แบบเสนอขอรับค่าตอบแทนในการตีพิมพ์วารสารวิชาการ

1. เอกสารประกอบการเสนอขอรับค่าตอบแทนในการตีพิมพ์วารสารวิชาการ

1. 1 แบบขอรับค่าตอบแทน

1. 2 หนังสือขออนุมัติค่าตอบแทน เรียน รองคณบดีฝ่ายวิจัยและบริการวิชาการผ่านหัวหน้าภาควิชา

1.3 สำเนาบทความวิจัยที่ได้รับการตีพิมพ์ในวารสารวิชาการ

1.4 รายละเอียดวารสาร

1.5 เอกสารแสดงค่า Impact factor ของวารสารที่ตีพิมพ์

2. รายละเอียดข้อมูลประกอบเสนอขอรับค่าตอบแทนในการตีพิมพ์วารสารวิชาการ

2.1 ผู้เสนอขอรับค่าตอบแทน ชื่อ-สกุล **ระพีพันธ์ ปิตาคะโส**

2.2 ชื่อบทความวิจัย (ภาษาไทย) .....

.....

1. **(ภาษาอังกฤษ)** Differential evolution algorithm for simple assembly line balancing type 1 (SALBP-1).....

2.3 รายละเอียดของวารสาร

ชื่อวารสาร **Journal of Industrial and Production Engineering**.....

อยู่ในฐานข้อมูล ○ ISI กรณี ISI ช่วยระบุ impact factor:..... ☑ SCOPUS ○ SJR.....

ปีที่ 32 ฉบับที่ 2 เดือน March ปี 2015 หน้า 104-114

2.4 สถานะในบทความวิจัยเป็น

☑ ชื่อแรก (first author) ☑ ผู้รับผิดชอบบทความ (corresponding author)

○ ผู้มีส่วนร่วมในบทความ

2.5 การมีส่วนร่วมในบทความของนักศึกษา

○ ใช้ขอจบการศึกษา ☑ ไม่ใช้ขอจบการศึกษา

การรับรองสัดส่วนผลงานทางวิชาการ กรุณากรอกข้อมูลตามแบบฟอร์มนี้ตามความเป็นจริง และรักษาไว้ซึ่ง **จรรยาบรรณ** และขอรับรองว่า บทความนี้ไม่เป็นส่วนหนึ่งของวิทยานิพนธ์ของผู้เสนอขอรับค่าตอบแทน

ลำดับที่	ชื่อ-สกุล	สัดส่วนผลงานทางวิชาการ (%)	ลงนามรับรองข้อมูล
1	Rapeepan Pitakaso	100%	
2			
3			
4			
5			

หมายเหตุ: กรณีผู้เสนอขอรับค่าตอบแทนเป็นชื่อแรก หรือ ผู้รับผิดชอบบทความสามารถรับรองแทนผู้เขียนร่วมได้

.....

ผู้เสนอขอรับค่าตอบแทน

The Scopus Author Identifier assigns a unique number to groups of documents written by the same author via an algorithm that matches authorship based on a certain criteria. If a document cannot be confidently matched with an author identifier, it is grouped separately. In this case, you may see more than 1 entry for the same author.

Back to results | **1 of 1** Print | E-mail

Pitakaso, Rapeepan

About Scopus Author Identifier | View potential author matches

Ubon Rajathanee University, Department of Industrial Engineering, , Thailand

Author ID: 14831530600

E-mail: enrapepi@ubu.ac.th

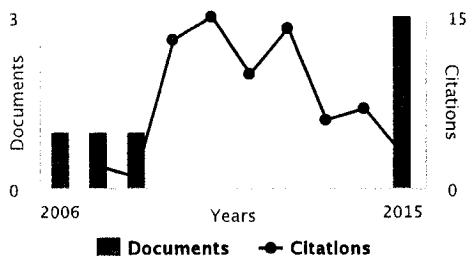
Documents: 6 Analyze author output

Citations: 71 total citations by 67 documents View citation overview

***h*-index: 3** View *h*-graph

Co-authors: 9

Subject area: Engineering , Decision Sciences View More



6 Documents | Cited by 67 documents | 9 co-authors

6 documents View in search results format Sort on: **Date** Cited by

Export all | Add all to my list | Set document alert | Set document feed

Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types	Pitakaso, R., Sethanan, K.	2015	Engineering Optimization Article in Press	0
View at Publisher				
A differential evolution algorithm for the capacitated VRP with flexibility of mixing pickup and delivery services and the maximum duration of a route in poultry industry	Dechampai, D., Tanwanichkul, L., Sethanan, K., Pitakaso, R.	2015	Journal of Intelligent Manufacturing Article in Press	0
View at Publisher				
Differential evolution algorithm for simple assembly line balancing type 1 (SALBP-1)	Pitakaso, R.	2015	Journal of Industrial and Production Engineering	0
View at Publisher				
The development of multi-objective optimization model for excess bagasse utilization: A case study for Thailand	Buddadee, B., Wirojanagud, W., Watts, D.J., Pitakaso, R.	2008	Environmental Impact Assessment Review	13
Full Text View at Publisher				

Author History

Publication range: 2006 - Present

References: 104

Source history:

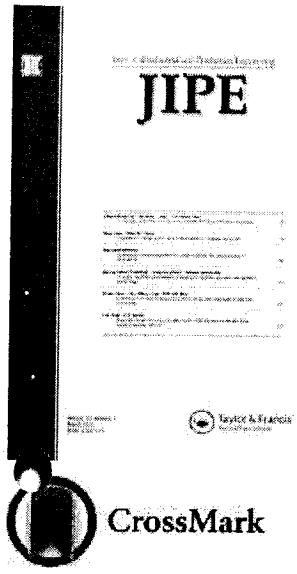
Journal of Industrial and Production Engineering View documents

Journal of Intelligent Manufacturing View documents

Engineering Optimization View documents

View More

Show Related Affiliations



Journal of Industrial and Production Engineering
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tjci21

Differential evolution algorithm for simple assembly line balancing type 1 (SALBP-1)

Rapeepan Pitakaso[a]
[a] Department of Industrial Engineering, Ubon Ratchathani University, 185 Sathonlamark Road, Warin Chamrap, Ubon Ratchathani, 34190, Thailand
Published online: 26 Mar 2015.

Journal of Industrial and Production Engineering, 2015
Vol. 32, No. 2, 104–114, http://dx.doi.org/10.1080/21681015.2015.1007094

Taylor & Francis
Taylor & Francis Group

Differential evolution algorithm for simple assembly line balancing type 1 (SALBP-1)

Rapeepan Pitakaso*

Department of Industrial Engineering, Ubon Ratchathani University, 185 Sathonlamark Road, Warin Chamrap, Ubon Ratchathani, 34190, Thailand

(*Received August 2013; revised May 2014; accepted January 2015*)

This article proposes a differential evolution algorithm (DE) for solving type 1 simple assembly line balancing problem (SALBP-1). The proposed heuristic composes of four main steps: (1) initialization, (2) mutation, (3) recombination, and (4) selection process. A new decoding scheme is proposed along with new recombination formulas besides those found in literatures. The computational results based on many tests using set of standard instances show that the proposed DE algorithm is very competitive for solving SALPB-1.

Keywords: simple assembly line balancing; differential evolution algorithm; binomial recombination; exponential recombination

1. Introduction

A typical assembly line balancing problem consists of series of jobs or tasks that must be performed on a production line. A task must be assigned to one workstation. The assignment of tasks is carried out to optimize some predefined objective function with predecessor constraints, cycle time constraints, etc. The objective functions found in literatures include minimal number of workstations, minimal cycle time, maximal assembly line efficiency, etc. (See, e.g. Kilincci [14], Peeters and Degrave [26] and Bowman [4] or simultaneously such as Scholl [31], Baybars [1], Nearchou [21] and Nourmohammadi and Zandieh [24].

Simple assembly line balancing problems (SALBPs) are normally represented in the form of precedence graph that is consisted of nodes and edges. Each node in the graph represents a specific task, while an edge connecting two nodes represents the precedence relation between the corresponding tasks. An example of a precedence graph for an eight-tasks ALB is given in Figure 1. The number inside each node represents task label while the number outside the node represent processing time for the corresponding task. In Figure 1, tasks 1, 2, 3, 4, 5, 6, 7, and 8 has processing time 5, 7, 4, 4, 9, 3, 8, and 3 time units, respectively. A precedence constraint is represented as edge to restrict production sequence of the tasks in the precedence graph. For instances, task 2 and task 3 can proceed only when task 1 has already finished.

The SALBP can be categorized into two types. The first type, SALBP-1, attempts to minimize number of stations for a given cycle time while the second type, SALBP-2, tries to minimize cycle time of a workstation for a given number of workstations. This article focuses mainly on solving SALBP-1, because it is widely used in textile industry to reduce number of workstations/machines/workers. These reductions lead to decrease production cost and improve the cost competitiveness of the plant.

SALBP is one of the NP-hard problems (Baybars [1]) that had been intensively studied by many researchers in the field of operational research and production planning (Scholl and Becher [32], Erel and Sarin [8], Becher and Scholl [3] and Baybars [1,2], Vilà and Pereira [39]). In the early years, the researchers mainly solve SALBP by trial-and-error method until Salveson [30] formulated mathematical model for the problems. Since then, various exact and heuristic methods such as linear programming, integer programming, dynamic programming, and branch-and-bound methods have been proposed.

Because the exact methods can only solve small size problems, many researchers turn their interest to develop heuristic method for solving practical size industrial problems. Iterative backtracking method (IBM) is a heuristic method for solving SALBP-1 that is simple and effective. The two main steps of IBM are: (1) form a feasible solution and (2) improve the solution by applying some backtracking methods such as branch and bound method, enumerative heuristics and linear programming to the problems. Notable IBM algorithms include FABLE by Johnson [12], OptPack by Nourie and Venta [23], EUREKA by Hoffmann [11], SALOME by Scholl and Klein [33], AGSA by Sprecher [35], Peeters and Degrave [26], Kilincci [14] and Fleszar and Hindi [9].

Recently, metaheuristic received huge attention from many researchers. For instances, Tabu search by Scholl and Voß [34], Chiang [6], and Lapierre et al. [16]; genetic algorithm by Kim et al. [15], Rubinovitz and

*Email: touch@rocketmail.com

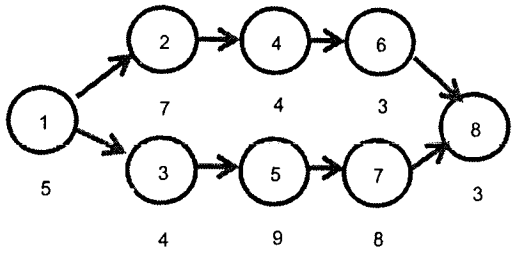


Figure 1. An example of precedence diagram of ALB.

Levitin [28], Bautista et al. [5] and Sabuncuoglu et al. [29]; particle swam optimization by Hamtaa et al. [10]; simulated annealing by Khorasanian et al. [13]; and differential evolution (DE) algorithm by Nearchou [18,20,21] and Nourmohammadi and Zandieh [24].

DE was proposed by Storn and Price [36] to solve continuous optimization problems. In recent years, DE has been applied to many combinatorial optimization problems such as job shop and flow shop scheduling problems (Wisittipanich and Kachitvichyanukul [41,42], Tasgetirena et al. [37], Onwubolu and Davendra [25] and Li and Yin [17], Nearchou [19]), scheduling and resource allocation problem (Tsai et al. [38]), joint replenishment and delivery problem (Wang et al. [40]), and multimode resource constrained project scheduling problem (Nguyen and Kachitvichyanukul [22]). These successful applications of DE to various NP-Hard problems are the main motivation to develop DE-based algorithm for solving SALBP-1.

A multiobjective simple assembly line balancing type 2 (SALBP-2) has been solved using DE by Nearchou [21] and it shows that DE outperforms genetic algorithm proposed by Kim et al. [15]. Nourmohammadi and Zandieh [24] also uses DE to solve the same problem as Nearchou [21] but by using different evaluation schemes than that of Nearchou [21]. Nourmohammadi and Zandieh [24] uses Pareto dominance concept and creates new evaluation scheme based on TOPSIS algorithm while Nearchou [21] uses a weight-sum of multiple objective as tool to evaluate solution generated from the algorithm.

Although, DE has been successfully applied to solve multiobjective SALBP-2 as proposed in Nourmohammadi and Zandieh [24], the only DE-based algorithm for SALBP-1 was proposed by Nearchou [18] and tested with 64 benchmark test problem instances found in the literature. Nearchou [18] constructs a solution by forming an order of items that will be assigned to workstations and then assigned them to a station according to that order. However, the ordering of items does not consider the precedence diagram so infeasible solutions may be easily formed when tasks are assigned to stations. An infeasible solution of SALBP-1 occurs when a task that is a successor of another task is assigned to an earlier workstation that a predecessor task being assigned. If the infeasible solution is formed, Nearchou [18] uses repair strategy to turn an infeasible solution into a feasible solution. The algorithm can find optimal solution in 61 out of 64 test problem instances. In this article, precedence graph will be considered in parallel, while an order of items is constructed so no infeasible solution will be formed. Detail of the decoding scheme proposed by Nearchou [18] and the proposed decoding scheme will be explained in Section 3.1.

The DE algorithm developed in this article modifies the decoding scheme from Nearchou [18] with several recombination formulas. Two recombination formulas from Qin et al. [27] are included along with one new recombination formula. Therefore, in this article, three recombination formulas are combined with five different mutation formulas taken from Qin et al. [27]. The most effective combination of different mutation and recombination formulas for DE to solve SALBP-1 is identified based on experiments.

This article is organized as following: Section 2 presents mathematical formulation of the SALBP-1, Section 3 presents the proposed heuristic (DE), Section 4 presents the computational result of the test instances, and Section 5 is the conclusion of the whole article.

2. Formulation of the problem

This section presents the mathematical model for SALBP-1 adapted from Bowman [4]. The indices, parameters, and decision variables as defined below.

Indices

n denotes index of task n while $n = 1 \ldots N$
k denotes index of work station k while $k = 1 \ldots M$
N denotes total number of task
M denotes total number of work station

Parameters

P_n denotes processing time of task n
CT denotes cycle time of work station
P_{nj} denotes relationship of task n to task j
F_{nj} $\begin{cases} 1 & \text{if task } n \text{ is predecessor of task } j \\ 0 & \text{otherwise} \end{cases}$

Decision variables

X_{nk} $\begin{cases} 1 & \text{if task } n \text{ is assigned to station } k \\ 0 & \text{otherwise} \end{cases}$

Y_k $\begin{cases} 1 & \text{station } k \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$

Objective function

$$\text{Min } z = \sum_{k=1}^{M} Y_k \qquad (1)$$

Subject to

$$\sum_{k=1}^{M} X_{nk} = 1 \quad \forall\, n = 1 \ldots N, \tag{2}$$

$$\sum_{k=1}^{M} (K \times X_{jk}) - (K \times X_{nk}) \geq 0 \quad \forall\, n = 1 \ldots N,\ j = 1 \ldots M, F_{nj} = 1 \tag{3}$$

$$\sum_{n=1}^{N} X_{nk} \times P_n \leq CT \times Y_k \quad \forall\, k = 1 \ldots M \tag{4}$$

$$Y_k \leq Y_{k-1} \ \forall\, k = 2 \ldots M \tag{5}$$

Formula (1) represents an objective function of the model to minimize number of stations. Formula (2) guarantees that task n must be assigned to exactly one workstation. Formula (3) enforces the precedence constraints. For example, if task n is predecessor of task j ($F_{nj} = 1$) task j cannot be assigned to a workstation before task n. Formula (4) ensures that total processing time used by all tasks assigned to a particular workstation must not exceed a prespecified cycle time (CT). Finally, formula (5) ensures that the station will be opened successively according to station number.

3. Proposed heuristics

General procedure of DE consists of several steps which are (1) construct a set of initial target vector, (2) perform mutation process, (3) perform recombination process, and (4) perform selection process to obtain a new generation of solutions and then step (2)–(4) will be performed repeatedly until a termination condition is met. Each step is explained as follow.

3.1. *Construct a set of initial target vector*

Initially, a population of NP target vectors must be generated. Each vector composes of D components (positions), while D equals to number of tasks that will be assigned to a certain number of workstations. For example, in Figure 1, the number of tasks shown is eight tasks; thus, a particular target vector will have dimension of 8 as shown in Figure 2.

Figure 2 shows a set of target vector which composes of 4 vectors (NP), and each vector has two rows, the first row denotes number of task while the second row contains values of randomly generated real numbers. Each vector represents one solution of SALBP-1. A new decoding scheme to transform vector to SALBP-1 solution is presented here named priority decoding method. It is modified from the sub-range decoding scheme proposed by in Nearchou [18] and it will be explained in Section 3.1.1. The new decoding scheme proposed will be explained in Section 3.1.2.

3.1.1. *Sub-range decoding scheme*

The sub-range decoding method introduced by Nearchou [18] has the following steps:

(1) For SALBP-1 with n tasks, a range is also applied as n sub-range. Each sub-range has the same range which is $1/n$. Sub-range array (SR) = [$1/n$, $2/n$, $3/n$, ... , n/n] as in Figure 1 showing precedent graph that has $n = 8$, consequently, the range is divided into eight sub-ranges as SR = 0.125,0.25,0.375,0.5,0.625,0.75,0.875,1.00].

(2) A sample, vector with value [0.92,0.02,0.78,0.38,0.69,0.30,0.59,0.13] can be decoded as follow. The value in position 1 is 0.92 which lie in sub-range 8 ($0.875 < 0.92 < 1.00$), so the 8th task will be aligned in the 1st order to be assigned to workstation. Do the same in the 2nd order until all works are aligned in the order. This order will be named *order p*. For examples, position value of position 2 is 0.02, which lie in 1st sub-range ($0.0 < 0.02 < 0.125$). As a result, order p will be [8,1,_,_,_,_,_,_]. When the process is in the 2nd order, order p will be [8, 1, 7, 4, 6, 3, 5, 2].

(3) Check to see if the task assignment is conflicting with precedence constraints. If so, changing of positions in the order to make them aligned in the correct order is necessary. For examples, task 8 cannot be in a position that is prior to task 1, the changing of these position is required making p = [1, 8, 7, 4, 6, 3, 5, 2]. Continue to switch the tasks until the sequence does not conflict with precedence constraints. The result is p = [1, 3, 2, 4, 5, 7, 6, 8].

1	2	3	4	5	6	7	8
0.75	0.14	0.95	0.73	0.97	0.78	0.98	0.94

1	2	3	4	5	6	7	8
0.37	0.25	0.95	0.92	0.78	0.17	0.00	0.33

1	2	3	4	5	6	7	8
0.23	0.27	0.54	0.94	0.89	0.47	0.02	0.45

1	2	3	4	5	6	7	8
0.92	0.78	0.020	0.38	0.69	0.30	0.59	0.13

Figure 2. Examples of randomly generated target vectors.

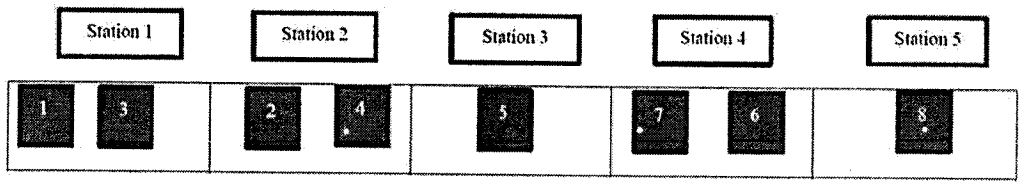


Figure 3. An outcome of task assignment according to Nearchou [18].

(4) Work will be assigned in accordance with p order. The first task to be assigned is task 1 to workstation 1. The conditions of workstation 1 include cycle time = 12. When task 1 is assigned to workstation 1, the remaining cycle time will be 7 units (12 − 5 = 7) hence task 3 is next assigned to workstation 1. Since workstation 1 will now have remaining time cycle as three units (7 − 4 = 3), task 2 cannot be further assigned to workstation 1 so workstation 2 will be opened for the next task assignment. The assignment can be shown in Figure 3.

3.1.2. *Priority decoding scheme*

Task assignment in priority decoding scheme is to assign tasks to workstations with consideration of precedence graph. If there are more than 1 task that can be assigned to a workstation, the selection is made using roulette wheel selection method must be done first. Given the same vector which is [0.92, 0.02, 0.78, 0.38, 0.69, 0.30, 0.59, 0.13], these are values of coordinate or task position of [1, 2, 3, 4, 5, 6, 7, 8]. From Figure 1, the only task ready for assignment is task 1 and as a consequence, task 1 will be assigned to workstation 1 with the remaining cycle time of 7 units (12 − 5 = 7). Based on the precedence diagram, the assignable tasks are task 2 and task 3 with their coordinate values as 0.02 and 0.78, respectively. Since there are more than 1 task to be assigned, the process can be done as followed.

(1) Calculate the possibility of task that will be assigned to workstations by utilizing coordinate to calculate. Task 2 and task 3 have coordinate values as 0.02 and 0.78, respectively. Consequently, the sum of coordinate values is 0.80 (0.02 + 0.78 = 0.8). Therefore, the possibility of task 2 is 0.025 (0.02/0.80 = 0.025), and the possibility of task 3 is 0.975 (0.78/0.8 = 0.975).
(2) Calculate cumulative probability of such works. For example, task 2 and task 3 have their possibility as 0.025 and 1.00, respectively.
(3) The last step, a random number is generated. If the random number lies in cumulative probability range of which task, such task will be assigned to current work station. For example, if the randomly generated number is 0.015 and it lies in cumulative probability of task 2. Therefore, task 2 will be assigned to workstation 1 following to task 1.
(4) The process will be repeated from step (1)–step (3) until all works are assigned.

An example of task assignment can be shown in Figure 4.

3.2. *Perform mutation process of a target vector ($X_{i,j,G}$)*

In mutation process, a mutant vector ($V_{i,j,G}$) will be calculated from one or more selected target vector ($X_{i,j,G}$). Traditionally, the mutation process of DE is performed using formula (6). In the proposed algorithm, five classic mutation formulas drawn from Qin et al. [27] are applied as shown in formula (6)–(10).

$$V_{i,j,G} = X_{r1,j,G} + F(X_{r2,j,G} - X_{r3,j,G}) \quad (6)$$

$$V_{i,j,G} = X_{best,j,G} + F(X_{r1,j,G} - X_{r2,j,G}) \quad (7)$$

$$V_{i,j,G} = X_{i,j,G} + F(X_{best,j,G} - X_{i,j,G}) + F(X_{r1,j,G} - X_{r2,j,G}) \quad (8)$$

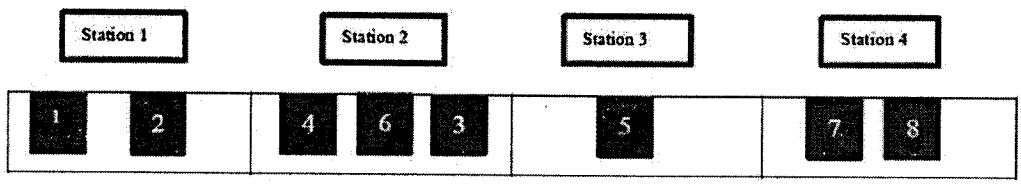


Figure 4. Result of transforming process of target vector 1.

$$V_{i,j,G} = X_{best,j,G} + F(X_{r1,j,G} - X_{r2,j,G}) + F(X_{r3,j,G} - X_{r4,j,G}) \quad (9)$$

$$V_{i,j,G} = X_{r1,j,G} + F(X_{r2,j,G} - X_{r3,j,G}) + F(X_{r4,j,G} - X_{r5,j,G}) \quad (10)$$

Let $r1$, $r2$, $r3$, $r4$, and $r5$ - denote the vectors which are randomly selected from a set of target vectors j. - represent the best vector found so far in the algorithm. F is a predefined integer parameter (scaling factor). In the proposed heuristics, F is set to 2 (Qin et al. [27]); i is vector number which starts from 1 to NP, and j is position of a vector which run from 1 to D.

3.3. *Perform recombination process on a mutant vector* $V_{i,j,G}$

The result of mutation process is a set of mutant vector $V_{i,j,G}$ (i run from 1 to NP). Then a mutant vector will apply recombination formula (11)–(13) to yield trial vector ($U_{i,j,G}$) as a product of recombination processes. In traditional DE for SALBP-1, a binomial recombination formula (formula (11)) is applied in the original version of DE proposed in Nearchou [18]. In the proposed algorithm, two recombination formulas are added with formula (12) drawn from Qin et al. [27], and a newly proposed recombination formula shown in formula (13). Formula 11 is a binomial recombination formula which is recombination formula that produces trail vectors from a selected target and mutant vector. Values in each position of a trail vector will equal to value in position of target or mutant vector depending on the sampling CR value. The sampling occurs in every position, therefore, every position of trial vectors have probabilities to copy value in position from target or mutant vector as shown in Figure 5(d). This example, CR is given to be *0.8* utilizing sampled numbers as shown in Figure 5(c). A selected target vector and a selected mutant vector are shown in Figure 5(a) and (b). Equation (12) will randomly select integer $randb_i$ which varies from 1 to D. The sampling will provide values for positions in trial vector that is transition position. Values for position in position 1 trial vector to transition position will copy value in positions from a selected mutant vector. The remaining positions will copy value in positions from a selected target vector as shown in Figure 5(e). Whereas Equation (13) will have 2 transition positions by sampling $randb_{i,1}$ and $randb_{i,2}$ integers which vary from 1 to D. Value in position of a trial vector will equal to a selected mutant vector from position 1 to position $randb_{i,1}$ while $randb_{i,1} + 1$ position to $randb_{i,2} - 1$ position will have value in positions that are copied from a selected target vector. The remaining position is from $randb_{i,2}$ position to D, their value in positions will equal to the selected mutant vector as shown in *5f*.

$$U_{i,j,G} = \begin{cases} V_{i,j,G} \text{ if } rand_{i,j} \le CR \text{ or } j = Irand \\ X_{i,j,G} \text{ if } rand_{i,j} > CR \text{ or } j \ne Irand \end{cases} \quad (11)$$

$$U_{i,j,G} = \begin{cases} V_{i,j,G} \text{ when } randb_i \le j \\ X_{i,j,G} \text{ if } randb_i > j \end{cases} \quad (12)$$

$$U_{i,j,G} = \begin{cases} V_{i,j,G} \text{ when } j \le randb_{i,1} \text{ and } j \ge randb_{i,2} \\ X_{i,j,G} \text{ when } randb_{i,1} < j < randb_{i,2} \end{cases} \quad (13)$$

Let $rand_{i,}$ to be random number between 0 and 1, and CR is recombination probability which is the predefined parameters in the proposed heuristics. $Irand$ is random integer number; $randb_{i,}$ $randb_{i,1}$ and $randb_{i,2}$ are random integer numbers which are used to represent position of a vector and these random numbers ranges from 1 to D.

The result of selection process is a set of target vectors in next generation which will be used as starting point of a mutation process of next iteration. The selection process is applied using formula (14)

$$X_{i,j,G+1} = \begin{cases} U_{i,j,G} & \text{if } f(U_{i,j,G}) \le f(X_{i,j,G}) \\ X_{i,j,G} & \text{otherwise} \end{cases} \quad (14)$$

1	2	3	4	5	6	7	8
0.18	0.53	0.22	0.50	0.26	1.00	0.05	0.02

(a) A selected target vector

1	2	3	4	5	6	7	8
0.54	0.28	0.50	0.39	0.93	0.19	0.07	0.29

(b) A selected mutant vector

1	2	3	4	5	6	7	8
0.92	0.67	0.08	0.40	0.18	0.55	0.37	0.17

(c) Value of $rand_{i,j}$

1	2	3	4	5	6	7	8
0.18	0.28	0.50	0.39	0.93	0.19	0.07	0.29

(d) a trial vector produced from a,b and c (CR=0.8)

1	2	3	4	5	6	7	8
0.54	0.28	0.50	0.50	0.26	1.00	0.05	0.02

(e) a trial vector when $randb_i$=3

1	2	3	4	5	6	7	8
0.54	0.28	0.50	0.50	0.26	1.00	0.07	0.29

(f) a trial vector when $randb_{i,1}$=3, $randb_{i,2}$=7

Figure 5. Example of Recombination processes *3.4 Perform selection process on a trial vector* $V_{i,j,G}$ *and a target vector* $X_{i,j,G}$.

Let $f(U_{i,j,G})$ and $f(X_{i,j,G})$ be objective functions of trial vector $(U_{i,j,G})$ and target vector $(X_{i,j,G})$. The objective functions of trial and target vector can be decoded using the procedure addressed in Section 3.1.2.

4. Computational framework and result

4.1. *Compare the proposed algorithm with existing algorithm found in literature*

This section is used to compare performance of the proposed heuristics against existing heuristics developed by many researchers. The comparisons have been carried out over different sets of test instances corresponding to those used in the compared methods and the availability of the test instances.

4.1.1. *Compare with genetic algorithm (Sabuncuoglu et al.* [29]*)*

Sabuncuoglu et al. [29] uses Tonge's 70 tasks problem which is real life problem from electronic industry to test their algorithm against other heuristics and this set of test instances is used to test the proposed algorithm against GA proposed by Sabuncuoglu et al. [29]. Tonge's 70 tasks problem has precedence graph which has 70 tasks. The tasks have different processing time varies from 170 to 364. The given cycle time values also vary from 170 to 364. The computational result is shown in Table 1. The algorithm is executed 5 times for each test instances. The stopping criterion is set to 500 iterations. Let's define following notations: *opt*: optimal solution found in literature, *GA*: number of workstation found by Sabuncuoglu et al. [29], *DE-Min*: minimum number of workstation generated by the proposed algorithm found in 5 runs, *DE-Max*: maximum number of station generated by the proposed algorithm found in 5 runs and *A-C-DE*: average computational time of 5 simulation runs (seconds). All optimal results found in the table are taken from *Kilincci* [14].

From Table 1, it can be seen that DE is better than the solution quality of GA proposed in Sabuncuoglu et al. [29]. DE can find two more optimal solutions than that of Sabuncuoglu et al. [29]. In other words, the genetic algorithm proposed by Sabuncuoglu et al. [29] can find ***75%*** optimal solutions, while the proposed heuristics can find ***100%*** optimal solutions. The average computational time of DE in this group of test instance is 7.28 s.

4.1.2. *Compare with Tabu search algorithms*

In this section, the proposed algorithm is compared with the algorithms found in Chiang [6] and Lapierre et al. [16] which are Tabu search-based heurisitcs. Data-set named Arcus 1 and Arcus 2 are used for the comparison. Arcus 1 is composed of 83 tasks and 7 different values of cycle time, while Arcus 2 is composed of 111 tasks and 6 different values of cycle time. Computational results are shown in Table 2. The algorithm is executed five times for each test instances. The termination condition is set to 100 iterations.

In Table 2, there are six columns, the first column shows values of cycle time of the test instances (Arcus 1 (83 tasks) and Arcus 2 (111 tasks)) which have value of cycle time varied from 5048 to 17,067. The second column is optimal solutions taken from the literature. The third (Chiang's), fourth (Lapierre's), and fifth (DE) column is number of stations that are found by Chiang [6], Lapierre et al. [16] and DE, respectively. The table only showed the minimum number of station generated by the proposed algorithm found in five runs because the maximum and minimum numbers of station found by the proposed heuristics are equal. The last column is the average computational time of DE. From Table 2, it can be seen that DE finds 100% optimal solutions which has equal solution quality as Lapierre et al. [16] but the solution quality is better than that of Chiang [6] which can find only 84.61% optimal solution (11 instances out of 13 instances). The average computational time of the proposed algorithm used to find optimal solution for Arcus 1 and Arcus 2 problems is 0.035 and 0.112, respectively.

4.1.3. *Comparison with other DE algorithm proposed by Nearchou* [18]

Nearchou [18] test his algorithm on Talbot's test instances. This set of instances is composed of 64 test instances (the number of tasks varies from 7 to 111 tasks and the cycle time values vary from 6 to 17,067). The computational result shows in Table 3. The algorithm is executed five times for each test instances. The execution is terminated when the optimal solution is found and the computation result is collected and reported.

From Table 3, the proposed heuristic outperforms the algorithm proposed in Nearchou [18]. The proposed heuristic find *100%* optimal solutions, while Nearchou [18] find only *95.31%* of optimal solutions. In the cases that

Table 1. Comparison with GA (Sabuncuoglu et al. [29]).

Cycle time	Opt	GA	DE – Min	DE – Max	A-C-DE	Cycle time	Opt	GA
170	22	23	22	22	7.418	170	22	23
173	22	23	22	22	7.199	173	22	23
176	22	22	22	22	7.119	176	22	22
179	22	22	22	22	7.208	179	22	22
182	22	22	22	22	7.895	182	22	22

Table 2. Comparison with Tabu search (Chiang [6] and Lapierre et al. [16]).

Cycle time	Optimal solution	Chiang's	Laperiere's	DE	Computational time of DE
Arcus 1					
5048	16	17	16	16	0.035
5853	14	14	14	14	0.035
6842	12	12	12	12	0.037
7571	11	11	11	11	0.035
8412	10	10	10	10	0.034
8898	9	9	9	9	0.035
10,816	8	8	8	8	0.036
Arcus 2					
5755	27	27	27	27	0.12
8847	18	19	18	18	0.11
10,027	16	16	16	16	0.11
10,743	15	15	15	15	0.11
11,378	14	14	14	14	0.11
17,067	9	9	9	9	0.11

Table 3. Compare with Nearchou [18].

Performance measure	Nearchou [18]	DE
#of optimum solutions found	61	64
#of optimal solutions found (%)	95.31	100
Average deviation from the optimal solution (%)	4.67%	0%
Average computational time (seconds)	2.00	0.52

the optimal solution cannot be found, Andreas has 4.67% deviated from the optimal soiutions. Nearchou [18] used average computational time 2.00 s, while the proposed algorithm used 0.52 s computational in average (the computational time is already convert using computer comparing factor found in Dongarra [7]).

4.1.4. *Compare with Kilincci* [14]

In this section, the proposed algorithm is compared with Kilincci [14] which is the most recent single pass procedure found so far in literature. The proposed algorithm is tested on two set of instances. The first set of test instances is similar to the test instances that are used in Section 4.1.2. Another set of test instances is Scholl's problem with 297 tasks and 422 precedence constraints (the cycle time starts from 1394 to 2787), which is known as the largest test instances found in literature. In total, the algorithm is tested on 46 test instances. The computational results are shown in Table 4. Let us denote following notations: *Data-set:* Name of test instances, *Cycle time:* given cycle time, *Kilincci's:* number of station found by *Kilincci* [14], *DE:* number of station found by the proposed heuristic, and *C-DE:* computational time of the proposed heuristic. The algorithm is executed five times for each test instances. The termination condition is set to 100 iterations.

From Table 4, it shows that the proposed algorithm performs almost the same as Kilincci [14]. There is one instance that better solutions than that of Kilincci [14] are obtained which is test instance of Acus 2 cycle time of 5755 and there is also one instances that the proposed algorithm cannot find the optimal solution when Kilincci [14] can find the optimal solution (Scholl's problem that has cycle time 1483). In this group of test instances, it can be concluded that the proposed algorithm has the same solution quality in finding the optimal solution as Kilincci [14].

4.2. *Compare different mutation and recombination formulas*

In this section, the efficiency of the different mutation and recombination formulas will be tested and the best combination will be reported. A total of 15 combinations are reported in Table 5 based on recombine five mutation formulas with three recombination formulas.

The tests reveal the best combination among all 15 combinations. The combinations are generated from five mutation and three recombination formulas. The algorithm is tested on Talbot's and Scholl's problems with 232 test instances. The 232 test instances are divided into three groups which are small (number of tasks less than 45), medium (number of tasks is between 45 and 100), and large (number of task is over 100). According to the classification above, there are 72 small test instances, 93 medium instances, and 67 large instances. All instances are tested, and the number of work stations found by each combination is recorded and the average number of optimal solutions found by the combinations

Table 4. Compare with Kilincci [14].

Data-set	Cycle time	Kilincci's	DE	C-DE
Arcus 1	5048	16	16	0.035
	5853	14	14	0.035
	6842	12	12	0.037
	7571	11	11	0.035
	8412	10	10	0.034
	8898	9	9	0.035
	10,816	8	8	0.036
Arcus 2	5755	28	27	0.12
	8847	18	18	0.11
	10,027	16	16	0.11
	10,743	15	15	0.11
	11,378	14	14	0.11
	17,067	9	9	0.11
Scholl's	1394	51	51	9.478
	1422	50	50	10.011
	1452	49	49	8.986
	1483	48	49	9.355
	1515	47	47	8.861
	1659	43	43	8.836
	1548	46	46	8.843
	1584	45	45	8.819
	1620	44	44	8.839
	1699	42	42	8.799
	1742	41	41	8.822
	1787	40	40	8.016
	1834	39	39	9.038
	1883	37	37	8.311
	1935	36	36	9.385
	1991	35	35	8.383
	2049	34	34	8.451
	2111	33	33	8.758
	2177	32	32	8.745
	2247	31	31	8.751
	2322	30	30	8.058
	2402	29	29	7.410
	2488	28	28	9.321
	2580	27	27	8.573
	2680	26	26	8.042
	2787	25	25	8.791

Table 5. Combination of mutation formula and recombination formulas.

#of combination	Mutation formula	Recombination formula	Name of the combination
1	(6)	(11)	DE-1
2	(6)	(12)	DE-2
3	(6)	(13)	DE-3
4	(7)	(11)	DE-4
5	(7)	(12)	DE-5
6	(7)	(13)	DE-6
7	(8)	(11)	DE-7
8	(8)	(12)	DE-8
9	(8)	(13)	DE-9
10	(9)	(11)	DE-10
11	(9)	(12)	DE-11
12	(9)	(13)	DE-12
13	(10)	(11)	DE-13
14	(10)	(12)	DE-14
15	(10)	(13)	DE-15

is calculated. The average computation time of each combination is also recorded in order to conclude the fastest convert combination among all 15 combinations. The computational results are shown in Table 6. The algorithm is executed five times for each test instances. The termination condition is set to 100 iterations.

In Table 6, the first column shows the name of the combinations (DE-1 to DE-15). It reports three performance measurements which are percentage of optimal solution found by a combination (*%opt*), percentage of deviation of the solution found by a combination, while it cannot find the optimal solution (%dev) and the average computational time (copT). All performance measurements are reported in the form of the group of instances (small, medium, and large size of instances).

The computational result shows that all DE's algorithms can find 100% optimal solution with 0.26 s

Table 6. Compare all combination of mutation and recombination formulas.

	Small size of instances			Medium size of instances			Large size of instances		
	% opt	% dev	copT.	% opt	% dev	copT.	% opt	% dev	copT.
DE-1	100	0	0.184	91.398	0.047	4.132	89.552	0.073	10.427
DE-2	100	0	0.109	91.398	0.045	3.964	89.552	0.125	10.217
DE-3	100	0	0.118	91.398	0.043	5.967	92.537	0.125	10.562
DE-4	100	0	0.257	91.398	0.047	4.04	89.552	0.073	20.755
DE-5	100	0	0.223	91.398	0.047	6.469	89.552	0.073	16.013
DE-6	100	0	0.192	91.398	0.047	3.686	89.552	0.073	10.23
DE-7	100	0	0.215	91.398	0.047	6.701	89.552	0.073	11.345
DE-8	100	0	0.302	91.398	0.047	8.699	89.552	0.073	12.561
DE-9	100	0	0.329	91.398	0.047	9.29	89.552	0.073	17.6
DE-10	100	0	0.313	91.398	0.043	6.247	89.552	0.073	14.551
DE-11	100	0	0.294	94.624	0.044	6.554	89.552	0.073	9.979
DE-12	100	0	0.289	95.699	0.04	7.55	95.522	0.022	14.561
DE-13	100	0	0.354	91.398	0.047	6.67	89.552	0.073	13.767
DE-14	100	0	0.411	91.398	0.047	9.107	89.552	0.073	17.767
DE-15	100	0	0.396	91.398	0.047	5.982	89.552	0.073	13.874
Average	100.00	0.00	0.26	91.899	0.05	6.47	90.15	0.08	13.61

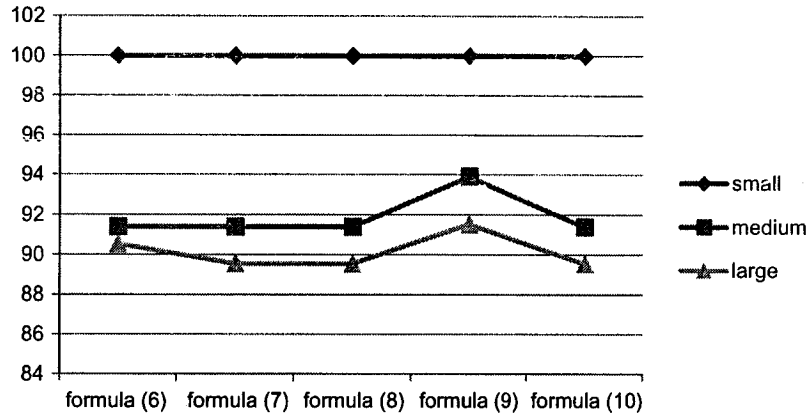


Figure 6. Average %opt found by the mutation formulas.

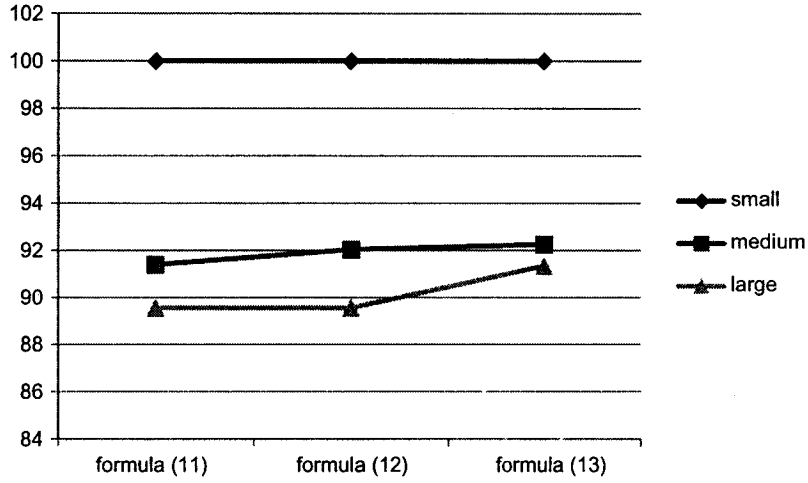


Figure 7. Average *%opt* found by the recombination formulas.

average computational time to solve small-sized instances. DE-2 use lowest computational time among all 15 combinations (0.109 s.). In medium and large-sized test instances, DE-12 seems to be the best combination in finding the optimal solution, because it finds the most number of optimal solutions (95.699%). When it cannot find the optimal solution, DE-12 has the lowest deviation from the optimal solution (0.04%).

In addition, the average values of result quality by utilizing mutation and recombination for every equation are shown in Figures 6 and 7.

By comparing the average values in Figure 6, Equation (9) is the best equation which was utilized in combination DE-10 to DE-12. On the other hand, this equation has the same efficiency as Equations (6)–(10) for solving small-scaled problems, but Equation (9) is the best for solving medium and large-scaled problems.

From Figure 7, the most efficient equation is Equation (13) which is newly proposed in this article. Equation (13) has the same efficiency as Equations (11) and (12) for solving small-scaled problems. However, for solving medium and large-scaled problems, Equation (9) has the same efficiency as other equations.

5. Conclusions

In this article, a DE algorithm to solve SALBP-1 is proposed. The proposed algorithm is compared to many different algorithms found in the literature. Efficiency of different combination of mutation and recombination formulas is also studied in order to reveal the best combination for solving SALBP-1.

By comparing to the metaheuristics found in literature, firstly, a comparison is made with genetic algorithm proposed by Sabuncuoglu et al. [29], the proposed heuristics yield better solution quality than those from genetic algorithm. GA finds optimal solution in only 75% of the test instances, while the proposed heuristic reached optimal solution in 100% of the test instances. Secondly, comparison is made with tabu search-based

heuristics using Arcus 1 and Arcus 2 problems. The results showed that the proposed heuristic is better than Chiang [6], but has the same solution quality as Lapierre et al. [16]. Chiang [6] can find 11 optimal solution out of 13 test instances, while the proposed heuristic and Lapierre et al.[16] can find optimal solution in all test instances (100%). The third comparison is executed over 64 test instances found in Narchou [18]. The computational result shows that the proposed algorithm outperforms Nearchou [18] which can find optimal solutions in 61 out of 64 test instances, while the proposed algorithm reached optimal solutions in all of the test instances.

Finally, the comparison has been made with Kilincci [14]. The comparison is made using Arcus 1, Arcus 2, and Scholl's problems. The computational results reveal that Kilincci [14] and the proposed heuristics yielded the same solution quality with both heuristics can find 97.82% optimal solutions (45 out of 46 instances).

Besides the comparison between the proposed heuristic and other heuristics found in the literature, a test for different mutation and recombination formulas is made. From the computational results, the best results are obtained using mutation formula 9 which calculates mutant vector from four sampled target vectors along with the best vector from previous iteration. However, for small-scaled problems, all mutation formula (Equations (6)–(10)) has the same efficiency for solving the problems. When only consider for recombination formula, the recombination formula (13) which is proposed in this article is the best recombination formula. However recombination formula (13) has the same efficiency as other recombination formulas when they are used to solve small-scaled problems.

When considering both mutation and recombination formulas (combination), the DE-12 which is the combination between mutation formula (9) and recombination formula (13) can yield the best result when comparing with other combinations for solving medium and large-scaled problems. All combinations yield the same efficiency for solving small-scaled problems.

Based on the success of DE proposed in this article, the same encoding, decoding scheme, and two points exponential recombination formula should be applied to other different assembly line balancing problem in order to prove that the contribution of this article can work well with other problems. Moreover, improve effectiveness of DE mechanism such as mutation and recombination method should be considered in order to enhance search capability of DE.

Notes on contributor

Rapeepan Pitakaso is an associate professor in the Department of Industrial Engineering at Ubon Ratchathani University, Thailand. He received his MS from Asian institute of Technology, Thailand, and PhD degrees from University of Vienna, Austria. His research activities include metaheuristics and other optimization techniques.

References

[1] Baybars, I., "A survey of exact algorithms for the simple assembly line balancing problem," *Management Science*, 32, 909–932 (1986a).

[2] Baybars, İ., "An efficient heuristic method for the simple assembly line balancing problem," *International Journal of Production Research*, 24, 149–166 (1986b).

[3] Becker, C. and A. Scholl, "A survey on problems and methods in generalized assembly line balancing," *European Journal of Operational Research*, 168, 694–715 (2006).

[4] Bowman, E. H., "Assembly-line balancing by linear programming," *Operations Research*, 8, 385–389 (1960).

[5] Buatista, J., R. Suarez, M. Mateo and R. Companys, "Local search heuristics for assembly line balancing problem with incompatibilities between tasks," Proceedings of the 2000 IEEE International Conference on Robotics and auTomation, San Francisco, CA, 2404–2409 (2000).

[6] Chiang, W. C., "The application of a tabu search metaheuristic to the assembly line balancing problem," *Annals of Operations Research*, 77, 209–227 (1998).

[7] Dongarra, J. J., Performance of Various Computers Using Standard Linear Equations Software, University of Tennessee, Knoxville, TN (2013).

[8] Erel, E. and S. Sarin, "A survey of the assembly line balancing procedures." *Production Planning & Control*, 9, 414–434 (1998).

[9] Fleszar, K. and K. S. Hindi, "An enumerative heuristic and reduction methods for the assembly line balancing problem," *European Journal of Operational Research*, 145, 606–620 (2003).

[10] Hamtaa, N., S. M. T. Fatemi Ghomia, F. Jolaib and M. Akbarpour Shirazia, "A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect," *International Journal of Production Economics*, 141, 99–111 (2013).

[11] Hoffmann, T. R., "EUREKA: A hybrid system for assembly line balancing," *Management Science*, 38, 39–47 (1992).

[12] Johnson, R. V., "Optimally balancing large assembly lines with "FABLE"," *Management Science*, 34, 240–253 (1988).

[13] Khorasanian, D., S. R. Hejazi and G. Moslehi, "Two-sided assembly line balancing considering the relationships between tasks," *Computers & Industrial Engineering*, 66, 1096–1105 (2013).

[14] Kilincci, O., "Firing sequences backward algorithm for simple assembly line balancing problem of type 1," *Computer and Industrial Engineering*, 60, 830–839 (2011).

[15] Kim, Y. K., Y. J. Kim, Y. Kim, "Genetic algorithms for assembly line balancing with various objectives," *Computers and Industrial Engineering*, 30, 397–409 (1996).

[16] Lapierre, S. D., A. Ruiz and P. Soriano, "Balancing assembly lines with tabu search," *European Journal of Operational research*, 168, 826–837 (2006).

[17] Li, X. and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software*, 55, 10–31 (2013).

[18] Nearchou, A. C., "A differential evolution algorithm for simple assembly line balancing," Paper present in the 16th Int. Federation of Automatic Control (IFAC) World Congress, Jul. 4–8, Prague, Czech Republic (2005).

[19] Nearchou, A. C., "Meta-heuristics from nature for the loop layout design problem," *International Journal of Production Economics*, 101, 312–328 (2006).

[20] Nearchou, A. C., "Balancing large assembly lines by a new heuristic based on differential evolution method," *The International Journal of Advanced Manufacturing Technology*, 34, 1016–1029 (2007).

[21] Nearchou, A. C., "Multi-objective balancing of assembly lines by population heuristics," *International Journal of Production Research*, 46, 2275–2297 (2008).

[22] Nguyen, S. and V. Kachitvichyanukul, "An efficient differential evolution algorithm for multi-mode resource constrained project scheduling problems," *International Journal of Operational Research*, 15, 466–481 (2012).

[23] Nourie, F. J. and E. R. Venta, "Finding optimal line balances with OptPack," *Operations Research letters*, 10, 165–171 (1991).

[24] Nourmohammadi, A. and M. Zandieh, "Assembly line balancing by a new multi-objective differential evolution algorithm based on TOPSIS," *International Journal of Production Research*, 49, 2833–2855 (2011).

[25] Onwubolu, G. O. and D. Davendra, "Scheduling flow shops using differential evolution," *European Journal of Operational Research*, 169, 1176–1184 (2006).

[26] Peeters, M. and Z. Degrave, "An linear programming based lower bound for the simple assembly line balancing problem," *European Journal of Operational Research*, 168, 716–731 (2006).

[27] Qin, A. K., V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on evolutionary computation*, 13, 398–417 (2009).

[28] Rubinovitz, J. and G. Levitin, "Genetic algorithm for assembly line balancing," *International Journal of Production economics*, 41, 343–354 (1995).

[29] Sabuncuoglu, I., E. Erel and M. Tanyer, "Assembly line balancing using genetic algorithms," *Journal of Intelligent Manufacturing*, 11, 295–310 (2000).

[30] Salveson, M. E., "The assembly line balancing problem," *Journal of Industrial Engineering*, 6, 18–25 (1995).

[31] Scholl, A., Balancing and sequencing of assembly lines, Physica-Verlag, Heidelberg (1999).

[32] Scholl, A. and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *European Journal of Operational Research*, 168, 666–693 (2006).

[33] Scholl, A. and R. Klein, "SALOME: A biderectional branch and bound procedure for assembly line balancing," *INFORMS Journal on Computing*, 9, 319–334 (1997).

[34] Scholl, A. and S. Voβ, "Simple assembly line balancing Heuristic approaches," *Journal of Heuristics*, 2, 217–244 (1996).

[35] Sprecher, A., "A competitive branch and bound algorithm for the simple assembly line balancing problem," *International Journal of Production Research*, 37, 1787–1816 (1999).

[36] Storn, R. and K. Price, "Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, 11, 341–359 (1997).

[37] Tasgetirena, M. F., Q. K. Pan, P. N. Suganthan and O. Buyukdagli, "A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem," *Computers & Operations Research*, 40, 1729–1743 (2013).

[38] Tsai, J. T.J. C., Fang and J. H., Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Computers & Operations Research*, 40, 3045–3055 (2013).

[39] Vilà, M. and J. Pereira, "A branch-and-bound algorithm for assembly line worker assignment and balancing problems," *Computers & Operations Research*, 44, 105–114 (2014).

[40] Wang, L., C. X. Dun, W. J. Bi and Y. R. Zeng, "An effective and efficient differential evolution algorithm for the integrated stochastic joint replenishment and delivery model," *Knowledge-Based Systems*, 36, 104–114 (2012).

[41] Wisittipanich, W. and V. Kachitvichyanukul, "Differential Evolution Algorithm for Job Shop Scheduling Problem," *Industrial Engineering and Management Systems*, 10, 203–208 (2011).

[42] Wisittipanich, W. and V. Kachitvichyanukul, "Two enhanced differential evolution algorithms for job shop scheduling problems," *International Journal of Production Research*, 50, 2757–2773 (2012).

แบบเสนอขอรับค่าตอบแทนในการตีพิมพ์วารสารวิชาการ

1. เอกสารประกอบการเสนอขอรับค่าตอบแทนในการตีพิมพ์วารสารวิชาการ

1. 1 แบบขอรับค่าตอบแทน

1. 2 หนังสือขออนุมัติค่าตอบแทน เรียน รองคณบดีฝ่ายวิจัยและบริการวิชาการผ่านหัวหน้าภาควิชา

1.3 สำเนาบทความวิจัยที่ได้รับการตีพิมพ์ในวารสารวิชาการ

1.4 รายละเอียดวารสาร

1.5 เอกสารแสดงค่า Impact factor ของวารสารที่ตีพิมพ์

2. รายละเอียดข้อมูลประกอบเสนอขอรับค่าตอบแทนในการตีพิมพ์วารสารวิชาการ

2.1 ผู้เสนอขอรับค่าตอบแทน ชื่อ-สกุล ระพีพันธ์ ปิตาคะโส

2.2 ชื่อบทความวิจัย (ภาษาไทย) .....

.....

1. **(ภาษาอังกฤษ)** Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types

2.3 รายละเอียดของวารสาร

ชื่อวารสาร. **Engineering Optimization**.....

อยู่ในฐานข้อมูล ☑ ISI กรณี ISI ช่วยระบุ impact factor: 1.230 ○ SCOPUS ○ SJR.....

ปีที่.....ฉบับที่.....เดือน.....ปี 2015 หน้า.....

2.4 สถานะในบทความวิจัยเป็น

☑ ชื่อแรก (first author) ☑ ผู้รับผิดชอบบทความ (corresponding author)

○ ผู้มีส่วนร่วมในบทความ

2.5 การมีส่วนร่วมในบทความของนักศึกษา

○ ใช้ขอจบการศึกษา ☑ ไม่ใช้ขอจบการศึกษา

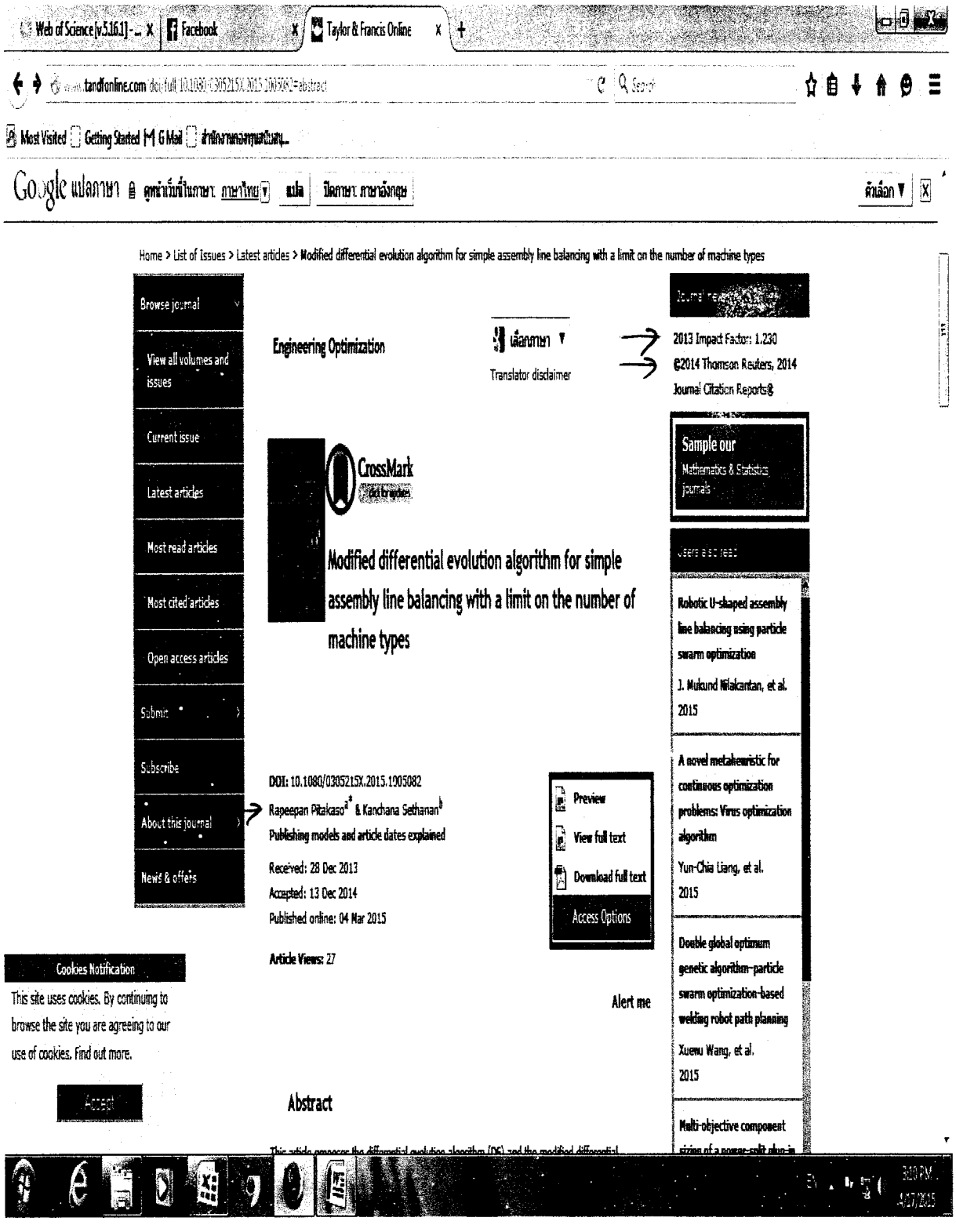
การรับรองสัดส่วนผลงานทางวิชาการ กรุณากรอกข้อมูลตามแบบฟอร์มนี้ตามความเป็นจริง และรักษาไว้ซึ่ง **จรรยาบรรณ** และขอรับรองว่า บทความนี้ไม่เป็นส่วนหนึ่งของวิทยานิพนธ์ของผู้เสนอขอรับค่าตอบแทน

ลำดับที่	ชื่อ-สกุล	สัดส่วนผลงานทางวิชาการ (%)	ลงนามรับรองข้อมูล
1	Rapeepan Pitakaso	60%	
2	Kanchana Sethanan	40%	
3			
4			
5			

หมายเหตุ: กรณีผู้เสนอขอรับค่าตอบแทนเป็นชื่อแรก หรือ ผู้รับผิดชอบบทความสามารถรับรองแทนผู้เขียนร่วมได้

.....

ผู้เสนอขอรับค่าตอบแทน


Web of Science [v.5.16.1] - ...
Facebook
Taylor & Francis Online
tandfonline.com
Most Visited
Getting Started
Gmail
Google แปลภาษา
Home > List of Issues > Latest articles > Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types
Browse journal
View all volumes and issues
Current issue
Latest articles
Most read articles
Most cited articles
Open access articles
Submit
Subscribe
About this journal
News & offers
Engineering Optimization
Translator disclaimer
2013 Impact Factor: 1.230
©2014 Thomson Reuters, 2014
Journal Citation Reports®
Sample our
Mathematics & Statistics
journals
CrossMark
Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types
DOI: 10.1080/0305215X.2015.1005082
Rapeepan Pitakaso & Kanchana Sethanan
Publishing models and article dates explained
Received: 28 Dec 2013
Accepted: 13 Dec 2014
Published online: 04 Mar 2015
Article Views: 27
Preview
View full text
Download full text
Access Options
Alert me
Robotic U-shaped assembly line balancing using particle swarm optimization
J. Mukund Nilakantan, et al.
2015
A novel metaheuristic for continuous optimization problems: Virus optimization algorithm
Yun-Chia Liang, et al.
2015
Double global optimum genetic algorithm–particle swarm optimization-based welding robot path planning
Xuewu Wang, et al.
2015
Cookies Notification
This site uses cookies. By continuing to browse the site you are agreeing to our use of cookies. Find out more.
Accept
Abstract

Results: 853
(from All Databases)

You searched for: PUBLICATION NAME: (Engineering Optimization)
...More

Sort by: **Publication Date -- newest to oldest**

☐ Select Page | **Save to EndNote online** | **Add to Marked List**

Create Citation Report

Refine Results

Search within results for...

Databases

Research Domains

☐ SCIENCE TECHNOLOGY

Refine

Research Areas

Refine

☐ 1. **Classifier-guided sampling for discrete variable, discontinuous design space exploration: Convergence and computational performance**

By: Backlund, Peter B.; Shahan, David W.; Seepersad, Carolyn Conner
ENGINEERING OPTIMIZATION Volume: 47 Issue: 5 Pages: 579-600 Published: MAY 4 2015

View Abstract

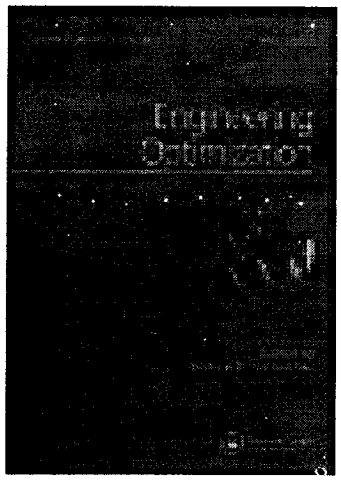
Times Cited: 0
(from All Databases)

☐ 2. **Multi-objective optimization in spatial planning: Improving the effectiveness of multi-objective evolutionary algorithms (non-dominated sorting genetic algorithm II)**

By: Karakostas, Spiros
ENGINEERING OPTIMIZATION Volume: 47 Issue: 5 Pages: 601-621 Published: MAY 4 2015

View Abstract

Times Cited: 0
(from All Databases)



Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types

Rapeepan Pitakaso[a] & Kanchana Sethanan[b]
[a] Department of Industrial Engineering, Research Unit on System Modelling for Industry, Ubon Ratchathani University, Ubon Ratchathani, Thailand;
[b] Industrial Engineering Department, Research Unit on System Modelling for Industry, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand

Published online: 04 Mar 2015.

To cite this article: Rapeepan Pitakaso & Kanchana Sethanan (2015): Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types, Engineering Optimization, DOI: 10.1080/0305215X.2015.1005082

To link to this article: http://dx.doi.org/10.1080/0305215X.2015.1005082

Gmail

Thitikan Boonkang <thitikan.b@ubu.ac.th>

Fw: Engineering Optimization - Decision on Manuscript ID GENO-2014-0378.R3
1 ข้อความ

Rapeepan Pitakaso <touch@rocketmail.com> 21 มิถุนายน 2558 20:03
ตอบกลับไปยัง: Rapeepan Pitakaso <touch@rocketmail.com>
ถึง: "thitikan.b@ubu.ac.th" <thitikan.b@ubu.ac.th>

****
****
** Dr.Rapeepan Pitakaso **
** P.O Box 3 Varinchamrab **
** Ubonratchathanee **
** Thailand 34190 **
****
****

On Saturday, December 13, 2014 10:01 PM, "morrisfan@ntut.edu.tw" <morrisfan@ntut.edu.tw> wrote:

12/13/2014

Dear Mr Pitakaso:

Ref: Modified Differential Evolution algorithm for Simple Assembly Line Balancing with limit on number of machines

Referees have now considered your paper and have recommended publication in Engineering Optimization. I am pleased to accept your paper in its current form which will now be forwarded to the publisher for copy editing and typesetting.

Please note, if you have provided a PDF file for the peer-review process, you will need to promptly supply your original source files when contacted by the publisher. These source files will prevent any delay in the copy editing and typesetting process and the eventual publication of your manuscript.

You will receive proofs for checking, and instructions for transfer of copyright in due course.

Thank you for your contribution to Engineering Optimization and we look forward to receiving further submissions from you.

Sincerely,
Professor Shu-Kai S. Fan
Associate Editor, Engineering Optimization
morrisfan@ntut.edu.tw