International Journal of Advanced Manufacturing Technology, 2011, Volume 59, Issue 1-4, pp335-349

A Review on Assembly Sequence Planning and Assembly Line Balancing Optimisation using Soft Computing Approaches

Mohd Fadzil Faisae Rashid, Windo Hutabarat, Ashutosh Tiwari

Abstract

Assembly optimisation activities occur across development and production stages of manufacturing goods. Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) problems are among the assembly optimisation. Both of these activities are classified as NP-hard. Several soft computing approaches using different techniques have been developed to solve ASP and ALB. Although these approaches do not guarantee the optimum solution, they have been successfully applied in many ASP and ALB optimisation works. This paper reported the survey on research in ASP and ALB that use soft computing approaches for the past 10 years. To be more specific, only Simple Assembly Line Balancing Problem (SALBP) is considered for ALB. The survey shows that three soft computing algorithms that frequently used to solve ASP and ALB are Genetic Algorithm, Ant Colony Optimisation and Particle Swarm Optimisation. Meanwhile, the research in ASP and ALB is also progressing to the next level by integration of assembly optimisation activities across product development stages.

Keywords: Assembly sequence planning; Assembly line balancing; soft computing

1 Introduction

In product development, current global market continuously gives pressure to manufacturer to compete with competitors from all over the world. Manufacturer needs to speed up the time to market and at the same time minimise the manufacturing cost to ensure that their products remain competitive [1]. Assembly is considered one of the important processes in manufacturing. It consumes up to 50% of total production time and account for more than 20% of total manufacturing cost [2].

Research in assembly optimisation can help manufacturer to speed up assembly process and reduce assembly cost. According to [3], research in assembly optimisation can be categorised based on which product development and production phases is being studied (Fig. 1).

In product conception and design stage, the aim of assembly optimisation is to reduce the assembly costs by applying design for assembly (DFA) approach in product design. Besides reducing cost, DFA may also bring about additional benefits in terms of increased quality, reliability and shorter manufacturing time. The approach shortens the product cycle and ensures a smoother transition from prototype to production [4].

Assembly optimisation in the production planning stage deals with determination of optimum assembly sequence and determination of optimum location of each resource. Solving the Assembly Sequence Planning (ASP) problem is crucial because it will determine many assembly aspects including tool changes, fixture design and assembly freedom. Assembly sequence also influences overall productivity because it determines how fast and accurate the product is assembled.

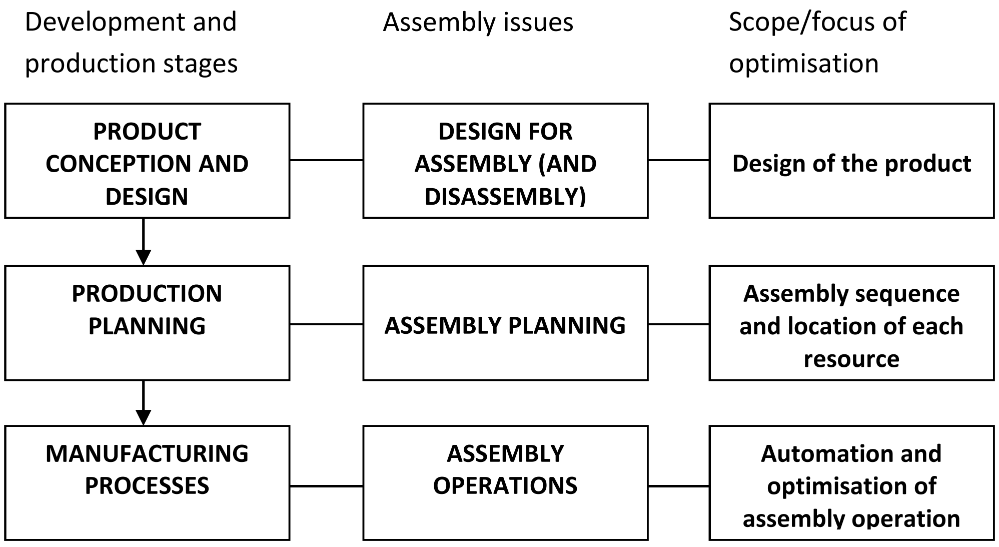

Development and production stages
Assembly issues
Scope/focus of optimisation
PRODUCT CONCEPTION AND DESIGN
DESIGN FOR ASSEMBLY (AND DISASSEMBLY)
Design of the product
PRODUCTION PLANNING
ASSEMBLY PLANNING
Assembly sequence and location of each resource
MANUFACTURING PROCESSES
ASSEMBLY OPERATIONS
Automation and optimisation of assembly operation

Fig. 1 Assembly related issues in different product development stages [3]

During manufacturing processes stage, assembly optimisation is focused on two major activities. The first activity is determining the optimum automation level in assembly. The purpose of this activity is to apply the appropriate automation level in assembly in order to balance the investment in automation and the output. The second activity in this stage is assigning the assembly tasks into workstations, such that workstations have equal or almost equal load [3]. This activity is usually known as Assembly Line Balancing (ALB). In this stage, research in assembly optimisation focuses more on ALB problem rather than optimisation of automation level. It can be observe through the number of publication in optimising both problems.

Both ASP and ALB problems are classified as NP-hard problem and cannot be solved in polynomial time even using a powerful computer [5–8]. In ASP and ALB optimisation, the soft computing approach is more acceptable because of their ability to handle more complex problems, larger size problems and numerous side constraints. Lendak *et al.* [9] define the soft computing method as an approach that is characterized by the use of inexact solutions to computationally hard tasks for which an exact solution cannot be derived in polynomial time.

This paper surveys the past 10 years of research that work on ASP and ALB optimisation using soft computing methods. Besides reviewing the current research pattern, this paper also explores the potential of ASP and ALB research. The rest of this paper is organised as follow: Sections 2 and 3 review the ASP and ALB problems starting with problem representation, constraints and optimisation objectives. Section 4 reviews on soft computing methods that is being used in ASP and ALB optimisation. Section 5 discussed the trends and research potentials in ASP and ALB. Finally, Section 6 summarises and concludes the survey.

2 Assembly sequence planning

ASP is one of important component in assembly planning. ASP refers to a task for which planners, on the basis of their particular heuristics in assembling all the components of a product, arrange a specific assembly sequence according to the product design description [10].

ASP is an NP-hard combinatorial problem [7, 8], where the solution space is excessively increased when the number of component increased. Consider a product with six components that can be assembled in any sequences. In this case, the number of possible solution for this product is given by $s = 6!$ which is equal to 720 solutions. When the number of component increased to seven, the possible solutions for the products excessively increased to $7! = 5{,}040$. Additionally, in a real assembly problem, there are some constraints that need to be considered when generating assembly sequences.

Previous research shows that many approaches were proposed and used by researchers to represent the ASP problem. The most prominent way to represent the ASP problem is by using directed graph method as used in [7, 11–13]. An assembly can be described by a directed graph $D = (P, C)$. P is a finite nonempty set of vertices, and C is a set of edges connecting them. Each vertex represents a component, and each edge represents a relationship between the two components. In some cases, the vertices and edges bring additional information such as assembly orientation, tool, assembly type and assembly time as used in [5].

In assembly representation, directed graph is specifically known as precedence diagram since the graph represents the precedence relation of assembly [14]. Figure 2 shows the assembly precedence diagram with additional assembly information. In this diagram, the vertices represent the assembly components. Meanwhile, the information $T1$, $T2$, $T3$ and $T4$ represent the assembly tools and $(+x, -x, +y, -y, +z, -z)$ represents the assembly direction for a particular component. Therefore, when an assembly sequence is established, it can be evaluated based on information in assembly precedence diagram.

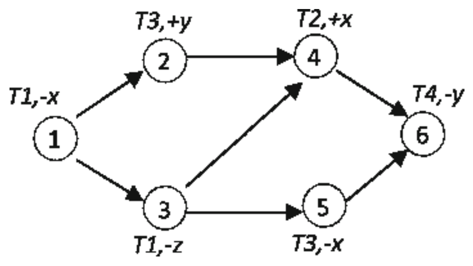


Fig. 2 Assembly precedence diagram with additional information

The assembly sequence is evaluated according to objective function. In previous research, to establish the objective function, the tool and direction variable is transformed to measurable format such as cost [15– 17], time [18] or penalty index [13]. Besides this approach, researchers like [10, 19, 20] use connector-based approach to represent ASP problem. In this method, each connector may assemble two or more components. Every vertex represents one connector in assembly process and brings information of fasteners type, assembly direction, tools type and standard assembly time.

2.1 ASP constraints

According to [3], there are two types of constraints in assembly, which are '*absolute constraints*' and '*optimisation constraints*'. The absolute constraints refer to constraints that, if violated, lead to infeasible assembly sequence. Meanwhile, the optimisation constraints are the constraints that lead to lower quality of assembly sequences when violated.

In ASP context, the absolute constraints that are usually considered include precedence and geometrical constraints. Precedence constraint shows the relation of predecessor and successor components for assembly process. The precedence constraint cannot be violated, otherwise the infeasible assembly sequence will be generated. Precedence constraint can be represented in precedence diagram (Fig. 2) or in matrix form. Table 1 presents the precedence matrix for precedence diagram in Fig. 2. In this matrix, when part *i* must be assembled after part *j*, *P(i, j)* = 1. Otherwise, the matrix will be left empty.

Meanwhile, geometrical constraint in assembly is about assembling the components without any collision. When mating two parts, there must be at least one collision-free trajectory that allows to bring components in contact. All valid assembly sequences must meet geometric constraints for a given structure. Chen and Liu [21] use a matrix to describe geometric constraints between components in an assembly. For each pair of components (*Pi, Pj*), the matrix records directions in which *Pi* can be assembled without colliding with *Pj*. Then, a set of valid assembly directions, for each (*Pi, Pj*) is defined, as the moving wedge of *Pi* with respect to *Pj* , denoted by MW*(Pi, Pj)*. They compute moving wedges for all pairs of components and store all moving wedges in the *MW* matrix.

Table 1 Precedence matrix (*P*) for Fig. 2

i	*j*					
	1	2	3	4	5	6
1						
2	1					
3	1					
4	1	1	1			
5	1		1			
6	1	1	1	1	1	

On the other hand, the constraints that classified as optimisation constraints are related with optimisation objectives of the problem. The constraints that are classified in this category include assembly tool constraint, assembly direction constraint and assembly stability constraint.

Wang and Liu [22] use the tool matrix TM = [*tij*]*nxm* to represent the tool constraint. Where, *n* is the number of parts and *m* is the number of practicable tools to assemble the corresponding part. After the optimal or near optimal assembly sequences have been generated, the corresponding tools are also confirmed and at the same time, the number of change (*nt*) of the assembly tools can be obtained.

Meanwhile, the assembly direction constraint is represented as penalty index in [19, 23, 24]. In these papers, when the subsequent assembly direction is different with the current direction, a penalty will be given to that particular assembly sequence.

The stability constraint is defined by [11] when the assembling parts maintain relative position and they do not break contact during the assembly operation. Wang and Liu [22] categorised the stability strengthened by the assembly connectors into three levels according to connection strength; strong, weak and unstable connection. This classification is also used by [25]. In this approach, strong connection will be assigned '0' index, weak connection '1' index and unstable connection '2' index.

2.2 ASP optimisation objectives

From the past 10 years, various objectives and soft computing techniques have been used to optimise ASP problem. The most popular ASP objective is to minimise the number of assembly direction changes. This objective is applied in 29 out of 39 cited research papers in ASP. In this objective, the assembly directions concerned are along the three principal axes *(+x, −x*; *+y, −y*; *+z, −z)*. When the direction of the next assembly part is different with the current direction, a penalty is given according to the magnitude of direction change. The optimum sequence according to this objective will have minimum penalty caused by direction change.

The second most popular ASP objective is to minimise number of tool changes which were used in 21 cited research papers. In assembly, tool change time is considered one of non-productive activity and may consume large time if not well managed. In this case, when the next assembly process requires different tool with the current assembly process, the penalty will be given. The most optimum sequence following this objective is the sequence with the least tool change penalty.

Besides these two most frequent objectives, the ASP objective to minimise assembly type change is used in nine cited research papers. This objective considers the physical assembly features change such as mating, aligning, screwing, reverting etc. [8, 26–28].

In the next place, four ASP optimisation objectives shares similar position with four cited papers. These objectives are to minimise assembly complexity, minimise connector similarity, maximise assembly stability and minimise geometrical constraint penalty. In ASP, some of research stated that the geometrical constraint is a compulsory restriction. When the generated assembly sequence did not match with geometrical constraint, the assembly sequence will not be evaluated. Therefore, this attribute is not included as objective. However, in a small number of ASP research as in [25, 29–31], the geometrical constraint is declared as one of ASP objective. When the assembly geometry is unfeasible, the large penalty index will be given to fitness function. Therefore, the unfeasible sequence will not appear as optimum assembly sequence in final results.

Figure 3 shows that three least frequent ASP objectives that were used in cited papers are to minimise assembly cost, assembly time and assembly tool travel distance. In ASP context, the objective to minimise assembly time is suitable to be applied in assembly cell. Meanwhile, the objective to minimise assembly tool travel distance is related with Printed Circuit Board assemblies that involved robotic pick and place arm.

It needs to bear in mind that more than half of cited ASP research are using multi-objective optimisation technique that employed more than one objective in their research. Therefore, the total objectives frequencies as shown in Fig. 3 is more than the total ASP cited research. Details on this information are available in Table 2.

3 Assembly line balancing

ALB is the decision problem of optimally partitioning the assembly work among the stations with respect to some objective [5]. ALB was first mathematically formulated by Salveson in 1955. This problem aims at grouping assembly operations, which have to be performed to produce final products, and assigning the groups of operations to stations, so that the total assembly time required at each station is approximately the same and the precedence constraints between operations are respected [32].

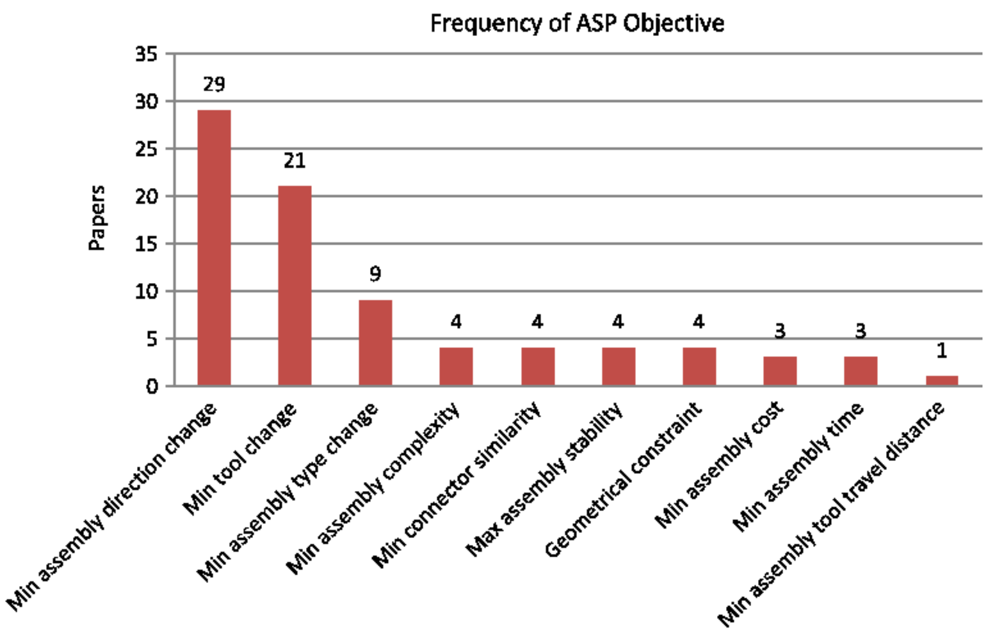


Fig. 3 Frequency of ASP objective in cited research

In general, researchers like [6, 33, 34] divide ALB problem into two categories; Simple Assembly Line Balancing Problem (SALBP) and Generalised Assembly Line Balancing Problem (GALBP). SALBP deals with a serial assembly line that processes a unique model of a single product with all input parameters known with certainty [35]. SALBP can be classified into three groups according to the objectives [36]:

– SALBP-1: the objective is to minimise the number of stations on the line for a given cycle time

– SALBP-2: the objective is to minimise the cycle time for a given number of stations on the line

– SALBP-E: the objective is to maximise the line efficiency for variable cycle time and number of stations

Meanwhile, GALBP includes all of the problems that are not SALBP, such as balancing of mixed model, parallel, U-shaped and two-sided lines with stochastic-dependent processing times [37]. In this paper, only SALBP will be considered since it has accumulated a large number of works.

The simple ALB problem can be represented in precedence diagram that contain n vertices and a set of edges. Each vertex represents an assembly task. Meanwhile, the vertices weight shows the assembly time and the edges reflecting the successor tasks.

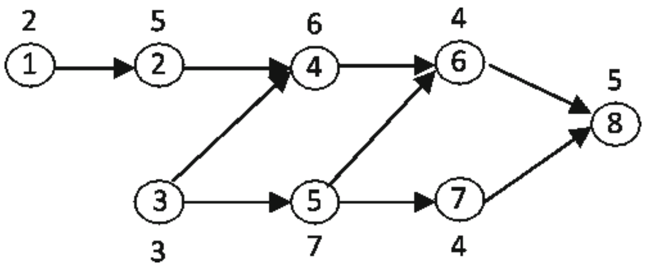
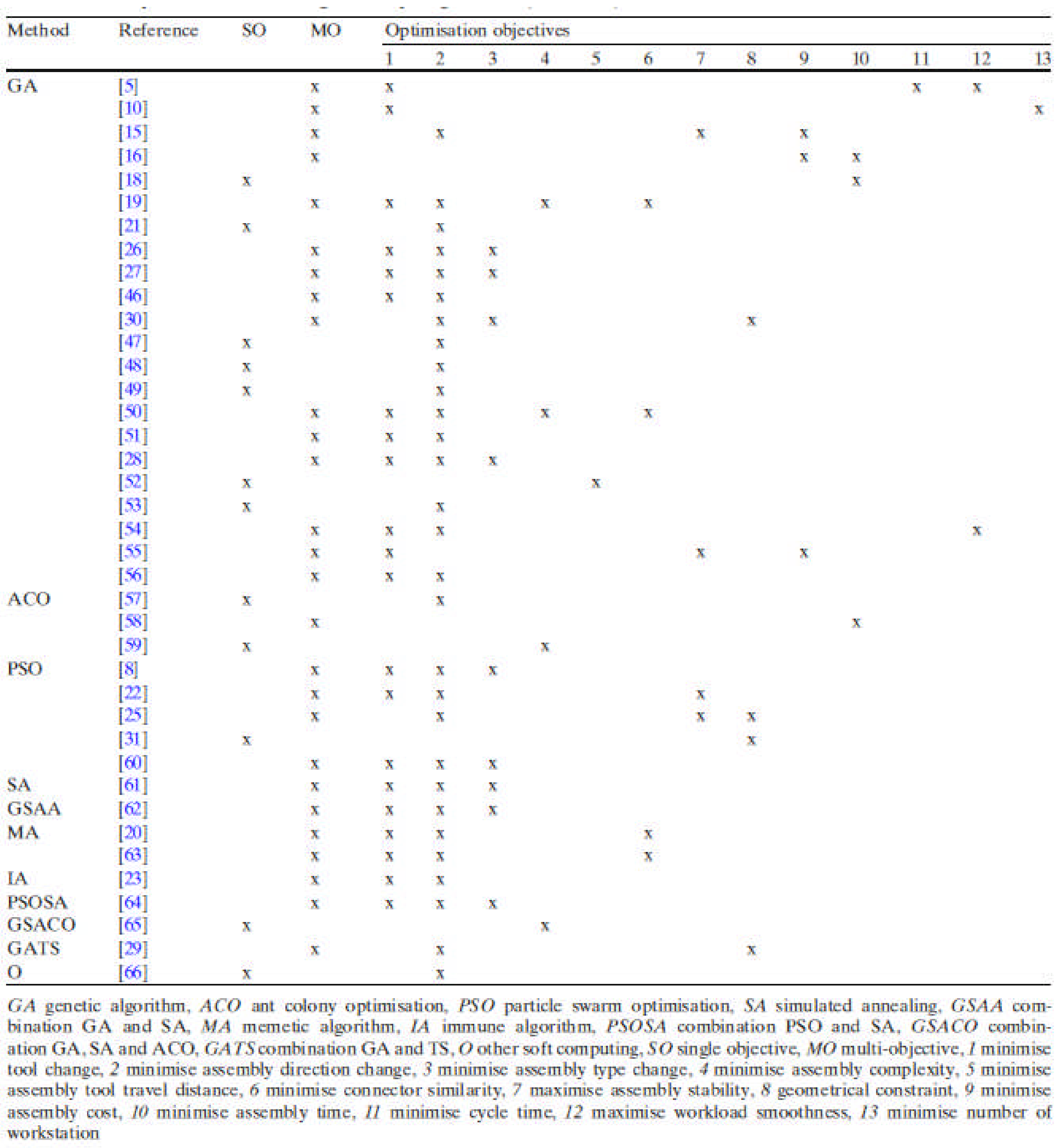


Fig. 4 Precedence diagram for ALB

Solving ALB problem is about assigning the tasks *Vi(i* = 1*,* 2*, ..., n)* into workstation *Wj(j* = 1*,* 2*, ..., m)*subjected to assembly constraints and optimisation objectives. In this problem, assembly time in each node is known as task time, *ti* that refers to task *i*. Meanwhile, the total task time in workstation *Wj* is named as processing time, *pj*. The highest processing time among all workstations then is defined as cycle time, *c*. In assembly line, the cycle time will determine the production rate, *R*, which is given as follows:

$$= \frac{1}{-} \qquad (1)$$

Table 2 Summary of ASP research using soft computing methods (2000–2010)



Method	Reference	SO	MO	Optimisation objectives												
				1	2	3	4	5	6	7	8	9	10	11	12	13
GA	[5]		x	x										x	x	
	[10]		x	x												x
	[15]		x		x					x		x				
	[16]		x									x	x			
	[18]	x											x			
	[19]		x	x	x		x		x							
	[21]	x			x											
	[26]		x	x	x	x										
	[27]		x	x	x	x										
	[46]		x	x	x											
	[30]		x		x	x					x					
	[47]	x			x											
	[48]	x			x											
	[49]	x			x											
	[50]		x	x	x		x		x							
	[51]		x	x	x											
	[28]		x	x	x	x										
	[52]	x						x								
	[53]	x			x											
	[54]		x	x	x										x	
	[55]		x	x						x		x				
	[56]		x	x	x											
ACO	[57]	x			x											
	[58]		x										x			
	[59]	x					x									
PSO	[8]		x	x	x	x										
	[22]		x	x	x					x						
	[25]		x		x					x	x					
	[31]	x									x					
	[60]		x	x	x	x										
SA	[61]		x	x	x	x										
GSAA	[62]		x	x	x	x										
MA	[20]		x	x	x				x							
	[63]		x	x	x				x							
IA	[23]		x	x	x											
PSOSA	[64]		x	x	x	x										
GSACO	[65]	x					x									
GATS	[29]		x		x						x					
O	[66]	x			x											

GA genetic algorithm, *ACO* ant colony optimisation, *PSO* particle swarm optimisation, *SA* simulated annealing, *GSAA* combination GA and SA, *MA* memetic algorithm, *IA* immune algorithm, *PSOSA* combination PSO and SA, *GSACO* combination GA, SA and ACO, *GATS* combination GA and TS, *O* other soft computing, *SO* single objective, *MO* multi-objective, *1* minimise tool change, *2* minimise assembly direction change, *3* minimise assembly type change, *4* minimise assembly complexity, *5* minimise assembly tool travel distance, *6* minimise connector similarity, *7* maximise assembly stability, *8* geometrical constraint, *9* minimise assembly cost, *10* minimise assembly time, *11* minimise cycle time, *12* maximise workload smoothness, *13* minimise number of workstation

Since cycle time is the highest processing time among all workstations, the difference between cycle time and processing time is unproductive time which also known as *idle time*. The total idle time in assembly line is calculated as follows:

$$= \quad . \quad - \tag{2}$$

As an example, the assembly tasks in Figure 4 are assigned into four workstations; W_1= {1,2}, W_2= {3,5}, W_3= {4,6} and W_4={7,8}. It was found that the processing time for each workstation is p_1 = 7, p_2 = 10, p_3 = 10 and p_4 = 9. The highest processing time is found in W_2 and W_3, therefore the cycle time for this problem is *c* = 10 time units. The idle time for this solution is calculated as follows:

Idle time = 4(10) − (7 + 10 + 10 + 9)

= 4 time units

3.1 ALB Constraints

In ALB, the important constraints that highlighted by [38] are occurrence constraint, precedence constraint and capacity constraint. The occurrence constraint refer to restriction that ensure each task be assigned to exactly one workstation. For this purpose, an assignment matrix that consists of task and workstation variables is established. For *i*th task and *j*th workstation, *xij* = 1 if task *i* is assigned to workstation *j* and 0 if otherwise. [38] formulated the precedence constraint as follows:

$$. \quad - \quad . \quad \leq 0 \tag{3}$$

In this case, *j* refers to workstation, *m* is number of workstation, *p* is task/s that immediately precede task *i*. Meanwhile, *xpj* = 1 if task *p* is assigned to workstation *j* and 0 if otherwise.

In the meantime, capacity constraint depends on SALBP problem. For SALBP-1, the capacity constraint refers to maximum allowable cycle time in workstation. It can be formulated as follows:

$$. \quad \leq \tag{4}$$

In this equation, *ti* refers to processing time for task *i* and *c* is predetermined cycle time for the assembly line. Meanwhile, in SALBP-2, the capacity constraint is represented by the maximum number of workstations in assembly line.

ALB research works have also addressed problems that consider additional restrictions apart from cycle time and precede constraints. For example, [39] considered a problem involving resource constraint, which defined the assembly space as one of constraint. Other examples include zoning constraint [40] and uniqueness constraint [41].

3.2 ALB Optimisation objectives

Figure 5 shows the frequencies of ALB objectives that has been recorded from 45 cited research papers. The most frequent objective is to maximise workload smoothness. In assembly line, the basic workload smoothness is measures by calculating workload devia tion as follows:

$$=\frac{\sum\quad(\quad-\quad)}{} \qquad (5)$$

where *n* is number of component, Ct is cycle time, Pt is processing time and *m* is number of workstation. In this case minimising the workload deviation will maximise the workload smoothness [5].

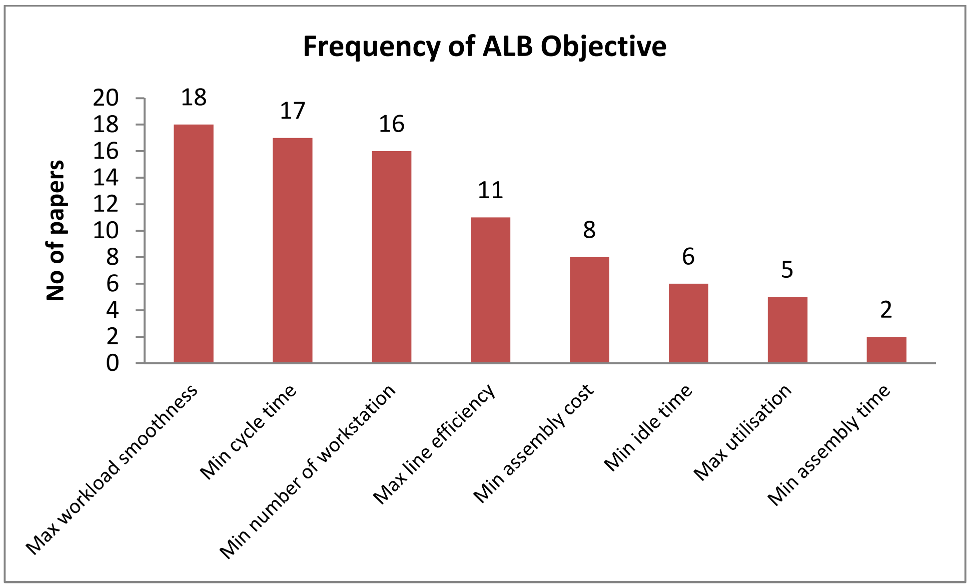


Fig. 5 Frequency of ALB objective in cited research

The objective to minimise cycle time is recorded in 17 cited research papers. Cycle time is available time in each workstation, to complete the required tasks to process a unit of product. It is also defined as the time interval between the processing of two consecutive units [42]. In ALB view, the cycle time is equal to the longest processing time on any worksta-tions. The ALB objective to minimise the number of workstation is also important as it had been used in 16 cited papers. Usually, this objective is used in combina tion with upper limit of cycle time and another objective to maximise workload smoothness. In this case, the smallest number of workstation is not always the most optimum sequence because the workload balance will need to be considered as well.

In ALB, the objective to maximise line efficiency has seen moderate frequency of usage as it appears in 11 cited research. Line efficiency is the ratio between total processing time in all workstations to the product of cycle time and number of workstation. The assembly line efficiency is given by the following equation;

$$=\frac{\sum\quad(\quad)}{\times}\times 100 \qquad (6)$$

With *m* is number of workstation, Pt is processing time in workstation *i*th and Ct is cycle time.

The objective to minimise assembly cost is also moderately popular. To use this objective in ALB optimisation, many assumptions need to be made such as labour cost, equipment utilisation cost and setup cost. Most of the related costs are dependent on variables like time, market price and geographical location. Therefore, this objective is only applicable to particular case studies.

Besides that, objective to minimise idle time and maximise utilisation in assembly has also been used in ALB optimisation. The idle time in each workstation is defined as the difference between processing time and allowed cycle time in assembly line [43]. In the meantime, the assembly utilisation measure has been implemented in variety of ways. McMullen and Tarasewich [44] use assembly utilisation and associate the objective with labour utilisation. Meanwhile, [45] combine the labour and space utilisation to represent assembly utilisation objective.

The least frequent objective to optimise ALB is to minimise the total assembly time. The total assembly time in an assembly line is defined as the total processing time in all workstations for a product. This objective is rarely used because in ALB context, the cycle time is more important than total processing time since it will determine the production rate of the assembly line.

Tables 2 and 3 show the summary of the research in ASP and ALB using soft computing methods for the last 10 years, respectively.

4 Optimisation methods

Previous research in ASP and ALB optimisation shows that various soft computing methods were used. Figure 6 shows the number of papers that used different soft computing methods to optimise ASP and ALB problems for the past ten years. According to the diagram, three most dominant optimisation methods which had been used in almost 70% of the cited research are genetic algorithm (GA), ant colony optimisation (ACO) and particle swarm optimisation (PSO).

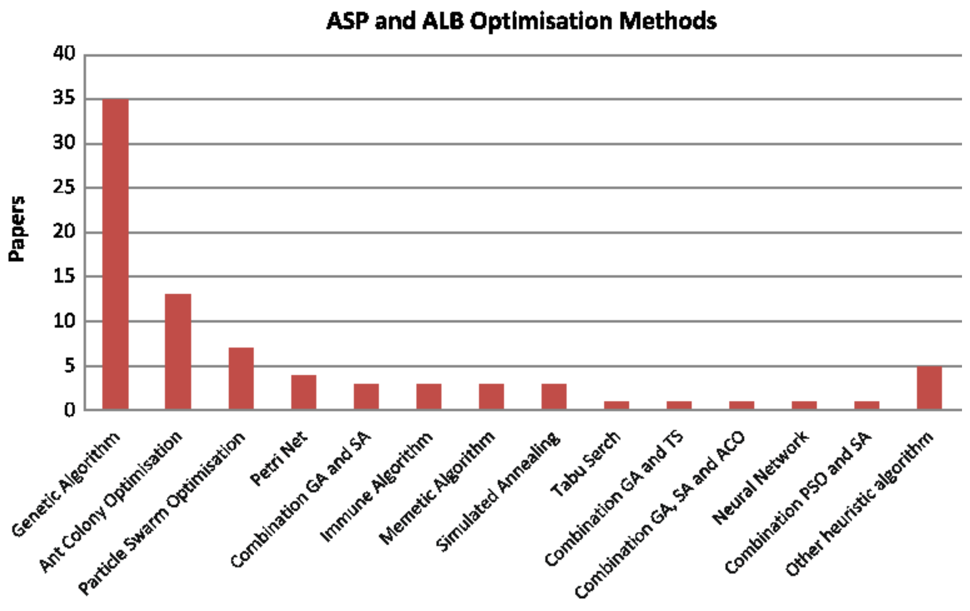
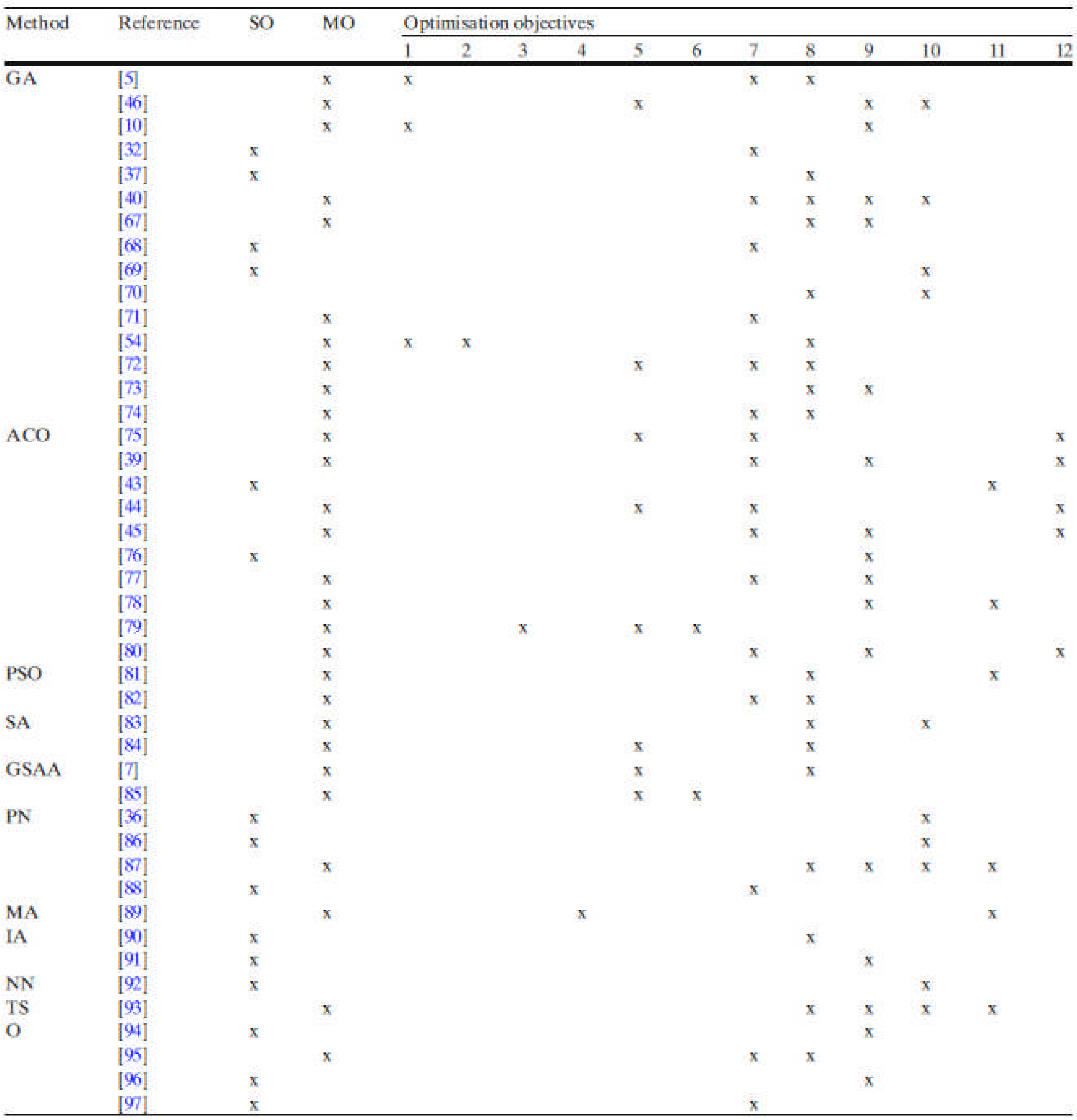


Fig 6: Number of paper used different soft computing methods

Table 3: Summary of ALB research using soft computing methods (2000-2010)



Method	Reference	SO	MO	Optimisation objectives											
				1	2	3	4	5	6	7	8	9	10	11	12
GA	[5]		x	x						x	x				
	[46]		x					x				x	x		
	[10]		x	x								x			
	[32]	x								x					
	[37]	x									x				
	[40]		x							x	x	x	x		
	[67]		x								x	x			
	[68]	x								x					
	[69]	x											x		
	[70]										x		x		
	[71]		x							x					
	[54]		x	x	x						x				
	[72]		x					x		x	x				
	[73]		x								x	x			
	[74]		x							x	x				
ACO	[75]		x					x		x					x
	[39]		x							x		x			x
	[43]	x												x	
	[44]		x					x		x					x
	[45]		x							x		x			x
	[76]	x										x			
	[77]		x							x		x			
	[78]		x									x		x	
	[79]		x			x		x	x						
	[80]		x							x		x			x
PSO	[81]		x								x			x	
	[82]		x							x	x				
SA	[83]		x								x		x		
	[84]		x					x			x				
GSAA	[7]		x					x			x				
	[85]		x					x	x						
PN	[36]	x											x		
	[86]	x											x		
	[87]		x								x	x	x	x	
	[88]	x								x					
MA	[89]		x				x							x	
IA	[90]	x									x				
	[91]	x										x			
NN	[92]	x											x		
TS	[93]		x								x	x	x	x	
O	[94]	x										x			
	[95]		x							x	x				
	[96]	x										x			
	[97]	x								x					

GA genetic algorithm, *ACO* ant colony optimisation, *PSO* particle swarm optimisation, *SA* simulated annealing, *GSAA* combination GA and SA, *PN* Petri net, *MA* memetic algorithm, *IA* immune algorithm, *NN* neural network, *TS* tabu search, *O* other soft computing, *SO* single objective, *MO* multi-objective, *1* minimise tool change, *2* minimise assembly direction change, *3* minimise assembly complexity, *4* minimise assembly tool travel distance, *5* minimise assembly cost, *6* minimise assembly time, *7* minimise cycle time, *8* maximise workload smoothness, *9* minimise number of workstation, *10* maximise line efficiency, *11* minimise idle time, *12* maximise utilisation

4.1 Genetic algorithm

GA is inspired by evolutionary processes that based on natural evolution. It was introduced by John Holland in 1975. This technique imitates the biological evolution theory, where by the concept of '*survival for the fittest*' exists. GA provides a method of searching which does not need to explore every possible solution in the feasible region to obtain a good result [98].

In ASP and ALB optimisation, 35 out of 81 cited researches used this algorithm to find optimum assembly sequence. Researchers like [15, 47] used GA in ASP problem because it can generate optimum or near optimum solution faster than exact algorithms. In GA, the number of considered solution is reduced compared with exact algorithms. Researchers also prefer to use GA in ASP and ALB because it can handle complex and multiple constraints problems well [21]. Other researchers like [32, 73] were influenced by the success of GA in solving a wide variety of problems.

Although there are numerous papers that used this algorithm, GA in an original and basic form is unsuitable to be directly used to solve and optimise ASP and ALB problems. The first reason is the original binary strings in chromosomes are less suitable for complex combinatorial problem such as ASP [99]. The second reason is regarding the feasibility of chromosome in handling assembly precedence constraint. The GA in basic form tends to generate infeasible offspring that violates precedence constraint because of crossover and mutation operators [3]. To handle this constraint, researchers used different approaches like penalty and repair strategy.

In [28, 48, 71], the penalty approach were used to handle precedence constraint. A penalty is given to chromosomes that are infeasible due to violation of precedence constraints, resulting in reduced fitness. Therefore, the chance of infeasible chromosome to be selected in next generation is reduced. Besides that, repair strategy is used in [15, 100] to handle precedence constraint. In this approach, the infeasible chromosome is repaired and transformed into feasible chromosome using an additional step in GA. Other than that, topo - logical sort concept that originated from graph theory is also applied in handling precedence constraint as used in [74].

Even though GA has been successfully implemented in ASP and ALB, researchers have highlighted some issues regarding this algorithm. The main issue is that standard GA is susceptible to early convergence [61, 62]. Besides that, another common issue is about high computational time [101]. Yu and Yin [73] proposed an adaptive GA to solve premature convergence and high computational time issues. In adaptive GA, dynamic probabilities for crossover and mutation operators are introduced to vary computational time and selection rate. Another work to reduce computational time is performed by [67] by introducing dynamic partitioning (DPa) in chromosome. DPa modifies chromosome structures by defining frozen and unfrozen task allocation in workstation for ALB. The task allocation is only made for unfrozen task, whereby the frozen task will remain unchanged as in previous generation. Therefore, the computational time of this problem is reduced because of lesser length of active chromosome. Besides improving basic GA operators, researchers have combined GA with other soft computing algorithms to improve its performance. In general, combination of GA with simulated annealing [61, 62], tabu search [29] and ant colony optimisation [65] has resulted in better performance compared with the original GA.

4.2 Ant colony optimisation

ACO was introduced by Marco Dorigo in 1992. It is inspired by the pheromone trail laying behaviour of real ant colonies. In ACO, a set of agents called artificial ants search for good solutions to a given optimisation problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The artificial ants incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a pheromone model, that is, a set of parameters associated with graph components (either nodes or edges) whose values are modified at runtime by the ants.

ACO has attracted 13 cited papers in ASP and ALB in the past ten years due to various reasons. Shuang et al. [59] and Wang et al. [57] used this algorithm to overcome a shortcoming of GA that highly depends on initial chromosomes. Besides that, the success of this algorithm to solve popular discrete problems such as Travelling Salesman Problem, machine scheduling problem and Vehicle Routing Problem also have inspired researchers to use ACO in ASP and ALB [45, 77]. Another reason of ACO implementation is that ASP and ALB problem can be directly been represented by a completed graph as in ACO [57].

In original ACO, one of the common drawback were that stressed by researchers is regarding positive feedback system that only accumulates good solutions. In original ACO, the better the solution, the greater amount of pheromone will be deposited. However, the pheromone trail for all paths is set to be evaporated when it generates a bad solution. According to [59], over emphasis of this rule will cause premature convergence. Meanwhile, [77] has proposed to re-evaluate the unfit solutions, because they might just be a few iterations away from global optimum.

The main focus of researchers to improve ACO algorithm is solving premature convergence. Zhang et al. [77] introduce a summation rule to replace the original pheromone '*drop and evaporate*' updating rule. In pheromone summation updating rule, the best trail is determined by summation of total pheromone that dropped without considering evaporation factor. Meanwhile, [59] adopted particle swarm position updating approach to overcome premature convergence in ACO. The hybridisation of ACO and particle swarm not only solves the premature convergence in ACO but also reduces computational time compared with original ACO.

4.3 Particle swarm optimisation

PSO, originated by Kennedy and Eberhart in 1995 [102]. It is inspired by social behaviour of bird flocking or fish schooling. The PSO is quite similar to GA, in which the system is initialised with a population of random solutions. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. The potential solutions, called particles, fly through the problem space by following the current optimum particles [103]. In ASP and ALB, only seven cited papers applied this algorithm. From this number, six papers used PSO to solve multi-objective problem, but only one paper used Pareto optimal approach to deal with multi-objective optimisation. Most of the researchers used traditional weighted approach to solve multi-objective optimisation.

The PSO is a relatively new algorithm compare with GA and ACO. Not many papers that applied PSO to ASP and ALB have been published. This has motivated researchers [8, 25, 82] to apply PSO to ASP and ALB optimisation. Besides that, PSO is a simple algorithm because it only uses a single velocity formula to evolve [60]. Therefore, PSO algorithm is easy to implement and requires less computational resources compared with GA.

However, similarly with GA, the original PSO is not suitable to be directly applied to ASP and ALB problems. Besides the precedence constraint issue, the original PSO is designed for continuous problem, where the solution is in real-value space while ASP and ALB solutions reside in discrete integer space [8, 22]. Another important issue of original PSO is that it is easily trapped in local optima [22]. To solve this problem, [25] introduced a new mechanism of updating velocity by using one of two formulas randomly instead of single formula. Meanwhile, [22] introduced a chaotic operator to diversify the updated particle position, which finally help to reduce premature convergence.

4.4 Other methods

Besides the three main algorithms above, the researchers in ASP and ALB also use other soft computing methods such as simulated annealing [61, 83, 84], Petri net [86–88], memetic algorithm [20, 63, 90], immune algorithm [23, 89, 91], neural network [92] and tabu search [93].

Other than that, some researchers also combined a few soft computing methods to solve the ASP and ALB. Qin et al. [7] Li and Shan [62] and Lin et al. [85] were combined GA with simulated annealing method and called them genetic simulated annealing algorithm. Shan et al. [64] combined PSO and simulated annealing to solve multi-objective ASP problem using Pareto approach. The other algorithm combination that found to optimise ASP and ALB problem are GA and tabu search [29] and GA, ACO and simulated annealing [65].

5 Discussions and research potentials

This paper studied research in ASP and ALB that used soft computing approaches for the past 10 years. From this study, the previous research patterns and trends were identified. Figure 7 shows the number of published ASP and ALB papers that used soft computing methods between 2000 until 2010. The number of published papers in ALB was significantly increased from 2006. Meanwhile, the ASP research papers show increment from 2009. This figure concludes that, although the research in ASP and ALB were started in earlier time, but it had been given special attention by researchers between 2–5 years ago. This trend is predicted to be maintained in the near future due to growth in computational technique.

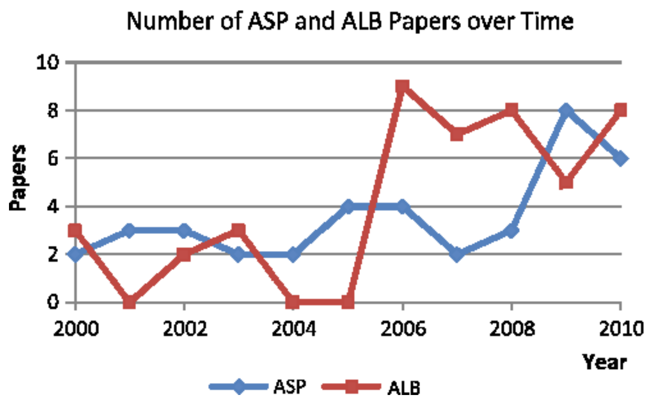


Fig. 7 Number of published papers in ASP and ALB for 2000- 2010

Meanwhile, Fig. 8 presents the trend of single- and multi-objective usage in ASP and ALB optimisation for 2000 until 2010. The trend shows research papers that used single objective were fluctuated from 2000 to 2010. Meanwhile, similar trend was also found in number of papers that used multi-objective optimisation for the first 5 years. However, this trend was changed for the second half of this period. The number of papers that used multi-objective optimisation was started to grow from 2006 until 2010. The multi-objective optimisation was attracted many researchers because of complexity of the problem and closer to the real assembly application.

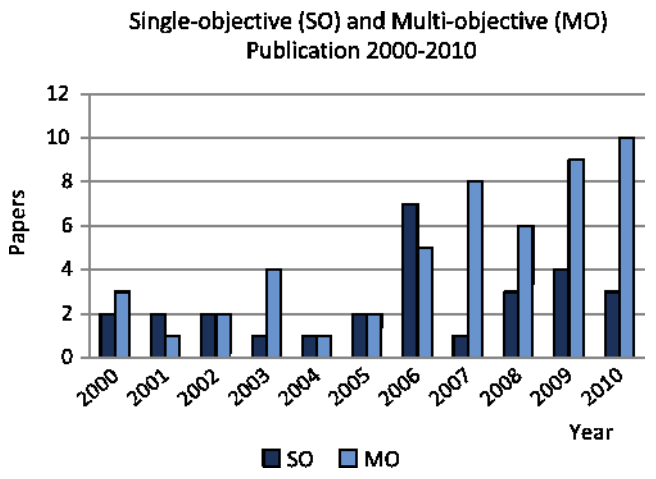


Fig. 8 Number of papers uses single and multi-objective

In terms of optimisation algorithms usage, application of GA in ASP and ALB papers between 2005 and 2010 is quite stable with an average of three papers per year (Fig. 9). For the same period, the ACO usage in the cited research papers fluctuates. Meanwhile, the PSO algorithm was first implemented in ASP and ALB research in 2009. The number of papers that applied PSO algorithm had shown rapid progress with two papers in 2009 and five papers in 2010. In 2010, papers using PSO in ASP and ALB optimisation was outnumbered papers that employed GA and ACO. If the current trend persists, it is possible for PSO to be widely used in this area as GA.

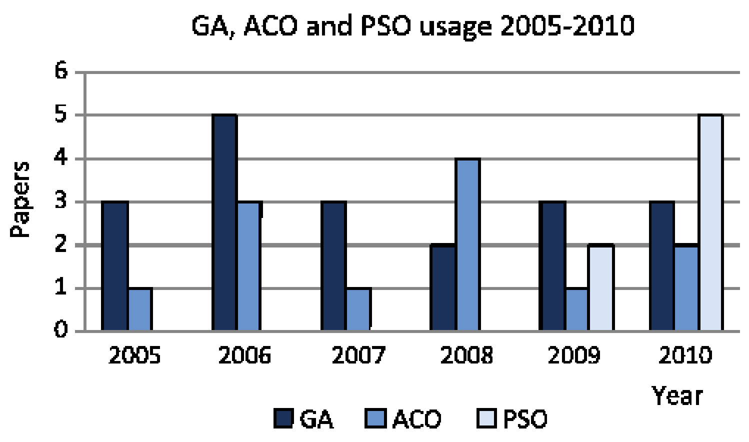


Fig. 9 Number of papers uses GA, ACO and PSO for 2005–2010

A number of issues had also been raised by researches regarding the ASP and ALB optimisation. One of the issues is related with high computational time for ASP and ALB. Researchers like [21, 55, 104, 105] agreed that the existing algorithms may inadequate to solve larger ASP and ALB problems due to computational limitation.

The second issue highlighted by researchers is about tedious data entry procedure into computer programme. The current approach requires the researchers to identify and key in a set of data such as precedence, geometrical character, direction etc. This process consumes a lot of time as stated by [66]; '*The man–computer interaction for constraint detection is the most manpower consuming process*'. To simplify the process, research on data extraction from computer-aided design (CAD) model is highly recommended by [20, 22, 66].

Besides that, researchers also made an argument on assumptions in ALB. The first assumption stated that all workstations have similar capability, therefore any assembly task can be assigned to any workstation. The idea of all workstations having similar facilities cannot be accepted because it did not imply the real situation [10]. Meanwhile, [95] disagreed on the assumption where most of ALB problem is discussed as a deterministic problem, but in reality, the processing times are rarely deterministic.

Researchers in assembly optimisation have contributed in various problems and applications. However, there are still a few unfulfilled potential and gaps. ALB research started with simple line balancing problem with basic precedence constraint. This field has progressed to a complex problem with other assembly constraints. In computational experiment research like ALB, the computational model is nearer to actual situation when less assumption is used. However, the problem will become more complicated and requires higher computational cost. The suggestion to facilitate particular assembly task into a particular workstation with facilities constraint had been discussed in earlier research, but it has not been implemented yet.

In ASP and ALB problem, optimisation algorithm plays an important role since both problems are classified as NP-hard. Research on algorithm improvement is important to handle more complicated ASP and ALB problems with larger size, various constraints and objectives. Currently, the algorithms to optimise ASP and ALB problem are dominated by GA, ACO and PSO. The researchers are more interested to explore and improve these algorithms although many other potential algorithms are available. However, the algorithm improvement works mainly in focusing on solving premature convergence issues rather than high-computational time or algorithm complexity issues.

In the next few years, the algorithm usage is likely that remain lead by main algorithms (GA, ACO and PSO) with modification to reduce premature convergence. Although there are many recent papers that focus on solving this problem, the definitive answer is still unclear. In the near future, the trend for algorithm hybridisation is also predicted to be the focus of re-searchers. The current success of hybrid algorithm as presented in [62, 65, 85] has motivated researchers to give additional attention to this approach. Although the algorithm hybridisation approach has been started earlier, the number of papers that use this approach have significantly increased since 2008. However, there are still plenty of opportunities in the algorithm hybridisation because many potential algorithm combinations are not tested yet.

On the other hand, further research on automation and integration of assembly optimisation also have a potential. At the moment, research on data extraction from CAD model are only being implemented in DFA level but not widely used in ASP and ALB [20, 22]. Meanwhile, integration of assembly optimisation consumes larger manpower to enter the data. In the same time, integration of assembly optimisation is also considered as a bridge to enable flows of extracted data from DFA level to ASP and ALB optimisation. Therefore, automation and integration of assembly optimisation are mutually dependent on enhancing each other.

6 Conclusions

This paper surveyed the ASP and ALB research that used soft computing methods for the past 10 years. The current research trend shows that ASP and ALB are progressing to a more complicated problem by increment in the number of papers that works on multi-objective optimisation. Besides that, growth in usage of relatively new algorithm like PSO shows that the researchers tend to explore and develop algorithm which manage to handle more complex problems.

In the future, one of the main challenges in ASP and ALB research is how to simplify and shorten assembly optimisation processes throughout different levels (Fig. 1). This is an important issue especially for manufacturers to be able to compete in the global market with shorter product life cycle. Another challenge in this field is how to make the ASP and ALB problem model closer to the actual situation in industry. This challenge is important to acquire accurate results from computational experiments. The challenge to reduce computational cost is another future research direction, since the ASP and ALB problems are getting more complicated. Therefore, it can be concluded that, although many works had been published, research in ASP and ALB optimisation still have a long way to go.

References

1. Padron M, de los AIM, Resto P, Mejia HP (2009) A methodology for cost-oriented assembly line balancing problems. J Manuf Technol Manag 20(8):1147–1165
2. Pan C (2005) Integrating CAD files and automatic assembly sequence planning. Ph.D. thesis, Iowa State University
3. Marian R (2003) Optimisation of assembly sequences using genetic algorithm. Ph.D. thesis, University of South Australia
4. Corallo A, Margherita M, Pascali G (2010) Digital mockup to optimize the assembly of a ship fuel system. J Model Simul Syst 1(1):4–12
5. Chen RS, Lu KY, Yu SC (2002) A hybrid genetic algorithm approach on multi-objective of assembly planning problem. Eng Appl Artif Intell 15(5):447–457
6. Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168(3):666–693
7. Qin YF, Xu ZG (2007) Assembly process planning using a multi-objective optimization method. In: Proceedings of the 2007 IEEE international conference on mechatronics and automation, ICMA 2007, 4303610, pp 593–598
8. Lv H, Lu C (2010) An assembly sequence planning approach with a discrete particle swarm optimization algorithm. Int J Adv Manuf Technol 50(5–8):761
9. Lendak I, Erdeljan A, Capko D, Vukmirovic S (2010) Algorithms in electric power system one-line diagram creation: the soft computing approach. In: IEEE international conference on systems, man and cybernetics, pp 2867–2873
10. Tseng HE, Tang CE (2006) A sequential consideration for assembly sequence planning and assembly line balancing using the connector concept. Int J Prod Res 44(1):97–116
11. Sinanoglu C, Boklu HR(2005) An assembly sequence-planning system for mechanical parts using neural network. Assem Autom 25(1):38–52
12. Chen G, Zhou J, Cai W, Lai X, Lin Z, Menassa R (2006) A framework for an automotive body assembly process design system. CAD Comput Aided Des 38(5):531–539
13. Chen WC, Tai PH, Deng WJ, Hsieh LF (2008) A three-stage integrated approach for assembly sequence planning using neural networks. Expert Syst Appl 34(3):1777–1786
14. Mitrovic-Minic S, Krishnamurti V (2006) The multiple TSP with time windows: Vehicle bounds based on precedence graphs. Oper Res Lett 34(1):111–120
15. DeLit P, Latinne P, Rekiek B, Delchambre A (2001) Assembly planning with an ordering genetic algorithm. Int J Prod Res 39(16):3623–3640
16. Choi YK, Lee DM, Cho YB (2009) An approach to multi-criteria assembly sequence planning using genetic algorithms. Int J Adv Manuf Technol 42(1–2):180–188

17. Tseng YJ, Chen JY, Huang FY (2010) A particle swarm optimisation algorithm for multi-plant assembly sequence planning with integrated assembly sequence planning and plant assignment. Int J Prod Res 48(10):2765–2791
18. Senin N, Groppetti R, Wallace DR (2000) Concurrent assembly planning with genetic algorithms. Robot Comput Integr Manuf 16(1):65–72
19. Wang WP, Tseng HE (2009) Complexity estimation for genetic assembly sequence planning. J Chin Inst Ind Eng 26(1):44–52
20. Chang CC, Tseng HE, Meng LP (2009) Artificial immune systems for assembly sequence planning exploration. Eng Appl Artif Intell 22(8):1218–1232
21. Chen and Liu Chen SF, Liu YJ (2001) An adaptive genetic assembly-sequence planner. Int J Comput Integr Manuf 14(5):489–500
22. Wang Y, Liu JH (2010) Chaotic particle swarm optimization for assembly sequence planning. Robot Comput Integr Manuf 26(2):212–222
23. Cao PB, Xiao RB (2007) Assembly planning using a novel immune approach. Int J Adv Manuf Technol 31(7–8):770– 782
24. Gao L, Qian W, Li X, Wang J (2010) Application of memetic algorithm in assembly sequence planning. Int J Adv Manuf Technol 49(9–12):1175–1184
25. Yu H, Yu J, Zhang W (2009) An particle swarm optimization approach for assembly sequence planning. Appl Mech Mater 1228:16–19
26. Lazzerini B, Marcelloni F (2000) Genetic algorithm for generating optimal assembly plans. Artif Intell Eng 14(4):319– 329
27. Guan Q, Liu JH, Zhong YF (2002) A concurrent hierarchical evolution approach to assembly process planning. Int J Prod Res 40(14):3357–3374
28. Lu C, Wong YS, Fuh JYH (2005) An enhanced assembly planning approach using a multi-objective genetic algorithm. Proc Inst Mech Eng, B J Eng Manuf 220(2):255–272
29. Li JR, Khoo LP, Tor SB (2003) A Tabu-enhanced genetic algorithm approach for assembly process planning. J Intell Manuf 14(2):197–208
30. Marian RM, Luong LHS, Abhary K (2006) A genetic algorithm for the optimisation of assembly sequences. Comp Ind Eng 50(4):503–527.
31. Xing Y, Wang Y, Zhao X (2010) A particle swarm algorithm for assembly sequence planning. Adv Mat Res 3243:97–101
32. Gu L, Hennequin S, Sava A, Xie X (2007) Assembly line balancing problems solved by estimation of distribution. In: Proceedings of the 3rd IEEE international conference on automation science and engineering, IEEE CASE 2007, pp 123–127
33. Baybars I (1986) Survey of exact algorithms for the simple assembly line balancing problem. Manage Sci 32(8):909–932
34. Boysen N, Fliedner M, Scholl (2007) A classification of assembly line balancing problems. Eur J Oper Res 183(2):674– 693
35. Betancourt L (2007) ASALBP: the alternative subgraphs assembly line balancing problem. Formalization and resolution procedures. Ph.D. thesis, Technical University of Catalonia
36. Kilincci O, Bayhan GM (2006) A Petri net approach for simple assembly line balancing problems. Int J Adv Manuf Technol 30(11–12):1165–1173
37. Tasan SO, Tunali S (2006) Improving the genetic algorithms performance in simple assembly line balancing. In: Lecture notes in computer science, LNCS vol 3984, pp 78–87
38. Nof SY, WE W, Warnecke H (1997) Industrial assembly. Chapman & Hall, London

39. Chica M, Cordon O, Damas S, Bautista J (2010) Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. Inf Sci 180(18):3465–3487
40. Ponnambalam SG, Aravindan P, Naidu GM (2000) Multi-objective genetic algorithm for solving assembly line balancing problem. Int J Adv Manuf Technol 16(5):341–352
41. Capacho L, Pastor R (2008) Asalbp: The alternative sub-graphs assembly line balancing problem. Int J Prod Res 46(13):3503–3516
42. Whitney DE (2004) Mechanical assemblies: their design, manufacture and role in product development. Oxford University Press, New York
43. Zhang Z, Cheng W, Song L, Yu Q (2009) An ant-based algorithm for balancing assembly lines in a mass customization environment. In: International workshop on intelligent systems and applications, ISA 2009, 5072706
44. McMullen PR, Tarasewich P (2006) Multi-objective assembly line balancing via a modified ant colony optimization technique. Int J Prod Res 44(1):27–42
45. Chica M, Cordon O, Damas S, Pereira J, Bautista J (2008) Incorporating preferences to a multi-objective ant colony algorithm for time and space assembly line balancing. In: Lecture notes in computer science. LNCS vol 5217, pp 331– 338
46. Moon DS, Park BY (2007) Genetic algorithms for concurrent assembly planning. In: Regional computational conference, pp 214–219
47. Smith SSF, Liu YJ (2001) The application of multi-level genetic algorithms in assembly planning. J Ind Technol 17(4):1
48. Smith GC, Smith SSF (2002) An enhanced genetic algorithm for automated assembly planning. Robot Comput-Integr Manuf 18(5–6):355–364
49. Smith SSF (2004) Using multiple genetic operators to reduce premature convergence in genetic assembly planning. Comput Ind 54(1):35–49
50. Tseng HE, Li JD, Chang YH (2004) Connector-based approach to assembly planning using a genetic algorithm. Int J Prod Res 42(11):2243–2261
51. Bai YW, Chen ZN, Bin HZ, Hun J (2005) An effective integration approach toward assembly sequence planning and evaluation. Int J Adv Manuf Technol 27(1–2):96–105
52. Udeshi T, Tsui K (2005) Assembly sequence planning for automated micro assembly. In: IEEE International symposium on assembly and task planning 2005, vol 2005, pp 98– 105
53. Pan C, Smith S, Smith G (2006) Automatic assembly sequence planning from STEP CAD files. Int J Comput Integr Manuf 19(8):775–783
54. Tseng HE, Chen MH, Chang CC, Wang WP (2008) Hybrid evolutionary multi-objective algorithms for integrating assembly sequence planning and assembly line balancing. Int J Prod Res 46(21):5951–5977
55. Tseng YJ, Chen JY, Huang FY (2010) A multi-plant assembly sequence planning model with integrated assembly sequence planning and plant assignment using GA. Int J Adv Manuf Technol 48(1–4):333–345
56. Zhou W, Zheng J, Yan J, Wang J (2010) A novel hybrid algorithm for assembly sequence planning combining bacterial chemotaxis with genetic algorithm. Int J Adv Manuf Technol 52(5–8):715–724
57. Wang JF, Liu JH, Zhong YF (2005) A novel ant colony algorithm for assembly sequence planning. Int J Adv Manuf Technol 25(11–12):1137–1143

58. Zhang J, Sun J, He Q (2010) An approach to assembly sequence planning using ant colony optimization. In: Proceedings of 2010 international conference on intelligent control and information processing, ICICIP 2010, vol part 2, pp 230– 233
59. Shuang B, Chen J, Li Z (2008) Microrobot based micro-assembly sequence planning with hybrid ant colony algorithm. Int J Adv Manuf Technol 38(11–12):1227–1235
60. Lv HG, Lu C, Zha J (2010) A hybrid DPSO-SA approach to assembly sequence planning. In: IEEE international conference on mechatronics and automation, ICMA 2010, 5589203, pp 1998–2003
61. Shan H, Li S, Gong D, Lou P (2006) Genetic simulated annealing algorithm-based assembly sequence planning. In: IET conference publications, vol 524, pp 1573–1579
62. Li SX, Shan HB (2008) GSSA and ACO for assembly sequence planning: a comparative study. In: Proceedings of the IEEE international conference on automation and logistics, ICAL 2008, pp 1270–1275
63. Tseng HE, Wang WP, Shih HY (2007) Using memetic algorithms with guided local search to solve assembly sequence planning. Expert Syst Appl 33(2):451–467
64. Shan H, Zhou S, Sun Z (2009) Research on assembly sequence planning based on genetic simulated annealing algorithm and ant colony optimization algorithm. Assem Autom 29(3):249–256
65. Hui C, Yuan L, Kai-Fu Z (2009) Efficient method of assembly sequence planning based on GAAA and optimizing by assembly path feedback for complex product. Int J Adv Manuf Technol 42(11–12):1187–1204
66. Su Q (2009) A hierarchical approach on assembly sequence planning and optimal sequences analyzing. Robot ComputIntegr Manuf 25(1):224–234
67. Sabuncuoglu I, Erel E, Tanyer M (2000) Assembly line balancing using genetic algorithms. J Intell Manuf 11(3):295– 310
68. Zhao ZY, Souza RD (2000) Genetic production line-balancing for the hard disk drive industry. Int J Adv Manuf Technol 16(4):297–302
69. Goncalves JF, Almeida JRD (2002) A hybrid genetic algorithm for assembly line balancing. J Heuristics 8(6): 629–642
70. Baykasoglu A (2006) Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. J Intell Manuf 17(2):217–232
71. Zhang R, Chen D, Wang Y, Yang Z, Wang X (2007) Study on line balancing problem based on improved genetic algorithms. In: International conference on wireless communications, networking and mobile computing, WiCOM 2007, 4340283, pp 2033–2036
72. Zhang W, Gen M, Lin L (2008) A multiobjective genetic algorithm for assembly line balancing problem with worker allocation. In: IEEE international conference on systems, man and cybernetics, 4811759, pp 3026–3033
73. Yu J, Yin Y (2010) Assembly line balancing based on an adaptive genetic algorithm. Int J Adv Manuf Technol 48(1–4):347–354
74. Zacharia PT, Nearchou AC (2010) Multi-objective fuzzy assembly line balancing using genetic algorithms. J Intell Manuf 1–13. doi:10.1007/s10845-010-0400-9
75. McMullen PR, Tarasewich P (2003) Using ant techniques to solve the assembly line balancing problem. IIE Trans (Institute of Industrial Engineers) 35(7):605–617
76. Blum C, Bautista J, Pereira J (2006) Beam-ACO applied to assembly line balancing. In: Lecture note in computer science (LNCS), vol 4150, pp 96–107

77. Zhang, ZQ, Cheng WM, Tang LS, Zhong B (2008) Ant algorithm with summation rules for assembly line balancing problem. In: International conference on management science and engineering, ICMSE'07 (14th), 4421875, pp 369– 374
78. Blum C, Bautista J, Pereira J (2008) An extended beamACO approach to the time and space constrained simple assembly line balancing problem. In: Lecture notes in computer science (LNCS), vol 4972, pp 85–96
79. Zhang ZQ, W-M C, B Z, Wang JN (2007) Improved ant colony optimization for assembly line balancing problem. Comput-Integr Manuf CIMS 13(8):1632–1638
80. Chica M, Cordon O, Damas S, Bautista J (2011) Including different kinds of preferences in a multi-objective ant algorithm for time and space assembly line balancing on different Nissan scenarios. Expert Syst Appl 38(1): 709–720
81. Lu JS, Jiang LL, Li XL (2009) Hybrid particle swarm optimization algorithm for assembly line balancing problem-In: Proceedings 2009 IEEE 16th International conference on industrial engineering and engineering management, pp 979–983
82. Nearchou AC (2010) Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. Int J Prod Econ 12(2):242
83. Ozcan U, Toklu B (2009) A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems. J Intell Manuf 20(1):123–136
84. Cakir B, Altiparmak F, Dengiz B (2010) Multi-objective optimization of a stochastic assembly line balancing: a hybrid simulated annealing algorithm. Comput Indu Eng 60(3): 376
85. Lin YY, Che ZH, Chiang TA, Che ZG, Chiang CJ (2009) A bi-objective model for concurrent planning of supplier selection and assembly sequence planning. In: Smith S (ed) Global perspective for competitive enterprise, economy and ecology. Springer, London, pp 573–580
86. Lapierre SD, Ruiz A, Soriano P (2006) Balancing assembly lines with tabu search. Eur J Oper Res 168(3):826– 837
87. Suwannarongsri S, Limnararat S,Puangdownreong D (2007) A new hybrid intelligent method for assembly line balancing. In: IEEE international conference on industrial engineering and engineering management, 4419365, pp 1115– 1119
88. Kilincci O (2010) A Petri net-based heuristic for simple assembly line balancing problem of type 2. Int J Adv Manuf Technol 46(1–4):329–338
89. Liu SB, Ong HL, Huang HC (2003) Two bi-directional heuristics for the assembly line type II problem. Int J Adv Manuf Technol 22(9–10):656–661
90. Khoo LP, Alisantoso D (2003) Line balancing of PCB assembly line using immune algorithms. Eng Comput 19(2– 3):92–100
91. Andrés C, Miralles C, Pastor R (2008) Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. Eur J Oper Res 187(3):1212–1223
92. Kilincci O, Bayhan GM (2008) A P-invariant-based algorithm for simple assembly line balancing problem of type-1. Int J Adv Manuf Technol 37(3–4):400–409
93. Suwannarongsri S, Puangdownreong D (2008) Multi-objective assembly line balancing via adaptive tabu search method with partial random permutation technique. In: IEEE international conference on industrial engineering and engineering management, IEEM 2008, 4737881, pp 312– 316
94. Tijo S, Numar R (2008) Heuristic programming for assembly line balancing. In: Regional conference of mathematical programming. Seoul, pp 226–230

95. Nearchou AC (2008) Multi-objective balancing of assembly lines by population heuristics. Int J Prod Res 46(8):2275– 2297
96. Yeh DH, Kao HH (2009) A new bidirectional heuristic for the assembly line balancing problem. Comput Indu Eng 57(4):1155–1160
97. Martino L, Pastor R (2010) Heuristic procedures for solving the general assembly line balancing problem with setups. Int J Prod Res 48(6):1787–1804
98. Goldberd D (2007) Genetic algorithms: the design of innovation. Springer, Berlin
99. Tasan S, Tunali S (2008) A review of the current applications of genetic algorithms in assembly line balancing. J Intell Manuf 19(1):49–69
100. Gao J, Sun L, Wang L, Gen M (2009) An efficient approach for type II robotic assembly line balancing problems. Comput Ind Eng 56(3):1065–1080
101. Moon I, Logendran R, Lee J (2009) Integrated assembly line balancing with resource restrictions. Int J Prod Res 47(19):5525–5541
102. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE international conference on neural networks, vol 4, pp 1942–1948
103. Sinavandam SN, Deepa SN (2008) Introduction to genetic algorithms. Springer, Berlin
104. Toksari MD, Isleyen SK, Guner E, Baykoc OF (2010) Assembly line balancing problem with deterioration tasks and learning effect. Expert Syst Appl 37(2):1223–1228
105. Capacho L, Pastor R (2006) The ASALB problem with processing alternatives involving different tasks: definition, formalization and resolution. In: Lecture notes in computer science, vol 3982 LNCS, pp 554–563.