doi:10.4186/ej.2010.14.2.53

The Impact of Walking Time on U-shaped Assembly Line Worker Allocation Problems

Ronnachai Sirovetnukul and Parames Chutima*

Department of Industrial Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok, Thailand 10330
E-mail : egrsr@mahidol.ac.th and cparames@chula.ac.th*

ABSTRACT

The one-piece flow manufacturing line of single and customized products is usually organized as a U-shaped assembly layout. In this study, the characteristics of a single U-line are described and modeled. The worker allocation problem is hierarchically concerned with the task assignment into a U-line and allocate task to workers in sequence. Several products are assembled in 7-task to 297-task problems, and each problem is performed with a given cycle time. The primary purpose is to identify the impact of walking time on both symmetrical and rectangular U-shaped assembly layouts. The minor purpose is to compare the number of workers between two fixed layouts. Coincidence algorithm demonstrates clarifying solutions. To respond to two previous aims, the primary objective function of a number of workers is used. Finally, with the Pareto-optimal frontier between the deviation of operation times of workers and the walking time, its computational study is exemplified to identify good task assignment and walking path.

KEYWORDS

coincidence algorithm, multiple objectives, U-shaped assembly lines, walking time, worker allocation.

doi:10.4186/ej.2010.14.2.53

I . Introduction

Traditionally, a straight assembly line is suitable for mass production. In the circumstance of product variety, an assembly line is replaced with the U-shaped line. The decision to transform straight-line assembly systems to U-shaped assembly line systems constitutes a major layout design change and investment for assembly operations. Proponents of the lean manufacturing and just-in-time (JIT) philosophies assert that U-shaped assembly systems offer several benefits over traditional straight-line layouts [1] especially an improvement in labor productivity. U-lines have become popular in order to obtain the main benefits of smoothed workload, multi-skilled workforce and other principles of the JIT philosophy. Many researchers agree that U-lines are one of the most important components for a successful implementation of JIT production systems [2],[3]. The U-line is equivalent to the straight line for any allocation of tasks or machines to workers so long as a worker does not work together with other stations (i.e. no crossing loop). The number of workers required on a U-line is never more than that required on a straight line [1]. However, a worker can be either in the similar line or across from the front line to the back line or vice versa. Thus, the historical results of a number of workers from the straight line balancing cannot be used due to lack of the addition of walking time. Walking time is also negligible for U-line balancing problems in most papers [4] - [8] but walking time should be considered as workers follow circular paths and walk on the beginning (front), the ending (back) and the middle (side) U-line to complete their tasks. There are two research gaps to be filled in this paper. The first research question is whether and how much the appropriate walking time between tasks should be considered in the U-line instead of the ignorance as often modeled in the straight line problem. Secondly, the research question is whether different fixed U-shaped layouts affect the number of workers. This paper remarks that adding the walking time at the early percentage of average processing time significantly increases the number of workers. The line balancing problem can be replaced with the worker allocation problem if a single worker can handle multiple tasks. Hierarchically after solving the minimum number of workers, the rest of multi-objective coincidence algorithm for tackling the U-shaped worker allocation problem is developed in this research. The objective of the described study is to allocate workers in single U-shaped assembly line problems having manually operated machines by determining the average processing time percentage.

The structure of this paper is as follows. In Section II, previous related literature is reviewed. Section III describes the physical and mathematical models. This model is primarily used to minimize a number of workers needed to perform tasks. Secondly, dual objectives are to minimize the deviation of operation times of workers and their walking time simultaneously with the development of evolutionary algorithm. The combinatorial optimization with the coincidence algorithm and its numerical example are explained in Section IV. Experimental results are presented in Section V. The last section concludes the work and discusses the potential future work.

II. Literature Review

2.1 U-shaped assembly line balancing problems

An assembly line is a manufacturing process in which component parts are added to a product in a sequential manner to create a finished product. Assembly lines are special flow-line production systems which are of great importance in the industrial production of high quantity standardized commodities. Recently, assembly lines have even gained importance in low volume production of customized products (mass customization). Balancing an assembly line means allocating the basic assembly tasks to be carried out to different stations to achieve specific goals and all in compliance with given constraints. The main assembly line balancing objective is to balance the task workload across workstations so that no workstation has an excessively high or low task workload. Becker and Scholl [9] have classified the main characteristics of assembly line balancing problems with several constraints and different objectives. Assembly line balancing problems consist of the simple and general types. SALBP-1, SALBP-2, SALBP-E, and SALBP-F are in the Simple Assembly Line Balancing Problems (SALBPs) whereas the U-shaped Assembly Line Balancing Problem (UALBP), Mixed-model Assembly Line Balancing Problem (MALBP) and so on are in the General Assembly Line Balancing Problems (GALBPs). Since this study focuses on the UALBP of type I minimizing a number of workers at a given cycle time, relevant papers on U-shaped assembly lines are reviewed.

Miltenburg and Wijngaard [10] were the first to compare a U-shaped line with a straight line. There are many papers that reveal advantages of the U-shaped layout over the linear layout [1],[6],[11],[12],[13] Cheng *et al.* [1] found collectively that the benefits and factors favoring U-lines are better volume flexibility, worker flexibility, number of workstations, material handling, visibility and teamwork, and rework. Miltenburg [14] analyzes the U-line facility problem where a multi-line station may include tasks from two adjacent U-lines. To solve the problem, a dynamic programming approach is used. Sparling [15] also investigates the multiple U-line problems and presents several heuristic approaches to solve the *N* U-line facility problem. More complex U-lines, which are not a single or simple U-line, are named multi-lines in a single U, double-dependent U-lines, embedded U-lines, figure-eight-pattern U-lines, and multi-U-line facility [16]. Travel time between tasks are hardly considered; however, at present only Miltenburg [3] in the issue of line balancing and Shewchuk [17] in the issue of worker allocation consider walking time. Miltenburg [3]'s 10-task problem of a single U-line was studied hierarchically in USALBP-1. It gives us the optimal number of workstations with walking distance (one unit for adjacent machines (at the same row) and two units for opposite machines). The paper of Shewchuk studied the same problem of 5-20 machines (or tasks) with walking time (one second for adjacent machines (same row) and two seconds for opposite machines). This research relaxes Shewchuk's assumption that does not guarantee minimum walking times on page 3,489 [17]. It will be studied in the following experiments in this research. However, it did not refer to the input of the precedence graph. As a result, its optimum number of workstations with walking time cannot be compared.

2.2 Data sets of U-shaped assembly line balancing problems

The well-known Talbot's data set based on 12 precedence networks with 8-111 tasks, each of which is combined with several cycle times is adapted to build a total of 64 instances [18]. Miltenburg [14] noted that U-line problem sets with more than 26 tasks may be too difficult to solve in more restricted constraints. However, Scholl's data sets are composed of 168 instances with 25-297 tasks [19]. All instances form the combined data set with 269 instances. Complete descriptions of all data sets are given in chapter 7.2 of Scholl [19] and can be downloaded from http://www.assembly-line-balancing.de. These sets are used for testing ULINO which is applied directly to the U-shaped Assembly Line Balancing Problem of type I (UALBP-I)

2.3 U-shaped worker allocation problems

The worker allocation problem consists of providing a simultaneous solution to a double assignment: (1) tasks to stations; and (2) available workers to stations [20]. There are three important differences between the U-line assembly line balancing problem (UALBP) and the worker allocation problem [17]. First, the task assignments are fixed on the U-line. Secondly, walking time is taken into account. Thirdly, there are no restrictions on what machines can be assigned to given workers. In manufacturing, the purpose of worker allocation is to minimize the labor costs, by telling a production facility what to make, when, by which staff, and on which equipment. The reason why this research focuses on the worker allocation is because it is one of the most important decisions that can achieve productivity gains and right sizing in a labor intensive manufacturing system. If one worker can only attend one machine, then the required number of workers is proportional to the number of machines in a workstation. However, one worker operating a few machines is more interesting in this research. Until now, rarely is the study of worker allocation in U-shaped assembly lines found. Much of the existing literature solves the worker allocation problem with mathematical programming by assuming both deterministic data and single objective [21] - [24]. Most papers assume a homogeneous skill in solving the worker allocation problem. Several papers [4],[6],[7],[10],[13],[15],[25] in Table 1 discard the movement of a worker from task location to another task location. Thus, it is determined to be zero: that means no movement within each workstation. However, in lean thinking for the U-line it is necessary to add walking time for operation times. The increase of a number of workers is obviously dependent on walking time, but no papers give results on how much walking time is significant for changing the number of workers. This paper extends more details found in Sirovetnukul and Chutima [26] by assuming one-second walking time from one task to another task. In brief, the summary of related papers on UALBP-I and the research gap of this paper are illustrated in Table 1.

Authors(Year)	Problem description	Problem set	Objectives	Walking time	Solution techniques
Miltenburg and Wijngaard (1994) [10]	Single model	Up to 11 tasks Up to 111 tasks	Number of workstations	No	DP formulation RPWT-based heuristic
Miltenburg and Sparling (1995) [21]	Single model	Up to 40 tasks	Number of workstations	No	DP-based exact algorithm Depth-first and breath-first B&B
Ajenblit and Wainwright (1998) [4]	Single model	Up to 111 tasks	Number of workstations and workload balance	No	Generic algorithm
Miltenburg (1998) [14]	U-line facility with several individual U-lines	Individual U-lines with up to 22 tasks	Number of workstations and idle time in a single station	No	DP-based exact algorithm
Sparling and Miltenburg (1998) [15]	Mixed model	Up to 25 tasks	Number of workstations	No	Heuristic
Urban (1998) [16]	Single model	Up to 45 tasks	Number of workstations	No	IP formulation
Scholl and Klein (1999) [7]	Single model	Up to 297 tasks	Number of workstations	No	B&B-based heuristic
Miltenburg (2001) [3]	Single model	10 tasks	Number of workstations	Yes*	ILP and DP formulation
Hwang and Katayama (2008) [13]	Mixed model	19, 61 and 111 tasks	Number of workstations and workload variation	No	Genetic algorithm
Shewchuk (2008) [16]**	Lean single U-lines	Up to 20 tasks	Number of workstations and maximize full work	Yes	Heuristic
This research study	*Single&mixed model with several single U-lines*	*Up to 297 tasks*	*Number of workstations, workload smoothness, and walking time*	*Yes*	*Multi-objective evolutionary algorithm*

***Walking time is set at one unit for adjacent tasks at the same row and two units for opposite tasks.**
**Shewchuk [16] did not use the standard problems of precedence constraints and given cycle time.

Table 1
Summary of the papers conducted on single U-shaped assembly lines.

2.4 Worker allocation objective functions

Historical single and multiple objective functions have been studied by several researchers. Efficiency and balance performance measures influencing just-in-time manufacturing (especially in UALBP) are reviewed and shown in Figure 1. In the study of decision making, terms such as multiple objectives, multiple attributes, and multiple criteria are used interchangeably. Multiple objectives decision making consists of a set of conflicting goals that cannot be achieved simultaneously. The motivation to consider the problem of generating the efficient set of the worker allocation problem comes from the variety of industrial cases where the criteria are related to the minimum number of workstations, smoothed workload in a sense of equity, and minimum walking time to save the space needed for the actual size of a U-shaped line and shorten distance for communication between workers.

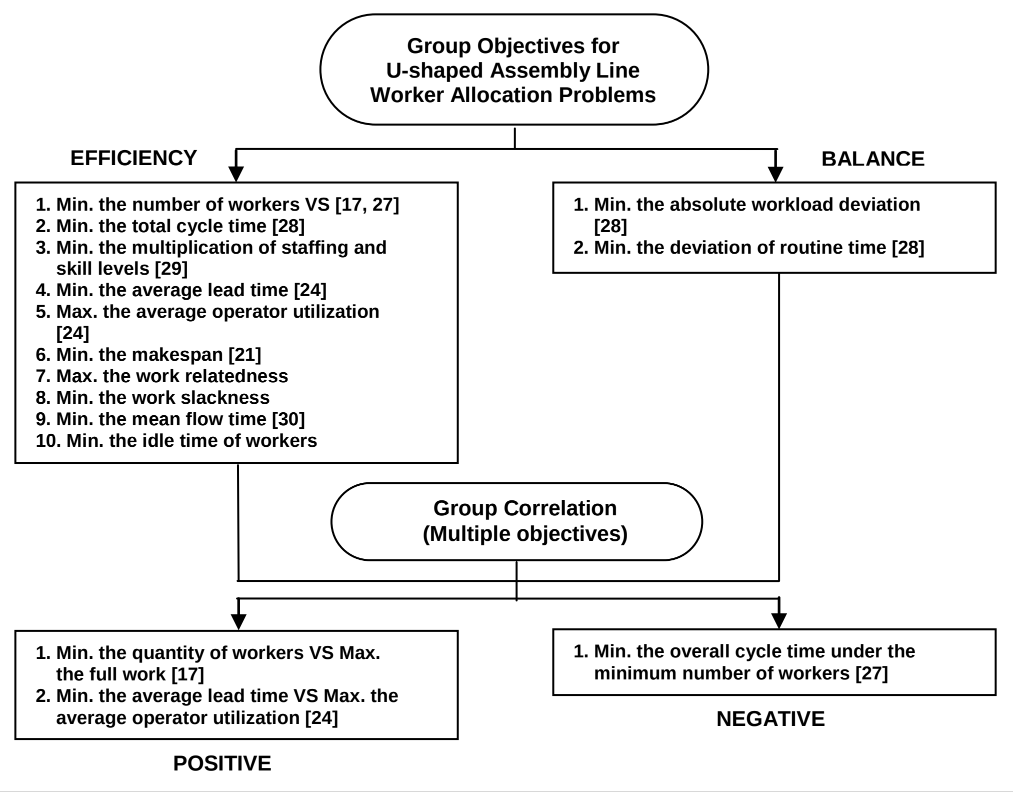

Group Objectives for U-shaped Assembly Line Worker Allocation Problems
EFFICIENCY
BALANCE
1. Min. the number of workers VS [17, 27]
2. Min. the total cycle time [28]
3. Min. the multiplication of staffing and skill levels [29]
4. Min. the average lead time [24]
5. Max. the average operator utilization [24]
6. Min. the makespan [21]
7. Max. the work relatedness
8. Min. the work slackness
9. Min. the mean flow time [30]
10. Min. the idle time of workers
1. Min. the absolute workload deviation [28]
2. Min. the deviation of routine time [28]
Group Correlation (Multiple objectives)
1. Min. the quantity of workers VS Max. the full work [17]
2. Min. the average lead time VS Max. the average operator utilization [24]
POSITIVE
1. Min. the overall cycle time under the minimum number of workers [27]
NEGATIVE

Figure 1
Comparisons of objective functions for the UALBP.

doi:10.4186/ej.2010.14.2.53

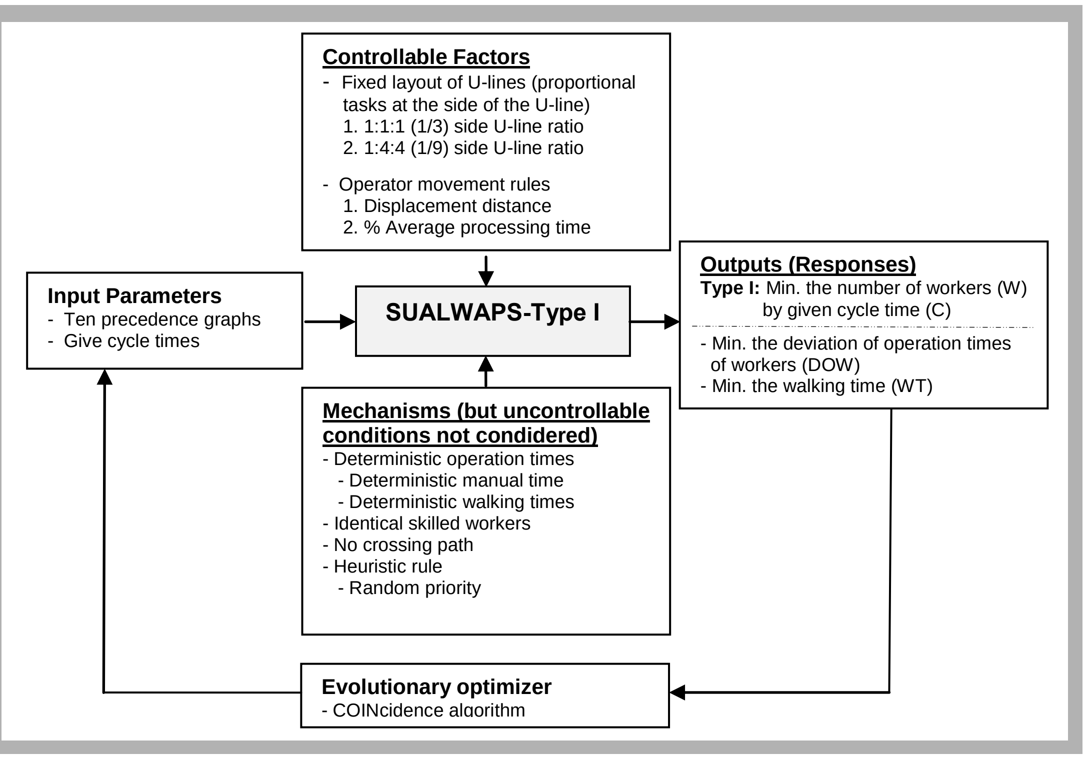
2.5 Solution techniques

Using exact solution for small size problems of U-shaped assembly line balancing is studied in some papers. It is well known that traditional assembly line balancing problems (ALBP) fall into the NP-hard class of combinatorial optimization problems [8] As a result of the problem complexity, it is difficult to find an optimal solution in polynomial time. Since both the MALBP and the UALBP are subsets of the ALBP, they are also NP-hard. Therefore, mathematical methods that evaluate the entire solution space are not suitable for large sized problems and heuristics need to be employed in order to efficiently search the solution space. Since last two decades there has been many multiple objective evolutionary algorithms as mentioned by Chutima and Pinkoompee [31]. However, the COINcidence algorithm [26] is preferable to Non-dominated Sorting Genetic Algorithm (NSGA-II) [32] due to the additional use of negative knowledge that renders faster convergence speed and less CPU time. The details of COIN are later described in Section VI.

III. Model Configurations

3.1 Problem development

The Just-in-time (JIT) production system has been adopted extensively in today's manufacturing industries such as the apparel industry to meet production demands. A U-shaped production line can be described as a special type of cellular manufacturing used in JIT production systems. In recent decades, many apparel manufactures have installed several production systems on their apparel assembly lines such as the traditional progressive bundle system and the automated unit production system [33]. The assembly line to be studied in this paper is a modular production system (or a single U-line). There are no automated processing machines in the production system. After each worker operates an item at a machine, a worker walks with several patterns such as a circular loop, a rectangular loop, or a straight-line loop and takes it to the next machine and at the end of each intra loop. Generally a worker hands it over to the adjacent worker along the sequence of U-line. From some of the sample companies, there is no equity of workload although line efficiency has been continuously improved. In practice, most companies manage the assembly line problem of type F (given number of workstations and given cycle time) and improve line efficiency by avoiding the complexity of the problem. However, this paper studies the problem of type I: the minimum number of workstations at given cycle times. The evolutionary combinatorial optimization process of Single U-shaped Assembly Line Worker Allocation Problems of type I (SUALWAPs-I) minimizing the number of workers given a target cycle time is illustrated in Figure 2.



Controllable Factors
- Fixed layout of U-lines (proportional tasks at the side of the U-line)
1. 1:1:1 (1/3) side U-line ratio
2. 1:4:4 (1/9) side U-line ratio
- Operator movement rules
1. Displacement distance
2. % Average processing time
Input Parameters
- Ten precedence graphs
- Give cycle times
SUALWAPS-Type I
Outputs (Responses)
Type I: Min. the number of workers (W) by given cycle time (C)
- Min. the deviation of operation times of workers (DOW)
- Min. the walking time (WT)
Mechanisms (but uncontrollable conditions not condidered)
- Deterministic operation times
- Deterministic manual time
- Deterministic walking times
- Identical skilled workers
- No crossing path
- Heuristic rule
- Random priority
Evolutionary optimizer
- COINcidence algorithm

Figure 2 Evolutionary combinatorial optimization process of SUALWAPs-I.

doi:10.4186/ej.2010.14.2.53

3.2 Input parameters

In any time period, the number of jobs is deterministic and job arrivals come from not only new customer orders but also remaining jobs from the previous planning period that were not completed. Each job is an entity worked on many tasks. No job priority (i.e. no preemption job) constraint is allowed: that is, each job is allowed to start its processing whenever it is ready. These jobs are sorted by the daily production order excluding the sequencing problem. The ten standard precedence graphs (7-task to 297-task assembly networks) and various cycle times (the time which is available at each station to perform all the tasks assigned to the station) are input in U-shaped assembly line worker allocation problems [26]. They are referred to in column one and two of Table 4 in Section 3.5. Given precedence graphs for an assembly line are produced from the process of making intermediate parts in the final assembly line.

3.3 Characteristics of a single assembly U-line

Although there are many types of U-lines, the configuration of this study is a single U-shaped assembly line only. The U-line arranges machines or tasks around a U-shaped line in the order in which production tasks are serial. The sequence of tasks on the U-line is not fixed, making it possible to reallocate tasks to different line locations. Thus, the assignment of tasks to line locations can be altered. The system is one-piece flow manufacturing moving one piece at a time between tasks within a U-line. One floating worker supervises both the entrance and the exit of the line. The task efficiency is proportional to the worker’s performance. Machine-work is not separated from worker-work. Standard operation charts specify exactly how all work is done. Workers can be reallocated periodically when production requirements change (or cycle time changes). This requires workers to have multi-functional skills to operate several different machines or tasks. It also requires workers to work standing up and walking because they need to operate at different locations. Whenever a worker arrives at a task, one performs any needed tasks at the task location, and then walks to the next task. Following the last task of a path, the worker returns to the starting point and works or waits for the start of the next cycle. The characteristics of the single U-shaped assembly line worker allocation problem of type I are shown in Figure 3(a).

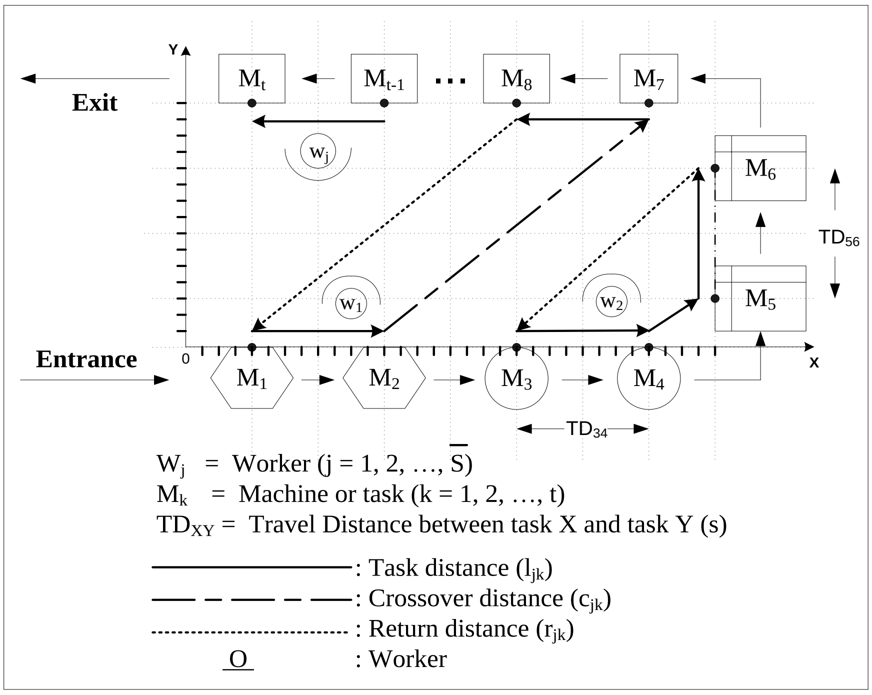

Y
Exit
Entrance
0
X
M_t
M_{t-1}
M_8
M_7
M_6
M_5
M_1
M_2
M_3
M_4
w_j
w_1
w_2
TD_{56}
TD_{34}
W_j = Worker (j = 1, 2, …, $\overline{S}$)
M_k = Machine or task (k = 1, 2, …, t)
TD_{XY} = Travel Distance between task X and task Y (s)
: Task distance (l_{jk})
: Crossover distance (c_{jk})
: Return distance (r_{jk})
O : Worker

Figure 3(a)
Mapping a diagram of a single U-shaped assembly line for *j* workers and *k* machines on grid arrangement.

The U-line layouts illustrated in Table 2 are configured with symmetrical and rectangular shapes at the side ratios of 1/3 (1:1:1) and 1/9 (1:4:4), respectively. They are representative enough for the workable single U-line. The line consists of *M* machines or k tasks in a U-shape arrangement with the side of M odd or M even. Walking distance and travel speed of each worker are considered into walking time in section 3.5. The values of network density (D) are also shown in column III of Table 2. Network density, a characteristic which measures the strength of this relationship, has been found to be an important factor in influencing heuristic performance in previous investigations of the line balancing problem [18]. The density of the assembly network is defined as the ratio of D , Let d denote the total number of precedence relations in the precedence graph, and N denote the number of task on a U-line. Then, $D = 2d/[N(N - 1)]$ where $D \leq 1.0$. Values of D close to 1 indicate a highly interconnected network with fewer alternatives available for assigning tasks to a work station. Values of D close to 0 indicate relatively fewer precedence relationships, and more opportunities for assigning tasks to a work station.

Table 2
Symmetrical (1/3 side ratio) and rectangular (1/9 side ratio) U-lines.

Problem	Number of tasks	Network density (D)	U-shaped layouts (side:front:back)					
			Symmetry (1:1:1)			Rectangle (1:4:4)		
			side	front	back	side	front	back
1. Merten (1967)	7	0.2857	1	3	3	1	3	3
2. Miltenburg (2001)	10	0.0667	2	4	4	2	4	4
3. Jackson (1956)	11	0.2364	3	4	4	1	5	5
4. Thomopoulos (1970)	19	0.1228	7	6	6	3	8	8
5. Heskiaoff (1968)	28	0.1032	10	9	9	4	12	12
6. Kilbridge&Wester (1961)	45	0.0626	15	15	15	5	20	20
7. Kim (2006)	61	0.0361	21	20	20	7	27	27
8. Tongue (1961)	70	0.0356	24	23	23	8	31	31
9. Arcus (1963)	111	0.0283	37	37	37	11	50	50
10. Scholl&Klein (1999)	297	0.0096	99	99	99	33	132	132

3.4 Mathematical model

In this section, a mathematical model for solving the worker allocation problem is described. The symbols used are listed and explained in the next section.

3.4.1 Notation used

The notation used in this section can be summarized as follows:

Indices

j	=	index on workers
k	=	index on tasks

Parameters

t	=	number of tasks in the U-line
C	=	given cycle time
P	=	$\{(k,l):$ task k must be completed before task l can begin$\}$ – precedence constraints
t_k	=	time to complete manual task at task k
α	=	coefficient of walking time

Variables

$\bar{S}$	=	number of workers in the U-line
C_{jk}	=	cycle time at task k assigned to worker j

doi:10.4186/ej.2010.14.2.53

m	=	minimum number of workers
l_{jk}	=	task distance at task k assigned to worker j
c_{jk}	=	crossover distance at task k assigned to worker j
r_{jk}	=	return distance at task k assigned to worker j
T_{jk}	=	task time at task k assigned to worker j
WT_{jk}	=	walking time at task k assigned to worker j

Decision variables

$$x_{jk} = \begin{cases} 1 & \text{if task k is located on front of line and assigned to worker j} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jl} = \begin{cases} 1 & \text{if task k is located on back of line and assigned to worker j} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{jl} = \begin{cases} 1 & \text{if task l is located on front of line and assigned to worker j} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jl} = \begin{cases} 1 & \text{if task l is located on back of line and assigned to worker j} \\ 0 & \text{otherwise} \end{cases}$$

$$z_j = \begin{cases} 1 & \text{if worker j is used} \\ 0 & \text{otherwise} \end{cases}$$

3.4.2 Model formulation

This section identifies the minimum number of workers (workstations) in Eq. (1) required in the U-line to obtain the optimum of dual objectives. Besides aiming to increase productivity (minimizing the number of workers or the cycle time), some other goals are important for the addition of high productivity achievements, i.e., a sense of equity among workers and the shortest travel path. Hierarchically both objective functions are calculated accordingly in the same unit of time from Eq. (2), (3), and (4). An ineffective allocation of workers to tasks and machines would yield long idle times (imbalance workload) and long walking time.

Then select x_{jk}, y_{jk}, z_j to,

$$\text{(ILP) Minimize } \sum_{j=1}^{\bar{S}} z_j \tag{1}$$

After computing the minimum number of workers in the first step, it is necessary to evaluate and minimize the deviation of operation times of workers (DOW) and the walking time (WT) as seen in Section 3.4.2.1 with Pareto-optimal frontier.

3.4.2.1 Objective functions

I. Min. the deviation of operation times of workers (DOW)

$$\text{DOW} = \sqrt{\frac{\sum_{j=1}^{\bar{S}} \sum_{k=1}^{t} (C - C_{jk})^2}{m}} \tag{2}$$

$$\underset{\textit{(U-Worker Balancing)}}{\text{Cycle Time } (C_{jk})} = \underset{\textit{(U-line Balancing)}}{\text{Task Time } (T_{jk})} + \text{Walking Time } (WT_{jk}) \tag{3}$$

II. Min. the walking time (WT)

$$WT = \sum_{j=1}^{\bar{s}} \sum_{k=1}^{t} \left(l_{jk} + c_{jk} + r_{jk} \right) \tag{4}$$

3.4.2.2 Constraints

Subject to:

$$\sum_j (x_{jk} + y_{jk}) = 1 \qquad \text{for each task } k \tag{5}$$

doi:10.4186/ej.2010.14.2.53

$$\sum_j (\bar{S} - j + 1)(x_{jk} - x_{jl}) \geq 0 \qquad \text{for all } (k,l) \in P \qquad (6)$$

$$\sum_j (\bar{S} - j + 1)(y_{jl} - y_{jk}) \geq 0 \qquad \text{for all } (k,l) \in P \qquad (7)$$

$$x_{jk}, y_{jk}, z_j \in \{0,1\} \qquad \text{for all j, k} \qquad (8)$$

The first constraint in Eq. (5) ensures that every task is located on the front or back of the line and is assigned to one worker. The next constraint in Eq. (6) ensures that the precedence constraints are satisfied for each task assigned to the front of the line. The following constraint in Eq. (7) does the same for the tasks of the equation (6) assigned to the back of the line. In other words, constraint (6) enforces task sequence assigned on the U-line by a set of ordered pairs of tasks reflecting the precedence relationships; for example, P = (3,1), (5,10) and (6,9) is the ordered pair of Miltenburg's 10-task problem indicating task k precedes task l. X_{jk} or/and X_{jl} is 1 when worker j does task k or/and task l. Otherwise, its value is 0 or their values are 0. Constraints (7) is the same, but is reversed because task on the U-line can be also assigned at the back line. The variables of x, y, and z are binary solution in Eq. (8). However, Miltenburg [3] does not take walking distance into account and may not find the best U-line design. Thus, the last constraint of walking time in Eq. (9) is essential to complete the worker allocation problem. The constraint proves that the sum of the manual task times for the tasks in each worker in the first term and the total walking distance in the second term does not exceed the cycle time, C. The coefficient of walking time (α) is varied by TD_{XY} in Figure 3(a) or the percentage of Average Processing Time (APT) from one task to another task. The average processing time is defined as APT = $\sum_{k=1}^{t} t_k / t$.

$$w_j = \sum_j T_{jk} + \alpha \sum_j (l_{jk} + c_{jk} + r_{jk}) \leq C \qquad \text{for all j,k} \qquad (9)$$

3.4.2.3 Illustrative example

For Miltenburg's 10-task problem, the precedence graph is (3,1), (5,10) and (6,9) and task$^{\text{Manual time}}$ is 1^3, 2^4, 3^2, 4^2, 5^4, 6^4, 7^3, 8^2, 9^2, 10^2. Suppose that Task Sequence (TS) on the U-line is -1_7_2_-10_6_5_8_9_4_3. This means task 1, which is assigned to the back location of the line. Then task 7 is assigned to the front location on the front of the line. Finally, the rest of task sequence is assigned. The sequence is feasible because the precedence constraints are satisfied with five tasks on the front and five tasks on the back. Suppose C = 10 seconds, and α = 1 at 35% average processing time. Then the assigned task to workers is [-1_7]_[2_-10]_[6_5]_[8_9_4]_[3] shown in Figure 3(b). Tasks 1 and 7 are in the worker 1 and the manual times (T_{jk}) of T_{11} and T_{17} are 3 and 3, respectively. From Eq. (3), $W_1 = (T_{11} + T_{12}) + \alpha \times WT_{1k}$; where $WT_{1k} = l_{11} + l_{12} + c_{11} + r_{12}$. Thus, $W_1 = (3+3) + 1 \times (0+0+2+2) = 10$ seconds. After that, the cycle times (C_{jk}) of worker 2 to worker 5 are computed as the same.

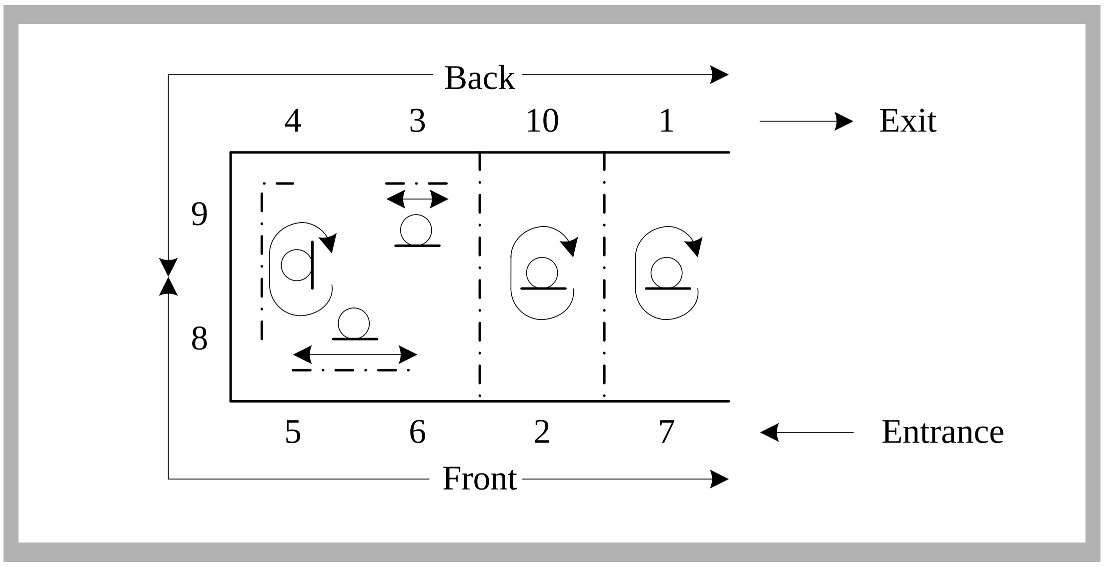


Figure 3(b)
An illustrative example of Miltenburg's 10-task problem.

doi:10.4186/ej.2010.14.2.53

3.4.3 Assumptions

In this study, the SUALWAPs-I is subjected to the following assumptions:

- A U-line comprises inexpensive and small non-automated machines. Several identical machines may be found and machines are enough to be allocated in a single U-line;
- Machines or tasks are located via a grid arrangement with the same distance of %APT between adjacent task locations in the same row. For other non-adjacent task locations, the walking distance is calculated by the displacement of Euclidean distance;
- Trained homogeneous skilled workers have the same efficiency and multi- functional skills and are able to operate any processes or machines. They walk in a circle inside the U-line (also called the zone constraint – machines allocated to each worker must be adjacently located within a loop);
- A worker is assigned to one station (or one loop) only;
- All parameters and variables such as processing times and walking times are deterministic (known and constant);
- The completion time of a machine or task summed with many subordinate tasks is known and a task cannot be split between two or more workers;
- Precedence relationships of the problem are consistent from model to model. That is, if task k precedes task l in any model there is no other model where task l must precede task k. Each unit of products is processed through all tasks in the same precedence order;
- Setup times (assumed to be less than 10% compared with processing time) are negligible. U-lines can be operated as single-model and mixed-model lines where each worker is able to produce any product in any cycle. Consequently, job sequence is regardless at any period;
- The mixed-model task times use the weight of composite demand to transform average task time into the task of a single model. However, a floating worker may be assisted unless task times in some model are feasible;
- Learning effect has no consideration since it is assumed that worker performance runs into steady state already;

The mathematical model of this research is not studied in depth because minimizing the number of workers, DOW and WT at the same time make the exact solution too complex to deal with.

3.5 Determination of walking time

In the previous papers [3], [17], [26] the coefficient of walking time (α) is required to travel a unit of distance or one time unit. Thus, in this study the adjacent matrix (From-To chart) of walking times under displacement distance for each problem of the symmetrical and rectangular shape is initially constructed at one time unit from one location to another location. Each of the walking times between a pair of tasks is directly proportional to Euclidean distance between locations. The example of walking times for the 10-task problem is shown in Table 3 (a). For example, the displacement gives us a travel distance of 0.7071 distance units (or time units), calculated as the sum of distance between location $\|(3,0),(3.5,0.5)\| = \sqrt{(3.5-3)^2 + (0.5-0)^2} = 0.7071$, where $\|\bullet\|$ is a Euclidean distance operator. *Note that*: Location 1 is assumed to be an origin (0,0).

In practice, a worker keeps walking more than one second. However, Balakrishnan *et al.* [34] assumingly use two values of travel time for each problem instance, (i.e. walking time = the five or ten percentage of Average Processing Times (APT), where APT is the expected value of processing times which is defined as $APT = \sum_{k=1}^{t} t_k / t$; where t_k is task time k and t is the summation of task 1 to k. In this study, the values of APT percentage are varied from 5% to 120% in Table 4 to find out the initial value that affect the addition of a number of workers for each problem. As an example, the adjacent matrix of walking times for the 10-task problem at the 2:4:4 U-shaped layout is exemplified at 5% APT and shown in Table 3(b). Afterwards, the matrix that specifies the minimum APT percentage is input to the solution of minimum walking time.

doi:10.4186/ej.2010.14.2.53

Table 3
3(a) Exemplified displacement distance for U-line $\frac{task(10)}{side(2)}$ at one time unit from one location to another location.

3(b) Exemplified displacement distance for U-line $\frac{task(10)}{side(2)}$ at 5% APT.

Walking Time		To									
		1	2	3	4	5	6	7	8	9	10
From	1	0.0000	1.0000	2.0000	3.0000	3.5355	3.8079	3.6056	2.8284	2.2361	2.0000
	2	1.0000	0.0000	1.0000	2.0000	2.5495	2.9155	2.8284	2.2361	2.0000	2.2361
	3	2.0000	1.0000	0.0000	1.0000	1.5811	2.1213	2.2361	2.0000	2.2361	2.8284
	4	3.0000	2.0000	1.0000	0.0000	0.7071	1.5811	2.0000	2.2361	2.8284	3.6056
	5	3.5355	2.5495	1.5811	0.7071	0.0000	1.0000	1.5811	2.1213	2.9155	3.8079
	6	3.8079	2.9155	2.1213	1.5811	1.0000	0.0000	0.7071	1.5811	2.5495	3.5355
	7	3.6056	2.8284	2.2361	2.0000	1.5811	0.7071	0.0000	1.0000	2.0000	3.0000
	8	2.8284	2.2361	2.0000	2.2361	2.1213	1.5811	1.0000	0.0000	1.0000	2.0000
	9	2.2361	2.0000	2.2361	2.8284	2.9155	2.5495	2.0000	1.0000	0.0000	1.0000
	10	2.0000	2.2361	2.8284	3.6056	3.8079	3.5355	3.0000	2.0000	1.0000	0.0000

Walking Time		To									
		1	2	3	4	5	6	7	8	9	10
From	1	0.0000	0.1400	0.2800	0.4200	0.4956	0.5334	0.5054	0.3962	0.3136	0.2800
	2	0.1400	0.0000	0.1400	0.2800	0.3570	0.4088	0.3962	0.3136	0.2800	0.3136
	3	0.2800	0.1400	0.0000	0.1400	0.2212	0.2968	0.3136	0.2800	0.3136	0.3962
	4	0.4200	0.2800	0.1400	0.0000	0.0994	0.2212	0.2800	0.3136	0.3962	0.5054
	5	0.4956	0.3570	0.2212	0.0994	0.0000	0.1400	0.2212	0.2968	0.4088	0.5334
	6	0.5334	0.4088	0.2968	0.2212	0.1400	0.0000	0.0994	0.2212	0.3570	0.4956
	7	0.5054	0.3962	0.3136	0.2800	0.2212	0.0994	0.0000	0.1400	0.2800	0.4200
	8	0.3962	0.3136	0.2800	0.3136	0.2968	0.2212	0.1400	0.0000	0.1400	0.2800
	9	0.3136	0.2800	0.3136	0.3962	0.4088	0.3570	0.2800	0.1400	0.0000	0.1400
	10	0.2800	0.3136	0.3962	0.5054	0.5334	0.4956	0.4200	0.2800	0.1400	0.0000

doi:10.4186/ej.2010.14.2.53

Problems / Number of tasks	Cycle Time (s.)	Average Processing (s.)	Walking time from one location to another location (s.)															
			5%	10%	15%	20%	25%	30%	35%	40%	50%	60%	70%	80%	90%	100%	110%	120%
1. Merten / 7	10	4.14	0.21	0.41	0.62	0.83	1.04	1.24	1.45	1.66	2.07	2.48	2.90	3.31	3.73	4.14	4.55	4.97
2. Miltenburg / 10	10	2.8	0.14	0.28	0.42	0.56	0.70	0.84	0.98	1.12	1.40	1.68	1.96	2.24	2.52	2.80	3.08	3.36
3. Jackson / 11	13	4.18	0.21	0.42	0.63	0.84	1.05	1.25	1.46	1.67	2.09	2.51	2.93	3.34	3.76	4.18	4.60	5.02
4. Thomsopoulos / 19	120	20.42	1.02	2.04	3.06	4.08	5.11	6.13	7.15	8.17	10.21	12.25	14.29	16.34	18.38	20.42	22.46	24.50
5. Heskiaoff / 28	256	36.57	1.83	3.66	5.49	7.31	9.14	10.97	12.80	14.63	18.29	21.94	25.60	29.26	32.91	36.57	40.23	43.88
6. Kilbridge & Wester / 45	110	12.27	0.61	1.23	1.84	2.45	3.07	3.68	4.29	4.91	6.14	7.36	8.59	9.82	11.04	12.27	13.50	14.72
7. Kim / 61	600	86.50	4.33	8.65	12.98	17.30	21.63	25.95	30.28	34.60	43.25	51.90	60.55	69.20	77.85	86.50	95.15	103.80
8. Tongue / 70	251	50.14	2.51	5.01	7.52	10.03	12.54	15.04	17.55	20.06	25.07	30.08	35.10	40.11	45.13	50.14	55.15	60.17
9. Arcus / 111	7,916	1,312.23	65.61	131.22	196.83	262.45	328.06	393.67	459.28	524.89	656.12	787.34	918.56	1,050	1,181	1,312	1,443	1,575
10. Scholl & Klein / 297	1,834	234.53	11.73	23.45	35.18	46.91	58.63	70.36	82.09	93.81	117.27	140.72	164.17	187.62	211.08	234.53	257.98	281.44

Table 4
Average processing time percentage of 5-120 for all problems.

doi:10.4186/ej.2010.14.2.53

IV. Multiple Objective Coincidence Algorithm

4.1 Proposed algorithm

Wattanapornprom *et al*. [35] developed a new effective evolutionary algorithm called *combinatorial optimization with coincidence* (COIN) originally aiming to solve traveling salesman problems. Several benchmarks are compared to the experiment of Robles et al. [36]. The idea is that most well-known algorithms such as Genetic Algorithm (GA) search for good solutions by sampling through crossover and mutation tasks without much exploitation of the internal structure of good solution strings. This may not only generate large number of inefficient solutions dissipated over the solution space but also consume long CPU time. In contrast, COIN considers the internal structure of good solution strings and memorizes paths that could lead to good solutions. COIN replaces high computation time of crossover and mutation tasks of GA and employs a joint probability matrix as a means to generate neighborhood solutions. It prioritizes the selection of the paths with higher chances of moving towards good solutions.

Apart from traditional learning from good solutions, COIN allows learning from below average solutions as well. Any coincidence found in a situation can be statistically described whether the situation is good or bad. Most traditional algorithms always discard the bad solutions without utilizing any information associated with them. In contrast, COIN learns from the coincidence found in the bad solutions and uses this information to avoid such situations to be recurrent; meanwhile, experiences from good coincidences are also used to construct better solutions in Figure 4. Consequently, the chances that the paths being parts of the bad solutions are always used in the new generations are lessened. This lowers the number of solutions to be considered and hence increases the convergence speed.

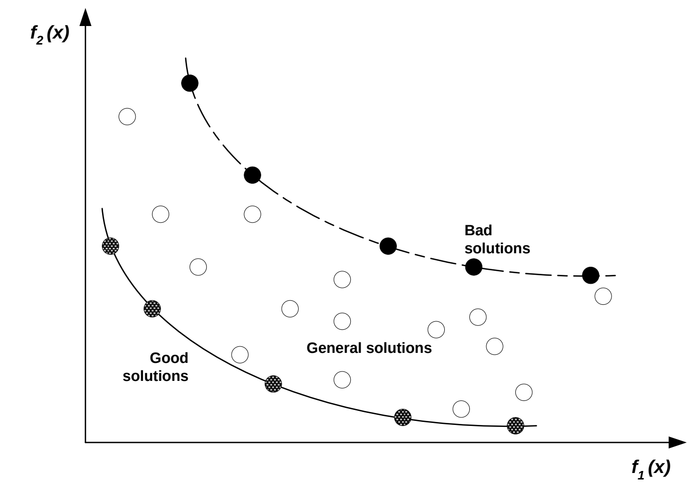


Figure 4
Good and bad solutions.

COIN uses a joint probability matrix (generator) to create the population. The generator is initialized so that it can generate a random tree with equal probability for any configuration. The population is evaluated in the same way as traditional evolutionary algorithms. However, COIN uses both good and bad solutions to update the generator. Initially, COIN searches from a fully connected tree and then incrementally strengthening or weakening the connections. As generations pass by, the probabilities of selecting certain paths are increased or decreased depending on the incidences found in the good or bad solutions. The algorithm of COIN can be stated as follows.

1. To initialize the joint probability matrix (generator);
2. To generate the population using the generator;
3. To evaluate the population;
4. To make diversity preservation;
5. To select the candidates according to two options: (a) good solution selection (select the solutions in the first rank of the current Pareto frontier), and (b) bad solution selection (select the solutions in the last rank of the current Pareto frontier);

6. For each joint probability matrix $H(x_i, x_j)$, to adjust the generator according to the reward and punishment scheme as Eq.(10);

$$x_{i,j}(t+1) = x_{i,j}(t) + \frac{k}{(n-1-np_i)}\{r_{i,j}(t+1) - p_{i,j}(t+1)\} + \frac{k}{(n-1-np_i)^2}\{\textstyle\sum_{j=1}^{n} p_{i,j}(t+1) - \sum_{j=1}^{n} r_{i,j}(t+1)\} \quad (10)$$

where $x_{i,j}$ = the element (i, j) of joint probability matrix $H(x_i / x_j)$, k = the learning coefficient, $r_{i,j}$ = the number of coincidences (x_i, x_j) found in the good solutions, $p_{i,j}$ = the number of coincidences (x_i, x_j) found in the bad solutions, t = generation number, n = the size of the problem, and np_i = the number of the direct predecessors of task i ;

7. To apply a strategy to maintain elitist solutions in the population, and then repeat step 2 until the terminating condition is met.

4.2 Numerical example

The 10-task problem of the single product with 10-minute cycle time originated by Miltenburg [3] is used to elaborate the algorithm of COIN. The manual task times are two time units for tasks 3, 4, 8, 9, 10, three time units for tasks 1, 7, and four time units for tasks 2, 5, 6. The precedence constraints are (3,1), (5,10), and (6,9). The fixed U-shaped layout of the side, front, and back is 2, 4, and 4 respectively. The walking time from one task to another task is five percent of average processing time. Since the total tasks time is 28, $0.05*(28/10) = 0.14$. As a result, the walking time matrix of 5% APT for each element $(x_{i,j})$ is shown in Table 5. The task assignment rule is randomized. Learning probability (k) and reward or punishment values are assumed to be 0.1. Population size is ten chromosomes and two generations are described step by step as follows.

x_i to x_j	1	2	3	4	5	6	7	8	9	10
1	0.00	0.14	0.28	0.42	0.50	0.53	0.50	0.40	0.31	0.28
2	0.14	0.00	0.14	0.28	0.36	0.41	0.40	0.31	0.28	0.31
3	0.28	0.14	0.00	0.14	0.22	0.30	0.31	0.28	0.31	0.40
4	0.42	0.28	0.14	0.00	0.10	0.22	0.28	0.31	0.40	0.50
5	0.50	0.36	0.22	0.10	0.00	0.14	0.22	0.30	0.41	0.53
6	0.53	0.41	0.30	0.22	0.14	0.00	0.10	0.22	0.36	0.50
7	0.50	0.40	0.31	0.28	0.22	0.10	0.00	0.14	0.28	0.42
8	0.40	0.31	0.28	0.31	0.30	0.22	0.14	0.00	0.14	0.28
9	0.31	0.28	0.31	0.40	0.41	0.36	0.28	0.14	0.00	0.14
10	0.28	0.31	0.40	0.50	0.53	0.50	0.42	0.28	0.14	0.00

Table 5
The walking time matrix of 5% APT.

4.2.1 Joint probability matrix initialization

The number of tasks to be considered is 10. Therefore, the dimension of From-To joint probability matrix $H(x_i, x_j)$ is the matrix 10x10. The value of each element ($x_{i,j}$) in the matrix is the probability of selecting task j after task i. In order to incorporate some precedence relationship into the matrix, in each row the element which belongs to the direct predecessor of the task is set to 0 to prohibit producing such a task before its direct predecessor. For example, the direct predecessor of task 1 is task 3; hence, $x_{1,3}$ = 0. Also, $x_{1,1}$ = 0, since it cannot move within itself. Initially, the value of the remaining elements in the first row of the matrix are equal to $1/(n-1-np_1)=1/(10-1-1)=0.125$. Continuing this computation for all the remaining tasks (rows), the initial joint probability matrix is shown in Table 6.

Table 6
Initial probability joint matrix.

Task	1	2	3	4	5	6	7	8	9	10
1	0.0000	0.1250	0.0000	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
3	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
4	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
5	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111
6	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111
7	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111
8	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111
9	0.1250	0.1250	0.1250	0.1250	0.1250	0.0000	0.1250	0.1250	0.0000	0.1250
10	0.1250	0.1250	0.1250	0.1250	0.0000	0.1250	0.1250	0.1250	0.1250	0.0000

4.2.2 Population generation

The order representation scheme is used to create chromosomes. The task order list in a chromosome is created by moving forward from one task to another task. If more than one possible task can be selected, the probability of selecting any task will depend on its value on the joint probability matrix. In each generation, the first task (or the first order pair) is selected from the current elitist of the first Pareto-ranked chromosome(s). The same probability of selection will be randomized. For example, task 7 is randomly selected for the first position. After selecting the row of task 7, the set of eligible tasks comprises tasks 1 to 6 and 8 to 10. From row 7 of the joint probability matrix, a task is randomly selected according to its probability of selection ($p_{7,j}=0.1111,$ for $j=1,2,3,4,5,6,8,9,10$). Supposing that task 6 is selected, the new set of eligible tasks becomes tasks 1 to 5 and 8 to 10. This mechanism is continued as long as all positions in the task order list are filled in and the task order list of L_1={7,6,2,3,1,4,8,9,5,10} is obtained. As the population size is assumed to be 10, the nine remaining initial population consists of chromosomes L_2={8,6,5,3,1,9,10,2,4,7}, L_3={5,4,6,8,9,3,2,1,7,10}, L_4={8,5,7,2,10,4,6,3,9,1}, L_5={3,1,7,6,8,5,4,9,10,2}, L_6={3,7,4,8,5,6,1,10,2,9}, L_7={5,3,2,6,4,10,9,1,8,7}, L_8={4,8,7,5,10,6,3,2,1,9}, L_9={2,4,7,8,6,3,9,5,10,1}, and L_{10}={2,3,6,8,4,1,7,5,10,9}.

4.2.3 Population evaluation

To find tentative tasks to be allocated on the U-line, all tasks have to be searched through the task order list in both forward and backward directions. The tentative task on forward or backward searching is found first. The task has its task time and walking time to the next task less than or equal to the remaining worker cycle time. If both forward and backward tentative tasks are found, either one is selected randomly. If any task from the task order list has not yet being allocated, a new workstation is opened. This procedure is repeated for the remaining task order list to obtain the number of workers (or workstations), walking time and worker load distribution for each of them. An example chromosome (L_1) is shown in Table 7. The deviation of operation times of workers is calculated from Eq. (3) with C_{jk}= 7.28, 9.46, 6.46, and 6.28 respectively. Thus, it is equal to 2.918 time units. The walking times of 0.14, 0.14, 0.14, 0.14, 0.10, 0.22, 0.10, 0.10, 0.14, 0.22, 0.14, and 0.14 are summed and the total walking time is equal to 1.72 time units. Having obtained feasible worker allocations three objectives have to be evaluated for each chromosome. Table 8 indicates that all chromosomes give the

doi:10.4186/ej.2010.14.2.53

same number of workstations; therefore, all of them are eligible for Pareto ranking based on Deviation of Operation times of Workers (DOW) and Walking Time (WT) objectives. The Pareto ranking technique proposed by Goldberg [32] is used to classify the population into non-dominated frontiers with a dummy fitness value that lower value is better. They are assigned to each chromosome in Figure 5.

Worker or Workstation	Task considered Front graph	Task considered Back graph	Task assignment on a U-line			(3) Total (1) + (2) Cycle time (time unit)	(4) WT to Origin (time unit)	(5) Given cycle time	Idle time (time unit) (5)-(3)-(4)
			Task	(1) WT (time unit)	(2) Task time (time unit)				
1	7	- 10	7	-	3	3	-	10	7
1	6	- 10	6	0.14	4	4.14 [7.14]	0.14		2.72
2	2	- 10	2	-	4	4	-	10	6
2	3	- 10	3	0.14	2	2.14 [6.14]	0.14		3.72
2	1	- 10	1	0.10	3	3.10 [9.24]	0.22		0.54
3	4	- 10	4	-	2	2	-	10	8
3	8	- 10	8	0.10	2	2.10 [4.10]	0.10		5.80
3	9	- 10	9	0.14	2	2.14 [6.24]	0.22		3.54
4	5	- 10	5	-	4	4	-	10	6
4	-	- 10	10	0.14	2	2.14 [6.14]	0.14		3.72

Table 7
An example of worker allocation in a single U-line.

Chromosome Number	Number of workers	DOW	WT	Pareto Frontier	Crowding Distance
L5	4	2.8608	1.7373	1	Infinite
L1	4	2.9179	1.7197	1	Infinite
L10	4	2.8608	1.7783	2	Infinite
L2*	4	3.1136	1.7197	2	Infinite
L3*	4	3.1136	1.7197	2	Infinite
L8	4	2.9605	1.7960	3	Infinite
L7	4	3.1297	1.8773	4	Infinite
L4**	4	3.6619	1.9593	5	Infinite
L9**	4	3.6619	1.9593	5	Infinite
L6	4	4.0981	2.2400	6	Infinite

Table 8
Objective functions of each chromosome from the first generation.

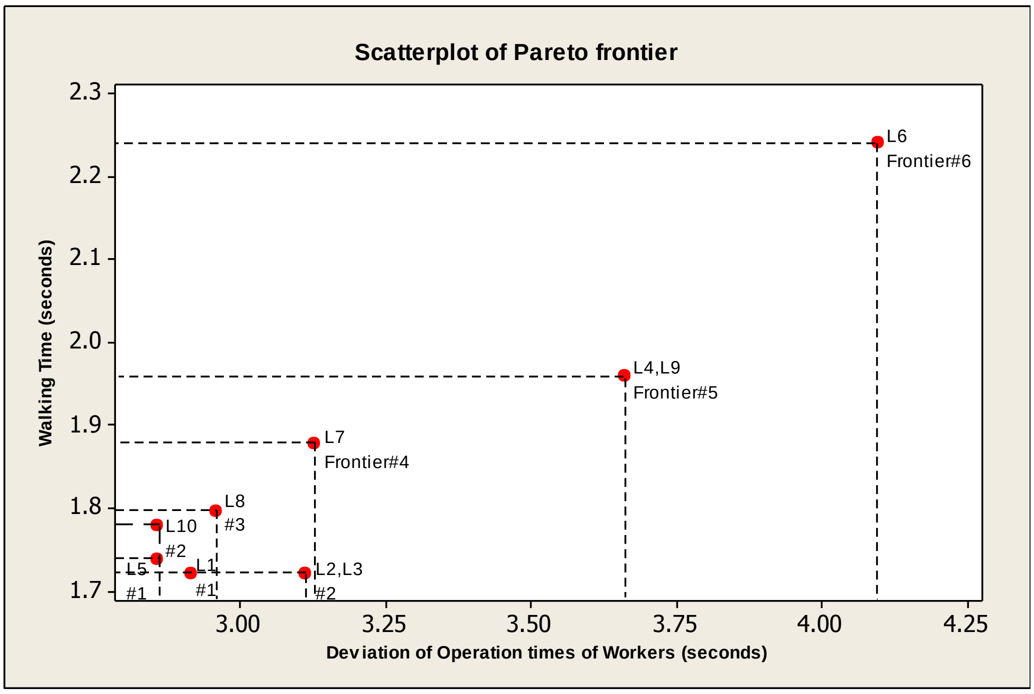


Figure 5
Pareto frontier of each chromosome.

4.2.4 Diversity preservation

COIN employs a crowding distance approach [32] to generate a diversified population uniformly spread over the Pareto frontier and avoids a genetic drift phenomenon (a few clusters of populations being formed in the solution space). The salient characteristic of this approach is that there is no need to define any parameter in calculating a measure of population density around a solution. The crowding distances computed for all solutions are infinite since at least two solutions are found for each frontier. Although both objectives of the chromosome L_2 are the same as L_3, and L_4 are the same as L_9, task sequence of each chromosome is different.

4.2.5 Solution selection

Having defined the Pareto frontier, the good solution is the chromosome located on the first Pareto frontier (dummy fitness = 1) and there is only one chromosome by the multiplication of reward value and population sizes, i.e. 0.1*10 = 1 solution. However, there are two solutions (L_1 and L_5) in the first rank. One of both is randomized, i.e. L_5 = {3,1,7,6,8,5,4,9,10,2}. In contrast, the bad solution is one located on the last Pareto frontier (dummy fitness = 6) and there is only one chromosome by the multiplication of punishment value and population sizes, i.e. 0.1*10 = 1, L_6={3,7,4,8,5,6,1,10,2,9}.

4.2.6 Joint probability matrix adjustment

The adjustment of the joint probability matrix is crucial to the performance of COIN. Reward will be given to $x_{i,j}$ if the order pair (i, *j*) is in the good solution to increase the chance of selection in the next round. For example, the first order pair (3,1) is the good solution of the chromosome L_5 = {3,1,7,6,8,5,4,9,10,2}. Assumed that k=0.1 ; therefore, the value of $x_{i,j}$ where i=3 and *j*=1 is increased by k/(n-1-np$_3$)=0.1/(10-1-0)=0.0111 The updated value of $x_{i,j}$ of the order pair (3,1) becomes 0.1111+0.0111=0.1222 The values of the other order pairs located in the same row of the order pair (3,1) is reduced by k/(n-1-np$_3$)^2=0.1/(10-1-0)^2=0.1/81=0.0012 For example, the value $x_{i,j}$ where i=3 and *j*=1 is 0.1222-0.0012=0.121 For the other positions of *j* in the third row, each adjusted joint probability is 0.1111-0.0012=0.1099 Previously, the summation of probability of $x_{3,j}$ is equal to one. Continuing this procedure to all order pairs located in the good solution; the revised joint probability matrix is obtained in Table 9.

Table 9
Revised joint probability matrix (good solution).

Task	1	2	3	4	5	6	7	8	9	10
1	0.0000	0.1238	0.0000	0.1238	0.1238	0.1238	0.1349	0.1238	0.1238	0.1238
2	0.1111	0.0000	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111
3	0.1210	0.1099	0.0000	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099	0.1099
4	0.1099	0.1099	0.1099	0.0000	0.1099	0.1099	0.1099	0.1099	0.1210	0.1099
5	0.1099	0.1099	0.1099	0.1210	0.0000	0.1099	0.1099	0.1099	0.1099	0.1099
6	0.1099	0.1099	0.1099	0.1099	0.1099	0.0000	0.1099	0.1210	0.1099	0.1099
7	0.1099	0.1099	0.1099	0.1099	0.1099	0.1210	0.0000	0.1099	0.1099	0.1099
8	0.1099	0.1099	0.1099	0.1099	0.1210	0.1099	0.1099	0.0000	0.1099	0.1099
9	0.1238	0.1238	0.1238	0.1238	0.1238	0.0000	0.1238	0.1238	0.0000	0.1349
10	0.1238	0.1349	0.1238	0.1238	0.0000	0.1238	0.1238	0.1238	0.1238	0.0000

On the contrary, if the order pair (i, *j*) is in the bad solution, $x_{i,j}$ will be penalized to reduce the chance of selection in the next round. For example, the first order pair (3,7) is in the bad solution of the chromosome L_6={3,7,4,8,5,6,1,10,2,9}. Assuming that $k = 0.1$; hence, the value of $x_{i,j}$ where i = 3 and $j = 7$ is decreased by k/(n-1-np$_3$) = 0.1/(10-1-0) = 0.0111 The updated value of $x_{i,j}$ of the order pair (3,7), which is later adjusted from Table 9 becomes 0.1099 – 0.0111 = 0.0988. The values of the other order pairs located in the same row of the order pair (3,7) is increased by k/(n-1-np$_3$)^2=0.1/(10-1-0)^2=0.1/81=0.0012 For example, the value $x_{i,j}$ where i=3 and *j*=7 is 0.0988+0.0012=0.1000. For the position of *j*=1 in the 3rd row, the adjusted joint probability is 0.1210+0.0012=0.1222. For the other positions of *j* in the third row, each adjusted joint probability is 0.1099+0.0012=0.1111. Continuing this procedure to all order pairs located in the bad solution, the revised joint probability matrix is obtained in Table 10.

Task	1	2	3	4	5	6	7	8	9	10
1	0.0000	0.1250	0.0000	0.1250	0.1250	0.1250	0.1361	0.1250	0.1250	0.1139
2	0.1123	0.0000	0.1123	0.1123	0.1123	0.1123	0.1123	0.1123	0.1012	0.1123
3	0.1222	0.1111	0.0000	0.1111	0.1111	0.1111	0.1000	0.1111	0.1111	0.1111
4	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111	0.1111	0.1000	0.1222	0.1111
5	0.1111	0.1111	0.1111	0.1222	0.0000	0.1000	0.1111	0.1111	0.1111	0.1111
6	0.1000	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1222	0.1111	0.1111
7	0.1111	0.1111	0.1111	0.1000	0.1111	0.1222	0.0000	0.1111	0.1111	0.1111
8	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.1111	0.0000	0.1111	0.1111
9	0.1238	0.1238	0.1238	0.1238	0.1238	0.0000	0.1238	0.1238	0.0000	0.1349
10	0.1250	0.1250	0.1250	0.1250	0.0000	0.1250	0.1250	0.1250	0.1250	0.0000

Table 10
Revised joint probability matrix (bad solution).

4.2.7 Elitism

To keep the best solutions found and to survive in the next generation, COIN uses an external list with the same size as the population size to store elitist solutions. All non-dominated solutions created in the previous population are combined with the current elitist solutions. Goldberg’s Pareto ranking technique is used to classify the combined population into several non-dominated frontiers. Only the solutions in the first non-dominated frontier are filled in the new elitist list. If the number of solutions in the first non-dominated frontier is less than or equal to the size of the elitist list, the new elitist list will contain all solutions of the first non-dominated frontier. Otherwise, tournament selection for Pareto domination [37] is exercised. Two solutions from the first non-dominated solutions are randomly selected and then the solution with larger crowding distance measure and not being selected before is added to the new elitist list. This approach not only ensures that all solutions in the elitist list are non-dominated solutions but also promoting diversity of the solutions. According to our example, the first-ranked elitist list from the first generation is L_1={7,6,2,3,1,4,8,9,5,10} and L_5={3,1,7,6,8,5,4,9,10,2}. From Table 11, the task order list of the population size is L_{11}={7,6,2,3,1,4,8,9,5,10}, L_{12}={8,2,7,5,10,4,3,6,1,9}, L_{13}={8,4,7,2,5,6,3,9,1,10}, L_{14}={3,5,7,4,2,10,1,6,8,9}, L_{15}={4,3,5,7,1,10,6,9,2,8}, L_{16}={5,4,3,2,10,7,6,9,1,8}, L_{17}={8,3,2,1,5,4,7,10,6,9}, L_{18}={6,5,4,2,9,7,8,10,3,1}, L_{19}={7,5,8,3,6,10,9,1,2,4}, and L_{20}={2,5,3,1,8,6,9,10,4,7}. The solutions in the current first non-dominated frontier is L_{11}={7,6,2,3,1,4,8,9,5,10}, L_{13}={8,4,7,2,5,6,3,9,1,10}, L_{15}={4,3,5,7,1,10,6,9,2,8} and L_{20}={2,5,3,1,8,6,9,10,4,7}. When the number of the combined solutions is less than the size of the elitist list, both solutions are added to the new elitist. Hence, five good solutions of current elitist list are L_1 or L_{11}={7,6,2,3,1,4,8,9,5,10}, L_5={3,1,7,6,8,5,4,9,10,2}, L_{13}={8,4,7,2,5,6,3,9,1,10}, L_{15}={4,3,5,7,1,10,6,9,2,8} and L_{20}={2,5,3,1,8,6,9,10,4,7}.

Table 11
Objective functions of each chromosome from the second generation.

Chromosome Number	Number of workers	DOW	WT	Pareto Frontier	Crowding Distance
L15	4	2.8338	1.8193	1	Infinite
L13	4	2.9090	1.7373	1	2.0000
L11*	4	2.9179	1.7197	1	Infinite
L20*	4	2.9179	1.7197	1	Infinite
L18	4	3.6878	1.9366	2	Infinite
L16**	4	4.0792	1.9593	3	Infinite
L17**	4	4.0792	1.9593	3	Infinite
L12***	4	4.0853	1.9593	4	Infinite
L14***	4	4.0853	1.9593	4	Infinite
L19***	4	4.0853	1.9593	4	Infinite

4.2.8 Worker allocation

Finally, the results of 10-task worker allocation of a chromosome L_1 or L_{11} in a single U-shaped assembly line are exemplified in Table 12 and Figure 6 and 7. Previously, its detailed calculation is clearly described in Table 7.

Table 12
Final exemplified results of Miltenburg's 10-task worker allocation problem for a chromosome L_1 or L_{11}.

Chromosome	Worker	Manual time	Travel distance time	Idle time	Allocated tasks (t_k)
1 or 11	1	7	0.14	2.72	7 (3), 6 (4)
	2	9	0.24	0.54	2 (4), 3 (2), 1 (3)
	3	6	0.24	3.54	4 (2), 8 (2), 9 (2)
	4	6	0.14	3.72	5 (4), 10 (2)

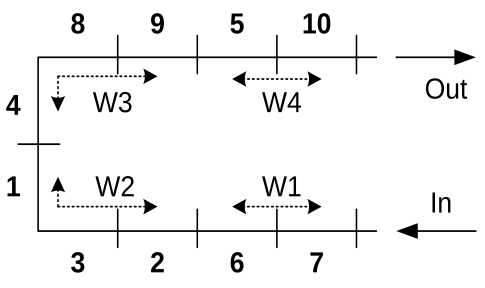


Figure 6
Final exemplified 10-task worker allocation results of a chromosome L_1 or L_{11} on a single U-line.

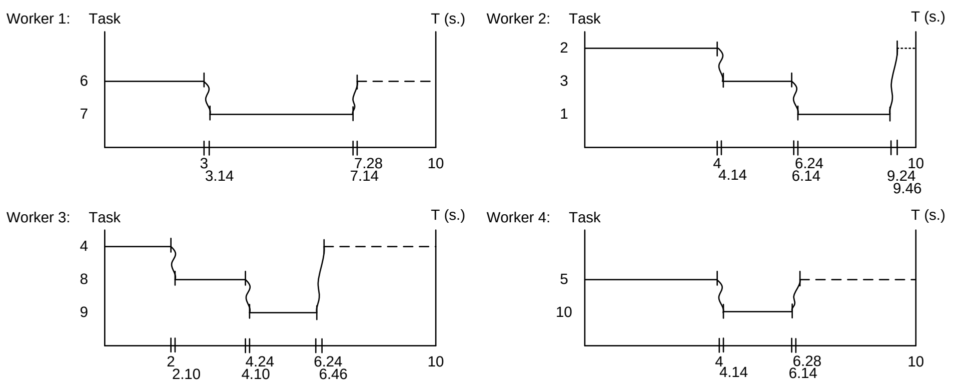


Figure 7
Work load routines, showing allocation of four workers: solid line = manual time; wavy line = walking time; dashed line = idle time.

V. Experimental Results

5.1 Multi-objective solutions

In order to demonstrate the effectiveness of the proposed approach, computational results with multiple objectives are obtained on a set of single U-shaped assembly line worker allocation problems, which have 7 to 297 tasks set. From the previous results of convergence to the Pareto-optimal set, spread and ratio of non-dominated solution, and CPU time, COIN seems to perform better than well-known NSGA-II for most problem sets [26]. The multi-objective solution in this study is solved with hierarchical procedure. First, the objective function of a number of workers is minimized. Secondly, only a minimum number of workers are selected to further evaluate a pair of minimum DOW and WT objective values as the Pareto-optimum frontier. In this paper, the multi-objective coincidence algorithm is encoded and programmed by using MATLAB R2008a on an AMD Athlon^{TM} 64 Processor 3500+ 2.21 GHz PC with 960 MB DDR-SDRAM., and the set of test problems are run at the 30 generations that are enough for attaining the major and minor aims in the paper.

5.2 A number of workers

In this section, a variety of proposed average processing time percentages is tested in every problem. The computational results of a number of workers at the symmetrical and rectangular layouts are shown in Table 13(a) and 13(b), respectively. A number of tasks that are representative in each problem are shown in the first column. The second column displays the summation of processing times. The cycle time is determined from test-bed problems [3],[7],[13],[26] in the third column. The theoretical number of workers in the fourth column is calculated with the second column divided by the third column. However, no paper displays the minimum number of workers for 7-task to 297-task U-shaped worker allocation problems with mathematical optimization techniques. Thus, the straight-line ULINO for the line balancing problem of type I [7] is benchmarked as the lower bound on quantity of workers in the fifth column. The values of a number of workers in various %APT display from column six to column twenty-one in every problem. The results show that the greater the walking time or %APT is, the larger the number of workers are. An example is shown in Figure 7.

From the experimental results of symmetrical and rectangular U-shaped layouts in Table 14, increasing only one worker in the first objective is sensitive to determining the walking time at only five percent of average processing time (or 0.14 to 65.61 seconds) in most problems. The 11-task problem is at the ten %APT (or 0.42 seconds.); the 19-task problem is at the 20 %APT (or 4.08 s.); and the 45-task problem is at the 15 %APT (or 1.84 seconds). It gives a conclusion that a decision to change a little walking time significantly effects the supplement of a larger number of workers in a single U-line. For every problem, the fixed average process time percentage shown in the second and third columns is representative to initial walking time that makes a number of workers different from the straight-line layout that is not taking walking time into account. The differences of a number of workers between both layouts for all problems are shown in the fourth column. A number of workers at the symmetrical U-line are the same as a number of workers at the rectangular U-line throughout a range of percentage for 7-task and 10-task problems due to their identical layouts at 1:3:3 and 2:4:4, respectively. From 11-task to 297-task problems, the first different values of number of workers between symmetrical and rectangular layouts are shown as each of %APT. However, from Table 13(a) and 13(b) they are different only one worker and for the rest of the first %APT of every problem a number of workers are mostly the same between symmetrical and rectangular U-line.

For an illustrative example, the experiment of 19-task problem in Figure 8 shows that walking time should be taken into account at the beginning of 20% average processing time (4.08 s.). After that, the difference of number of workers between the symmetrical and rectangular layouts is cut off at the distinguished line at 60 %APT in the same Figure.

Table 13
13(a) Theoretical, straight-line and U-line number of workers at the symmetrical layout.
13(b) Theoretical, straight-line and U-line number of workers at the rectangular layout.

(1) No. of tasks	(2) Sum. Of processing times (s.)	(3) Cycle time (s.)	(4) Number of workers at the symmetrical layout																	
			Theory (2) / (3)	Straight Line ULINO	U-shaped line plus walking															
					5%	10%	15%	20%	25%	30%	35%	40%	50%	60%	70%	80%	90%	100%	110%	120%
7	29	10	2.9	3	4	4	4	5	5	5	5	5	6	6	6	7	7	7	7	7
10	28	10	2.8	3	4	4	4	4	4	4	5	5	5	5	5	6	6	6	8	8
11	46	13	3.54	4	4	5	5	5	5	6	6	6	6	8	8	8	9	9	9	9
19	388	120	3.23	4*	4	4	4	5	5	5	5	6	6	7	7	7	8	8	9	9
28	1,024	256	4.00	4	5	5	5	6	6	6	7	7	8	8	9	9	10	10	11	11
45	552	110	5.02	6	6	6	7	7	8	8	8	9	10	10	11	12	13	14	14	15
61	5,274	600	8.79	9*	10	11	12	13	13	14	15	15	17	18	20	20	22	23	25	27
70	3,510	251	13.98	14	17	18	19	20	22	22	23	25	26	28	31	33	34	36	38	39
111	145,657	7,916	18.4	19	22	24	25	26	29	30	30	30	40	40	40	40	50	50	50	50
297	69,655	1,834	37.98	38	44	49	52	55	59	62	65	69	74	81	87	93	100	106	109	115

*Wattanapornprom et al., 2009 [36]

(1) No. of tasks	(2) Sum. Of processing times (s.)	(3) Cycle time (s.)	(4) Number of workers at the rectangular layout																	
			Theory (2) / (3)	Straight Line ULINO	U-shaped line plus walking															
					5%	10%	15%	20%	25%	30%	35%	40%	50%	60%	70%	80%	90%	100%	110%	120%
7	29	10	2.9	3	4	4	4	5	5	5	5	5	6	6	6	7	7	7	7	7
10	28	10	2.8	3	4	4	4	4	4	4	5	5	5	5	5	6	6	6	8	8
11	46	13	3.54	4	4	5	5	5	5	5	6	6	6	8	8	8	9	9	9	10
19	388	120	3.23	4*	4	4	4	5	5	5	5	6	6	6	7	7	8	8	9	9
28	1,024	256	4.00	4	5	5	6	6	6	6	7	7	8	8	9	9	10	10	11	11
45	552	110	5.02	6	6	6	7	7	8	8	8	9	10	11	11	12	13	14	15	15
61	5,274	600	8.79	9*	10	11	12	12	13	14	15	15	17	18	19	20	22	23	25	27
70	3,510	251	13.98	14	16	18	19	20	21	22	23	25	27	28	30	32	34	36	38	40
111	145,657	7,916	18.4	19	22	24	25	27	28	30	31	33	36	39	41	44	47	50	52	54
297	69,655	1,834	37.98	38	44	48	52	55	58	62	65	69	75	81	87	93	100	106	109	114

*Wattanapornprom et al., 2009 [36]

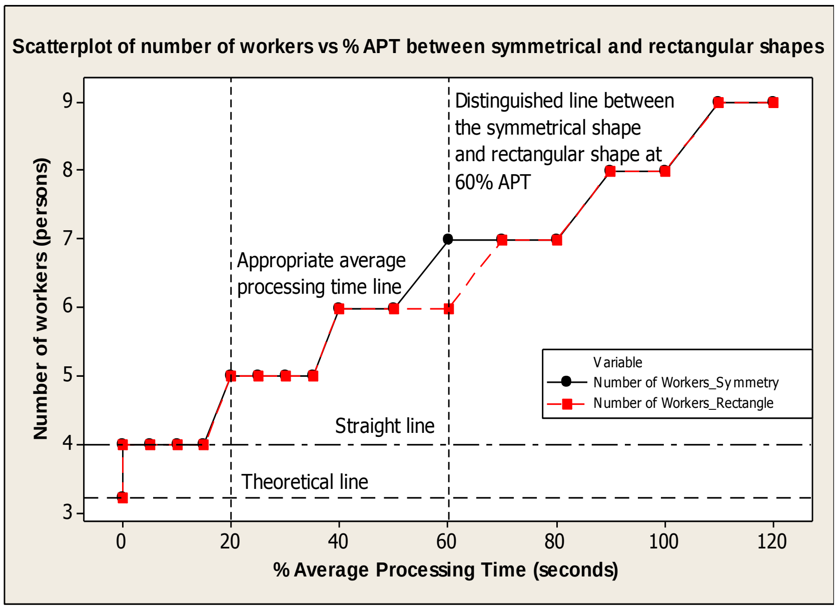


Figure 8
Appropriate average processing time line and distinguished line between the symmetrical and rectangular layouts for the 19-task problem.

Problem / Number of tasks	% Average processing time		
	Symmetry	Rectangle	Difference
1. Menten / 7	5% = 0.21 s.	5% = 0.21 s.	-
2. Miltenburg / 10	5% = 0.14 s.	5% = 0.14 s.	-
3. Jackson / 11	10% = 0.42 s.	10% = 0.42 s.	30% = 1.25 s.
4. Thomopoulos / 19	20% = 4.08 s.	20% = 4.08 s.	60% = 12.25 s.
5. Heskiaoff / 28	5% = 1.83 s.	5% = 1.83 s.	15% = 5.49 s.
6. Kilbridge & Wester / 45	15% = 1.84 s.	15% = 1.84 s.	60% = 7.36 s.
7. Kim / 61	5% = 4.33 s.	5% = 4.33 s.	20% = 17.30 s.
8. Tongue / 70	5% = 2.51 s.	5% = 2.51 s.	5% = 2.51 s.
9. Arcus / 111	5% = 65.51 s.	5% = 65.61 s.	20% = 262.45 s.
10. Scholl & Klein / 297	5% = 1 1.73 s.	5% = 11.73 s.	10% = 23.45 s.

Table 14
Fixed and different average processing time percentage for the 7-task to 297-task problems.

5.3 Pareto-optimal frontier

Similar to the 10-task worker allocation example in the section 4.2.8, this section needs to discover the results of DOW and WT good frontier on the first rank after running thirty generations. The Pareto-optimum frontier between DOW and WT of Arcus' 111-task problem at the 5% APT is exemplified and shown in Figure 9. Nine points (or nine chromosomes) give the best solutions indifferently. After that, each of them is essential to assign tasks to all workers. There are a lot of task assignments for many worker allocation problems. Repeatedly, the steps for assigning task are the same as described in Section 4.2.8.

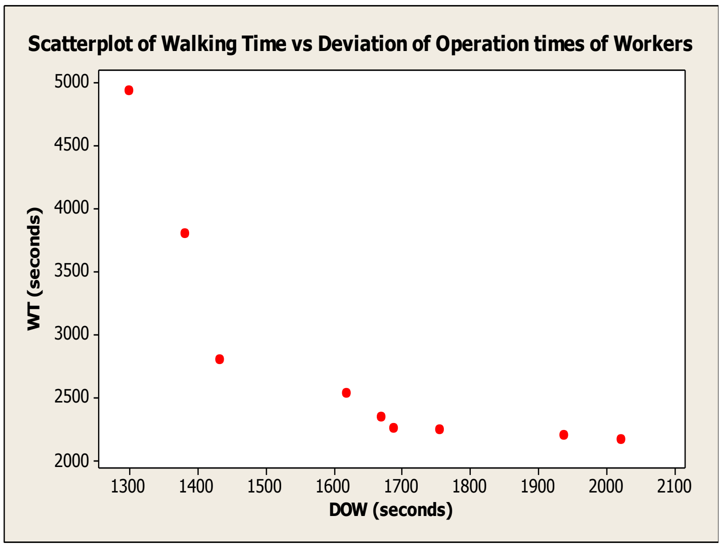


Figure 9
Pareto-optimum frontier of DOW and WT: an exemplified Arcus' 111-task problem at the 5% APT.

VI. Conclusions and future work

This paper presents the impact of walking time on the SUALWAPs of type I in a one-piece flow production environment under the 7-task to 297-task problems. Combinatorial optimization with multi-objective coincidence algorithm (COIN) as a novel evolutionary algorithm recognizes the positive knowledge appearing in the order pairs of the good solution by giving a marginal reward (increased probability) to its related element of the joint probability matrix. In contrast, the negative knowledge found in the order pairs of the bad solution, which is often remiss in most algorithms, is also utilized in COIN (reduced probability) to prevent undesirable solutions coincidentally found in this generation to be less recurrent in the next generation. The performances of COIN and its variances are solved with hierarchical procedure. First, the objective function of a number of workers is minimized. Secondly, only a minimum filtered number of workers are brought to discover the minimization of DOW and WT objective functions as the Pareto-optimum frontier. From the experimental results of symmetrical and rectangular U-shaped layouts, incrementing a number of workers in the former objective is sensitive to determining the walking time at only five percent of average processing time (or 0.14 to 65.61 seconds) in most problems. Just a few problems are at the ten and twenty percentage of APT (or 0.42 to 4.08 seconds). It gives the conclusion that a decision to change a little walking time significantly effects the supplement of a larger number of workers in a single U-line. After getting the fixed %APT from one task to another task of all problems, every worker is assigned to do task(s) by the consideration of the latter dual objectives simultaneously. In the future, the authors are going to conduct two research topics. First, the problem of U-line will be developed in other models such as the shared-center U-line. Secondly, the negative knowledge of the coincidence algorithm will be extended into other renowned evolutionary techniques such as the particle swarm optimization

.

Acknowledgements

The authors would like to thank to Professor Prabhas Chongstitvatana and Warin Wattanapornprom for sharing their idea about COIN, and also Charat Jirakomate for his help in verification and validation of the programs. We are grateful to Graham Keith Rogers for his editing suggestions. We would also express our gratitude to anonymous reviewers for their comments and proofreading.

REFERENCES

[1] C. H. Cheng, J. Miltenburg, and J. Motwani, "The effect of straight- and U-shaped lines on quality," *IEEE Transactions on Engineering Management*, vol. 47, no. 3, pp. 321-334, 2000.
[2] Y. Monden, *Toyota Production System*. Norcross, Georgia: Engineering and Management Press, 1993.
[3] J. Miltenburg, "One-piece flow manufacturing on U-shaped production lines: a tutorial," *IIE Transactions*, vol. 33, no. 4, pp. 303-321, 2001.
[4] D. A. Ajenblit and R. L. Wainwright, "Applying genetic algorithms to the U-shaped assembly line balancing problem," *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, 1998.
[5] D. Sparling and J. Miltenburg, "The mixed-model U-line balancing problem," *International Journal of Production Research*, vol. 36, no. 2, pp. 485-501, 1998.
[6] T. L. Urban, "Note. Optimal Balancing of U-shaped Assembly Lines," *Management Science*, vol. 44, no. 5, pp. 738-741, 1998.
[7] A. Scholl and R. Klein, "ULINO: Optimally balancing U-shaped JIT assembly lines," *International Journal of Production Research*, vol. 37, no. 4, pp. 721-736, 1999.
[8] R. K. Hwang, H. Katayama, and M. Gen, "U-shaped assembly line balancing problem with genetic algorithm," *International Journal of Production Research*, vol. 46, no. 16, pp. 4637-4649, 2008.
[9] C. Becker and A. Scholl, "A survey on problems and methods in generalized assembly line balancing," *European Journal of Operational Research*, vol. 168, no. 3, pp. 694-715, 2006.
[10] G. J. Miltenburg and J. Wijngaard, "The U-line balancing problem," *Management Science*, vol. 40, no. 10, pp. 1378-1388, 1994.
[11] K. Ohno and K. Nakade, "Analysis and optimization of a U-shaped production line," *Journal of the Operations Research Society of Japan*, vol. 40, no. 10, pp. 90-104, 1997.
[12] G. R. Aase, J. R. Olson, and M. J. Chniederjans, "U-shaped assembly line layouts and their impact on labour productivity: an experimental study," *European Journal of Operational Research*, vol. 156, no. 3, pp. 698-711, 2004.
[13] R. K. Hwang and H. Katayama, "A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems," *International Journal of Production Research*, vol. 47, no. 14, pp. 3797-3822, 2009.
[14] J. Miltenburg, "Balancing U-lines in a multiple U-line facility," *European Journal of Operational Research*, vol. 109, no. 1, pp. 1-23, 1998.
[15] D. Sparling and J. Miltenburg, "The mixed-model U-line balancing problem," *International Journal of Production Research*, vol. 36, no. 2, pp. 485-501, 1998.
[16] J. Miltenburg, "U-shaped production lines: A review of theory and practice," *International Journal of Production Economics*, vol. 70, no. 3, pp. 201-214, 2001.
[17] J. P. Shewchuk, "Worker allocation in lean U-shaped production lines," *International Journal of Production Research*, vol. 46, no. 13, pp. 3485-3502, 2008.
[18] F. B. Talbot, J. H. Patterson, and W. V. Gehrlein, "A comparative evaluation of heuristic line balancing techniques," *Management Science*, vol. 32, no. 4, pp. 430-454, 1986.
[19] A. Scholl, *Balancing and Sequencing of Assembly Lines*. 2nd ed. Germany: Physica-Verlag Heidelberg Company, 1999.
[20] C. Miralles, J. P. Garcia-Sabater, C. Andres, and M. Cardos, "Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled," *Discrete Applied Mathematics* vol. 156, no. 3, pp. 352-367, 2008.
[21] S. Vembu and G. Srinivasan, " Heuristics for operator allocation and sequencing in product-line-cells with manually operated machines," *Computers & Industrial Engineering*, vol. 32, no. 2, pp. 265-279,1997.
[22] K. Nakade and K. Ohno, "An optimal worker allocation problem for a U-shaped production line,"*International Journal of Production Economics*, vol. 60-61, no. 1, pp. 353-358, 1999.
[23] K. Nakade and K. Ohno, "Separate and carousel type allocations of workers in a U-shaped production line," *European Journal of Operational Research*, vol. 145, no. 2, pp. 403-424, 2003.
[24] T. Ertay and D. Ruan, "Data envelopment analysis based decision model for optimal operator allocation in CMS," *European Journal of Operational Research*, vol. 164, no. 3, pp. 800-810, 2005.
[25] J. Miltenburg and D. Sparling, "Optimal solution algorithms for the U-line balancing problem," *Working paper*, McMaster University, Hamilton, 1995.
[26] R. Sirovetnukul and P. Chutima, "Worker allocation in U-shaped assembly lines with multiple objectives," *Proceedings of the 2009 IEEE International Conference on Industrial Engineering and Engineering Management*, Hong Kong, pp. 105-109, 2009.
[27] K. Nakade and R. Nishiwaki, "Optimal allocation of heterogeneous workers in a U-shaped production line," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 432-440, 2008.
[28] P. Ghinato, S. Fujii, and H. Morita, "A basic approach to the multifunction workers assignment problem in U-shaped production lines," *Proceedings of the 3rd International Congress of Industrial Engineering, Gramado, Brazil, CD-ROM*, vol. October, 1997.
[29] Y. Kuo and T. Yang, "Optimization of mixed-skill multi-line operator allocation problem," *Computers and Industrial Engineering*, vol. 53, no. 3, pp. 386-393, 2007.
[30] V. R. Kannan and J. B. Jensen, "Learning and labor assignment in a dual resource constrained cellular shop," *International Journal of Production Research*, vol. 42, no. 7, pp. 1455-1470, 2004.
[31] P. Chutima and P. Pinkoompee, "An investigation of local searches in Memetic Algorithms for multi-objective sequencing problems on mixed-model assembly lines," *Proceedings of Computers and Industrial Engineering*, Beijing, China, 2008.
[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[33] R. Sirovetnukul, P. Chutima, and D. Kritchanchai, "Analysis Guidelines for Customised Orders in an Apparel Chain," *Thammasat International Journal of Science and Technology*, vol. 12, no. 1, pp. 10-19, 2007.

[34] J. Balakrishnan, C. H. Cheng, K. C. Ho, and K. K. Yang, "The application of single-pass heuristics for U-lines," *Journal of Manufacturing Systems*, vol. 28, no. 1, pp. 28-40, 2009.

[35] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multi-objective Combinatorial Optimisation with Coincidence Algorithm," *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pp. 1675-1682, 2009.

[36] V. Robles, P.D. Miguel, and P. Larrañaga, "Solving the Traveling Salesman Problem with EDAs," *In Estimation of Distribution Algorithm: A New Tool for Evolutionary Computation*, 2002.

[37] J. Horn, N. Nafpliotis, and D.E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1, pp. 82-87, 1994.