European Journal of Operational Research 242 (2015) 942–950

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Integer programming techniques for solving non-linear workforce planning models with learning

Mike Hewitt [a,*], Austin Chacosky [b], Scott E. Grasman [b], Barrett W. Thomas [c]

[a] Department of Information Systems and Operations Management, Loyola University Chicago, Chicago, IL 60611, USA
[b] Department of Industrial and Systems Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA
[c] Department of Management Sciences, University of Iowa, Iowa City, IA 52242, USA

ARTICLE INFO

Article history:
Received 30 October 2013
Accepted 28 October 2014
Available online 15 November 2014

Keywords:
Production planning and scheduling
Human learning
Nonlinear programming

ABSTRACT

In humans, the relationship between experience and productivity, also known as learning (possibly also including forgetting), is non-linear. As a result, prescriptive planning models that seek to manage workforce development through task assignment are difficult to solve. To overcome this challenge we adapt a reformulation technique from non-convex optimization to model non-linear functions with a discrete domain with sets of binary and continuous variables and linear constraints. Further, whereas the original applications of this technique yielded approximations, we show that in our context the resulting mixed integer program is equivalent to the original non-linear problem. As a second contribution, we introduce a capacity scaling algorithm that exploits the structure of the reformulation model and reduces computation time. We demonstrate the effectiveness of the techniques on task assignment models wherein employee learning is a function of task repetition.

1. Introduction

Peter Senge famously wrote that the learning organization is "continually expanding its capacity to create its future" (Senge, 2006). Citing employee learning as an effective way to create capacity from existing resources, Levinthal and March (1993) argue that effective management of employee learning can lead to a sustainable competitive advantage. Some go further and claim that, in the modern business environment, the only sustainable competitive advantage will come from a company's ability to learn more effectively than its competitors (Kapp, 1999). Even if it is not the only source of competitive advantage, Moustaghfir (2009) identifies learning as a core component of organizational capabilities that are "immune to competitive duplication." In this paper, we address the issue of learning at the operational level by developing solution techniques for workforce planning models that incorporate individual on-the-job learning.

In particular, this paper focuses on a class of multi-period workforce planning models that recognize human learning and forgetting over a fixed planning horizon in environments that can be characterized as task assignment. We consider a workforce that produces a product, the production of which requires the completion of a sequence of tasks. Inventory is allowed to accumulate in between each task in the sequence. We assume that the workforce's current skill levels and thus their productivity are heterogeneous. As workers perform a particular task, they gain experience that correspondingly leads to a productivity improvement as they learn on the job. In addition, when workers are not doing a particular task, they forget some of what they have learned and their productivity on that particular task erodes. We assume that each worker has his or her own rate of learning and forgetting for each task. The objective is to maximize production over a fixed time horizon.

In humans, the relationship between experience and productivity, also known as learning (possibly also including forgetting), is non-linear. Thus, prescriptive models incorporating learning are difficult to solve. This paper helps overcome this challenge in two ways.

First, it adapts a reformulation technique for non-convex optimization problems that enables linearization of the learning curves. For general non-convex optimization, the reformulation yields an approximation of the original non-linear program (NLP). However, in our application, the non-linear functions have a specific structure that has not previously been explored. Specifically, they are functions of the number of times an individual has performed a task over a fixed number of periods, and thus have discrete and finite domains. In this case, solutions to the mixed integer program (MIP) resulting from the reformulation are optimal for the original non-linear program. As a result, we can solve MIPs instead of NLPs, and with the superior

* Corresponding author. Tel.: +1 585 789 8034.
E-mail addresses: mhewitt3@luc.edu (M. Hewitt), atc7417@mail.rit.edu (A. Chacosky), segeie@rit.edu (S. E. Grasman), barrett-thomas@uiowa.edu (B. W. Thomas).
URL: http://tippie.uiowa.edu/barrett-thomas (B. W. Thomas)

http://dx.doi.org/10.1016/j.ejor.2014.10.060
0377-2217/

capabilities of MIP solvers to those of NLP solvers, we are able to solve problems that are significantly larger than what has been previously possible.

The reformulation technique associates a binary variable with each element of the domain of the non-linear function we wish to linearize. In our application this translates to representing each potential skill level for each worker in each period of the planning horizon with a binary variable. As a result, as instance sizes grow, the resulting MIPs become large, increasing solve times. Thus, as a second contribution, we derive structural properties of the feasible region of the MIP that in turn motivate a capacity scaling algorithm wherein these variables are generated dynamically. With an extensive computational study, we see that this algorithm reduces solution time. The capacity scaling algorithm can be applied to any MIP resulting from the reformulation technique.

The remainder of this paper is organized as follows. In Section 2, we review the relevant literature. Section 3 presents our reformulation technique, and Section 4 introduces the capacity scaling heuristic that exploits the structure of the reformulation. Section 5 presents the non-linear and reformulated math programs for the workforce planning model on which we perform computational tests. The section also demonstrates how to implement the capacity scaling algorithm with the reformulated workforce planning model. Using instances of different sizes and characteristics, Section 6 studies the computational times associated with the reformulated math program and the scaling algorithm. Section 7 offers conclusions and opportunities for future work.

2. Literature review

Given its importance, a significant body of research has been devoted to developing quantitative models of individual learning and forgetting. These models are often called learning curves. Of particular interest for this paper is research devoted to on-the-job or experiential learning, which is sometimes also called autonomous learning. Dar-El (2000) provides a comprehensive review of both learning and forgetting models as well as parameter estimation for learning models developed before 2000. Jaber and Sikström (2004) and Anzanello and Fogliatto (2011) provide overviews that account for more recent work on learning and forgetting models. Taking advantage of improved data collection capabilities resulting from bar code readers and other similar devices, Nembhard and Uzumeri (2000a) conclude that a three-parameter hyperbolic function provides best fits observations of individual learning. Nembhard and Uzumeri (2000b) extend the three-parameter hyperbolic function to incorporate forgetting. Nembhard (2001) examines several models of forgetting and identifies those with robust performance. Shafer, Nembhard, and Uzumeri (2001) introduce a model that accounts for the recency of a task. Jaber and Sikström (2004) provide a numerical analysis that, for three models, identifies in which particular applications each model has the best statistical fit of the learning and forgetting measured in the application. While our work is amenable to other models of learning and forgetting, we focus this work on the exponential function used in Nembhard and Norman (2007, chap. 4). The function was introduced by Thomas and Nembhard (2005) as an exponential learning and forgetting function designed to achieve the performance of the three-parameter hyperbolic function while allowing for improved tractability in optimization models.

A number of authors have incorporated models of learning or of learning and forgetting into optimization models for workforce management. While the existing work covers a variety of different applications, a common theme is the challenge in solving large-sized instances. The success of exact solution approaches has been particularly limited. Nembhard and Norman (2007) introduce a task-assignment model that includes learning and forgetting. Computational results are presented for a two workers, four tasks, and 10 time periods example. For ease of exposition, we choose a model similar to what is presented in Nembhard and Norman (2007) to illustrate our reformulation technique. Heimerl and Kolisch (2010) also consider an assignment model with the addition of constraints that represent the firm's desired skill composition at the end of a specified period of time. Kim and Nembhard (2010) use a non-linear mixed integer program to test the effects of experimental factors on cross-training policy selection. While the number of tasks is difficult to determine, the largest problem size considered has three workers and 24 periods. For the situation in which each station has infinite buffers and a problem similar to that studied in this paper, Nembhard and Bentefouet (2012) identify the structure of the optimal policy for the case where the number of tasks and workers is the same. The result allows the authors to solve the problem up to 96 workers, 96 tasks, and 246 periods. Using the flow-line production scenario used in this paper as an application and for small problems of two to three workers and two to three tasks, Bentefouet and Nembhard (2013) identify structural properties that characterize the optimal solution. The results do not generalize.

In addition to the specific structural results, however, Bentefouet and Nembhard (2013) present a model that provides an upper bound on the production that can be achieved from the application discussed in this paper. The nature of the formulation of the upper bound problems allows the authors to linearize the learning functions in a way that is a special case of the reformulation presented in Section 3. As noted in Bentefouet and Nembhard (2013), solutions to their presented model do not generally offer implementable task assignment schedules. The techniques discussed in this paper do offer implementable task assignments. The authors of this paper are not aware of any papers that offer a mixed integer reformulation of a non-linear function with a discrete domain.

To overcome the non-linearities of the learning and forgetting functions, a number of authors consider approximation schemes. In the literature that follows, none of the authors addresses how well their schemes approximate the learning and forgetting functions nor do they discuss whether or not the approximation impacts solution quality relative to using the actual learning and forgetting functions. Nembhard and Bentefouet (2012) introduce a rectangular approximation for learning/forgetting functions. For a problem similar to the application used in this paper, the approximation can solve 10 worker, 10 task, and 40 period problems. Corominas, Olivella, and Pastor (2010) introduce a piece-wise linear transformation of a convex learning function for a task assignment problem that incorporates learning resulting from experience with related tasks. The largest problem solved in that paper has five tasks, four workers, and 20 time periods. Olivella, Corominas, and Pastor (2013) combines piecewise linearization with constraint relaxation. Sayin and Karabati (2007) also use a piecewise linear approximation of the learning function, but in a problem in which they first try to maximize utility and then skill improvement. In addition to the linearization, the problem is solved for only a single period with each period's solution implemented as part of a simulation. Problems with at most 18 workers and four different skills are solved. Gutjahr, Katzensteiner, Reiter, Stummer, and Denk (2008) consider maximizing a weighted average of economic gains and skill development for a project selection problem. As part of the selection process, assignments to selected projects are optimized. Gutjahr et al. (2008) introduce a first-order approximation of the non-linear learning curve and are resultantly able to solve the approximate model to optimality for an example with 14 candidate projects requiring a mix of 40 skills (analogous to tasks in our discussion) with 28 workers over 24 periods.

A number of heuristic approaches to task assignment with learning can be found in the literature. Yan and Wang (2011) consider a model with the same learning function considered in this paper. They introduce a genetic algorithm and solve a problem with six

workers and five tasks over 40 periods. Other heuristic approaches can be found in Wirojanagud, Gel, Fowler, and Cardy (2007) and Fowler, Wirojanagud, and Gel (2008).

3. A mixed integer reformulation of a function with a discrete and finite domain

In this section, we present a technique that uses a set of binary variables and constraints to reformulate a non-linear function. This technique was first presented in Beale and Tomlin (1970) and more recently put into context in Burer and Letchford (2012). The technique was initially developed for general, non-convex optimization problems, and its application results in a mixed integer program that approximates the original non-linear program. Our use of the reformulation technique yields an exact reformulation (solving the resulting mixed integer program yields an optimal solution to the original non-linear program). Finally, while our motivation is non-linear learning curves, we provide a presentation that demonstrates the reformulation for any non-linear function with a discrete domain.

Let X be a vector of binary decision variables and $g(X)$ a linear function whose range is characterized by a discrete and finite set $\mathcal{K}$. As an example, in the workforce planning application we will consider later, $g(X)$ is the sum of variables that determine workers' accumulated experience. We consider a function $f(g(X))$, which, as in the application we discuss later, is such that $f(g(X)) > 0$, $\forall$ feasible vectors X. We do not assume a particular form for $f(\cdot)$, but are particularly interested in cases where it is non-linear. In our application, $f(\cdot)$ is a learning curve that takes workers' accumulated experience as input. Our goal is to model $f(\cdot)$ with linear constraints and binary variables so that we can solve a mixed integer program instead of a non-linear program.

Given our definition of $f(\cdot)$ and its discrete and finite domain, the range of the function $f(\cdot)$ can be enumerated (although we do not assume that the values in the range are integer). Thus, we introduce a single continuous variable r, with which we will model the value of $f(k)$, and a binary variable z_k for each potential value $k \in \mathcal{K}$. We then define constraints to ensure that when $z_k = 1$, r takes on the value $f(k)$. Formally, we use the following constraints:

$$r = \sum_{k\in\mathcal{K}} f(k) z_k, \tag{1}$$

$$g(X) = \sum_{k\in\mathcal{K}} k z_k, \tag{2}$$

$$\sum_{k\in\mathcal{K}} z_k = 1, \tag{3}$$

$$z_k \in \{0,1\}\ k \in \mathcal{K}, r \in \Re. \tag{4}$$

Together, constraints (1), (2), and (3) ensure that $r = f(g(X))$. Note that constraint (3) is necessary for the formulation to be correct. For example, consider the case in which $\mathcal{K} = \{1,2,3\}$. Without constraint (3), for values of X such that $g(X) = 3$, $z_1 = z_2 = 1$, $r = f(1) + f(2)$ would be feasible, when in fact we should have $r = f(3)$.

The correctness of the reformulation is given in the following theorem:

Theorem 1. *If $\bar{z}, \bar{r}$ satisfy constraints (1), (2), and (3) for $\bar{X} \in$ domain of $g(X)$, then $\bar{r} = f(g(\bar{X}))$.*

Proof. By definition, $g(\bar{X}) \in \mathcal{K}$, and there is a variable $\bar{z}_{g(\bar{X})}$ defined. Because exactly one variable z_k can equal one, we must have that $\bar{z}_{g(\bar{X})} = 1$ by constraint (2). As a result, by constraint (1) we have that $\bar{r} = f(g(\bar{X}))$. □

If $f(k) = a + h(k)$, we may rewrite constraint (1) as $r = a + \sum_{k\in\mathcal{K}} h(k) z_k$. Lastly, we note that when solving a maximization problem wherein the objective is non-decreasing in r we can relax constraint (1) to $r \le \sum_{k\in\mathcal{K}} f(k) z_k$ without changing the set of optimal solutions.

4. Capacity scaling algorithm

Capacity scaling is a well-known algorithmic strategy, often applied to network flow models, wherein an optimization problem that is similar to, yet easier to solve, than the original is repeatedly solved in order to produce high quality solutions to the original problem (Ahuja & Orlin, 1995). We use facet-defining valid inequalities of the above reformulation model to create these simpler problems.

Specifically, we consider the set

$$S = \left\{ z \in \{0,1\}^n, x \in \Re, y \in \Re : x = \sum_{j=1}^{n} a_j z_j, y \le \sum_{j=1}^{n} f(a_j) z_j, \sum_{j=1}^{n} z_j = 1, y \le b \right\} \tag{5}$$

and seek to find facets of its convex hull, $conv(S)$. As above, we have assumed that $f(a_j) > 0\ \forall j = 1,\ldots,n$. Thus, we limit our attention to cases wherein $b > 0$. We note, that in the definition of the set S, we have modeled $g(X)$ with the single variable x. We next show in Theorem 2 that, by tightening the right-hand side coefficients in the inequality $y \le \sum_{j=1}^{n} f(a_j) z_j$ based on the value of b to

$$y \le \sum_{j=1}^{n} \min(f(a_j), b) z_j, \tag{6}$$

one arrives at a facet of $conv(S)$.

Theorem 2. *The inequality $y \le \sum_{j=1}^{n} \min(f(a_j), b) z_j$ defines a facet of $conv(S)$.*

Proof. It is easy to see that the inequality is valid. We first prove that $dim(conv(S)) = n$. We note that the two equations defining S,

$$0 = x - \sum_{j=1}^{n} a_j z_j$$

and

$$1 = \sum_{j=1}^{n} z_j,$$

are linearly independent, and thus $dim(conv(S)) \le (n+2) - 2 = n$. We next consider the following n points $p_i, i = 1,\ldots,n$ in S, such that

$$p_i : z_i = 1, z_j = 0, j \ne i, x = a_i, y = \min(f(a_i), b)$$

and the point

$$p_0 : z_1 = 1, z_j = 0, j > 1, x = a_1, y = 0.$$

One can easily see that these $n+1$ points are linearly (and hence affinely) independent. Thus, we have that $dim(conv(S)) \ge n$ and ultimately that $dim(conv(S)) = n$. We next consider the face

$$F = \left\{ (z,y,x) \in S : y = \sum_{j=1}^{n} \min(f(a_j), b) z_j \right\} \tag{7}$$

induced by the valid inequality. We note that $p_0 \notin F$ and thus $F \subset conv(S)$ and $dim(F) < n$. However, the n points $p_i, i = 1,\ldots,n$ above are in F. We see that they are linearly independent and thus $dim(F) >= n-1$. Coupled with the observation that $dim(F) < n$, we have that $dim(F) = n-1$ and the inequality induces a facet. □

Algorithm 1 Capacity scaling algorithm.

Set iteration counter $k = 0$
Calculate initial value of $\underline{b}_0$
while do not stop **do**
  Solve $P(\underline{b}_k)$
  Update $\underline{b}_k$ to $\underline{b}_{k+1}$
  Set $k = k + 1$
end while

Next, we consider an optimization problem P resulting from the use of the reformulation technique presented in Section 3 and of the form

maximize y

subject to

$$y \leq \sum_{j=1}^{n} f(a_j) z_j,$$

$$x = \sum_{j=1}^{n} a_j z_j,$$

$$\sum_{j=1}^{n} z_j = 1,$$

$$(x, y) \in \mathcal{F},$$

where $\mathcal{F}$ is a set representing other constraints involving and potentially linking the x and y variables. Theorem 2 suggests that, by artificially constraining y, one may be able to solve the resulting optimization problem in much less time than required to solve P. With this insight, we define the problem $P(\underline{b})$ to be P, albeit with the extra constraint $y \leq \underline{b}$. We next present in Algorithm 1 a capacity scaling algorithm for problems of the form P where $P(\underline{b})$ is repeatedly solved for different values of $\underline{b}$.

We note that, if one knew an upper bound $\bar{y}$ on the optimal value of y (say by solving the linear relaxation of P), then when $\underline{b}_k \geq \bar{y}$, solving $P(\underline{b}_k)$ is equivalent to solving P. Thus, Algorithm 1 could be converted to an exact algorithm by first determining $\bar{y}$ and then iterating until $\underline{b}_k \geq \bar{y}$. Also, if we always have $\underline{b}_k \leq \underline{b}_{k+1}$ then the solution to $P(\underline{b}_k)$ will be feasible for $P(\underline{b}_{k+1})$.

To offer a concrete example of the techniques presented in this section, we next present a non-linear task assignment model that we will ultimately reformulate to a mixed integer program (MIP). We also provide details for how to apply Algorithm 1 to the resulting MIP.

5. A task assignment model with human learning

We consider a model that prescribes the assignment of individuals to tasks over a discretized planning period, where the tasks are ordered sequentially such that task j requires the finished work of task $j-1$ and the objective is to maximize the output of finished product (the last task, $|\mathcal{J}|$). The rate at which an individual is able to produce output of task j in period τ is governed by a functional model of learning and forgetting. The input to this function is the number of times the individual has performed task j in periods up to and including τ.

We denote the set of individuals by $\mathcal{I}$, the set of tasks by $\mathcal{J}$, and the periods in the planning horizon by $\mathcal{T}$. We assume that production begins in period 1. We define the binary variable x_{ij}^t to indicate whether individual $i \in \mathcal{I}$ performs task $j \in \mathcal{J}$ in period $t \in \mathcal{T}$, the continuous variable o_{ij}^t to indicate the amount produced by individual $i \in \mathcal{I}$ of task $j \in \mathcal{J}$ in period $t \in \mathcal{T}$, and the continuous variable r_{ij}^t to indicate the productivity rate of individual $i \in \mathcal{I}$ at task $j \in \mathcal{J}$ in period $t \in \mathcal{T}$. Finally, we define the continuous variable b_j^t to indicate the amount of work-in-process inventory available at task $j \in \mathcal{J}$ at the end of period $t \in \mathcal{T}, t \geq 2$. We let b_j^0 denote the initial inventory available at task $j \in \mathcal{J}$.

In this work, we illustrate our reformulation using the following learning and forgetting curve:

$$r_{ij}^{\tau}\left(\sum_{t=1}^{\tau} x_{ij}^t\right) = I_{ij} + K_{ij}\left[1 - e^{-\frac{\sum_{t=1}^{\tau} x_{ij}^t}{L_{ij}}}\right] e^{\frac{\sum_{t=1}^{\tau} x_{ij}^t - \tau}{F_{ij}}}. \tag{8}$$

While we illustrate our techniques on the learning function found in Nembhard and Norman (2007), they can be applied to others, including the hyperbolic learning function used in Sayin and Karabati (2007), the logistic learning function used in Gutjahr et al. (2008), and the exponential learning function of Heimerl and Kolisch (2010) (if assignments are restricted to being binary). The curve represents the productivity of individual i on task j in period τ as a function of i's experience on task j up to time τ. Individual i's experience level in period τ is dictated by $\sum_{t=1}^{\tau} x_{ij}^t$ or the number of repetitions individual i has had on task j by that time. In the τ periods, individual i also has $\sum_{t=1}^{\tau} x_{ij}^t - \tau$ periods in which forgetting occurs. For worker i and job j, I_{ij} is i's initial expertise at task j, K_{ij} is i's steady-state production rate for task j, L_{ij} is i's learning rate for task j, and F_{ij} is i's forgetting rate for task j. In practice, parameters I_{ij}, K_{ij}, L_{ij}, and F_{ij} are statistically determined based on observations collected on individual i on task j or a related task.

The Assignment with Learning (AwL) model, which was first presented in Nembhard and Norman (2007), seeks to

$$\text{maximize} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} o_{i|\mathcal{J}|}^t$$

subject to

$$b_1^1 = b_1^0 - \sum_{i \in \mathcal{I}} o_{i1}^1 \tag{9}$$

$$b_1^t = b_1^{t-1} - \sum_{i \in \mathcal{I}} o_{i1}^t \quad \forall t \in \mathcal{T}, t \geq 2, \tag{10}$$

$$b_j^t = b_j^{t-1} + \sum_{i \in \mathcal{I}} o_{ij-1}^t - \sum_{i \in \mathcal{I}} o_{ij}^t \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, t \geq 2, j \geq 2 \tag{11}$$

$$b_j^1 = b_j^0 + \sum_{i \in \mathcal{I}} o_{ij-1}^1 - \sum_{i \in \mathcal{I}} o_{ij}^1 \quad \forall j \in \mathcal{J}, j \geq 2, \tag{12}$$

$$\sum_{j \in \mathcal{J}} x_{ij}^t \leq 1 \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, \tag{13}$$

$$\sum_{i \in \mathcal{I}} x_{ij}^t \leq 1 \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{14}$$

$$o_{ij}^t \leq r_{ij}^t x_{ij}^t \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{15}$$

$$r_{ij}^t = I_{ij} + K_{ij}[1 - e^{-\frac{1}{L_{ij}} \sum_{t=1}^{\tau} x_{ij}^t}] e^{\frac{1}{F_{ij}}(\sum_{t=1}^{\tau} x_{ij}^t - (\tau))} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, \tau \in \mathcal{T} \tag{16}$$

$$x_{ij}^t \in \{0, 1\}, o_{ij}^t \in \Re_+, r_{ij}^t \in \Re_+ \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{17}$$

$$b_j^t \in \Re_+ \quad \forall j \in \mathcal{J}, t \in \mathcal{T}. \tag{18}$$

The objective maximizes the output of finished product (task $|\mathcal{J}|$). Constraints (9) and (10) represent the relationship between inventory levels and production quantities at the first task. Constraints (11) define the relationship between production and inventory at all except the first task, with initial conditions considered in constraints (12). Constraints (13) ensure that each individual performs at most

one task in a period and constraints (14) ensure that at most one individual can do each task in a period. Constraints (15) bound an individual's output of a task in a period by their productivity rate (we assume the rates are normalized such that all periods are of length one). Constraints (16) dictate an individual's productivity rate at a task in a period based on prior performance of that task. Finally, constraints (17) and (18) restrict the domains of the decision variables.

Before discussing whether this model is computationally tractable, we first make an observation. This model allows for, and even encourages "practicing." That is, an individual can perform a task ($x_{ij}^t = 1$) without producing any output ($o_{ij}^t = 0$). While such a combination is not beneficial when learning is not recognized, in this model, practicing can lead to greater output. The model can be adapted to disallow practicing by assuming a parameter μ that represents the minimum amount of output an individual can possibly produce and adding the constraint

$$\mu x_{ij}^t \leq o_{ij}^t.$$

Because of the nonlinearities in the constraints, particularly in constraints (16), instances of this model can prove very difficult to solve for all but small instances. For example, we developed a test set of instances with five workers, 10 tasks, and 10 periods, and found that KNITRO 7.0 (Waltz & Plantenga, 2009) was unable to produce a feasible solution for any of them in 2 hours of execution.

5.1. Applying the reformulation

In this section, we present a reformulation of AwL using the technique described in Section 3. Due to its discrete nature, the learning and forgetting function given in Eq. (8) is a candidate for our reformulation technique. Productivity is a function of the number of times individual i has performed task j in periods up to and including τ (the $\sum_{t=1}^{\tau} x_{ij}^t$ and $\sum_{t=1}^{\tau} x_{ij}^t - \tau$ terms in Eq. (8)). Thus, in period τ, we can compute the productivity rate of the worker knowing only how many times the task was performed in periods $t = 1, \ldots, \tau$. As there are only τ possible values for how many times the task was performed, we can enumerate the potential values of r_{ij}^{τ} and apply the reformulation technique.

We next present our reformulation, called the Assignment with Reformulated Learning model (AwRL), of the AwL. As noted previously, for a fixed time period, the domain of Eq. (8) is the possible numbers of times a task has been performed for an individual and is discrete and finite. As a result, the range of Eq. (8), the potential productivity rates, is also discrete and finite. Thus, for constraints (16), we can enumerate the set of potential productivity rates for individual i at task j in period t (there are t of them given that at most one task may be done per period) and define the binary variable z_{ij}^{kt} to indicate whether individual i is prescribed the productivity rate $\bar{r}_k$ at task j in period t.

Specifically, assume we have calculated the pairs $(k, \bar{r}_{ij}^{kt})$ where $\bar{r}_{ij}^{kt} = I_{ij} + K_{ij}[1 - e^{-\frac{k}{L_{ij}}}]e^{\frac{k-t}{F_{ij}}}$ and $\bar{r}_{ij}^{tt} = I_{ij} + K_{ij}(1 - e^{-\frac{t}{L_{ij}}})$ is used as an upper bound in a constraint that linearizes constraints (15). We reformulate the previous model, using the variables z_{ij}^{kt} by removing constraint sets (15) and (16) and adding the following:

$$o_{ij}^t \leq \bar{r}_{ij}^{tt} x_{ij}^t \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{19}$$

$$o_{ij}^t \leq r_{ij}^t \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{20}$$

$$r_{ij}^t = \sum_{k=1}^{\tau} \bar{r}_{ij}^{kt} z_{ij}^{kt} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, \tag{21}$$

$$\sum_{k=1}^{\tau} k z_{ij}^{k\tau} \leq \sum_{t=1}^{\tau} x_{ij}^t \quad \forall i \in \mathcal{I}, \tau \in \mathcal{T}, \tag{22}$$

$$\sum_{k=1}^{\tau} z_{ij}^{kt} = x_{ij}^t \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}. \tag{23}$$

Constraints (19) linearize constraints (15) based on the maximum possible productivity rate for individual i at task j in period t. Constraints (20) bound the output of an individual at a task by their productivity rate at that task in that period. Constraints (21) and (22) ensure that the productivity rate of an individual in period τ corresponds to the number of times they performed the task in the periods up to and including τ. Finally, constraints (23) ensure that each individual is assigned exactly one rate in each period. We derive the values $\bar{r}_{ij}^{kt}$ that parameterize our reformulation through enumeration.

5.2. Implementing the capacity scaling algorithm

We next discuss how we adapt the capacity scaling algorithm presented in Section 4 to produce high quality solutions to the AwRL. We first relax the constraints (21) to $r_{ij}^t \leq \sum_{k=1}^{\tau} \bar{r}_{ij}^{kt} z_{ij}^{kt}$. We then enforce artificial bounds $\underline{b}_{ij}^{t\nu}$ on r_{ij}^t through the constraints

$$r_{ij}^t \leq \sum_{k=1}^{\tau} \min(\bar{r}_{ij}^{kt}, \underline{b}_{ij}^{t\nu}) z_{ij}^{kt} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T} \tag{24}$$

in the hope that, because by Theorem 2 constraints (24) are facet-defining, the resulting integer program will solve quickly.

Thus, we let AwRL($\underline{b}$) denote the AwRL with the constraints (24) defined by the vector $\underline{b}$. We then iteratively create the vector $\underline{b}$ and solve the resulting instance of the AwRL($\underline{b}$). While any task assignment that is feasible for the AwRL is also feasible for the AwRL($\underline{b}$) (and vice-versa), constraints (24) may lead a task assignment to have a lower objective function value in the AwRL($\underline{b}$) than in the AwRL.

To create the value $\underline{b}_{ij}^{t\nu}$ at iteration ν of the heuristic, we choose an experience level $c_{ij}^{t\nu}$ and then set $\underline{b}_{ij}^{t\nu} = r_{ij}^{c_{ij}^{t\nu} t} = r_{ij}^t(c_{ij}^{t\nu})$. Thus, we artificially constrain the impact learning can have on an individual's productivity rate. To generate the initial values c_{ij}^{t1}, we solve the optimization problem

$$\text{maximize} \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} o_{i|\mathcal{J}|}^t,$$

subject to constraints (9), (10), (11), (12), (13), (14), (19), and the variable definitions in constraints (17) and (18). This problem, which we call TA-MAX, is the task assignment model without learning, wherein workers can always produce at their maximum productivity rate. We then set $c_{ij}^{t1} = \sum_{\tau=1}^{t} x_{ij}^{\tau}$ and $\underline{b}_{ij}^{t1} = r_{ij}^t(c_{ij}^{t1})$. We also note that, as TA-MAX is a relaxation of the AwRL, it also provides a bound on the optimal value of the AwRL.

We present the Capacity Scaling for the AwRL (CS-AwRL) algorithm in Algorithm 2. In the for-loop entered in Step 5, the allowed experience level is updated based on the most recent task assignment. However, because there is no guarantee that the task assignment will change from one iteration to the next, we include the if-statement in Step 10 wherein all allowable experience levels are increased by one level. We then update the maximum allowable production rates in Step 18 and repeat.

With the bound z^{UB} provided by solving TA-MAX, we can calculate both absolute and relative gaps for each primal solution found. If tolerances are given for each of those gaps, we can terminate the CS-AwRL when one of those gaps is within the stated tolerance. Similarly, if the condition in Step 12 is never true, *i.e.* the level of learning allowed for each individual on each task in each period is not artificially constrained, then solving AwRL($\underline{b}^{\nu}$) is equivalent to solving

Algorithm 2 CS-AwRL.

1: Solve TA-MAX to get bound z^{UB}
2: Set $c_{ij}^{t1} = \sum_{\tau=1}^{t} x_{ij}^{\tau}$ and $\underline{b}_{ij}^{t1} = r_{ij}^{t}(c_{ij}^{t1})$
3: **while** do not stop **do**
4: Solve AwRL($\underline{b}^{\nu}$) for task assignment solution x^{ν}
5: **for all** $i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}$ **do**
6: **if** $\sum_{\tau=1}^{t} x_{ij}^{\tau\nu} > c_{ij}^{t\nu}$ **then**
7: Set $c_{ij}^{t\nu+1} = c_{ij}^{t\nu} + 1$
8: **end if**
9: **end for**
10: **if** $c_{ij}^{t\nu+1} = c_{ij}^{t\nu} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}$ **then**
11: **for all** $i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}$ **do**
12: **if** $c_{ij}^{f\nu} + 1 < t + 1$ **then**
13: Set $c_{ij}^{t\nu+1} = c_{ij}^{t\nu} + 1$
14: **end if**
15: **end for**
16: **end if**
17: **for all** $i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}$ **do**
18: Set $\underline{b}_{ij}^{t\nu+1} = r_{ij}^{t}(c_{ij}^{t\nu+1})$
19: **end for**
20: **end while**

AwRL and the algorithm can terminate. Because of Steps 18 and 12 in each iteration, at least one rate will be increased unless they have all reached their upper limit. Thus, the CS-AwRL is an exact algorithm.

6. Computational analysis

To study the computational effectiveness of our reformulation technique and scaling algorithm, we solve instances of the AwRL using Gurobi 5.1 (Gurobi Optimization, 2012) to a relative optimality tolerance of 1 percent and an absolute optimality tolerance of 1. Experiments were performed on a computer cluster where each node has 32–64 cores with AMD Opteron 2.2 gigahertz or AMD Interlagos 2.6 "bulldozer" processors and 128–256 gigabytes of memory. When solving instances of the AwRL, Gurobi was given a time limit of 3600 seconds. All computational times reported are in seconds.

We give the instance sizes we consider (in terms of $\mathcal{I}, \mathcal{J}, \mathcal{T}$) in Table 1.

We only consider instances where there are at least as many tasks as workers ($|\mathcal{J}| \geq |\mathcal{I}|$) and at least as many periods as tasks ($|\mathcal{T}| \geq |\mathcal{J}|$). We initially report results for instances where the initial inventory levels are set as $b_1^1 = |\mathcal{T}|, b_j^1 = 5, j \geq 2$, but study the impact of this parameter on instance solve time later. We chose $b_1^1 = |\mathcal{T}|$ because productivity rates in most instances would typically be less than one and thus with this much raw material the production line will not be starved.

Table 1
Instance sizes.

$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{T}\rvert$
5,10,15,20	5,10,15,20,25,30,35,40	10,15,20,25,30,35,40

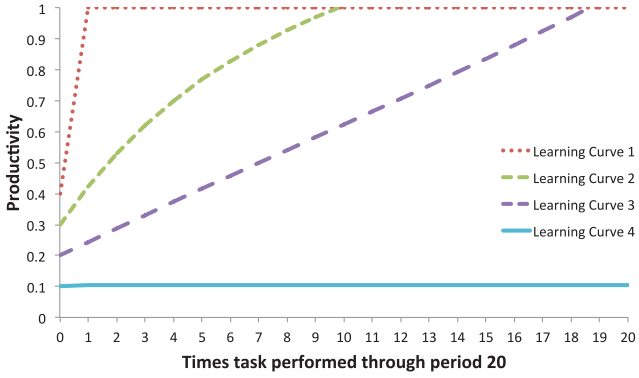
Along with studying the time required to solve instances of the AwRL of varying sizes, we also consider different workforce compositions with respect to the learning factors l_{ij}, K_{ij}, F_{ij}, and L_{ij}. We derive six different workforce composition cases from the four learning curves depicted in Fig. 1a, with parameter values detailed in Fig. 1b, and describe these cases in Table 2.

Our primary purpose for constructing these learning curves is to see the performance of the reformulation both in settings that are amenable to approximation techniques and those that are not. For example, in learning curves 1 and 4, the number of times an individual performs a task has little or no impact on future productivity. Thus, one way to deal with the non-linearity of the learning function would be to simply ignore learning and fix each individual's productivity rates *a priori*. In curve 3, learning occurs, but in a way that is nearly linear, suggesting that a piecewise linear approximation could be used (and very accurate). Finally, in curve 2, learning occurs in a way that a piecewise linear approximation would not be as accurate.

Symmetry is an issue that often plagues branch-and-bound-based solution methods for integer programming formulations of scheduling problems. Thus, we have chosen these six workforce compositions to see whether our reformulated model exhibits similar behavior. Specifically, Cases 1 through 4 and 6 represent homogeneous workforces, wherein symmetry could be an issue. Case 5 represents a heterogenous workforce, but assumes that the learning rate is independent of the task. Case 6 represents a homogenous workforce, but assumes that the learning rate does depend on the task. Thus, one would expect that symmetry will be less of an issue with Case 5.

6.1. Effectiveness of reformulation

Table 3 reports the average solution times (Solve time) and absolute optimality gaps (Abs. Gap) reported by Gurobi. We focus on absolute gaps because the objective function values in our instances are often small enough that we believe relative gaps do not give an accurate picture of the quality of solutions produced. We report



(a) Learning curves

Learning curve	I	K	L	F
1	.40	1	1	1,000
2	.30	1	5	50
3	.20	1	10	25
4	.10	1	50	10

(b) Parameter values

Fig. 1. Different learning curves considered.

Table 2
Workforce compositions used to derive instances.

Workforce composition	Description
Case 1	All workers learn each task according to learning curve 1
Case 2	All workers learn each task according to learning curve 2
Case 3	All workers learn each task according to learning curve 3
Case 4	All workers learn each task according to learning curve 4
Case 5	The $\mathcal{I}$ workers are divided into 4 (nearly) equal sized groups Workers in group g learn each task according to learning curve g
Case 6	The $\mathcal{J}$ tasks are divided into 4 (nearly) equal sized groups All workers learn each task in group g according to learning curve g

Table 3
Performance by worker composition and number of workers.

	5 workers		10 workers		15 workers		20 workers	
Worker composition	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap
Case 1	1385.80	5.15	118.74	0.11	233.77	0.10	619.62	0.13
Case 2	1367.79	4.04	232.63	0.25	466.67	0.30	902.16	0.23
Case 3	1032.48	1.60	196.22	0.14	397.20	0.37	776.41	0.23
Case 4	19.00	0.14	79.61	0.14	145.98	0.14	646.03	0.06
Case 5	2651.58	5.29	936.68	0.77	695.84	0.77	1407.14	1.00
Case 6	26.75	0.13	90.87	0.12	205.79	0.08	425.64	0.15
Average	1080.57	2.73	275.79	0.25	357.54	0.29	796.17	0.30

Table 4
Performance by number of periods and number of workers.

	5 workers		10 workers		15 workers		20 workers	
$\lvert\mathcal{T}\rvert$	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap
10	0.42	0.00	0.99	0.08	–	–	–	–
15	1.16	0.21	2.88	0.08	7.59	0.19	–	–
20	5.18	0.27	9.65	0.23	19.84	0.27	34.55	0.20
25	573.50	0.41	28.69	0.14	64.13	0.20	92.67	0.07
30	675.95	0.70	77.20	0.28	137.11	0.28	294.81	0.21
35	1893.62	3.26	262.98	0.18	377.29	0.26	754.97	0.21
40	2202.03	8.08	801.11	0.45	805.64	0.41	1563.66	0.53

Table 5
Performance by number of tasks and number of workers.

	5 workers		10 workers		15 workers		20 workers	
$\lvert\mathcal{J}\rvert$	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap
5	345.71	0.53	–	–	–	–	–	–
10	835.71	1.96	45.95	0.15	–	–	–	–
15	963.63	2.72	195.78	0.27	169.41	0.25	–	–
20	1111.70	2.20	253.03	0.22	279.34	0.24	436.94	0.29
25	1478.85	3.38	351.41	0.52	334.45	0.18	730.47	0.42
30	1841.25	4.77	495.39	0.16	465.51	0.33	874.58	0.24
35	2171.10	7.28	586.22	0.20	569.07	0.28	1239.33	0.32
40	2428.28	8.31	896.42	0.27	1222.79	1.20	1733.53	0.05

these results by number of workers and worker composition cases, averaging over instances with different numbers of tasks and periods. We observe that Gurobi is able to produce high-quality solutions and is particularly effective with homogeneous workforces, suggesting that symmetry is not an issue when solving instances of the reformulated model. We also see that the approach is robust with respect to the underlying learning curve. However, instances that represent a heterogeneous workforce (Case 5) are the hardest to solve.

Table 4 reports solver performance by number of periods ($|\mathcal{T}|$), where we average over instances with different numbers of tasks, and Table 5 reports solver performance by number of tasks ($|\mathcal{J}|$), where we average over instances with different numbers of periods. We see that with 10, 15, or 20 workers we are able to solve all instance sizes to within our tolerances. While solve time grows much more rapidly with respect to number of tasks than number of periods this growth is likely due to the nature of our set of instances, *i.e.* for instances with 30 tasks we only consider planning horizons of 30, 35, or 40 periods, whereas for instances with 40 periods we consider all numbers of tasks given in Table 1.

We next examine the impact of initial buffer levels on instance solve time. Specifically, we consider the same set of instances but with the initial buffer levels $b_1^0 = |\mathcal{T}|$ (to not starve the line), and for later stations, $b_j^0 = 1, j \geq 2$. We report aggregate solve times and absolute gaps over all instances in Table 6. We see that initial buffer levels have a significant impact on solve time and solution quality.

Ultimately, recalling our computational study with KNITRO, wherein KNITRO was unable to produce a feasible solution in two hours for instances with five workers, 10 tasks, and 10 periods, we conclude from Tables 4, 5, and 6 that solving instances of the AwRL

Table 6
Performance by number of workers and initial buffer level.

	5 workers		10 workers		15 workers		20 workers	
Initial buffer	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap	Solve time (seconds)	Abs. Gap
1	2588.33	5.60	2294.49	7.37	2333.39	8.21	2494.34	9.74
5	1080.57	2.73	275.79	0.25	357.54	0.29	796.17	0.30

Table 7
Root gap by initial buffer level.

	5 workers	10 workers	15 workers	20 workers
1	5.63	7.38	8.22	9.74
5	2.73	0.25	0.29	0.30

Table 10
Experience levels enumerated by CS-AwRL.

Initial buffer	Number iterations	Pct. exp. levels	Pct. exp. levels for best
1	48.65	0.34	0.17
5	75.67	0.51	0.15

is much less computationally intensive than the original non-linear program.

We next seek to understand why initial buffer levels and the number of workers have such an impact on solution time. One diagnostic for the difficulty in solving an integer program is the gap between the optimal value of the linear programming relaxation solved at the root node of a branch-and-bound-based algorithm and the final optimal solution, as this gives a sense of how often branching is likely to be required to solve the integer program. Thus, Table 7 presents averages over all instances for each initial buffer level of the absolute gap (Root gap) between the bound produced at the root node and the final primal solution produced. These two parameters (initial buffer levels and the number of workers) partially determine the production capability for an instance. Comparing the two rows, we conclude that the greater the production capability of an instance the closer the root node bound is to the value of the optimal solution. A similar observation can be made when comparing the columns of the row containing results for an initial buffer equal to five for stations after the first.

6.2. *Effectiveness of the CS-AwRL*

We next compare the effectiveness of the CS-AwRL to solving the AwRL for both initial buffer levels. To do so, we execute the CS-AwRL for 30 minutes (the only stopping criteria) for each instance and report the results, again for different initial buffer levels, in Tables 8 and 9. First, because the CS-AwRL is an exact algorithm, we present the percentage of instances the CS-AwRL solves to optimality in the time limit (Pct. solved). Next, we report the average absolute optimality gap (Abs opt gap) of the best solution produced by the CS-AwRL as measured against the dual bound produced when solving the AwRL and the time required to find that solution (Time best). The results show that CS-AwRL is able to produce high-quality solutions quickly and solve some instances to optimality.

We also compare in Tables 8 and 9 the performance of the CS-AwRL with solving the AwRL. Specifically, we present the percentage of instances for which the CS-AwRL produces an equivalent or better solution (Pct. beat/tie AwRL), the average of the time required to produce an equivalent or better solution (Time beat/tie AwRL), and the absolute gap in the two solutions (Abs gap AwRL). Across all instances, the CS-AwRL beats or ties the reformulation alone 88 percent of the time in both the one and five buffer cases. Further, CS-AWRL is able to achieve these results faster than is possible for the reformulation alone.

Finally, we note that, at any iteration, the CS-AwRL works with a limited set of potential experience levels, with new levels added at each iteration until the algorithm can conclude that all necessary levels have been enumerated or the time limit has been reached. We report in Table 10 the average number of iterations executed by the CS-AwRL (Number iterations), the average number of the percentage

Table 8
Capacity scaling heuristic—initial buffer = 5.

Worker composition	Pct. solved	Abs opt gap	Time best	Pct. beat/tie AwRL	Time beat/tie AwRL	Abs gap AwRL
Case 1	0.19	0.32	446.96	0.81	212.06	1.57
Case 2	0.22	0.89	538.29	0.88	337.87	0.71
Case 3	0.25	1.11	523.00	0.85	305.46	−0.39
Case 4	0.26	0.04	253.18	0.98	235.19	0.09
Case 5	0.14	1.14	718.11	0.76	219.66	1.27
Case 6	0.27	0.11	266.80	0.98	250.55	0.01
Average	0.22	0.60	457.72	0.88	260.13	0.54

Table 9
Capacity scaling heuristic—initial buffer = 1.

Worker composition	Pct. solved	Abs opt gap	Time best	Pct. beat/tie AwRL	Time beat/tie AwRL	Abs gap AwRL
Case 1	0.13	1.30	958.32	0.95	82.85	10.67
Case 2	0.12	3.72	1238.25	0.91	130.35	5.88
Case 3	0.11	4.20	1095.56	0.94	213.16	3.19
Case 4	0.18	2.18	748.67	0.82	310.05	0.58
Case 5	0.03	4.24	1506.27	0.94	226.87	6.23
Case 6	0.21	1.50	651.58	0.74	276.57	−0.01
Average	0.13	2.86	1033.11	0.88	206.64	4.42

of all possible experience levels generated at termination (Pct. exp. levels), and the average number of the percentage of experience levels generated when the best solution was found (Pct. exp. levels for best). We again report these results by initial buffer level. We conclude from these results that the CS-AwRL is able to produce high quality solutions with very few potential experience levels.

7. Conclusions and future work

In this paper, we adapt a technique for modeling a (non-linear) function with binary variables and linear constraints to a case where the function has a finite and discrete domain. We show that, in this case, solving the resulting mixed integer program yields an optimal solution to the original non-linear program (unlike the case for general, non-convex optimization problems wherein the resulting MIP is an approximation). With this technique, workforce planning models that contain non-linear models of human learning can be reformulated as mixed integer programs. Our computational study indicates that the resulting instances are much easier to solve and thus much larger instances can be solved than when the original non-linear program is solved with commercial software. We also use the resulting mixed integer program to develop an integer programming-based exact algorithm that can quickly produce a high quality solution.

While this reformulation technique does reduce solve times and enable larger instances to be solved, there is room for improvement. Our results indicate that strengthening the formulation with valid inequalities may greatly reduce solution times for instances that are tightly capacitated. Further, this paper is intended to demonstrate the general value of the reformulation technique. We have not addressed any particular workforce planning problem, and future studies demonstrating the impact of the reformulation on solving such practical problems would be valuable. Also, we are developing models that are similar to the AwRL, but that recognize there may be uncertainty with respect to a worker's learning parameters.

Acknowledgments

We would like to thank the Associate Editor and two anonymous referees for their helpful comments. This material is based upon work supported by the National Science Foundation under Grant No. CMMI-1266010.

References

Ahuja, R. K., & Orlin, J. B. (1995). A capacity scaling algorithm for the constrained maximum flow problem. *Networks*, *25*(2), 89–98.

Anzanello, M. J., & Fogliatto, F. S. (2011). Learning curve models and applications: Literature review and research directions. *International Journal of Industrial Ergonomics*, *41*(5), 573–583. doi: 10.1016/j.ergon.2011.05.001.

Beale, E. M. L., & Tomlin, J. A. (1970). Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence (Ed.), *Proceedings of the 5th national conference in operational research* (pp. 447–454). London: Tavistock.

Bentefouet, F., & Nembhard, D. A. (2013). Optimal flow-line conditions with worker variability. *International Journal of Production Economics*, *141*(2), 675–684. doi: 10.1016/j.ijpe.2012.10.008.

Burer, S., & Letchford, A. N. (2012). Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, *17*(2), 97–106.

Corominas, A., Olivella, J., & Pastor, R. (2010). A model for the assignment of a set of tasks when work performance depends on experience of all tasks involved. *International Journal of Production Economics*, *126*(2), 335–340. doi: 10.1016/j.ijpe.2010.04.012.

Dar-El, E. M. (2000). *Human learning: From learning curves to learning organizations*. Boston: Kluwer Academic Publishers.

Fowler, J., Wirojanagud, P., & Gel, E. (2008). Heuristics for workforce planning with worker differences. *European Journal of Operational Research*, *190*(3), 724–740. doi: 10.1016/j.ejor.2007.06.038.

Gurobi Optimization (2012). *Gurobi reference manual: Technical report*. Gurobi Optimization.

Gutjahr, W. J., Katzensteiner, S., Reiter, P., Stummer, C., & Denk, M. (2008). Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research*, *16*(3), 281–306. doi: 10.1007/s10100-008-0057-z.

Heimerl, C., & Kolisch, R. (2010). Work assignment to and qualification of multi-skilled human resources under knowledge depreciation and company skill level targets. *International Journal of Production Research*, *48*(13), 3759–3781. doi: 10.1080/00207540902852785.

Jaber, M. Y., & Sikström, S. (2004). A numerical comparison of three potential learning and forgetting models. *International Journal of Production Economics*, *92*(3), 281–294. doi: 10.1016/j.ijpe.2003.10.019.

Kapp, K. M. (1999). Transforming your manufacturing organization into a learning organization. *Hospital Material Management Quarterly*, *20*(4), 46–54.

Kim, S., & Nembhard, D. A. (2010). Cross-trained staffing levels with heterogeneous learning / forgetting. *IEEE Transactions on Engineering Management*, *57*(4), 560–574.

Levinthal, D. A., & March, J. G. (1993). The myopia of learning. *Strategic Management Journal*, *14*, 95–112.

Moustaghfir, K. (2009). How knowledge assets lead to a sustainable competitive advantage: Are organizational capabilities a missing link? *Knowledge Management Research & Practice*, *7*(4), 339–355. doi: 10.1057/kmrp.2009.26.

Nembhard, D., & Uzumeri, M. V. (2000a). An individual-based description of learning within an organization. *IEEE Transactions on Engineering Management*, *47*(3), 370–378. doi: 10.1109/17.865905.

Nembhard, D., & Uzumeri, M. V. (2000b). Experiential learning and forgetting for manual and cognitive tasks. *International Journal of Industrial Ergonomics*, *25*(4), 315–326. doi: 10.1016/S0169-8141(99)00021-9.

Nembhard, D. A. (2001). Heuristic approach for assigning workers to tasks based on individual learning rates. *International Journal of Production Research*, *39*(9), 1955–1968. doi: 10.1080/00207540110036696.

Nembhard, D. A., & Bentefouet, F. (2012). Parallel system scheduling with general worker learning and forgetting. *International Journal of Production Economics*, *139*(2), 533–542. doi: 10.1016/j.ijpe.2012.05.024.

Nembhard, D. A., & Norman, B. A. (2007). Cross training in production systems with human learning and forgetting. In D. A. Nembhard (Ed.), *Workforce cross training* (pp. 111–129). Boca Raton, FL, USA: CRC Press.

Olivella, J., Corominas, A., & Pastor, R. (2013). Task assignment considering cross-training goals and due dates. *International Journal of Production Research*, *51*(3), 952–962.

Sayin, S., & Karabati, S. (2007). Assigning cross-trained workers to departments: A two-stage optimization model to maximize utility and skill improvement. *European Journal of Operational Research*, *176*(3), 1643–1658. doi: 10.1016/j.ejor.2005.10.045.

Senge, P. (2006). *The fifth discipline: The art & practice of the learning organization* (Revised and updated ed.). New York: Doubleday.

Shafer, S. M., Nembhard, D. A., & Uzumeri, M. V. (2001). The effects worker learning, forgetting, and heterogeneity on assembly line productivity. *Management Science*, *47*(12), 1639–1653.

Thomas, B. G., & Nembhard, D. A. (2005). Preference based search approach for scheduling workers with learning and forgetting. In *Presentation to manufacturing & service management sponsored cluster of the INFORMS annual meeting in Denver, Colorado*.

Waltz, R. A., & Plantenga, T. D. (2009). *Knitro 6.0 user's manual: Technical report*. Ziena Optimization, Inc.

Wirojanagud, P., Gel, E. S., Fowler, J. W., & Cardy, R. (2007). Modelling inherent worker differences for workforce planning. *International Journal of Production Research*, *45*(3), 525–553. doi: 10.1080/00207540600792242.

Yan, J.-H., & Wang, Z.-M. (2011). GA based algorithm for staff scheduling considering learning-forgetting effect. In *2011 IEEE 18th international conference on industrial engineering and engineering management* (pp. 122–126). doi: 10.1109/ICIEEM.2011.6035120.