R}$ form a complete partitioning of $\mathcal{O}$ and have a feasible assignment to the pickers $\mathcal{W}$, such that the sum of costs of all pickers is minimized and constraints (6, 7, 8 and 9) are upheld.

The objective function (5) minimizes the sum of all the pickers' batch execution times. The binary decision variable $X_{w,r}$ is one if batch r is assigned to picker w, otherwise it is zero. Constraint (6) enforces that each batch is picked at most once. Each picker's maximum working time, M_w is enforced by (7). Constraint (8) enforces that no more than N orders are contained in any one batch. Each order

must be allocated once to any chosen batch (9) and integrality of the decision variable $X_{w,r}$ is enforced by (10). See Appendix A for a discussion of the complexity of the problems solved in this paper.

As a by-product of the optimization, the effect on picker productivity is also of interest. Let l_r be the number of order lines in batch r. Productivity L_w of a picker $w \in \mathcal{W}$ is defined as

$$L_w = \sum_{r \in \mathcal{R}} X_{w,r} l_r / \sum_{r \in \mathcal{R}} X_{w,r} t_{w,r} \tag{11}$$

or the average number of order lines picked per unit of time worked (usually minutes). Notice that the total work time differs from the shift length.

5 Solving the Joint Batching and Generalized Assignment Problem

5.1 Routing Heuristic

The total travel distance of a picking tour is needed to calculate total batch time. The travel distance is assumed to be independent of the picker. The sample warehouse has multiple drop-off locations (more than one of which can be visited during a tour) and a middle cross-aisle (see Figure 2). The drop-off locations can only be visited after all items of an order have been collected. Each aisle can only be traveled in a single direction, while the cross-aisles are bidirectional. Tours start and end at the depot. The number of potential batch evaluations for the problem introduced in Section 4 is potentially very large. For each batch evaluation, the routing problem needs to be solved. We use a computationally light heuristic to calculate total travel distance.

In Algorithm 1, a version of the aisle-by-aisle routing heuristic (Vaughan, 1999) for warehouses with unidirectional travel in the aisles is presented. In this version of the algorithm, aisles do not need to be traveled completely if there are no picks past the middle aisle. Aisles span over the middle aisle, i.e., the aisle numbering is not affected by the middle aisle. It also incorporates drop-offs before returning to the depot. The basic idea is to take advantage of the unidirectional travel in the warehouse and the middle cross-aisle. Once a path to pick all order lines has been formed (steps 1-4 and 6-8), the tour is optimally completed by adding the drop-off locations with Dijkstra's algorithm.

5.2 First-Free Assignment of Batches

In many warehouses, the current practice is to assign the next batch in the queue to a picker who is first available to execute it. No planning or scheduling is involved. This method of assigning jobs to pickers is henceforth called the *first-free* assignment method. It is assumed that *first-free* is used in the example warehouse.

Before assigning batches to pickers, an initial feasible batching solution must be found. This is done with a combination of a batching and an assignment algorithm. We use either either C&W(i) (Clarke and Wright, 1964) or VNS (Albareda-Sambola et al., 2009) as the batching algorithm, and use free-first to assign batches to pickers. This algorithm is used as the comparison baseline to assign batches to pickers when no knowledge of the pickers' productivity is available beforehand.

Input: A batch r, order drop-off locations $\mathcal{L}_r$, depot D, distance matrix B, set of aisles $\mathcal{Z}$

Output: A tour y, travel distance d_r

Sort all orders in r according to the aisle index.
Make a vector $\boldsymbol{v}_z$ of corresponding order lines for each relevant aisle $z \in \mathcal{Z}$ to batch r.
Sort each aisle vector $\boldsymbol{v}_z$ according to the travel direction of the aisle.
Insert all vectors from the smallest aisle index to the largest index to a path y_{small}.
Use Dijkstra's algorithm to make y_{small} into a tour by adding $\mathcal{L}_r$ and D to it.
Store the total path of y_{small} cost in d_{small} using B.
As in 4 but iterate from big to small aisle index and store the path in y_{big}.
Store the total path of y_{big} cost in d_{big} using B.
Use Dijkstra's algorithm to make y_{big} into a tour by adding $\mathcal{R}_r$ and D to it.
if $d_{big} < d_{small}$ **then**
    $y \leftarrow y_{big}$
    $d_r \leftarrow d_{big}$
end
else
    $y \leftarrow y_{small}$
    $d_r \leftarrow d_{small}$
end

Algorithm 1: Middle aisle multi-dropoff routing heuristic

5.3 Assignment of Customer Orders Based on Average Picker Productivity

Based on the data, the average picker productivity can be calculated using (11). This statistic is tracked in many warehouses. A warehouse manager can choose to assign work to the most productive pickers. To reflect this, and to justify the benefits of skill based assignment, the following heuristic, *fastest-first*, is used to assign work to the most productive pickers first. This algorithm assumes that there are as many pickers available as needed. When knowledge of the pickers' average productivity is available, this algorithm is used as the comparison baseline.

Step 1. Batch all orders based on travel distance using VNS.

Step 2. Sort all available pickers based on productivity in descending order.

Step 3. Sort all batches based on the number of order lines in descending order.

Step 4. Assign the first unassigned batch to the picker who has the highest productivity value and ensure the picker's maximum worktime is not exceeded. Repeat until all batches are assigned, otherwise exit.

5.4 Adaptive Large Neighborhood Search Algorithm

After forming a feasible initial solution, Adaptive Large Neighborhood Search (ALNS) (Ropke and Pisinger, 2006) is used to search for good local optima. Pisinger and Ropke (2010) offer guidelines for designing ALNS algorithms, which have been mostly followed in this section. The ALNS algorithm uses a set of neighborhood heuristics $\mathcal{H}$. For the ALNS presented in this section, $\mathcal{H}$ is composed of five

different heuristics. One heuristic is selected with each iteration, with a weighted random selection criterion called the *roulette wheel method* (Pisinger and Ropke, 2010). In our case, a heuristic is composed of a *destroy* and *repair* method. The destroy method of a heuristic $h \in \mathcal{H}$ breaks down a solution s in a predefined manner, while the repair methods try to find good alternatives to s by constructing another feasible solution $s' \in \mathcal{N}(s)$ based on on the repair method associated with h.

The probability of choosing heuristic $h \in \mathcal{H}$ with the roulette wheel method is

$$p_h = \frac{w_h}{\sum_{h' \in \mathcal{H}} w_{h'}}, \tag{12}$$

where $w_h \in \mathbb{R}^+$ is the *weight* of heuristic h. As the algorithm runs, some heuristics will perform better than others. Good performance of a heuristic will increase its weight and thus the probability of it getting selected again by (12). Each of the heuristics has a *score*, which is updated whenever the heuristic is selected based on the heuristic's search success. Each score is updated every δ iterations by adding the a *performance score* Ψ_h to it:

$$\Psi_h = \Psi_h + \begin{cases} \rho_1 & \text{if the new solution is the global best.} \\ \rho_2 & \text{if the new solution is better than the current one.} \\ \rho_3 & \text{if the new solution is accepted.} \end{cases} \tag{13}$$

An update to a heuristic's weight w_h is done by

$$w_h = \lambda w_h + (1 - \lambda)\Psi_h, \tag{14}$$

where λ is the *decay* parameter, which controls the rate of change in the weights. At the beginning and after an update of the scores, i.e., every δ iterations, the values Ψ_h are set to zero $\forall h \in \mathcal{H}$. If a (steepest) descent algorithm is used, updates to the scores happen only when a new global best solution is found, i.e., Ψ_h can only get the value ρ_1. Otherwise scores do not receive updates.

Following Ropke and Pisinger (2006), a simulated annealing-type hill climbing scheme is implemented to complement the search. This scheme uses a geometric cooling schedule (Cohn and Fielding, 1999). At each iteration, the temperature is updated by $T \leftarrow \phi T$, where T is the temperature variable and ϕ the cooling coefficient. A vector collecting all the adjustable parameters of the ALNS algorithm is defined as $\boldsymbol{v} = (\rho_1, \rho_2, \rho_3, \lambda, \delta, \phi)$. The actual parameter set is given in Section 6.1.

Next, the neighborhood search heuristics are detailed. All of them maintain the feasibility of the solution by enforcing $t_{w,\mathcal{R}_w} \leq M_w$, where $t_{w,\mathcal{R}_w}$ is the total time to pick the set of batches $\mathcal{R}_w \subseteq \mathcal{R}$ assigned to picker w and M_w is the maximum working time for picker w. Only those solutions that are feasible according to the model in Section 4 are accepted.

1 Random destroy, random repair

Step 1. Randomly choose $Q \sim U(2, 7)$ batches to be destroyed from any of the pickers, resulting in the set of $\mathcal{O}_d$ customer orders, where $|\mathcal{O}_d| \geq Q$.

Step 2. Form a set $\mathcal{W}_d$ of pickers currently assigned to $\mathcal{O}_d$.

Step 3. Randomly choose a number of orders up to the maximum batch size of N orders from $\mathcal{O}_d$. Form a new batch from the chosen orders and assign it to a random worker $w \in \mathcal{W}_d$ if constraint (7) can be upheld. Remove the chosen orders from $\mathcal{O}_d$..

Step 4. Repeat Step 3 until there are no orders left to assign.

Step 5. Calculate the tour costs using Algorithm 1 for the new batches and form completion time forecasts for each new picker batch pair.

2 Savings-based destroy, random repair The aim of this heuristic is to destroy those batches that have the least savings value (the time difference of picking each order separately minus picking them together in a batch). Let s be a current solution to the BatchGAP and $S_{tot} \in \mathbb{R}$ be the sum of time-savings of all batches in s. Furthermore, let $S_{max} \in \mathbb{R}$ be the maximum batch savings value in s, and S_r be the savings of batch $r \in \mathcal{R}_s$. Then the probability of destroying a batch r is

$$p_r = (S_{max} - S_r)/S_{tot}. \tag{15}$$

Step 1. Calculate the batch savings for all batches in s.

Step 2. Use the roulette wheel method from (12) with batch probabilities calculated from Equation (15) to choose $Q \sim U(2, 7)$ batches to destroy from any of the pickers, resulting in $M \geq Q$ total customer orders.

Step 3. Form a set $\mathcal{W}_d$ of pickers currently assigned to the batches to be destroyed.

Step 4. Randomly choose a maximum of N orders from the destroyed batches, form a new batch and assign it to a random worker $w \in \mathcal{W}_d$ if constraint (7) can be upheld.

Step 5. Repeat Step 4 until there are no orders left to assign.

Step 6. Get the tour costs for the new batches and form completion time estimates for each new picker batch pair.

3 Move batches This heuristic tries to move a set of batches $\mathcal{T} \subseteq \mathcal{R}$ between pickers while leaving the order composition of the batches intact. A batch $r \in \mathcal{T}$ from picker $w_i \in \mathcal{W}$ is moved to another picker $w_j \in \mathcal{W} - \{w_i\}$. No routing of batches is necessary, the work time estimates of the involved pickers just need to be calculated.

4 Empty picker and greedy repair With this heuristic, all batches are moved from a single picker to other workers. First, the picker is selected randomly from all pickers with batches. Then, each of the selected picker's batches is assigned to other pickers with a greedy heuristic.

5 Move all batches from a picker to another This final heuristic moves all batches from one random picker to the picker that executes them the fastest.

5.4.1 Acceptance of new solutions

After applying a neighborhood search heuristic to a current solution s, a new solution s' is found. Let s^* be the best solution found so far. Furthermore, let $f(s)$ be the cost of solution s. If $f(s') < f(s^*)$, the solution is accepted setting $s^* = s'$ and $s = s'$ The current score of the active heuristic is updated by $\Psi_h = \Psi_h + \rho_1$. Otherwise, if $f(s') < f(s)$, the solution is accepted as the current best one setting $s = s'$ and $\Psi_h = \Psi_h + \rho_2$. A worse solution than the current one can be accepted by the simulated annealing rule. Let $Q \sim U(0,1)$ and T be the current temperature of the algorithm. If $Q \leq \exp(\frac{f(s')-f(s)}{T})$, a hill climb is performed and we set $s = s'$ and $\Psi_h = \Psi_h + \rho_3$.

6 Results

In this section, first the real execution times of the current batches are compared with the forecasted execution times. Second, solutions are computed using state-of-the-art batching algorithms and improved upon by the ALNS algorithm detailed in Section 5.4. Results can be found in sections 6.4 and 6.5.

All algorithms were coded in C++ and run single-threaded on an Intel Xeon 3.2 GHz with 16GB of memory.

6.1 Parameter Calibration

Following Ropke and Pisinger (2006), the calibration parameter vector is initialized to $\boldsymbol{v} = (\rho_1, \rho_2, \rho_3, \lambda, \delta, \phi) = (33, 13, 9, 0.95, 200, 0.999995)$ with the exception of δ, which was not used by these authors; its value was set according to Gharehgozli et al. (2013). In order to calibrate the parameters, two sample instances were run of three pickers with 78 orders, and 10 pickers with 360 orders. One parameter was varied at a time on a uniform interval for each parameter around the initial value. The best performing parameters were chosen and set as $\boldsymbol{v} = (65, 13, 9, 0.95, 91, 0.999995, 0.25)$. The starting temperature T_0 is set such that the probability of an uphill climb is 0.5 if the new solution is 3% worse than the sum of batch execution times in the starting solution s_0 by $T_0 = -0.03 f(s_0)/\ln 0.5$.

6.2 Data Preparation

During a daily eight-hour shift in our sample warehouse, the number of available pickers can vary between 20 and 40. Since the data do not contain due times for the batches, and only a part of all the pickers appearing in the data can be used due to data cleaning, comparing the allocations and batches of the real shifts and the ones reconstructed by the ALNS is not necessarily meaningful. To still allow a comparison, virtual days are constructed by partitioning the data of 24,669 batches into 12 (about) equally sized subsets. Each partition forms one virtual day, resulting in approximately 2056 orders (685 batches) per day. A picker qualifies for inclusion in the workforce of the virtual day if both the sum of real execution times of the batches he or she performed and the sum of the forecast batch execution times exceed the minimum threshold of M_{min}. This is done to guarantee that each picker included carries out sufficient work. The total execution time of all the batches executed by a picker should not exceed the maximum threshold time M_{max}. If adding a batch were to exceed this threshold, it would be discarded for this picker.

The used data only contain Hamiltonian Paths starting from the first pick line to the last line, leaving out the roll container drop-offs and eventual return of the picker to the depot. Each order has its own unique drop-off location. Figure 2 shows the different locations of these drop-offs. Dijkstra's algorithm is applied to find the combined minimum distance of visiting the order drop-off locations and further traveling from the last drop-off location to the depot and from the depot to the first pick location. Using this distance, the extra time to complete the pick tour is now calculated, assuming an average truck speed of 1 m/s, and taken into consideration when calculating sums of batch execution times in constructing a virtual day.

Now the real batch execution times and their forecasts for these newly created virtual days are known. These can be used to compare the ALNS batching, routing, and assignment results with the original solutions.

6.3 Experimental Setup

Virtual days are constructed as detailed in Section 6.2. For each virtual day, the constraint in (7) is set to 8h. Two different instances are run by setting M_{min} to 7.15h or 7.40h and M_{max} to 7.75h in both real time and forecast time. This results in 29 and 20 pickers working on the virtual days on average, respectively. Out of the possible 99 pickers, 93 qualify for the simulations with $M_{min} = 7.15$h and 86 for $M_{min} = 7.4$h. The ALNS is run for 5,000,000 iterations in both cases. We focus on the results for the 20-picker experiments as most of the 29-picker results are very similar.

Work is currently issued according to the *first-free* allocation. Two batching algorithms, C&W(i) and the Variable Neighborhood Search (VNS) by Albareda-Sambola et al. (2009), are used to generate an initial solution for the ALNS. C&W(i) provides a quick solution to the batching problem, while the VNS method is shown to provide good results with the trade-off of exploring large parts of the solution space (Albareda-Sambola et al., 2009, Matusiak et al., 2014). Using these initial solutions, the ALNS is used in two ways: (1) including all neighborhood search heuristics, and (2) allowing only those heuristics which do not break down batches (i.e., excluding heuristics 1 and 2).

In the ALNS heuristic presented in this section, a computationally light routing heuristic (see Section 5.1) is used, as it has to be invoked numerous times. In order to understand how far this routing is off a near-optimal heuristic, it is compared with LKH (Helsgaun, 2000, Theys et al., 2010) for 10,000 random batches constructed from customer orders from the dataset. LKH has a gap of about 0.1% with optimal routing (Theys et al., 2010), at the expense of much longer computation times. LKH does not handle precedence constraints, so drop-offs are left out of the tours for this comparison. On average, LKH leads to solutions of about 11.7% shorter tours.

6.4 Time Savings and Comparison to Original Solution

In this section, the algorithmic solutions are compared to the original and initial batching allocation, as well as to each other. The following abbreviations are used for the solutions in Figure 3. Unless otherwise noted, batch execution times are forecast:

BF batching done with VNS and allocation using *fastest-first*;

CWI batching done with C&W(i) and allocation using *first-free*;

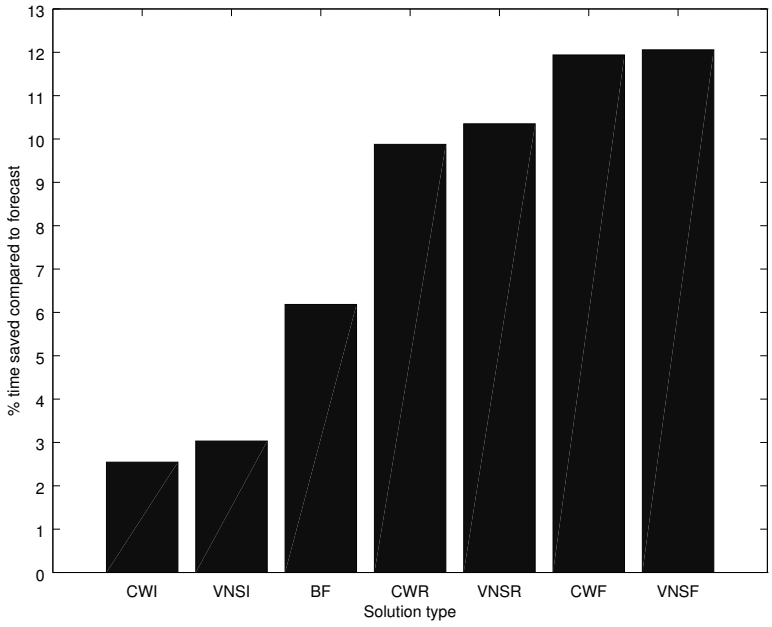

Figure 3: Savings generated by different algorithms, compared to the original batching with forecast execution times during a virtual day. 7.4h minimum time to qualify, 20 pickers. The real batch execution times for the original batching are 1% smaller than the forecast in this case.

CWR improvement of CWI using the ALNS algorithm with no further rebatching (only heuristics 3, 4, and 5 are used);

CWF as CWR, but with all neighborhood search heuristics used in the ALNS;

VNSI batching done with VNS and allocation using *first-free*;

VNSR improvement of VNSI using the ALNS algorithm with no further rebatching (only heuristics 3, 4, and 5 are used);

VNSF as VNSR, but with all neighborhood search heuristics used in the ALNS.

The comparison baseline in Figure 3 is the forecast batch execution time with real batches during a virtual day with first-free allocation, which represents the current situation. If jobs are assigned based on the average productivity (category BF), savings of around 6% can be achieved in both cases. The initial solution CWI gives improvements of over 2% compared to forecast batch execution times with real batches. The best solution is provided by the combination of VNS batching and the ALNS, i.e., solution VNSF. However, CWF is very close in terms of solution quality and much faster to execute. In the 29-picker case, CWF took on average 80 minutes to solve the problem of batching and assigning 2048 orders, whereas VNSF took 120 minutes. For the 20-picker case, these times are 35 minutes and 70 minutes, respectively. A different number of pickers is present during each virtual day. Further investigation shows that the running time of the ALNS algorithm increases linearly with the number of pickers if the number of iterations is kept constant. In the sample warehouse, orders arrive several of hours before the morning shift starts, so there is ample time to run the algorithm. Comparing CWR to CWF and VNSR to VNSF, Figure 3 shows that it is beneficial to rebatch using the neighborhood search

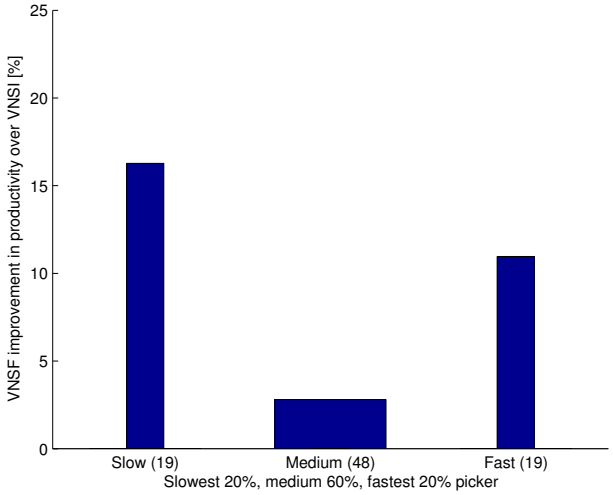

Figure 4: Change in average productivity by picker category. 7.4h minimum time to qualify, 20 pickers

heuristics 1 and 2 in addition to 3, 4, and 5, which just move batches between pickers, resulting in an additional 2% savings in batch execution time.

Most importantly, Figure 3 shows that significant savings in total batch execution time can be obtained by assigning the right batches to the right pickers. Almost 10% more time can be saved when comparing VNSF to CWI and to VNSI. Compared to the forecast and real time for the original allocation, between 10% and 12% can be saved. Finally, when comparing VNSF with BF, i.e., assigning based on picker skill vs. picker productivity, savings of 6% in total batch execution time can be achieved.

6.5 Impact on Picker Productivity

Figure 4 shows the effect of the VNSF solution compared to a VNSI solution, which implies state-of-the art batching with a *first-free* picker assignment. Pickers are divided into three categories based on their productivity (lines picked per time unit worked) in the VNSI solution: the slowest 20%, the medium 60%, and the fastest 20%, with a total of 20, 53, and 20 pickers, (19, 48, and 19 in Figure 4), respectively. As noted below, the slowest pickers receive much less work. Here productivity is calculated by the total number of assigned lines divided by the total time to process all assigned batches, i.e., actual work time.

All categories improve productivity, while most improvements can be observed in the slow and fast picker categories. For pickers in the slow, medium, and fast categories for the case of 29 pickers, the initial average productivity values are 1.14, 1.43, and 1.81 lines/minute, respectively. The VNSF solution improves productivity by 17%, 3%, and 8% on average compared to the VNSI for the same categories. For the 20-picker case, VNSI productivities are 1.12, 1.44, and 1.83 lines/minute, which improve by 16%, 3%, and 11% (Figure 4). The improvement in all categories is a result of a reduction in total working time compared to the initial allocation as picker skills are taken into consideration. Additionally, some pickers, particularly those in the slow category, are assigned less work, and in many cases do not get to work full shifts. Since VNSI is used as a reference, this improvement in productivity must result from better picker assignment. However, the slowest pickers receive fewer orders to be picked, resulting in 40% to 60% fewer pick lines, as can be seen from Figure 5. The medium and fast pickers tend to get more work in both the 29- and the 20-picker cases.

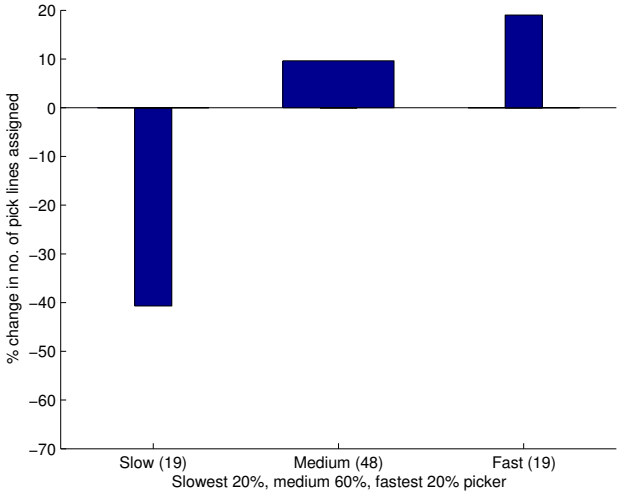

Figure 5: Change in number of lines assigned by picker category. 7.4h minimum time to qualify, 20 pickers.

To better understand which of the pickers are included in the ALNS assignment over the entire horizon of eight virtual days, we divide the picker categories used above into subcategories, indicated by two characters, XY. X stands for the main categories S, M, F (Slow, Medium, Fast), based on the productivity in the VNSI solution. Y has three categories, 0, 1, 2, indicating whether the picker: is left out of the assignment completely, or receives less than a fifth of his or her total number of order lines assigned in the VNSI solution, or receives more than a fifth of his or her total number of order lines assigned in the VNSI solution, respectively.

Figure 6 shows the division of pickers over these categories compared to VNSI for the 20-picker case. The pickers in the slow category are the most affected: in the 20-picker case, one picker is left out. In the 29-picker per day case, two out of 15 pickers are completely left without work for all eight days. In the medium category, five and four pickers are left out of the assignment, whereas all pickers in the fast category are assigned to work. Apparently, it pays off to not use some of the workers at all and to use some of the workers as little as possible.

As can be seen, the method leaves out some pickers, and assigns little work to some. This is an opportunity for the warehouse, as much of its workforce is hired on a daily basis from a (more or less) fixed labor pool. Based on the available daily orders, the warehouse manager can now decide better who to hire for the day, and for how many hours.

7 Discussion

This paper introduces a method to model and solve the combined batching, routing, and picker assignment problem, based on properties of the batches to be executed. The method consists of two parts: first the batch execution times are estimated with multilevel analysis based on properties of historical batches, such as total number of lines, distance between items, total mass and volume, and pick level. Second, the integrated batching, routing, and picker assignment problem is solved using an ALNS heuristic. The effect of total number of order lines on total batch time is characterized as general ability of the picker, that is not explained by other physical characteristics of a batch. It explains the

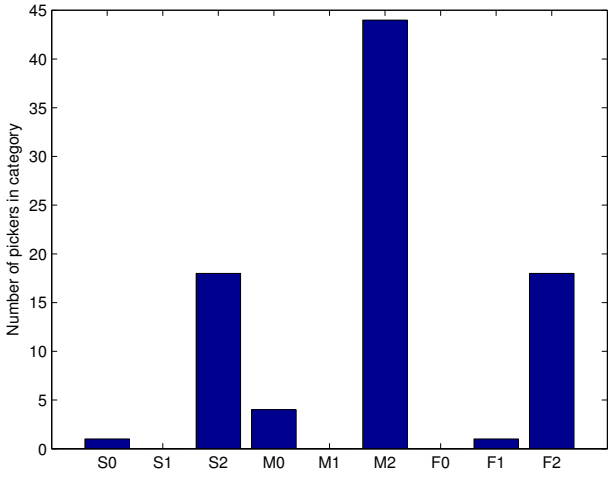

Figure 6: Picker categories after final assignment. 7.4h minimum time to qualify, an average of 20 pickers per virtual day for a total of 86 qualifying pickers in the experiment.

largest part of the variance in the data. The modeled picker skills of agility, picker strength, driving ability, preferred pick height, and ability to handle large volumes are all shown to be significant and to significantly impact the forecasting ability of the formed model. Picker models are clustered to illustrate the differences in the skills of the pickers, and to show that some of the pickers have similar skill sets.

We test the method for a large retail warehouse for which we could obtain three months of complete pick data. The multilevel modeling of the pickers shows that 13.1% of the variance in batch execution time can be explained by differences among pickers. Subsequent application of the ALNS heuristic shows that improvements of almost 10% in total batch execution time can be achieved compared to state-of-the art batching with VNS, but ignoring picker skills. Compared to batching with VNS and assignments made based on picker productivity, 6% is saved by considering picker skills.

Our method is generally applicable in picker-to-parts warehouses, independent of the layout. However, the choice of the routing heuristic should be made according to the case. The main requirement for applying our method is the availability of historical pick data to build the forecasting models of the pickers. These data are gathered in most modern warehouses, by pick-by-voice, pick-by-light, and pick-by-terminal systems. Using these data, Warehouse Management Systems can easily be extended by including forecasted pick times in the batching and assignment of batches.

Although we tested our method in only one warehouse, we expect that savings can be achieved in other picker-to-parts picking facilities. The magnitude of the savings will depend on several factors, such as the number of order lines in a batch (more lines yield a higher saving opportunity), the size of the pick force (more workers means more opportunity for savings), the product characteristics (larger variety in sizes and volume requires more stacking skills of the workers and potentially larger saving opportunities), and variation in the height levels at which the picking takes place. We leave this as a topic for further research.

Our approach has some limitations. The data obtained are extensive, but they do not contain the departure times of trucks shipping the orders. Therefore, we lack time windows for the orders, which are to some extent included in the real batch data. This means the calculated savings slightly overestimate the real savings as the search space for selecting the next orders to the batch shrinks. However, this only

affects savings compared to the original batching and allocation.

Our method implies that managers should consider employee skills and capabilities during the decision making process. Currently, worker to work assignment is primarily based on qualifications. However, when multiple workers qualify for the job, individual differences might still be exploited. Based on our experience in warehouses, when workers are qualified for the same work, the next job is assigned to the first available worker. It may pay off to select the right job from the stack to be executed by the best skilled worker. In addition, when hiring a flex workforce and past performance data are available, the warehouse manager can use them to hire the best performing pickers given a set of orders. Our findings suggest that impact of the right worker is almost 10% in execution time, or 6% if *apriori* knowledge of picker productivity is taken into account when assigning work.

Our study opens up an avenue of further research directions. First, the results should be further validated for various picker-to-parts warehouse types, with varying layouts, order, batch, and product properties. However, it is difficult to obtain realistic picker profile data from companies. If data logs are available, they are commonly huge, polluted, and prone to interpretation in order to construct valid models. Second, our method can be extended in a straightforward fashion to include order time windows. It would require adding some extra constraints and probably heuristics in the ALNS. It might also be extended to include online assignment of batches to pickers, in a rolling horizon setting. Third, although we can explain about 86% of the variance in batch execution time based on the current variables included, we may increase forecasting accuracy by including other, e.g., behavioral, variables. Based on previous studies (Bendoly et al., 2006, De Koster et al., 2011, De Vries et al., 2013), it is likely that also behavioral variables may have an additional effect. These can include motivational variables such as prevention and promotion focus, or personality traits. Fourth, the insights obtained in this study are not limited to picker-to-parts order picking situations. They are equally applicable to part-to-picker environments, where products are retrieved by machines and given to the right picker to execute the job. In general, any job type that can be parameterized and of which the execution times can be accurately forecasted based on these parameters for the different workers, qualifies for careful assignment. The classical vehicle routing problem is an example. Parcel carriers like DHL, TNT, UPS have to make many deliveries in different types of areas, urban as well as rural. Based on the types of deliveries, neighborhood, number of parcels, distances, and weight, it might pay off to select the best driver to fit the job circumstances.

A Note on Problem Complexity of the Joint Batching and Generalized Assignment Problem

The complexity of assigning orders to batches and batches to pickers can be illustrated by the number of combinations of a related bin-packing problem, where batches and pickers are both homogeneous. The number of possible batch combinations of up to the maximum batch size of N can be found from

$$C = \sum_{i=1}^{N} \begin{pmatrix} |\mathcal{O}| \\ i \end{pmatrix} \tag{16}$$

Let $|\mathcal{P}|$ be the total number of batches, and let L be the number of batches a picker can complete during a shift. Assuming $|\mathcal{P}| = |\mathcal{W}|L$, the total number of possible bin-packing combinations for $|\mathcal{W}|$ pickers

is

$$C_{assign} = \sum_{i=1}^{|\mathcal{W}|} \left(\begin{array}{c} |\mathcal{P}| - iL \\ L \end{array} \right). \tag{17}$$

A regular shift at the sample warehouse employs 30 pickers. Let one picker take half an hour to process one batch, and work eight hours each shift, for a maximum of 16 batches/shift. If all pickers and batches are to filled to capacity, there are 480 batches and 1440 orders (the maximum batch size is thus 3), and the number of joint bin-packing and order combinations is

$$C_{total} = \left(\begin{array}{c} 1440 \\ 3 \end{array} \right) \sum_{i=0}^{29} \left(\begin{array}{c} 480 - i16 \\ 16 \end{array} \right), \tag{18}$$

which is a very big number, approximately 10^{152}. To ground this number somewhat, the total number of atoms in the universe is estimated to be 10^{82}. Optimally solving the joint problem is impractical even for a small number of pickers.

References

Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., and De Blas, C. (2009). Variable neighborhood search for order batching in a warehouse. *Asia-Pacific Journal of Operational Research*, 26(05):655–683.

Bartholdi, J. J. and Eisenstein, D. D. (1996). A production line that balances itself. *Operations Research*, 44(1):21–34.

Baumann, H. (2013). *Order picking supported by mobile computing*. PhD thesis, University of Bremen.

Bendoly, E., Donohue, K., and Schultz, K. L. (2006). Behavior in operations management: Assessing recent findings and revisiting old assumptions. *Journal of Operations Management*, 24(6):737–752.

Berger, S. M. and Ludwig, T. D. (2007). Reducing warehouse employee errors using voice-assisted technology that provided immediate feedback. *Journal of Organizational Behavior Management*, 27(1):1–31.

Bliese, P. D. (2002). *Multilevel random coefficient modeling in organizational research: Examples using SAS and S-PLUS.* Jossey-Bass.

Bryk, A. S. and Raudenbush, S. W. (1992). *Hierarchical linear models: Applications and data analysis methods.* Sage Publications, Inc.

Campbell, G. M. and Diaby, M. (2002). Development and evaluation of an assignment heuristic for allocating cross-trained workers. *European Journal of Operational Research*, 138(1):9–20.

Cattrysse, D. G. and Van Wassenhove, L. N. (1992). A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Cohn, H. and Fielding, M. (1999). Simulated annealing: Searching for an optimal temperature schedule. *SIAM Journal on Optimization*, 9(3):779.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409.

Coyle, J., Bardi, E., and Langley, C. (1996). *The Management of Business Logistics*. West Publishing Company.

De Koster, R., Le-Duc, T., and Roodbergen, K. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

De Koster, R., Stam, D., and Balk, B. M. (2011). Accidents happen: The influence of safety-specific transformational leadership, safety consciousness, and hazard reducing systems on warehouse accidents. *Journal of Operations Management*, 29(7):753–765.

De Koster, R. and Van der Poort, E. (1998). Routing order pickers in a warehouse: a comparison between optimal and heuristic solutions. *IIE Transactions*, 30(5):469–480.

De Koster, R., Van der Poort, E., and Wolters, M. (1999). Efficient orderbatching methods in warehouses. *International Journal of Production Research*, 37(7):1479–1504.

De Vries, J., De Koster, R., and Stam, D. (2013). Safety does not happen by accident: How to manage a safe warehouse? *Working paper, Erasmus University, Rotterdam*.

Doerr, K. H. and Arreola-Risa, A. (2000). A worker-based approach for modeling variability in task completion times. *IIE Transactions*, 32(7):625–636.

Drury, J. (1988). Towards more efficient order picking. IMM Monograph 1, The Institute of Materials Management, Cranfield, U.K.

Duan, N. (1983). Smearing estimate: a nonparametric retransformation method. *Journal of the American Statistical Association*, 78(383):605–610.

Fan, J. and Huang, L.-S. (2001). Goodness-of-fit tests for parametric regression models. *Journal of the American Statistical Association*, 96(454):640–652.

Fisher, M. L., Jaikumar, R., and Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103.

Gademann, N. and Van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 37(1):63–75.

Gelman, A. and Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Gharehgozli, A., Laporte, G., Yu, Y., and De Koster, R. (2013). Scheduling twin yard cranes in a container block. *Transportation Science, forthcoming*.

Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Henn, S. and Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, 222(3):484–494.

Hong, S., Johnson, A. L., and Peters, B. A. (2012). Batch picking in narrow-aisle order picking systems with consideration for picker blocking. *European Journal of Operational Research*, 221(3):557–570.

Juran, D. C. and Schruben, L. W. (2004). Using worker personality and demographic information to improve system performance prediction. *Journal of Operations Management*, 22(4):355–367.

Kreft, I. and De Leeuw, J. D. (1998). *Introducing Multilevel Modeling*. Sage, London, UK.

Larco Martinelli, J. (2010). *Incorporating Worker-Specific Factors in Operations Management Models*. PhD thesis, Erasmus Research Institute of Management, Erasmus University Rotterdam.

Matusiak, M., de Koster, R., Kroon, L., and Saarinen, J. (2014). A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *European Journal of Operational Research*, 236(3):968 – 977. Vehicle Routing and Distribution Logistics.

Nakagawa, S. and Schielzeth, H. (2013). A general and simple method for obtaining R^2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, 4(2):133–142.

Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793.

Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer.

Powell, S. G. and Schultz, K. L. (2004). Throughput in serial lines with state-dependent behavior. *Management Science*, 50(8):1095–1105.

Randolph, W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, 25(4):76–87.

Ratliff, H. and Rosenthal, A. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521.

Roodbergen, K. and De Koster, R. (2001). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Theys, C., Braysy, O., Dullaert, W., and Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763.

Vaughan, T. (1999). The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37(4):881–897.

Weaver, K. A., Baumann, H., Starner, T., Iben, H., and Lawo, M. (2010). An empirical task analysis of warehouse order picking using head-mounted displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1695–1704. ACM.

ERIM Report Series *Research in Management*	
ERIM Report Series reference number	ERS–2015–008–LIS
Date of publication	2015–06–29
Version	29–06–2015
Number of pages	28
Persistent URL for paper	http://hdl.handle.net/1765/78318
Email address corresponding author	rkoster@rsm.nl
Address	Erasmus Research Institute of Management (ERIM) RSM Erasmus University / Erasmus School of Economics Erasmus University Rotterdam PO Box 1738 3000 DR Rotterdam, The Netherlands Phone: +31104081182 Fax: +31104089640 Email: info@erim.eur.nl Internet: http://www.erim.eur.nl
Availability	The ERIM Report Series is distributed through the following platforms: RePub, the EUR institutional repository Social Science Research Network (SSRN) Research Papers in Economics (RePEc)
Classifications	The electronic versions of the papers in the ERIM Report Series contain bibliographic metadata from the following classification systems: Library of Congress Classification (LCC) Journal of Economic Literature (JEL) ACM Computing Classification System Inspec Classification Scheme (ICS)