Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-3 (2015) 314–319

Calculation of the Throughput-Time in Simple Assembly Lines with Learning Effect

Tamás Koltai* Noémi Kalló**
Rita Györkös***

** Budapest University of Technology and Economics, Magyar tudósok körútja 2. 1117 Budapest, Hungary (Tel: +36-14632432; e-mail: koltai@mvt.bme.hu).*
*** Budapest University of Technology and Economics, Magyar tudósok körútja 2. 1117 Budapest, Hungary (Tel: +36-14631057; e-mail: kallo@mvt.bme.hu).*
**** Budapest University of Technology and Economics, Magyar tudósok körútja 2. 1117 Budapest, Hungary (Tel: +36-14631092; e-mail: gyorkos@mvt.bme.hu).*

Abstract: Calculating the throughput-time of a production run in simple assembly lines is important for several reasons. For example, the length of time to complete a production order is required for planning and scheduling production or the evaluation of throughput-time is required by any effort to improve assembly operations. Classical assembly line balancing techniques assume a constant cycle time. In this case, the calculation of the throughput-time is straightforward. In case of the presence of learning effect, however, cycle time changes for two main reasons. First, cycle time continuously decreases because of the decrease of operation time of the bottleneck station as a consequence of learning. Second, bottleneck may shift from one station to another, causing further changes of the cycle time. In this paper, an algorithm is presented to determine the throughput-time of a production run at the presence of learning based on the residence time of the workstations in the bottleneck. The algorithm and its application are illustrated with a simple example.

Keywords: Capacity and Performance Evaluation; Line Design and Balancing; Human-Automation Integration; Learning Curve; Bottleneck Analysis.

1. INTRODUCTION

A simple assembly line consists of several consecutive workstations. At each station, an operator performs the same tasks repetitively. Tasks assigned to the workstations are determined by precedence constraints and by several other conditions like capacity, cycle time, workforce and zoning conditions. Finished parts/products leave the assembly line at the final station. The classical assembly line balancing models assume constant station times, that is, regardless of how frequently a task is repeated, the task time is unchanged (see for example Scholl and Becker (2006); Battaïa and Dolgui (2013) or Koltai, Tatay and Kalló (2014)).

In case of the presence of learning, however, station time decreases when tasks assigned to a station are repeated several times. The first description of the decrease of task time is attributed to Wright (1936) who analyzed the operation times in an airplane assembly process. Since then, the existence of the learning function (or progress function) and its practical relevance have been illustrated in several industrial areas. See, for example, Conway and Schultz (1959), Hirschmann (1964) and Yelle (1979).

When learning is present, several classical models of operations management must be revised. The effect of learning on cost-volume-profit analysis has been studied by McIntyre (1977) and a nonlinear model for break-even analysis has been suggested. The economic order quantity (EOQ) model has been revised by Jaber and Bonney (1999) and a new approach has been proposed when learning and forgetting is present (Jaber and Bonney, 2007). The effect of learning and forgetting on bottleneck shifts was studied by Glock and Jaber (2013) in a two-stage production system. The widespread effect of learning on scheduling problems has been reviewed by Biskup (2008).

The effect of learning on assembly line balancing has also been studied previously. Cohen and Dar-El (1998) formulated several nonlinear mathematical programming models for optimizing the number of stations in an assembly line with learning. In these models, however, they assumed that the line is balanced, and the bottleneck station is the final station. Cohen, Vitner and Sarin (2006) showed, that when the throughput-time is minimized in an assembly line with learning, then the optimal solution belongs to an unbalanced line, and bottleneck shifts from the final station toward the first station. They assumed, however, that total production time can be distributed among the stations without any limitation. They also noted that when precedence relations and other combinatorial problems limit workload allocation, the problem gets very complicated. Assembly line balancing problems with learning has also been considered by Toksary et al. First, they proposed some heuristics for the minimization of total flow time in simple and U-shaped lines (Toksary et al, 2008). Next, they presented a nonlinear integer programing model when learning and task time deterioration is present (Toksary et al, 2010). They illustrated the performance of their model with the solution of the classical Jackson 11 problem with a heuristics.

It can be concluded that when total workload can be allocated to the stations without any precedence and combinatorial restrictions, then line optimization is tractable but the results

10.1016/j.ifacol.2015.06.100

have little direct practical relevance. When precedence relations and other combinatorial problems are considered, then the problem is practically relevant but only heuristic solutions can be found.

The objective of this paper is to determine the throughput-time of the production of a given quantity in simple assembly lines with learning. Since bottlenecks determine the output rate of the line, the description of the shifts of bottleneck is important information for the calculation. Consequently, the residence time of the stations in the bottleneck is also determined, which may provide useful information when the start-up period of a line is examined.

The paper is structured as follows. First, the basic definitions, notations and the limiting conditions of the analysis are presented. Next, some underlying properties of the exponential learning curve and characteristics of the bottleneck shifts in simple assembly lines are described. An algorithm is provided that determines those periods, in which different stations are in the bottleneck during the completion of a production order. Based on the residence time of the stations in the bottleneck the throughput-time is calculated. The performance of the algorithm is illustrated with a numerical example. Finally, the practical relevance of the algorithm is discussed and some future research possibilities are outlined.

2. NOTATIONS AND DEFINITIONS

Let us consider a simple assembly line depicted in Figure 1 (notations used in Figure 1 and in the following part of this paper are listed in Table 1). Parts proceed from the first station to the last station visiting all the stations. Stations are numbered with consecutive integers, denoted by j (j=1,…,J). The stations following station j are the upstream stations, and the stations preceding station j are the downstream stations. For the sake of simplicity, it is assumed that work-in-process inventory cannot accumulate between the stations.

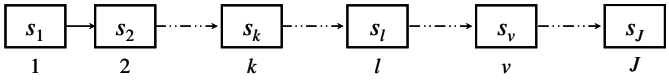

Fig. 1. Illustration of station indices and station times of a simple assembly line.

The station time of station j is denoted by s_j. The values of s_j can be determined with assembly line balancing methods. In this paper, it is assumed that s_j is given. If the presence of learning effect is assumed, then s_j is the station time at the first performance of the operation at station j. The $s_j(Q)$ function shows the station time at station j at the Qth performance of the operations. Consequently, if Q indicates production quantity, then $s_j(Q)$ is the operation time of the last part at station j when Q number of parts are produced. Applying the classical exponential learning function, the value of $s_j(Q)$ is the following,

$$s_j(Q) = s_j Q^b, \qquad (1)$$

where b<0 determines the decrease of station time in case of learning effect, and it is assumed identical for each station. In the following part of the paper, s_j denotes the duration of the first performance of the tasks at station j, which will be briefly called initial station time of station j.

Table 1. List of notations

Indices:	
j	– index of workstations,
k	– index of workstations,
l	– index of workstations,
v	– index of workstations,
i	– index of the workstation entering first in the bottleneck,
f	– index of the workstation entering last in the bottleneck.
Parameters and variables:	
J	– total number of workstations,
Q	– total production quantity,
Q_j	– quantity produced at station j,
$Q(k,l)$	– production quantity at which station k enters and station l leaves the bottleneck,
$Q_j(k,l)$	– production quantity of station j, at which station k enters and station l leaves the bottleneck,
s_j	– initial station time at station j,
$s_j(Q)$	– station time of station j as a function of production quantity,
L_l	– production quantity at which station l leaves the bottleneck,
E_k	– production quantity at which station k enters the bottleneck,
T_c	– cycle time of the assembly line,
d	– difference of the station indices of two workstations,
b	– power of the learning curve function,
L	– learning rate (L=2^b),
$TH(Q)$	– throughput time function.
Sets:	
S	– index set of those workstations which might be in the bottleneck,
I	– set of those index pairs which belong to potential bottleneck changes,
I_{Sol}	– set of those index pairs which belong to real bottleneck changes.

Note, that each station of an assembly line processes different parts at the same time. When station j works on part Q, then the preceding station (station j–1) works on the next part (part Q+1) and the succeeding station (station j+1) works on the previous part (part Q–1). To identify the parts manufactured at a station, the station index is used together with the production quantity. Consequently, Q_j indicates the part manufactured at station j. When the identification of the station is not significant, then the station index is ignored to simplify notation.

Finished parts leave the assembly line one after the other. The time between two consecutive finished parts is the cycle time (T_c). The cycle time of an assembly line is determined by the bottleneck station, that is, by the station with the longest