African Journal of Business Management Vol. 4(10), pp. 2051-2058, 18 August, 2010
Available online at http://www.academicjournals.org/AJBM
ISSN 1993-8233 ©2010 Academic Journals

Full Length Research Paper

An optimal algorithm for type-I assembly line balancing problem with resource constraint

Hsiu-Hsueh Kao[1]*, Din-Horng Yeh[2], Yi-Hsien Wang[3] and Jui-Cheng Hung[4]

[1]Department of Finance, Yuanpei University, No.306, Yuanpei St., Hsin-Chu 30015, Taiwan.
[2]Department of Business Administration, National Chung Cheng University, No.168, University Rd., Min-Hsiung Chia-Yi 62102, Taiwan.
[3]Department of Banking and Finance, Chinese Culture University, No.55, Hwa-Kang Road, Yang-Ming-Shan, Taipei 11114, Taiwan.
[4]Department of Finance, Lunghwa University of Science and Technology, No.300, Sec.1, Wanshou Rd., Guishan Shiang, Taoyuan County 33306, Taiwan.

Accepted 31 May, 2010

In this paper we consider resource-constrained type-I assembly line balancing problem (RCALBP-I) in which resource of different types, such as machines and workers, are required in processing tasks. The objective of RCALBP-I is to minimize not only the number of workstations needed, but also the number of resource types required. A shortest route algorithm is proposed to find the optimal solution for RCALBP-I. An illustrative example is also given to show the effectiveness of the proposed algorithm.

Key words: Layout, line balancing, resource constraint, shortest route.

INTRODUCTION

Assembly line balancing problem (ALBP) has been studied extensively since the pioneer work of (Bryton, 1954; Salveson, 1955; Jackson, 1956). During the past decades numerous optimal approaches have been developed to solve ALBP with different characteristics, including parallel, U-type, mixed-model, two-sided, etc. For extensive surveys (Baybars, 1986; Ghosh and Gagnon, 1989; Erel and Sarin, 1998; Scholl, 1999; Becker and Scholl, 2006). In the literature, the so-called type-I ALBP (ALBP-I) considers an assembly line that consists of a set of tasks with given processing times and precedence relationships that define the permissible ordering of tasks. The objective of ALBP-I is to assign the set of tasks to successive workstations in order to minimize the number of workstations needed for a given cycle time Ghosh and Gagnon (1989). The following definition of ALBP-I is adapted from Gutjahr and Nemhauser (1964).

Given a set *A* of tasks to be assigned, a positive real valued function *T* defined on *A* representing processing times of tasks, and a partial order *P* defined on *A* denoting precedence relationship among tasks. Let *C* be the given cycle time and *N* be the number of workstations needed in the assembly line. The objective of ALBP-I is to find a partition of the set *A* into successive subsets $A_i \subseteq A$, $i = 1,...,N$, so that the number of workstations *N* is minimized. Let $T(x)$ denote the processing time of task *x*, and $T(A_i) = \sum_{x \in A_i} T(x)$ be the total processing time of the tasks in subset A_i. A partition of subsets A_i is feasible if the following conditions hold:

$$\bigcup_{i=1}^{N} A_i = A, \tag{1}$$

*Corresponding author. E-mail: simonkao@mail2000.com.tw.

$$A_i \cap A_j = \phi,\ i \neq j, \tag{2}$$

$$T(A_i) \leq C,\ i = 1,...,N, \tag{3}$$

If xPy (that is, *x* precedes *y*) and $x \in A_i$, $y \in A_j$, then

$$i \leq j. \tag{4}$$

Conditions (1) and (2) simply state that all the tasks have to be assigned, and each task is assigned to one and only one subset; condition (3) ensures that, for any subset, the total processing time of the tasks in the subset does not exceed the given cycle time, while condition (4) maintains precedence relationship among tasks. Conventionally, subset A_i is usually called workstation *i*, and a task *x* is said to be assigned to workstation *i* if $x \in A_i$.

Recently, some researchers studied assembly system design problems (ASDP) in which the objective was to optimize some economic criteria (e.g., total cost) with machine selections (Nicosia et al., 2002; Yamada and Matsui, 2003). However, only limited researches discuss the practical situation of resource constraints that arises often in assembly line balancing. In the real-world ALBP, different types of resources (such as machines and workers) are often required in task processing; as pointed out by Ağpak and Gökçen (2005), the issue of line balancing with limited resources has always been a serious problem in industry. In their recent paper Ağpak and Gökçen (2005) developed a 0 - 1 integer programming model for a resource-constrained ALBP, and the objective was to balance the assembly line so that the number of *resources* required was minimized for a given number of workstations.

In this paper, we revisit the same ALBP considered by Ağpak and Gökçen (2005), for convenience, we call the problem resource-constrained type-I ALBP, or simply RCALBP-I; however, the objective is to minimize not only the number of workstations needed, but also the number of resource types required. Motivation of this paper arises from practical needs. In practice, a workstation in the assembly line usually consists of one or more dedicated machines as well as workers and tools. Thus, minimizing both the number of workstations and resources is equivalent to utilizing the least number of machines, workers, and tools. We propose an optimal approach, based on the shortest route algorithm developed by Gutjahr and Nemhauser (1964), to solve RCALBP-I considered in this paper. The paper is organized as follows: first, problem description of RCALBP-I is given; then, the proposed optimal approach is described, followed by an illustrative example given by Ağpak and Gökçen (2005), conclusion and future research are discussed in the last.

METHODOLOGY

Definition of RCALBP-I

As stated above, RCALBP-I differs from the traditional ALBP-I defined previously in that different types of resource are required in task processing in RCALBP-I. The definition of RCALBP-I is described as follows. Consider the ALBP-I with a given cycle time *C*, and let *A* be the set of tasks to be assigned and *N* be the number of workstations needed. Also, let $T(x)$ and $R(x)$ be the processing time and the resource type required for task $x \in A$, respectively. For simplicity, we assume in this paper that each task requires only one type of resource; the extension to multiple types of resource is straightforward. Let $A_i \subseteq A$ be the subset consisting of tasks that are assigned to workstation *i*, then $T(A_i)$ is the total processing time and $R(A_i)$ is the set of resource types required for workstation *i*. Let $|R(A_i)|$ denote the number of resource types required for workstation *i*. Therefore, for RCALBP-I considered in this paper, the objective is to find a feasible partition A_i in order to minimize both the number of workstations *N* and the number of resources $\sum_{i=1}^{N} |R(A_i)|$ for the assembly line.

The shortest route algorithm

In order to develop the shortest route algorithm, we need to show how to construct the network diagram for RCALBP-I. The construction procedure consists of two parts: nodes generation and arcs generation, which are explained in details below.

Nodes generation

In this paper we adapt the procedure developed by Gutjahr and Nemhauser (1964), in nodes generation. In the network diagram, a node is represented by the so-called *state* that is simply a subset of tasks. These subsets (that is, states) are generated stage by stage, and satisfy the following properties:

(i) No subsets are duplicated during the generation procedure.
(ii) All subsets generated are states.
(iii) Every subset is generated.

Conceptually, the proposed shortest route algorithm enumerates all the feasible partitions in order to find the optimal ones that achieve the desired objective.

The procedure starts from stage 0 with the empty set as the first state generated. The set of tasks with no predecessors are placed in stage 1 and are considered as "marked" tasks. All the subsets of the marked tasks are generated and are defined as states. For each generated state *S*, an unmarked immediate follower is defined as a task that is an immediate successor of at least one task in *S* and is not preceded by any tasks not in *S*. In general, for any state S_k generated in stage *n*, the unmarked immediate followers of S_k are placed in a set $F(S_k)$. For each subset $S_l \subset F(S_k)$, the union $S_k \cup S_l$ is generated as a new state in stage $n+1$. When all the states in stage *n* have been considered, the tasks in

$F(S_k)$ are then marked and the procedure is repeated for stage $n+1$. The procedure of nodes generation is finished when all the tasks are marked and all states are therefore generated.

Arcs generation

In their paper, Gutjahr and Nemhauser (1964) developed a procedure of arcs generation in which only one optimal solution was given because just a necessary portion of feasible arcs was generated. In this paper, we propose a modified arcs generation procedure in which we enumerate all the feasible arcs so that the desired solution for RCALBP-I can be found from among all the alternative optimal solutions of ALBP-I. The proposed arcs generation procedure is described as follows.

As mentioned above, a state S_k represents a node *k* in the network diagram constructed (that is, the number of states generated is equal to the number of nodes). In the proposed procedure, arcs are also generated stage by stage. The procedure starts from stage 0 with state 0 as node 0, which is considered as "marked" node. A directed arc is generated from node 0 to node *k* if a state S_k satisfies $T(S_k) \le C$, and all the nodes having directed arcs from node 0 are placed in stage 1 as marked nodes. In general, for any marked node *k* in stage *n*, a directed arc is generated from node *k* to an unmarked node *l* if a state S_l satisfies $S_k \subset S_l$ and $T(S_l) - T(S_k) \le C$; and node *l* is placed in stage $n+1$ as marked node. The procedure is terminated when all the nodes have been marked. Note that, in the proposed procedure, all feasible paths are enumerated from node 0 (that is, the empty state) to the last node (i.e., the state containing all the tasks). According to Gutjahr and Nemhauser (1964), the number of directed arcs needed for a path from node 0 to the last node is equal to the minimum number of workstations needed for the assembly line. The remaining issue is to find the paths that also minimize the number of resources required, which is accomplished as follows.

Computation of shortest routes

In order to find the desired solution, the length of arcs is defined as follows. Consider the directed arc (k,l) from node *k* to node *l*, and let S_k and S_l be the associated states respectively. Let $A_i = S_l / S_k$, that is, A_i denotes a workstation *i* containing task $x \in S_l / S_k$. Then, the length of arc (k,l) is defined to be $|R(A_i)|$, the number of resource types required for workstation *i*. Let *N* be the minimum number of workstations obtained in the above procedure of nodes generation, then $\sum_{i=1}^{N} |R(A_i)|$ is equal to the total number of resource types required for a path from node 0 to the last node. Therefore, the path with the minimum value of $\sum_{i=1}^{N} |R(A_i)|$ is the desired solution that not only gives the minimum number of workstations but also requires the minimum number of resource types.

Illustrative example

We use the example presented in Ağpak and Gökçen (2005), as depicted as in Figure 1 with the given cycle time $C=9$, to illustrate the proposed shortest route algorithm. As shown in the figure, there are 11 tasks (represented by nodes) and their processing times as well as the required resource type are given next to the nodes. For instance, processing time of task 1 is 6 and the required resource is type A (Figure 1 Illustrative example by Ağpak and Gökçen 2005).

Nodes generation

Starting from stage 0, we have the first generated $S_0 = \phi$ (the empty set containing no task). Since task 1 is the only unmarked immediate follower, we have $F(S_0) = \{1\}$ and place task 1 in stage 1 as marked. The only subset of $F(S_0)$ is $\{1\}$; by taking the union of S_0 and $\{1\}$, we generate the state $S_1 = S_0 \cup \{1\} = \{1\}$ and proceed to stage 1. In stage 1, the unmarked immediate followers of state S_1 are tasks 2, 3, 4 and 5, and we have $F(S_1) = \{2,3,4,5\}$ and these tasks are then placed in stage 2 as marked. By taking the union of S_1 and each subset of $F(S_1)$, we generate the states $S_2 = \{1,2\}$ to $S_{16} = \{1,2,3,4,5\}$ as shown in Table 1. The procedure continues and the result of nodes generation is summarized as in Table 1, in which state time is the total processing time of tasks in the state. As seen in the table, there are totally 51 states generated (in which the last state S_{51} contains all tasks) (Table 1, Nodes generation).

Arcs generation

Starting from stage 0 with the empty state S_0 as node 0, we generate directed arcs from node 0 to nodes 1, 2, 5, and 8 (that is, states $S_1 = \{1\}$, $S_2 = \{1,2\}$, $S_5 = \{1,5\}$, and $S_8 = \{1,2,5\}$ respectively) because their state times are less than or equal to the cycle time 9 and they contain state S_0 as subset. These nodes are placed in stage 1 as marked, and the procedure proceeds to stage 1. Take node 5 with state $S_5 = \{1,5\}$ as example, directed arcs are generated from node 5 to nodes 10, 11, 13, 20, 14, and 22 because (i) the differences between their state times and that of the state S_5 are less than or equal to the cycle time 9, and (ii) the state S_5 is subset of these states. The procedure continues and the result of arcs generation is summarized as in Table 2 (Table 2 Arcs generation).

Computation of shortest routes

The resulting network diagram obtained from the procedures is as depicted in Figure 2 (Figure 2 Network

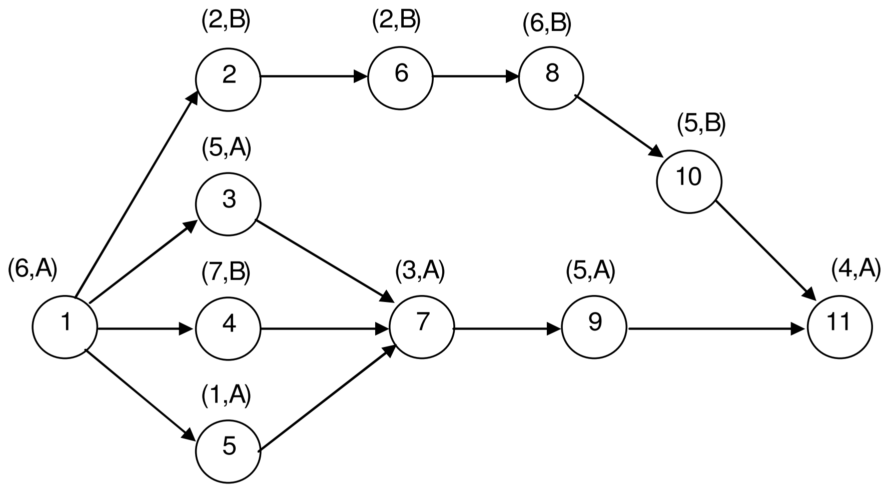

Figure 1. Illustrative example by (Ağpak and Gökçen, 2005).

Table 1. Nodes generation.

Stage	Marked tasks	State	Node	Tasks in state	State time	Immediate followers
0		S_0	0	ϕ	0	1
1	1	S_1	1	1	6	2, 3, 4, 5
2	2, 3, 4, 5	S_2	2	1, 2	8	6
		S_3	3	1, 3	11	
		S_4	4	1, 4	13	
		S_5	5	1, 5	7	
		S_6	6	1, 2, 3	13	6
		S_7	7	1, 2, 4	15	6
		S_8	8	1, 2, 5	9	6
		S_9	9	1, 3, 4	18	
		S_{10}	10	1, 3, 5	12	
		S_{11}	11	1, 4, 5	14	
		S_{12}	12	1, 2, 3, 4	20	6
		S_{13}	13	1, 2, 3, 5	14	6
		S_{14}	14	1, 2, 4, 5	16	6
		S_{15}	15	1, 3, 4, 5	19	7
		S_{16}	16	1, 2, 3, 4, 5	21	6, 7
3	6, 7	S_{17}	17	1, 2, 6	10	8
		S_{18}	18	1, 2, 3, 6	15	8
		S_{19}	19	1, 2, 4, 6	17	8
		S_{20}	20	1, 2, 5, 6	11	8
		S_{21}	21	1, 2, 3, 4, 6	22	8
		S_{22}	22	1, 2, 3, 5, 6	16	8

Table 1. Nodes generation (continued).

Stage	Marked tasks	State	Node	Tasks in state	State time	Immediate followers
		S_{23}	23	1, 2, 4, 5, 6	18	8
		S_{24}	24	1, 3, 4, 5, 7	22	9
		S_{25}	25	1, 2, 3, 4, 5, 6	23	8
		S_{26}	26	1, 2, 3, 4, 5, 7	24	9
		S_{27}	27	1, 2, 3, 4, 5, 6, 7	26	8, 9
4	8, 9	S_{28}	28	1, 2, 6, 8	16	10
		S_{29}	29	1, 2, 3, 6, 8	21	10
		S_{30}	30	1, 2, 4, 6, 8	23	10
		S_{31}	31	1, 2, 5, 6, 8	17	10
		S_{32}	32	1, 2, 3, 4, 6, 8	28	10
		S_{33}	33	1, 2, 3, 5, 6, 8	22	10
		S_{34}	34	1, 2, 4, 5, 6, 8	24	10
		S_{35}	35	1, 3, 4, 5, 7, 9	27	
		S_{36}	36	1, 2, 3, 4, 5, 6, 8	29	10
		S_{37}	37	1, 2, 3, 4, 5, 6, 7, 8	32	10
		S_{38}	38	1, 2, 3, 4, 5, 7, 9	29	
		S_{39}	39	1, 2, 3, 4, 5, 6, 7, 9	31	
		S_{40}	40	1, 2, 3, 4, 5, 6, 7, 8, 9	37	10
5	10	S_{41}	41	1, 2, 6, 8, 10	21	
		S_{42}	42	1, 2, 3, 6, 8, 10	26	
		S_{43}	43	1, 2, 4, 6, 8, 10	28	
		S_{44}	44	1, 2, 5, 6, 8, 10	22	
		S_{45}	45	1, 2, 3, 4, 6, 8, 10	33	
		S_{46}	46	1, 2, 3, 5, 6, 8, 10	27	
		S_{47}	47	1, 2, 4, 5, 6, 8, 10	29	
		S_{48}	48	1, 2, 3, 4, 5, 6, 8, 10	34	
		S_{49}	49	1, 2, 3, 4, 5, 6, 7, 8, 10	37	
		S_{50}	50	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	42	11
6	11	S_{51}	51	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	46	

diagram of the illustrative example).

As seen in the figure, six arcs are needed for a path from node 0 to the last node (that is, state S_{51}); according to Gutjahr and Nemhauser (1964), this means that the minimum number of workstations needed is 6. Also we see that many alternative paths exist, indicating that there are alternative optimal solutions using the same minimum number of workstations. In order to find the desired solutions for RCALBP-I that also minimizes the number of resource types required, we need to compute the length of arcs. Take the arc $(2,14)$ directed from node 2 to node 14 for example, we have $S_{14}/S_2 = \{4,5\}$. As defined above, the length of arc $(2,14)$ is given by $|R(\{4,5\})| = 2$ because task 4 and task 5 require resource type A and B respectively. After

Table 2. Arcs generation.

Stage	Marked nodes	Node	Arcs to nodes
0	0	0	1, 2, 5, 8
1	1, 2, 5. 8	1	3, 4, 6, 7, 10, 11, 13, 17, 18, 20
		2	6, 7, 13, 17, 18, 20, 14, 19, 22, 28, 31
		5	10, 11, 13, 20, 14, 22
		8	13, 20, 14, 22, 31, 23
2	3, 4, 6, 7, 10, 11, 13, 17, 18, 20, 14, 19, 22, 28, 31, 23	3	9, 12, 15
		4	9, 12, 15, 16, 21, 24
		6	12, 16, 21, 29, 33
		7	12, 16, 21, 25, 26, 30, 34
		10	15, 16
		11	15, 16, 24, 25
		13	16, 25, 33
		17	
		18	21, 25, 29, 33
		20	
		14	16, 25, 26, 34
		19	21, 25, 30, 34, 27
		22	25, 33
		28	30, 34, 33, 29, 41, 44
		31	34, 33, 44
		23	25, 34, 27
3	9, 12, 15, 16, 21, 24, 25, 26, 30, 34, 33, 29, 27, 41, 44	9	35
		12	32, 36, 38
		15	35
		16	36, 38
		21	39, 32, 36
		24	35, 38, 39
		25	37, 36, 39
		26	37, 38, 39
		30	43, 47, 32, 36, 37
		34	36, 37, 47
		33	46, 36
		29	42, 32, 36, 46
		27	37, 39
		41	42, 43, 46, 47
		44	46, 47
4	35, 32, 36, 38, 39, 37, 43, 47, 46, 42	35	
		32	40, 45, 48, 49
		36	40, 48, 49
		38	40
		39	40
		37	40, 49
		43	48, 45, 49
		47	48, 49
		46	48
		42	48, 45
5	40, 45, 48, 49	40	50, 51
5		45	50
5		48	50

Table 2. Cont'd.

Stage	Marked nodes	Node	Arcs to nodes
5		49	50, 51
6	50, 51	50	
6		51	

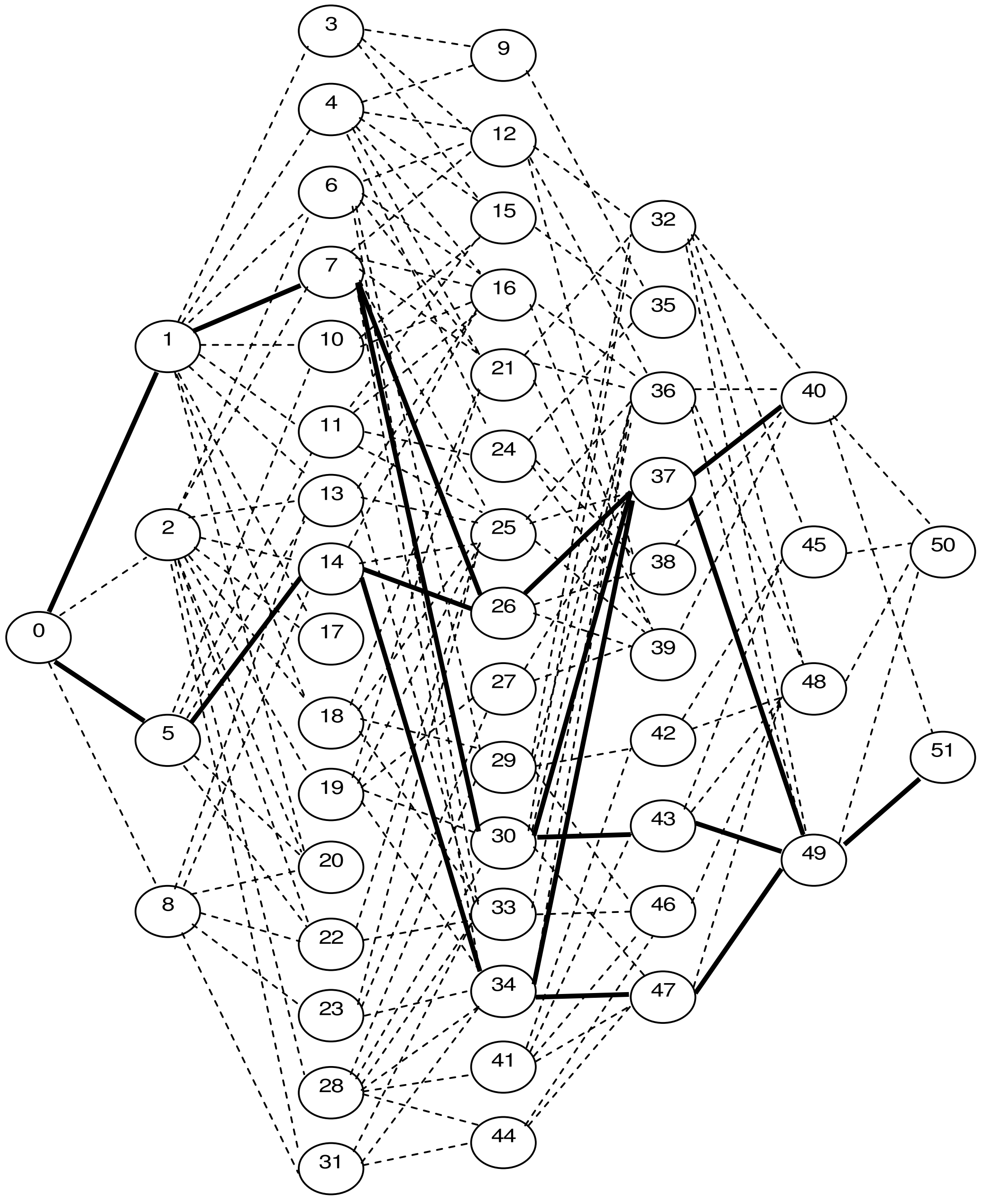

Figure 2. Network diagram of the illustrative example.

computing the length of all directed arcs, we may apply any shortest route algorithm such as Dijkstra (1959) to find the shortest routes from node 0 to the last node. Computational result is also shown in Figure 2, in which there are totally six shortest routes represented in bold lines; the minimum number of resource types required is given by $\sum_{i=1}^{6}|R(A_i)| = 6$. Take the route $S_0 \rightarrow S_1 \rightarrow S_7 \rightarrow S_{26} \rightarrow S_{37} \rightarrow S_{49} \rightarrow S_{51}$ for example to show how the result of task assignment can be obtained. Tasks assigned to workstations 1 through 6 are given by $A_1 = S_1 / S_0 = \{1\}$, $A_2 = S_7 / S_1 = \{2,4\}$, $A_3 = S_{26} / S_7 = \{3,5,7\}$, $A_4 = S_{37} / S_{26} = \{6,8\}$, $A_5 = S_{49} / S_{37} = \{10\}$ and $A_6 = S_{51} / S_{49} = \{9,11\}$ respectively, which is the same optimal solution obtained in Ağpak and Gökçen (2005). Note that, although there are five other shortest routes, only one possible different task assignment can be obtained (for example, from the path $S_0 \rightarrow S_5 \rightarrow S_{14} \rightarrow S_{34} \rightarrow S_{47} \rightarrow S_{49} \rightarrow S_{51}$): $A_1 = S_5 / S_0 = \{1,5\}$, $A_2 = S_{14} / S_5 = \{2,4\}$, $A_3 = S_{34} / S_{14} = \{6,8\}$, $A_4 = S_{47} / S_{34} = \{10\}$, $A_5 = S_{49} / S_{47} = \{3,7\}$ and $A_6 = S_{51} / S_{49} = \{9,11\}$.

Conclusion and future research

In this paper we consider type-I assembly line balancing problem with resource constraint (RCALBP-I) in which the objective is to find the optimal task assignment that minimizes not only the number of workstations needed, but also the number of resource types required. It is of practical importance that the number of resource types should be minimized as possible in order to utilize the least number of resource requirements.

We propose a shortest route algorithm in which all alternative optimal solutions are enumerated in order to find the desired solution of RCALBP-I. Future research may include extensions of the proposed algorithm to solve different types of assembly line balancing problem and the development of efficient algorithms to solve practical large-size problems.

REFERENCES

Ağpak K, Gökçen H (2005). Assembly line balancing: two resource constrained cases. Int. J. Prod. Econ. 96: 129-140.

Baybars I (1986). A survey of exact algorithms for the simple assembly line balancing problem. Manage. Sci. 32: 909-932.

Becker C, Scholl A (2006). A survey on problems and methods in generalized assembly line balancing. Eur. J. Oper. Res. 168: 694-715.

Bryton B (1954). Balancing of continuous production line. MS thesis, Northwestern University.

Dijkstra EW (1959). A note on two problems in connection with graphs. Numerical Mathematics, 1: 269-271.

Erel E, Sarin SC (1998). A survey of the assembly line balancing procedures. Prod. Plan. Control, 9(5): 414-434.

Ghosh S, Gagnon RJ (1989). A comprehensive literature review and analysis of the design balancing and scheduling of assembly systems. Int. J. Prod. Res. 27: 637-670.

Gutjahr AL, Nemhauser GL (1964). An algorithm for the line balancing problem. Manage. Sci. 11(2): 308-315.

Jackson JR (1956). A computing procedure for a line balancing problem. Manage. Sci. 2: 261-271.

Nicosia G, Pacciarelli D, Pacifici A (2002). Optimally balancing assembly lines with different workstations. Discrete Appl. Mathematics, 118: 99-113.

Salveson ME (1955). The assembly line balancing problem. J. Industrial Eng. 6: 18-25.

Scholl A (1999). Balancing and sequencing of assembly lines. Physica-Verlag Heidelberg, New York.

Yamada T, Matsui M (2003). A management design approach to assembly line systems. Int. J. Prod. Econ. 84: 193-204.