Engineering Journal issn: 0125-8281

Article

Solving Many-Objective Car Sequencing Problems on Two-Sided Assembly Lines Using an Adaptive Differential Evolutionary Algorithm

Parames Chutima[1,2,3,a,*] **and Trirat Kirdphoksap**[1,b]

1 Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
2 Regional Centre for Manufacturing Systems Engineering, Chulalongkorn University, Bangkok 10330, Thailand
3 The Royal Society of Thailand
E-mail: [a]cparames@chula.ac.th (Corresponding author), [b]trirat.ki@gmail.com

Abstract. The car sequencing problem (CSP) is addressed in this paper. The original environment of the CSP is modified to reflect real practices in the automotive industry by replacing the use of single-sided straight assembly lines with two-sided assembly lines. As a result, the problem becomes more complex caused by many additional constraints to be considered. Six objectives (i.e. many objectives) are optimised simultaneously including minimising the number of colour changes, minimising utility work, minimising total idle time, minimising the total number of ratio constraint violations and minimising total production rate variation. The algorithm namely adaptive multi-objective evolutionary algorithm based on decomposition hybridised with differential evolution algorithm (AMOEA/D-DE) is developed to tackle this problem. The performances in the Pareto sense of AMOEA/D-DE are compared with COIN-E, MODE, MODE/D and MOEA/D. The results indicate that AMOEA/D-DE outperforms the others in terms of convergence-related metrics.

Keywords: Car sequencing, two-sided assembly line, many objectives, evolutionary algorithm, decomposition, differential evolutionary algorithm.

ENGINEERING JOURNAL Volume 23 Issue 4
Received 24 September 2018
Accepted 17 May 2019
Published 8 August 2019
Online at http://www.engj.org/
DOI:10.4186/ej.2019.23.4.121

DOI:10.4186/ej.2019.23.4.121

1. Introduction

A car sequencing problem (CSP) has attracted much attention from car manufacturers and researchers for many decades. The CSP practised by automotive manufacturers was first elaborated in a scientific manner by Parrello, et al. [1]. The classical version of the CSP consists in establishing the daily launching order of a set of cars with different types and options (e.g. sunroof, a navigation system, camera, etc.) to be produced on the mixed-model final assembly line to achieve a predefined goal without violating capacity constraints. The CSP can be broadly viewed as a subset of the job shop scheduling problem.

RENAULT, the renowned French car manufacturer, articulated in the 2005's ROADEF challenge that the CSP in a modern car factory must consider three major shops rather than taking into account just the constraints and objectives related to the assembly shop like in the classical CSP. These three shops ordered sequentially include the body fabrication shop where the chassis of the cars are produced, the paint shop where the cars are painted by workers and robots, and the assembly shop where different options are installed in the cars [2]. Moreover, most car manufacturers put great efforts in the order-taking process to please customers by shortening production lead time and offering a large variety of options.

The CSP is a strong NP-hard problem. The CSP was used to consider as a constraint satisfaction problem in which different priorities could be given to the constraints. Recently, Kis [3] demonstrated that the problem was rather well-match with a combinatorial optimisation problem, even the ratio-constraint alone was taken into account. Adding more constraints involving in the body fabrication shop and paint shop into the classical CSP enhances the complexity of the problem dramatically. The brute force approaches of constraint programming or integer programming may not be effective enough to solve the problem since the software could reach its limit when facing practical-sized problems [4].

Gagné, et al. [5] argue that the production equipment such as robots installed in the body fabrication shop is flexible enough to adjust its output rate to match with the demands given by the other two manufacturing shops. As a result, to reduce the complexity of the CSP, the constraints from the body fabrication shop could be neglected since their impacts on the daily car sequencing plan are not significant. In other words, the optimal solution obtained from solving the CSP in which taken into consideration only the objectives and constraints of the paint shop and assembly shop is sufficient to reflect the operations of the car production facility as a whole [6].

Most research on the CSP normally attempts to optimise only a single objective. However, in reality, various objectives are applicable and they should be optimised simultaneously. Moreover, the layout of the assembly line used in the traditional model is simplified by adopting the straight-shaped form. The assumption as such is unpractical because the car size is big which normally needs two workers to work on the opposite sides of the workstation to assemble different components on the same car to shorten line length and improve operations efficiency. Moreover, task times and precedence relationships of various car models are disregarded in the traditional model. Consequently, the capacity problem involved in the necessity of utility workers employed to help clear up all unfinished tasks leftover by regular workers is ignored.

Recently, a more practical platform of the CSP was proposed by Chutima and Olarnviwatchai [7]. The major modifications given to the traditional model included using the two-sided assembly line (2SAL) to replace the straight-shaped assembly line (or the one-sided assembly line, 1SAL), explicitly considering precedence relationships and task times of different car models, and optimising multi-objectives simultaneously. This modified model was called "a car sequencing problem on the two-sided assembly line (CSP2SAL)".

The contributions of this paper are as follows. First, the environment of the CSP2SAL as presented in [7] is adopted and the problem with many (six) objectives is optimised simultaneously. Note that the term "many" in this context means more than three [8]. Second, since the size of the objective space is gigantic in many objective optimisation problems (MaOPs), the conventional evolutionary algorithms that bias their search directions based on non-dominated solutions on the Pareto fronts may not be a useful approach. Alternatively, the solution approach of multi-objective evolutionary optimization based on decomposition (MOEA/D) that partitions the MaOP into a set of single-objective subproblems and solve them simultaneously could be much more effective. Therefore, MOEA/D is adapted as a main structure of the solution technique. Third, the adaptive MOEA/D hybridised with the differential evolution algorithm (AMOEA/D-DE) is also proposed to effectively tackle the many-objective CSP2SAL (MaCSP2SAL) and prevents the premature convergence of the algorithm. To our best knowledge, no research paper has been attempted in this area before.

The remaining of this paper is organised as follows. A comprehensive review related to CSP and 2SAL is elaborated in Section 2. The distinct characteristics of the MaCSP2SAL are discussed in Section 3. Section 4 explains the proposed algorithm (AMOEA/D-DE) to solve the MaCSP2SAL, followed by Section 5 that shows the experimental design. The experimental results and concluding remarks are given in Sections 6 and 7, respectively.

2. Literature Review

2.1. CSP

The CSP has attracted attention from practitioners and researchers for many decades since it paves the way for the advanced studies of formulations and solution searching strategies. The CSP was first formally elaborated by Parrello, et al. [1]. Initially, the CSP was formulated through a constraint satisfaction model to find an optimal sequence of different car models manufactured along an assembly line without violating contiguity constraints, while satisfying their demands. However, in practice, more appropriated formulation of the CSP should be a combinatorial optimisation problem rather since some of their constraints could be violated, but with additional penalty cost [6, 9].

The CSP was proved to be in the class of NP-hard combinatorial optimisation problems [3, 10]. As a result, the exact solution to large-scaled CSPs is difficult to achieve. Solnon, et al. [11] presented a comprehensive review of the exact and heuristic approaches for solving the CSP. It was indicated that the approaches that could provide a relatively good solution fairly quickly were more practical and useful than those that achieved an optimal solution at the expense of irrational computational time. Hence, unsurprisingly, various approximated solution approaches such as heuristics or metaheuristics became dominated in the good solution searching process of the CSP.

Warwick and Tsang [9] classified the CSP into two types, i.e. solvable and unsolvable (no solution). In addition, previous research on the CSP was mainly emphasised on the solvable problem. However, the CSP should rather be considered as a partial constraint satisfaction problem in which constraints could be violated but at some predetermined cost. A generic genetic algorithm (GAcSP) in which the repair and hill climbing mechanisms were combined with the genetic algorithm (GA) to find the solutions of both solvable and unsolvable CSPs. Jaszkiewicz, et al. [12] described an adaptation of the genetic local search algorithm (GLS) in which a systematic approach to the construction of recombination operators, a heuristic for efficient initial solution creations, and a fast local search method was developed.

Smith, et al. [13] formulated the CSP using a nonlinear integer programming and proposed two heuristics, i.e. steepest descent and simulated annealing (SA), and Hopfield neural network to solve the problem approximately. Briant, et al. [14] tackled the CSP via SA. The dynamic probabilities taken into account the best success rate of each move so far were used to find predominant neighbours for the given instance.

Puchta and Gottlieb [15] compared three permutation-based local search algorithms which employed different acceptance criteria for moves. Two variants of threshold accepting approaches were superior to the greedy approach to solution quality and robustness. Prandtstetter and Raidl [2] proposed a new integer linear programming (ILP) formulation for small- and medium-sized CSP instances. For the large-sized problem, a general variable neighbourhood search (VNS) approach encompassed the large neighbourhoods examined using ILP technique were shown promising. Two local search approaches, i.e. very large neighbourhood search and very fast local search, were compared by Estellon, et al. [4]. They mentioned that sophisticated metaheuristics were impractical to solve the CSP. Ribeiro, et al. [16], Ribeiro, et al. [17] approximately solved the CSP using a set of heuristics based on the paradigms of the VNS and iterated local search (ILS) metaheuristics which were further enhanced with the intensification and diversification strategies. The optimised data structure for the effective implementation of heuristics was also proposed.

Gottlieb et al. [6] compared a few heuristics used in the CSP, i.e. greedy heuristics, local search and ant colony optimisation (ACO). It was revealed that the ACO outperformed the others in obtaining better solution quality for smaller time limits. Gagné, et al. [5] developed the ACO as a new and powerful solution engine which contained effective transition rules for solving a multi-objective CSP. Solnon [18] introduced an ACO working under two different pheromone structures, each of which aimed at learning for good car sequences or learning for critical cars. Since the operations of these pheromone structures were complementary to each other, their combination could significantly improve the algorithm speed.

Gavranović [19] developed the constructive greedy algorithm and VNS developed for the CSP with colours. Tabu search (TS) was then applied to improve the results initially obtained from the VNS. Cordeau,

et al. [20] developed an iterated TS heuristic which combined the classical TS with perturbation operators in order to avoid the local optima.

Zinflou, et al. [21] introduced three new crossover operators using with a GA to solve the CSP, i.e. interest based crossover (IBX), uniform interest crossover (UIX) and non-conflict position crossover (NCPX). Zinflou and Gagné [22] proposed an algorithm, which is a hybridisation of a GA and an artificial immune system, namely GISMOO, to tackle the multi-objective CSP in a Pareto sense. The Pareto solutions obtained from GISMOO dominated those of the non-dominated sorting genetic algorithm (NSGA II) and Pareto memetic strategy for multiple-objective optimisations (PMS^{MO}). Atiker et al. [23] proposed a heuristic to tackle the CSP under the scenario in which high priority ratio constraints are primary, and colour constraints are secondary.

2.2. 2SAL Sequencing

Prior to launching orders to the assembly line, the assembly sequence of parts and subassemblies must be determined. Bahubalendruni and Biswal [24] reviewed and discussed research articles on the assembly sequence generation published over the past four decades. Obviously, an appropriate assembly sequence results in minimal lead time and low cost of assembly [25].

The mixed-model assembly line sequencing problem has been one of the research topics that challenges researchers during the last decade. Obviously, most research was mainly conducted on one-sided straight-shaped assembly lines. This environment is applicable to small-sized products, e.g. printed circuit board, television, etc. where each worker is comfortable to work effectively with a workpiece and its assembly components. Since the size of cars is too big to manipulate by a single worker, two-sided assembly lines where two workers perform their assigned tasks cooperatively in a workstation tend to be a more effective alternative for doing assembly works.

Boysen et al. [26] give a detailed review of mixed-model sequencing algorithms and illustrated a number of potential performance measurements commonly used in literature, e.g. workload variation, part usage rate variation, amount of utility work, etc. In addition, most research in the early years attempted to optimise only a single objective such as the articles of Miltenburg, et al. [27] and Stiener and Yeomans [28].

During the last decade, the trend of the research has moved towards multi-objective optimisation of mixed-model sequencing problems which much more reflects the decision maker's point of view in practice. The challenging issue arises from several objectives to be optimised simultaneously are often conflict with one another; therefore, a compromised solution becomes implicitly acceptable. In addition, Pareto-based solution techniques are more appropriate than weighted sum techniques [29].

A new constraint occurred in the 2SAL, but not in the 1SAL, is sequence-dependent finish time of tasks. Since two workers have to work concurrently and cooperatively on the same workpiece in a mated station of 2SAL, the finish time of tasks done by one worker could affect the start time of tasks to be performed by the other. As a result, to find the optimal operations of the CSP2SAL, the launching sequence of mixed-model cars and the cooperation of workers in the same mated station must be contemplated carefully.

Since the CSP2SAL is NP-hard by nature, the evolutionary algorithm seems to be promising solution techniques. Several researchers employed this algorithm to solve the 2SAL sequencing problem including TS [30], Pareto stratum-niche cubicle genetic algorithm [31], SA [32], genetic algorithm [33], ant colony optimisation [34], memetic algorithm [35], multi-objective scatter search [36], coincidence algorithm [37], extended coincidence algorithm [7], etc.

3. Problem Definition

According to the proposal in the ROADEF'05 challenge on the CSP, the car production facility in industry consists of three main steps serially interconnected as the body fabrication shop, paint shop and assembly shop [2]. Car chassis, metal fabrication, bodywork and welding process are manufactured in the body fabrication shop. Various predefined colours are painted on the bodies of cars according to customer orders in the paint shop. In the assembly shop, engine, suspension parts, electrical parts, underbody parts as well as specific options, e.g. sunroof, alloy wheel, sports seat, satellite navigation, etc. are installed into the car.

Gagné, et al. [5] mentioned that manufacturing flexibility in the body fabrication shop stemming from the widespread use of programmable robotic equipment resulting in its daily production schedule could be easily adjusted to fit with the requirements imposed by the two remaining shops. Consequently, the shops

which must be taken into account in optimising the CSP are limited to the paint shop and the assembly shop only [6].

As mentioned earlier, the CSP should be considered as a combinatorial optimisation problem rather than a constraint satisfaction problem since in practice their constraints could be violated. When a constraint is violated, the number of occurrences in a particular type will be notified and accumulated. Cordeau, et al. [20] recommended that some types of violations happening in a certain production day J were affected by the actual production in the previous day ($J-1$) and the planned production in the next day ($J+1$). As a result, these three consecutive days must be considered all together to effectively assess the constraint violations.

Two constraints are normally emphasised in the CSP, i.e. ratio constraint (RC) and number of paint colour changes (NCC). The RC, a capacity related constraint, is the primitive constraint of the traditional CSP. The simplified version of this constraint is formalised by a ratio p/q meaning that for each particular option at most p cars could be installed with that option in any subsequence production segment of q cars. The RC is a soft constraint meaning violations are feasible but with a cost. Ribeiro, et al. [16], Ribeiro, et al. [17] subdivided the RC into two categories, i.e. high and low priorities. The low priority RC has the value of the RC less than the high priority RC. In fact, the option with high priority RC will impose heavy workload on the assembly line and demonstrating the sense of production difficulty.

To count the number of RC violations, the sliding window method is normally used. This method counts all RC violations whenever found (Gottlieb et al. 2003). However, this method does not only double counting the RC violations, but also it could be biased by the violations occurring around the beginning and the end of the daily production sequence. To mitigate this issue, Fliedner and Boysen [37] suggested counting only the actual number of occurrences leading to the RC violations. An numerical example is demonstrated in Chutima and Olarnviwatchai [7].

Another constraint of the CSP recently introduced by the ROADEF challenge is the number of colour change (NCC). This constraint occurred in the paint shop where various colours are painted on the bodies of cars. Since unproductive setups are inevitable when changing paint colours from one to another, cars should be painted with the same colour in a batch as many as possible. However, the maximum batch size (CBS_{max}) is controlled by two events, i.e. number of cars continuously painted with the same colour before the paint gun agglutinates and before visual colour inspection becomes ineffective, depending on which one is less. Obviously, CBS_{max} is a hard constraint; hence, any car sequence that violates the CBS_{max} is unfeasible. To count the NCC correctly, the colour of the last car produced in day $J-1$ must be compared with the first car of day J to check if the colour change is occurred or not.

Most literature published on the CSP always assumes that one-sided assembly lines (1SALs) are used in the assembly shop which is unrealistic in practice. The more pragmatic assumption is the utilisation of 2SALs. Recently, Chutima and Olarnviwatchai [7] introduced the new version of the CSP where the 2SAL was employed in the system namely the CSP2SAL. In their research, apart from the RC and NCC, the utility work (UTW) arisen due to the employment of the 2SAL was also optimised. This research extends the previous work of Chutima and Olarnviwatchai [7] to include more pragmatic objectives related to the 2SAL and just-in-time production environment called MaCSP2SAL. Feasible solutions of the MaCSP2SAL is a permutation of daily car production sequence in which all hard constraints are satisfied and the violations of the soft constraints are minimised.

In this research, six objective functions normally used in automotive industry and literature are optimised simultaneously to assess the effectiveness of the car sequence in 2SAL including minimising number of colour changes, minimising utility work, minimising total idle time, minimising number of RC violations, minimising total production rate variation and minimising total parts usage variation. The detailed formulation of each objective is elaborated as follows.

(a) Minimising number of colour changes (NCC): In the paint shop, cars are consecutively painted with various colours. Setup is necessary when changing from one colour to another. As a result, consecutive painting with the same colour is an effective way to save the setup time and cost. However, the same colour painting batch size could not be greater than the predefined maximum number (CBS_{max}) since the paint gun needs to be purged at a regular interval to reduce the quality problems on the painted surface. In addition, the paint colour is often changed during paint gun purging to control visual overload and substandard indistinguishable in the quality inspection process. The first objective can be formulated as follows [7].

$$f_1 = \text{minimise } NCC = \sum_{k=0}^{nc-1} \delta_{COL_{k,k+1}} \quad (1)$$

where

k	position of the car sequence in day J, ($k \in [1, nc]$)
nc	number of the cars to be produced in day J
COL_k	colour of the car located at the position k
$\delta_{COL_{k,k+1}}$	difference in colours of cars in positions k and $k+1$, ($\delta_{COL_{k,k+1}}$ = 1 if the colours of cars in positions k and $k+1$ are different; and 0 otherwise)

(b) Minimising utility work (UTW): In 2SAL, the utility work represents the unfinished tasks leftover by regular workers due to the cycle time limitation. Utility workers have to help complete these unfinished utility works to avoid line stoppage and allow the worker in the next workstation to start his operation at the earliest start time. The pictorial explanation is in Fig. 1. The second objective can be formulated as follows [38].

$$f_2 = \text{minimise } UTW = \sum_{n_m=1}^{N_M} \left(\sum_{i=1}^{nc} U_{i,n_m} + Z_{(i+1),n_m} / s_c\right) \quad (2)$$

subject to

$$U_{i,n_m} = \left\{ \begin{array}{l} \max\left[0, \frac{\left(Z_{i,n_m} + s_c \sum_{m=1}^{M} X_{i,m}\{t_{2n_m-1,m} + Y_{2n_m-1,m}\} - L_{n_m}\right)}{s_c}\right] \\ + \max\left[0, \frac{\left(Z_{i,n_m} + s_c \sum_{m=1}^{M} X_{i,m}\{t_{2n_m,m} + Y_{2n_m,m}\} - L_{n_m}\right)}{s_c}\right] \end{array} \right\} \quad (3)$$

$$Z_{(i+1),n_m} = \max \left\{ \begin{array}{l} \max\left[0, \min\left(Z_{i,n_m} + s_c \sum_{m=1}^{M} X_{i,m}\{t_{2n_m-1,m} + Y_{2n_m-1,m}\} - \gamma s_c, L_{n_m} - \gamma s_c\right)\right] \\ , \max\left[0, \min\left(Z_{i,n_m} + s_c \sum_{m=1}^{M} X_{i,m}\{t_{2n_m,m} + Y_{2n_m,m}\} - \gamma s_c, L_{n_m} - \gamma s_c\right)\right] \end{array} \right\} \quad (4)$$

where

n_m	sequential number for each mated station (n_m = 1, 2, …, N_M)
nc_v	number of cars in model $v \in V$ to be produced in the current day
nc	total daily production order ($nc = \sum_{v=1}^{V} nc_v$)
γ	rate of the product launch interval into the conveyer
$t_{j,v}$	total operation time for car model v at station j
$Y_{j,v}$	total unavoidable idle time caused by the sequence-dependent finish time of tasks for car model v at station j
s_c	conveyor speed
CT	cycle time of the line
L_{n_m}	fixed line length of the mated station n_m ($L_{n_m} = s_c * CT$)
U_{i,n_m}	amount of utility work required for the ith product ($i = 1 \ldots I$) in a sequence at the mated station n_m
$X_{i,v}$	existent of car model v at the ith product sequence ($X_{i,v}$= 1, if the ith car in a sequence is model v; and $X_{i,v}$= 0, otherwise)
Z_{i,n_m}	starting position of the work on the ith car in a sequence at the mated station n_m

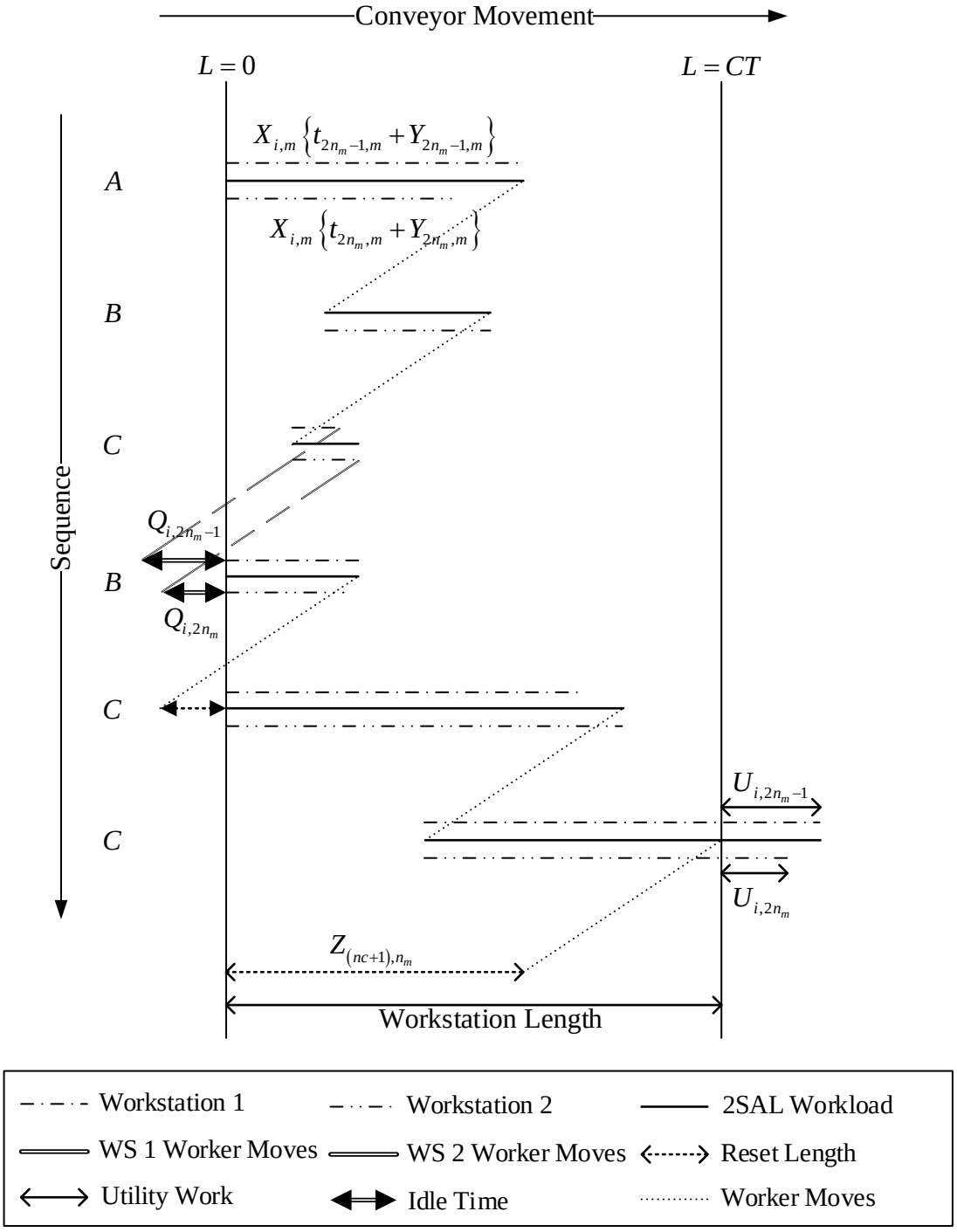

Fig. 1. Pictorial description of utility work and idle time in 2SAL.

(c) Minimising total idle time (IT): In 2SAL, the duration which the worker has to wait before the next car entering the workstation is to be minimised in order to increase worker utilisation. Since two workers are needed in the mated stations of 2SAL, the total idle time of this kind is the sum of the idle time from each worker. The pictorial explanation is illustrated in Fig. 1. The third objective can be formulated as follows [39].

$$f_3 = \text{minimise } IT = \frac{1}{v_c}\sum_{n_m=1}^{N_M}\sum_{i=1}^{nc} Q_{i,n_m} \tag{5}$$

$$Q_{(i+1),n_m} = \begin{Bmatrix} \max\left[0, \gamma s_c - \left(Z_{i,n_m} + s_c \sum_{m=1}^{M} X_{i,m}\left\{t_{2n_m-1,m} + Y_{2n_m-1,m}\right\}\right)\right] \\ + \max\left[0, \gamma s_c - \left(Z_{i,n_m} + s_c \sum_{m=1}^{M} X_{i,m}\left\{t_{2n_m,m} + Y_{2n_m,m}\right\}\right)\right] \end{Bmatrix} \tag{6}$$

subject to

$$s_c \sum_{m=1}^{M} X_{i,m}\left\{t_{2n_m-1,m} + Y_{2n_m-1,m}\right\} > 0 \qquad \forall i \forall n_m \tag{7}$$

$$s_c \sum_{m=1}^{M} X_{i,m}\left\{t_{2n_m,m} + Y_{2n_m,m}\right\} > 0 \qquad \forall i \forall n_m \tag{8}$$

where

IT total idle time to wait for the next car entering the mated stations

Q_{i,n_m} distance before the ith car sequence to enter the mated station n_m

(d) Minimising a total number of RC violations ($TNRCV$): In the assembly shop, the same standard body of cars could be installed with different options to satisfy the personalised demands of specific customers. For some difficult-to-install options, workers have a limited capacity to handle such heavy workload continuously before feeling tired. Therefore, these options have to distribute uniformly along the production sequence so that workers can reduce their fatigues by installing difficult options alternatively with those easy ones. The RC of the option $o \in O$ is defined as p_o/q_o meaning that for any segment with length q_o of the car production sequence, at most p_o cars requiring option o to be produced. Obviously, for an easy option the ratio of p_o/q_o is low. Because the subsequent segment of q_o cars must be considered in the RC violation, the last $q_o - 1$ cars produced in day $J-1$ and the first $q_o - 1$ cars produced in day $J+1$ must be used in determining the RC violation in day J. Since the RC is a soft constraint, violations are possible but at a cost. The modified sliding window technique proposed by Fliedner and Boysen [16] is used in this research since it can count the actual positional occurrences of violations effectively. Besides revealing the number of violations correctly without double counting, this technique could reveal the maximum number of overloaded workstations also. The fourth objective can be formulated as follows (note: a calculation sample can be seen in [7]).

$$f_4 = \text{minimise } TNRCV = \sum_{o \in O}\left\{NPV_{o;\, i\, \epsilon\, day\, J}\left[\!\left[\bigcup_{i=-q_0+1}^{nc} PV_o(i, \ldots, i+q_o-1)\right]\!\right]\right\} \tag{9}$$

where

$\cup$ union set operator

J current production day

nc number of cars to be produced in day J

$PV_o(i, \ldots, i+q_o-1)$ set of positions where violations occurred under a sliding window starting from position i to $i+q_o-1$

$NPV_{o;\, i\, \epsilon\, day\, J}[\![\ ]\!]$ number of positions only in day J in which violations occurred for option o

(e) Minimising total production rate variation (PRV): In the assembly shop, the production rate variation is the variation in the rate at which different car models are sequentially assembled on the assembly line. If the production of any car model is uniformly distributed on the daily car production sequence, its production rate variation will be low, and vice versa. This objective is very important in the just-in-time production system since the mixed-model production is necessary in responding to customer demands for various car models. The fifth objective can be formulated as follows [36].

$$f_5 = \text{minimise } PRV = \sum_{i=1}^{nc} \sum_{m=1}^{M} \left| \left(\sum_{l=1}^{i} \frac{X_{l,m}}{i} \right) - \frac{d_m}{nc} \right| \quad (8)$$

where

m	car models to be produced by the assembly line ($m = 1, \ldots, M$)
d_m	demand of the car model m
nc	number of cars to be produced in the current day
$X_{l,m}$	existent of car model m at the lth product sequence (equal to 1 if car lth in the assembly sequence is of model m, and 0 otherwise)

(f) Minimising total parts usage variation (PUV): In the assembly shop, the parts usage variation directly affects the preparation of parts for each product model to be assembled in the assembly line. If parts usage variation is high or parts consumption is uneven, the production planning will become difficult and the possibility of parts shortage is high. The sixth objective can be formulated as follows [40].

$$f_6 = \text{minimise } PUV = \sum_{i=1}^{nc} \sum_{j=1}^{\beta} \left(\frac{i \times N_j}{nc} - UT_{j,i} \right)^2 \quad (9)$$

$$UT_{j,i} = \sum_{m=1}^{M} \left(\left(\sum_{l=1}^{i} X_{l,m} \right) \times b_{m,j} \right) \quad (10)$$

$$N_j = \sum_{m=1}^{M} \left(d_m \times b_{m,j} \right) \quad (11)$$

where

β	number of parts variety
N_j	total quantity of part j required by all cars
d_m	demand of car model m
$UT_{j,i}$	total number of part j being assembled from the first product sequence until ith product sequence ($i = 1, \ldots, nc; j = 1,2, \ldots, \beta$)
$b_{m,j}$	number of units of part j required per unit of car model m (equal to 1 if car model m uses part j; 0 otherwise)

4. Proposed Solution Approach

4.1. AMOEA/D-DE

The CSP has long been considered as a multi-objective optimization problem (MOP) in practice since various conflicting objectives may need to be achieved simultaneously by car manufacturers. Previously, due to the absence of advanced metaheuristic theories as well as computer hardware and software, the multi-objective CSP had to be solved as a single objective optimisation by defining an aggregate utility function which takes a weighted sum of various objectives or alternatively solving these multi-objectives in a lexicographic order based on their preferential priorities. However, during the past two decades, the concepts of evolutionary algorithms and Pareto optimality have become more acceptable among researchers resulting in simultaneously optimising objectives used in the CSP under more realistic production environment is realisable [7].

According to Coello and Zacatenco [41], the formal formulation of the MOP is as follows.

$$minimise\ F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T \quad (12)$$
subject to $\boldsymbol{x} \in R^n$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of decision variables located in the feasible landscape of R^n, and $f_i(\boldsymbol{x})$ is the value of the ith objective function ($i = 1,2, \ldots, m$) in the objective space of R^m. Due to the objectives of the MOP are often conflicted in nature, trade-off decisions among them are always unavoidable when attempting to optimise them all together. As a result, the resulting solutions will be represented as a set of efficient trade-off solutions rather than a single one.

A solution $\boldsymbol{x}$ is said to dominate a solution $\boldsymbol{y}$ (mathematically written as $\boldsymbol{x} \prec \boldsymbol{y}$) if $\boldsymbol{x}$ is not worse than $\boldsymbol{y}$ in any objective and at least in one objective $\boldsymbol{x}$ is strictly better than $\boldsymbol{y}$. However, if $\boldsymbol{x}$ is better than $\boldsymbol{y}$ in one or more objectives and $\boldsymbol{y}$ is better than $\boldsymbol{x}$ in one or others, these two solutions are non-dominated (indifferent) to each other. Moreover, we said that $\boldsymbol{x}$ is a Pareto optimal solution (POS) or an efficient solution if $\boldsymbol{x}$ is non-dominated over the set of solutions under consideration. The vector of POSs plotted in the objective space forms the Pareto front (PF). The ideal PF should contain a reasonable number of POSs, all of which are diverse and spread uniformly throughout the whole area of the objective space.

One of the effective approaches to tackle the MOP is the application of multi-objective evolutionary algorithms (MOEAs) since the shape or discontinuity of PFs affects their performance very little. Most conventional MOEAs employ POSs to guide their search trajectories to improve the fitness of the solutions. However, this approach may be ineffective particularly in case of more than three objectives (i.e. many objectives) are dealt with simultaneously. When the number of the objectives escalates, the number of POSs will increase accordingly due to the lessening of the Pareto domination strength among the obtained solutions [42].

A novel MOEA to solve many-objective optimisation problems (MaOPs) was proposed by Zhang and Li [58] namely a multi-objective evolutionary algorithm based on decomposition (MOEA/D). The algorithm was found to perform very well on a number of MOPs. The MOEA/D decomposes the original MaOP into a number of subproblems in which different single objectives formulated by the scalar aggregations of uniformly distributed weight vectors are assigned to each of them. Unlike Pareto-based MOEAs, these diverse weight vectors are the key parameter used in projecting the search direction of the algorithm towards the optimal solution. In each generation, these subproblems are collaboratively optimised simultaneously. After the evolutionary process completes, the optimal solution to each of these subproblems is brought together to form the PF of the original MaOP.

To generate weight vectors for subproblems in the MOEA/D, the simplex-lattice design, a conventional decomposition method of the mixture experimental design, is often used. Let m be the number of objectives to be simultaneously optimised, $\lambda^i = (\lambda_1^i, \ldots, \lambda_m^i)^T$ be a weight vector of the ith subproblem (where $\lambda_j^i \geq 0$ for all $j = 1, \ldots, m$; and $\sum_{j=1}^{m} \lambda_j^i = 1$). The number of subproblems (i.e. population size N_p) in the (m, H) simplex-lattice design in which each objective takes H+1 equally distanced between 0 and 1 is C_{H+m-1}^{m-1}.

The MOEA/D utilises the information from each subproblem and its neighbourhoods to improve the fitness of solutions. Let $B^i(N_n)$ be a set of neighbourhood sized N_n of the i th subproblem. The subproblems $\{i_1, i_2, \ldots, i_{N_n}\}$ belong to the neighbourhood of the subproblem i if the Euclidean distances between their weight vectors $\lambda^{i_1}, \lambda^{i_2}, \ldots, \lambda^{i_{Nn}}$ and the weight vector λ^i are shortest comparing with the rest. Since the values of λ^i of the subproblem i and their neighbourhoods are not much different and closely related, the solutions within this area tend to have quite similar characteristics and hence they are used as a mechanism to guide searching direction towards the optimal solution.

Although various approach could be used to transform many objectives in the subproblems of the original MOP into scalar single-objective optimization problems, the Tchebycheff approach seems to be more effective because of its computational simplicity and effectiveness in handling non-convex PF [43]. The Tchebycheff function $g(x)$ is expressed as follows.

$$\min g(x|\lambda^i, z^*) = max_{1 \leq j \leq m}\{\lambda_j^i | f_j(x) - z_j^*|\} \tag{13}$$

where x is the decision variable, Ω is the feasible region of the decision variable x, $f_j(x)$ is the real value of the jth objective, $z^* = (z_1^*, \ldots, z_m^*)^T$ is a utopian point, and $z_j^* = min\{f_j(x)|x \in \Omega\}$ is the reference point (i.e. so-far best value of the jth objective). The function $\{\lambda_j^i | f_j(x) - z_j^*|\}$ is the weighted absolute deviation between the value of the jth objective and its reference point. The Tchebycheff function attempts to reduce the highest weighted absolute deviation of the subproblem as much as possible.

If the scales of the objectives are different, the normalisation to the same scale ranging between [0, 1] to provide a uniform platform of the solution space is necessary. The formulation of the normalised objective is given as follows.

$$\bar{f}_j = \frac{f_j - z_j^*}{z^{nad} - z_j^*} \tag{14}$$

where $\bar{f}_j$ is the normalised value of the jth objective, and $z^{nad} = max\{f_j(x)|x \in POS\}$ is the nadir point in the objective space.

In order to create new tentative solutions (offspring), the original version of the MOEA/D uses the genetic operator, i.e. one point crossover, which is normally used in the GA. However, the GA was outperformed by the differential evolution (DE) algorithm in terms of global search ability [44, 45]. Li and Zhang [46], Li and Zhang [47] proposed a multi-objective differential evolution based decomposition (MODE/D) in which the genetic operator was replaced by the differential operators and revealed that the MODE/D performed better than several other MOEAs on many test problems.

The DE, a subset of evolutionary algorithms, was first introduced by Storn [48] for global optimisation problems over continuous-valued landscapes. The DE has been recognised as a simple yet effective algorithm for successfully implemented in many engineering practices, e.g. power generation, engineering design, etc. [49]. The DE uses a real number to represent each variable in the solution. To produce tentative offspring, two of DE's operators are applied to solutions in the current generation in the order of mutation first and then crossover. The mutation operator creates a trial vector for each individual parent by mutating each variable in a target vector with a weighted differential of different vectors. After that, the crossover operator manipulates discrete recombination between the trial vector and the parent vector to form a new offspring. The fitness of the offspring is compared with its parent to determine which one will be survived in the next generation.

Various DE strategies, in which the main differences are on mutation and crossover operators, have been developed in the literature [50]. To articulate their variations, a general notation as DE/x/y/z is normally adopted; where x is the target vector selection method, y is the number of difference vectors used in the mutation, and z is the crossover method. The MODE/D uses the DE/rand/1/bin strategy to create offspring which works as follows. Under the conventional MOEA/D platform, for each individual subproblem i th (i.e. parent vector $\boldsymbol{x}_i$), three solutions are selected from the population (i.e. the neighbourhood of the subproblem, $B^i(N_n)$) at random. One of them ($\boldsymbol{x}_{r_1}$) acts as a target vector (x = rand) and the remaining twos ($\boldsymbol{x}_{r_2}$ and $\boldsymbol{x}_{r_3}$) are used to find the differential variation (y = 1). The mutant vector $\boldsymbol{v}_i$ is computed by perturbing the target vector with the weighted differential as follows.

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F\left(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}\right) \tag{15}$$

where $\boldsymbol{v}_i$ is the mutant vector of the parent vector $\boldsymbol{x}_i$ ($i \neq r_1 \neq r_2 \neq r_3$) and $F \in (0, \infty)$ is the scale factor. Once the mutation is finished, the crossover operator creates the trial vector $\boldsymbol{x}_i'$ from the recombination of the mutant vector $\boldsymbol{v}_i$ and its target vector (parent) $\boldsymbol{x}_i$. For the binomial crossover (z = bin) which is normally used in literature, the crossover points are randomly selected from the set of feasible crossover points $j \in [1, n_x]$ as follows.

$$x_{i,j}' = \begin{cases} v_{i,j}, & \text{if } random[0,1] < CR \text{ or } j = j_{random} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{16}$$

where $\boldsymbol{x}_i'$ is the trial vector, $x_{i,j}'$, $v_{i,j}$ and $x_{i,j}$ are the j^{th}element of the vectors $\boldsymbol{x}_i'$, $\boldsymbol{v}_i$ and $\boldsymbol{x}_i$, respectively, n_x is the number of elements in the vector, and CR is the crossover probability. Obviously, the higher the value of CR, the more element of the mutant vector will be included in the offspring than the parent (base vector), and vice versa. To ensure that at least one element of the trial vector is distinct from its parent, the crossover point j_{random} is randomly selected and assigning the value of $v_{i,j_{random}}$ to $x_{i,j_{random}}'$. The polynomial mutation is applied to the initial mutation vector to become the (final) mutation vector $\boldsymbol{x}_i''$ in the following way [49].

$$x_{i,j}'' = \begin{cases} x_{i,j}' + \sigma_n\left(x_{i,j}^{'Up} - x_{i,j}^{'Lw}\right) & \text{with probability } p_m \\ x_{i,j}' & \text{with probability } 1 - p_m \end{cases} \tag{17}$$

$$\sigma_n = \begin{cases} (2 \times rand_\sigma)^{\frac{1}{\eta+1}} - 1 & \text{if } rand_\sigma < 0.5 \\ 1 - [2(1 - rand_\sigma)]^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \quad (18)$$

where

$x_{i,j}^{\prime Up}$	the lower bound of the j^{th}element of the vectors $\boldsymbol{x}_i'$
$x_{i,j}^{\prime Lw}$	the lower bound of the j^{th}element of the vectors $\boldsymbol{x}_i'$
$rand_\sigma$	random number sampling from a uniform distribution [0, 1]
p_m	mutation probability
η	distribution index of polynomial mutation

Once a trial vector is created, its fitness is compared with its base vector's fitness and the one with better fitness will be survived as an offspring of the DE operations.

Apart from the DE strategy, the behaviour of the algorithm could be controlled by other parameters including population size, mutation factor $\boldsymbol{F}$ and crossover probability $\boldsymbol{CR}$. If the population size is big, a large number of individuals will participate in the mutation resulting in a greater chance to create a better offspring. The mutation factor $\boldsymbol{F}$ affects the searching ability of the DE in generating potential trail vectors. The values of $\boldsymbol{F}$ directly affect the search landscape of the DE, i.e. local search for small $\boldsymbol{F}$ values and global search for large $\boldsymbol{F}$ values. Moreover, the crossover probability $\boldsymbol{CR}$ controls the proportion of elements in the offspring to be inherited from the trial vector or the parent. The offspring is more resemble to the trial vector than the parent if the values of $\boldsymbol{CR}$ is high, and vice versa.

Conventionally, the DE strategy, as well as the values of the control parameters, are determined beforehand and they are kept unchanged during the course of the evolutionary search. As a result, the same searching pattern is repeatedly applied regardless of current fitness landscape causing a high risk of being trapped in local optima. In fact, the processes to obtain an appropriate DE strategy and the best values for individual control parameters are laborious. Although prior knowledge from previous research (if any) may be a good starting point for finding appropriate control parameter settings, a trial-and-error investigation which requires tedious experimental trials is necessary for the tuning process since these control parameters are mostly problem dependent. Moreover, the DE strategies and the values of the corresponding control parameters that make the algorithm perform best in one stage of the evolutionary search may not be preferred in another stage [51]. This shortcoming could be overcome by embedding an adaptive mechanism in the main structure of the algorithm.

Zhang and Sanderson [52] categorised parameter control mechanisms into three classes, i.e. deterministic, adaptive and self-adaptive. While the adaptive and self-adaptive parameter controls take feedback information from the evolutionary process to dynamically update the control parameters, the deterministic parameter control simply uses some deterministic rules to adjust the control parameters. The adaptive and self-adaptive parameter controls differ in that the latter uses a method of the evolution of evolution through mutation and crossover in the self-adaptation of control parameters. If the adaptive or self-adaptive parameter control is well planned, the robustness of the algorithm could be strengthened. Besides no tiresome trial-and-error trail is needed, the convergence rate of the algorithm could be improved when allowing the values of its control parameters to be dynamically updated to suit various search characteristics at different stages of the evolutionary process.

The DEs with adaptive or self-adaptive mechanism showed faster and more reliable convergence than those without [53, 54]. Among them, the randomisation-adaptation based DE usually outperforms the others [55]. Specifically, the randomisation-adaptation based DE uses some forms of probability distribution function, whose probability is derived from the successful offspring generation rate, to randomly provide various appropriate values of the control parameters, i.e. $\boldsymbol{F}$ and $\boldsymbol{CR}$. In this research, the concept of randomisation-adaptation based DE is adopted under the framework of MODE/D, namely AMOEA/D-DE, to solve the MaCSP.

The AMOEA-DE is an adaptive algorithm in which the algorithm could learn and utilise the feedback information acquired during the evolutionary search to dynamically adjust the control parameters without user's intervention. The adaptive mechanism of AMOEA/D-DE consists of two components, i.e. (1)

mutation strategy adaptation, and (2) control parameter adaptation. The detail of each mechanism is explained as follows.

Mutation strategy adaptation: AMOEA/D-DE adapts the concept of simultaneous implementation of multi-mutation strategies from SaDE [53]. Three mutation strategies ($s1$, $s2$ and $s3$) are in the candidate pool to allow the algorithm to generate a variety of search patterns with respect to the current requirement of the fitness landscape. As a result, at generation g, a trial vector $\boldsymbol{v}_{i,g}$ can be generated by one of these strategies.

$s1$: DE/rand/1: $$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{r_1,g} + F(\boldsymbol{x}_{r_2,g} - \boldsymbol{x}_{r_3,g}) \quad (19)$$

$s2$: DE/rbest/1: $$\boldsymbol{v}_{i,g} = \boldsymbol{r}\text{best}_{i,g} + F\left(\boldsymbol{x}_{r_1,g} - \boldsymbol{x}_{r_2,g}\right) \quad (20)$$

$s3$: DE/rand-to-rbest/1: $$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{r_1,g} + \gamma(\boldsymbol{r}\text{best}_{i,g} - \boldsymbol{x}_{r_1,g}) + F\left(\boldsymbol{x}_{r_2,g} - \boldsymbol{x}_{r_3,g}\right) \quad (21)$$

These mutation strategies are chosen in this research since they are simple to implement, widely used in DE literature, and reported to perform well on diverse problems. The mutation strategy DE/rand/1 is quite robust in generating diverse solutions but in contrast, its convergence rate is quite slow since it searches around a relatively small landscape without bias towards any particular direction. Gämperle et al. [56] indicated that the utilisation of the information from the best solution in developing a trial vector, e.g. DE/best/1, could be beneficial in fast convergence by focusing the search in the vicinity of the best solution. However, this strategy may cause premature convergence from reduced population diversity. To alleviate the weak point of fast but less reliable convergence, the mutation strategy DE/rand-to-best/1 which is the blending between DE/rand/1 and DE/best/1 is developed so that the trial vector could gain mutual benefits from both mutation strategies by searching around a wider landscape and biasing its searches towards potential optimum directions.

Under the base structure of MOEA/D, the vector located at a particular position in the neighbourhood of subproblem i currently being the best-so-far solution that subproblem i has ever found till generation g could be replaced by another solution during the evolutionary process of generation g. As a result, in generation g+1, the information on the real best solution of subproblem i will be lost if such replacement occurs in generation g especially with a poorer solution. Therefore, the information of the best-so-far solution associated with subproblem i must be constantly updated and kept throughout the evolutionary process. Moreover, in order to find the real best ($\boldsymbol{r}$best) of subproblem i to be used in Eq. (18) and (19), its best-so-far solution must also be taken into account along with the solutions in the neighbourhood in the competition arena. Where $\boldsymbol{r}\text{best} = (r\text{best}_1, r\text{best}_2, \dots, r\text{best}_{n_x})^T$

The mutation strategy adaptation of the AMOEA/D-DE is modified from Qin and Suganthan [50] to suit the context of MaCSP. Initially, the selection probability of each strategy is set to be equal (1/3). After completing the evolutionary process in each generation, the number of trial vectors successfully and unsuccessfully surviving in the next generation by each mutation strategy is recorded. In addition, the number of survival trial vectors becoming new non-dominated solutions is also counted. This information is used to update the probabilities of strategy selection in the next generation as follows.

$$p_{s,g+1} = p_{min} + (1 - S \times p_{min}) \cdot \left[(1-\alpha)p_{s,g} + \alpha \cdot \left(\frac{Suc_{s,g}}{\sum_{s=1}^{S} Suc_{s,g}}\right)\right] \quad (22)$$

$$Suc_{s,g} = \frac{ns_{s,g}}{ns_{s,g}+nf_{s,g}} \quad (23)$$

$$ns_{s,g} = \sum_{i=1}^{N_p}\sum_{j=1}^{N_n} DSR_{i,j,g}^{s} \quad (24)$$

$$nf_{s,g} = \sum_{i=1}^{N_p}\sum_{j=1}^{N_n}\left(1 - SR_{i,j,g}^{s}\right) \quad (25)$$

where

s	selectable strategy, $s \in \{s1, s2, s3\}$
S	total number of selectable strategy (S = 3 in this research)
α	scaling factor based on the proportion of offspring vectors successfully surviving ($0 \leq \alpha \leq 1$)

N_p population size
N_n neighborhood size
$p_{s,g}$ probability of selecting strategy s in generation g
p_{min} minimum probability of selecting any strategy (p_{min} = 0.05 in this research)
$Suc_{s,g}$ proportion of strategy s successfully creating survival offspring vectors in generation g
$ns_{s,g}$ number of strategy s successfully creating survival offspring vectors in generation g
$nf_{s,g}$ number of strategy s unsuccessfully creating survival offspring vectors in generation g
$SR^s_{i,j,g}$ number of strategy s successfully creating offspring vector that are better than the parent vector and its neighboring solutions of subproblem i ($SR^s_{i,j,g}$ = 1, if the offspring vector $\boldsymbol{x}_i''$ is better than its parent vector $\boldsymbol{x}_i$ and the other neighboring vectors $\boldsymbol{x}_j$ of subproblem i in generation g; 0, otherwise)
$DSR^s_{i,j,g}$ number of strategy s successfully creating offspring vector that are better than the parent vector and its neighbouring solutions of subproblem i, and the offspring vector becoming a new non-dominated solution ($SR^s_{i,j,g}$ = 1, if the offspring vector $\boldsymbol{x}_i''$ is better than its parent vector $\boldsymbol{x}_i$ and the other neighboring vectors $\boldsymbol{x}_j$ of subproblem i, and the offspring vector becoming a new non-dominated solution in generation g; 0, otherwise)

The detailed numerical example of MOEA/D-DE to solve MaCSP2SAL is given in [57].

Control parameter adaptation: The control parameters used in the DE could affect the convergence rate and diversity exploration performances significantly. As mentioned earlier, the values of the control parameters should not be fixed but allowed to vary based on the progress step and direction of the current search requirement. The first DE control parameter the value of which is adjusted during the evolutionary process is the mutation factor F which is a key parameter used for generating mutant vectors. The values of F have a direct impact to the exploitation and exploitation performances of the algorithm. A relatively small F renders fast convergence rate but may lead to premature convergence.

The concept to adjust F in this research is adapted from Zhang and Sanderson [52]. At each generation, the mutation factor of each subproblem i is independently updated based on a Cauchy distribution depending on the number of successful offspring creations in the previous generation. The Cauchy distribution is used because it could diversify F to avoid premature convergence especially in greedy mutation strategies as DE/rbest/1 and DE/rand-to-rbest/1. The formulation is as follows.

$$F^s_{i,g+1} = \begin{cases} \text{Cauchy}\left(\mu^s_{F,i,t+1}, 0.1\right), & \text{if } rand_1 < \tau^s_{i,g} \\ F^s_{i,g}, & \text{otherwise} \end{cases} \tag{26}$$

$$\mu^s_{F,i,g+1} = \begin{cases} (1-LR)\cdot\mu^s_{F,i,g} + LR\cdot F^s_{i,g} & \text{if } \sum_{j=1}^{N_p} DSR^s_{i,j,g} > 0 \\ \mu^s_{F,i,g} & \text{otherwise} \end{cases} \tag{27}$$

$$\tau^s_{i,g} = 1 - \left[(1-\beta)\cdot\frac{\sum_{k=1}^{g} ASR^s_{i,j,k}}{\sum_{k=1}^{g} X^s_{i,k}}\right] - \left[\beta\cdot\frac{\sum_{k=1}^{g}\sum_{j=1}^{N_p} ASR^s_{i,j,k}}{NR\sum_{k=1}^{g} X^s_{i,k}}\right] \tag{28}$$

$$ASR^s_{i,j,g} = \frac{1}{2}\left(SR^s_{i,j,g} + DSR^s_{i,j,g}\right) \tag{29}$$

where

$F^s_{i,g}$ mutation factor of strategy s in subproblem i in generation g
$\mu^s_{F,i,g}$ average mutation factor of strategy s in subproblem i in generation g
$\tau^s_{i,g}$ ratio of unsuccessfully creating offspring vector that are better than the parent vector from using strategy s in subproblem i generation g
LR learning rate
$ASR^s_{i,j,k}$ average of $SR^s_{i,j,g}$ and $DSR^s_{i,j,g}$

β scaling factor for weighting the successfully creating offspring at the subproblem i and the successfully creating offspring in the neighborhood of subproblem i, $0 \leq \beta \leq 1$
$X_{i,g}^{s}$ binary number, $X_{i,g}^{s} = 1$ if strategy s is used in subproblem i in generation g, 0 otherwise
NR maximum number of successfully creating offspring vector
$rand_1$ random number sampling from a uniform distribution [0, 1]

The crossover probability CR is another DE parameter that affects the performance of the algorithm. CR approximates the number of elements from the mutant vector will be inherited to the trial vector. If the value of CR is high, the trial vector will be more closely similar to the mutant vector than the base vector crating more solution diversity. Lin, et al. [58] suggested that the values of CR should be set at high during the beginning phase of the evolutionary search (exploration) and continuously decrease to a low value during the termination phase (exploitation). The regression equation was proposed to find appropriate values of CR. The value of CR is subject to adjustment in each generation which can be formalised as follows.

$$CR_{i,g+1}^{s} = \begin{cases} 0.55 + \left[\frac{1}{\pi} \times \arctan\left(\frac{1-NA_{i,g}^{s}/RG-0.8}{0.1}\right)\right], & \text{if } rand_2 < \tau_{i,g}^{s} \\ CR_{i,g}^{s}, & \text{otherwise} \end{cases} \quad (30)$$

$$NA_{i,g}^{s} = \sum_{k=1}^{g} na_{i,k}^{s} \quad (31)$$

where
$CR_{i,g}^{s}$ crossover probability of strategy s in subproblem i in generation g
$NA_{i,g}^{s}$ cumulative number of crossover probability adjustments by strategy s in subproblem i until generation g
$na_{i,k}^{s}$ binary number, $na_{i,g}^{s} = 1$ if crossover probability of strategy s is adjusted in subproblem i in generation k; and 0 otherwise
RG remaining number of generations yet to be run
$rand_2$ random number sampling from a uniform distribution [0, 1]

The last parameter to be adjustable in this research is the greediness scale of the base vector γ which is used in the DE/rand-to-$r$best/1 strategy ($s3$). This research applies the method to adjust γ similar to Brest, et al. [55] by altering γ when $rand_3$ is less than the probability of unsuccessfully creating a new offspring vector that is better than the base vector so as to generate more diverse trial vectors. The adjustment is carried out in every generation using the uniform distribution. The formulation is as follows.

$$\gamma_{i,g+1}^{s3} = \begin{cases} rand_4, & \text{if } rand_3 < \tau_{i,g}^{s3} \\ \gamma_{i,g}^{s3}, & \text{otherwise} \end{cases} \quad (32)$$

where
$\gamma_{i,g}^{s3}$ value of γ used by mutation strategy $s3$ in subproblem i in generation g
$rand_3$ random number sampling from a uniform distribution [0, 1]
$rand_4$ random number sampling from a uniform distribution [0, 1]

In this research, although the population size could be an adjustable parameter, it is assumed to be properly tuned in advance and fixed throughout the evolutionary process. Pseudo code for the AMOEA/D-DE is presented as follows.

Pseudo code of AMOEA/D-DE

/* *Initialisation*

Generate N_p weight vectors $\lambda^i = \left(\lambda_1^i, \lambda_2^i, \dots, \lambda_m^i\right)^T$, $i = 1, \dots, N_p$

For $i = 1, \dots, N_p$, define the set of indexes $B^i(N_n) = \left\{i_1, \dots, i_{N_n}\right\}$ where $\left\{\lambda^{i_1}, \dots, \lambda^{i_{N_n}}\right\}$ are the N_n closest weight vectors to λ^i (by the Euclidean distance)

Generate an initial population $P_0 = \left\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_p}\right\}$, $\boldsymbol{x}_i = \left(x_{i,1}, x_{i,2}, \dots, x_{i,n_x}\right)^T$

Evaluate each individual in the initial population P_0 and associate $\boldsymbol{x}_i$ with λ^i

For $i = 1, \dots, N_p$, set $\boldsymbol{r}\text{best}_i = \boldsymbol{x}_i$

Initialize $z^* = (z_1^*, \dots, z_m^*)$ by setting $z_k^* = \min\limits_{1 \le i \le N_p} \{f_k(\boldsymbol{x}_i)\}$, $k = 1,2, \dots, m$

Set $g = 1$

For all strategies $s = 1, \dots, S$, set $p_{s,g} = 1/S$

For all $SR_{i,j,g}^s$ and $DSR_{i,j,g}^s$ are set zero

/* *Main computation loop*

repeat

  for each parent vector $\boldsymbol{x}_i$, $i = 1, \dots, N_p$ do

    Select strategy s from the pool according to $p_{s,g}$

    /* *Parameter adaptation*

    if $rand_1 < \tau_{i,g-1}^s$ then //Adaptation of F ($rand$ in $U[0,1]$)

      Generate $F_{i,g}^s = \text{Cauchy}\left(\mu_{F,i,g+1}^s, 0.1\right)$

    else

      $F_{i,g}^s = F_{i,g-1}^s$

    end if

    if $rand_2 < \tau_{i,g-1}^s$ then //Adaptation of CR

      Calculate $CR_{i,g}^s = 0.55 + \left[\frac{1}{\pi} \times \arctan\left(\frac{1 - NA_{i,g}^s/RG - 0.8}{0.1}\right)\right]$

    else

      $CR_{i,g}^s = CR_{i,g-1}^s$

    end if

    if $rand_3 < \tau_{i,g-1}^s$ then //Adaptation of γ

      Generate $\gamma_{i,g}^s = rand$

    else

      $\gamma_{i,g}^s = \gamma_{i,g-1}^s$

    end if

    /* *Update real best solution*

    If the Tchebycheff value of $\boldsymbol{x}_j$ is better than $\boldsymbol{r}\text{best}_i$, $j \in B^i(N_n)$ then

      Set $\boldsymbol{r}\text{best}_i = \boldsymbol{x}_j$

    end if

    /* *Reproduction*

    Generate a new solution $\boldsymbol{x}_i'$ by DE operator (repair it if necessary)

    Apply polynomial mutation to produce $\boldsymbol{x}_i''$ (repair it if necessary)

    Update z^*, $z_k^* = \min\left(z_k^*, f_k(\boldsymbol{x}_i'')\right)$ and set $n_r = 0$

45: /* *Selection*
46: for each subproblem $j \in B^i(N_n)$ do
47: if $n_r < NR$ then
48: if $g^{te}(\boldsymbol{x}_i''|\lambda^j, z^*) \leq g^{te}(\boldsymbol{x}_j|\lambda^j, z^*)$ then
49: Replace $\boldsymbol{x}_j$ by $\boldsymbol{x}_i''$, increment n_r and set $SR_{i,j,g}^s = 1$
50: if $g^{te}(\boldsymbol{x}_i''|\lambda^j, z^*) < g^{te}(\boldsymbol{x}_j|\lambda^j, z^*)$ then
51: Set $DSR_{i,j,g}^s = 1$//true is equal to one
52: end if
53: end if
54: end if
55: end for
56:
57: /* *Update mean of scaling factor*
58: if any $DSR_{i,j,g}^s$ of subproblem $j \in B^i(N_n)$ be true then
59: Update $\mu_{F,i,g+1}^s$
60: else
61: $\mu_{F,i,g+1}^s = \mu_{F,i,g}^s$
62: end if
63:
64: Calculate all $ASR_{i,j,g}^s$ and $\tau_{i,g}^s$ for each strategy
65: end for
66: Calculate and update the probability $p_{s,g+1}$ for each strategy
67: $g = g + 1$
68: until $g > G$

4.2. Many-Objective Performance Measurements

To assess the performance of various algorithms in solving the MaCSP2SAL in Pareto sense, five groups of performance measurements suggested by Jiang, et al. [59] and Chutima and Olarnviwatchai [7] are used in this paper. Obviously, a good algorithm should create various POSs, most of which are located on the approximated true PF. In addition, it would be even better if these POSs are distributed uniformly along the approximated true PF. Since the true PF of the MaCSP2SAL is unknown, the approximated true PF is derived from combining all POSs of all algorithms and applying non-dominated sorting on them. The resultant first PF is an approximated true PF.

(a) *Convergence metric*: This metric computes the generational distance of algorithm j (GD) which is the least distance between the obtained POSs and the POSs on the approximated true PF. It indicates the proximity level between the non-dominated solutions of algorithm j and those in the approximated true PF. If this metric is close to zero, the PF of algorithm j is close to the approximated true PF. The formulation of GD is as follows.

$$GD(S_j, S^*) = \frac{1}{|S_j|}\sum_{x \in S_j} \min\{d_{xy} | y \in S^*\} \tag{33}$$

$$d_{xy} = \sqrt{\sum_{k=1}^{K}\left(\frac{f_k(x)-f_k(y)}{f_k^{\max}-f_k^{\min}}\right)^2} \tag{34}$$

where

GD_j	generational distance of algorithm j
S_j	non-dominated solutions on the first PF of algorithm j
S^*	non-dominated solutions on the approximated true PF
$\|S_j\|$	number of non-dominated solutions on the first PF of algorithm j
$\|S^*\|$	number of non-dominated solutions on the approximated true PF
d_{xy}	Euclidean distance between the obtained solution (x) and solution (y) on the approximated true PF
$f_k^{\max}$	maximum value of the objective k of non-dominated solutions on the approximated true PF
$f_k^{\min}$	minimum value of the objective k of non-dominated solutions on the approximated true PF
$f_k(x)$	value of objective k of the obtained solution (x)
$f_k(y)$	value of objective k of the solution (y) on the approximated true PF
x	obtained solution
y	solution on the approximated true PF

(b) *Convergence and diversity metric*: This metric indicates both the convergence and diversity of the first PF of the algorithm on a single scale. It computes the inverted generational distance of algorithm j (IGD) which is the Euclidian distance between the non-dominated solutions on the first PF of algorithm j and the non-dominated solutions on the approximated true PF. If this metric is close to zero, the non-dominated solutions of the algorithm j are not only converged to the approximated true PF but also they are diverse. The formulation of IGD is as follows.

$$IGD(S^*, S_j) = \frac{1}{|S^*|}\sum_{y \in S^*} \min\{d_{xy} | x \in S_j\} \tag{35}$$

(c) *Capacity metric*: This metric computes the ratio between the non-dominated solutions on the first PF of algorithm j which are on the approximated true PF and its owned non-dominated solutions on the first PF (R_{NDS1}), or between the non-dominated solutions on the first PF of algorithm j which are on the approximated true PF and the non-dominated solutions on the approximated true PF (R_{NDS2}). The higher the value of R_{NDS1} or R_{NDS2} is the better algorithm. The formulations of R_{NDS1} and R_{NDS2} are as follows. (Note: $y \prec x$ meaning x is not dominated by y)

$$R_{NDS1}(S_j) = \frac{|S_j - \{x \in S_j | \exists y \in S^* : y \prec x\}|}{|S_j|} \quad (36)$$

$$R_{NDS2}(S_j) = \frac{|S_j - \{x \in S_j | \exists y \in S^* : y \prec x\}|}{|S^*|} \quad (37)$$

(d) *Diversity metric*: This metric indicates both the distribution and spread of non-dominated solutions on the first PF of algorithm j simultaneously. The good algorithm should have non-dominated solution distributed uniformly and covered all the extreme points of the approximated true PF. The formulation of the diversity metric is as follows. (Note: the lower the value of this metric the better algorithm)

$$\Delta^*(S_j, S^*) = \frac{\sum_{c=1}^{C} d(E_c) + \sum_{i=1}^{|S_j|} |d(x_i) - \bar{d}|}{\sum_{c=1}^{C} d(E_c) + |S_j|\bar{d}} \quad (38)$$

$$d(x_i) = \min_{y \in S_j} \sqrt{\sum_{k=1}^{K} \left(\frac{f_k(x_i) - f_k(x_y)}{f_k^{\max} - f_k^{\min}} \right)^2} \quad (39)$$

where

$d(x_i)$ Euclidean distance between consecutive non-dominated solutions

$\bar{d}$ average of $d(x_i)$

$d(E_c)$ Euclidean distance from the non-dominated solutions of the algorithm to the extreme solutions of the approximated true PF.

C number of extreme solutions

(e) *Computational time metric*: This metric indicates the CPU time spent by the algorithm until the final solution is reached. A good algorithm should be able to search for a good solution by using less CPU time.

5. Experimental Design

5.1. Problem Sets

In order to evaluate the effectiveness of AMOEA/D-DE against its competitor algorithms on various production situations, 18 testbed MaCSP2SAL problems with various characteristics are employed (Table 1). The problems are grouped into three different sizes, i.e. small (S1-S6), medium (M1-M6) and large (L1-L6). The sizes of the problems are classified based on the daily number of cars to be produced which is ranged from 15-200. In each problem size, different production requirements are used including a) order-related requirements (i.e. number of car models, MPS, daily production orders), b) car-related requirements (i.e. number of options, number of colours, CBS_{max}), and c) production-related requirements (i.e. ratio constraints, number of parts, number of tasks in the precedence diagram, number of mated stations and number of stations).

All the requirements used in setting various testbed platforms are adapted from problems published in literature, i.e. colour, option and RC from Solnon, et al. [11]; mixed-model assembly line sequencing problems from McMullen [34] and Mansouri [33]; and 2SAL from Bartholdi [60], Kim, et al. [61] and Özcan and Toklu [62]. Table 1 also shows the feasible number of solutions for each problem as suggested by Hyun, et al. [28] to indicate the complexity of search space. In addition, line balancing is conducted for each particular 2SAL using the PSONK algorithm proposed by Chutima and Chimklai [63] to determine the number of mated stations, number of workstations and task-to-station assignments. These prerequisite input information is necessary before solving MaCSP2SALs.

DOI:10.4186/ej.2019.23.4.121

Table 1. Testbed problems.

Problem		Number of car models	MPS	Daily production orders	Number of tasks	Number of mated stations	Number of workstations
Small	S1	5	7:3:2:2:1	15	24	11	22
	S2	5	4:3:3:3:2	15	24	11	22
	S3	5	8:7:2:2:1	20	35	11	22
	S4	5	5:4:4:4:3	20	35	11	22
	S5	10	7:5:1:1:1:1:1:1:1:1	20	53	9	18
	S6	10	4:4:4:2:1:1:1:1:1:1	20	53	8	16
Medium	M1	10	6:5:5:5:4:4:3:3:3:2	40	65	8	16
	M2	10	11:7:5:5:4:2:2:2:1:1	40	65	7	14
	M3	12	8:7:5:5:5:4:4:4:3:3:1:1	50	83	17	34
	M4	12	12:9:8:6:5:3:2:1:1:1:1:1	50	83	16	32
	M5	14	8:7:7:7:6:6:5:4:4:2:1:1:1:1	60	100	9	18
	M6	14	15:15:5:4:4:4:3:2:2:2:1:1:1:1	60	100	10	20
Large	L1	15	20:20:20:15:15:1:1:1:1:1:1:1:1:1:1	100	148	14	28
	L2	15	15:15:10:10:10:10:10:10:4:1:1:1:1:1:1	100	148	13	26
	L3	15	20:20:20:15:15:10:20:10:5:20:5:10:10:10:10	200	205	14	28
	L4	15	25:20:15:20:10:10:15:5:15:15:10:15:10:10:5	200	205	15	30
	L5	18	45:30:30:15:10:10:10:8:7:7:7:7:5:3:3:1:1:1	200	297	39	78
	L6	18	25:25:20:20:20:20:15:10:10:5:5:5:5:5:5:2:2:1	200	297	39	78

DOI:10.4186/ej.2019.23.4.121

Table 1. Testbed problems (Continue).

Problem		Number of options	RC Constraint (p/q)	Number of colours	CBS_{max}	Number of parts	Possible number of solutions
Small	S1	4	1/1,1/2,2/5,1/2	5	3	10	1.08E+07
	S2	4	1/1,1/2,2/5,1/2	5	3	10	1.26E+08
	S3	4	1/3,1/2,1/3,1/2	5	3	10	2.99E+09
	S4	4	1/3,1/2,1/3,1/2	5	3	10	2.44E+11
	S5	7	2/5,5/9,10/13,1/2,1/2,1/2,2/3	9	4	20	4.02E+12
	S6	7	2/5,5/8,2/3,1/2,1/2,1/2,1/2	9	4	20	8.80E+13
Medium	M1	7	1/3,1/2,1/5,5/9,1/4,1/2,5/8	10	4	30	2.64E+33
	M2	7	1/3,1/2,1/6,5/9,2/9,1/2,2/3	10	4	30	1.47E+30
	M3	8	1/3,1/2,5/12,1/2,2/3,5/11,1/3,5/9	10	4	30	1.74E+44
	M4	8	1/3,1/2,5/12,1/2,2/3,5/11,1/3,5/9	10	4	30	4.19E+39
	M5	10	10/19,10/21,1/6,5/14,2/3,1/2,10/17,10/17,2/5,10/13	12	5	45	2.25E+55
	M6	10	5/8,5/8,1/5,5/9,2/3,5/13,5/7,5/7,10/21,10/11	12	5	45	6.11E+49
Large	L1	10	2/3,10/21,5/7,5/11,2/9,2/3,5/6,1/2,5/9,1/5	15	7	60	3.79E+78
	L2	10	5/9,2/5,10/21,5/14,2/9,2/3,2/3,1/2,2/7,1/2	15	7	60	9.96E+92
	L3	12	5/9,10/13,1/4,1/2,2/5,1/2,5/12,1/2,5/16,5/11,1/2,10/21	15	7	60	>1.00E+100
	L4	12	1/2,5/7,1/3,1/2,2/5,1/2,5/12,1/2,10/13,10/21,1/2,10/19	15	7	60	>1.00E+100
	L5	12	5/22,5/13,10/33,5/7,1/12,1/3,5/7,5/8,5/8,10/19,5/16,5/9	18	9	75	>1.00E+100
	L6	12	2/11,5/13,5/17,2/3,1/12,2/5,5/9,10/17,10/17,10/19,5/19,5/8	18	9	75	>1.00E+100

5.2. Parameter Settings of Algorithms

The proposed algorithm, AMOEA/D-DE, and its contestants (i.e. COIN-E, MODE, MOEA/D, MODE/D) are carefully programmed in MathLab R2015b software and running on an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 16 GB RAM, 64 bit operating system personal computer with Microsoft Windows 10 Pro. To make all algorithms worked at their best performance, the key controllable parameters of each algorithm are initially identified. The parameter tuning process starts with the adaptation of the key parameters suggested in the literature as initial setting points and then fine tuning all significant parameters are systematically conducted by the designs of experiments. Table 2 shows the resultant parameter settings of all algorithms.

Table 2. Parameter settings of all algorithms.

Parameter settings	COIN-E	MODE	MOEA/D	MODE/D	AMOEA/D-DE
Population size (N_p)	133	133	133	133	133
Maximum number of generation (G)	3000	3000	3000	3000	3000
DE Strategy	-	DE/rand/1/bin	-	DE/rand/1/bin	DE/rand/1/bin & DE/*r*best/1/bin & DE/rand-to-*r*best/1/bin
Crossover method	-	-	PMX	-	-
Mutation method	-	-	-	PM	PM
Probability of crossover	-	-	1.0	-	-
Probability of mutation (p_m)	-	-	-	$1/nc$	$1/nc$
Distribution index of polynomial mutation	-	-	-	20	20
Scale of the base vectors (γ)	-	-	-	-	0.5
Number of weight vectors in the neighbourhood (N_n)	-	-	5	S1: 5 S2-9.2: 20	20
Maximal number of solutions replaced by each offspring (NR)	-	-	S1-S2: 5 S3-M1: 2 M2: 5 M3: 5 M4: 2 M5: 2 M6: 2 L1: 5 L2: 5 L3: 5 L4: 2 L5: 5 L6: 5	S1: 2 S2: 5 S3-S4: 2 S5: 5 S6: 2 M1: 5 M2: 2 M3-M4: 5 M5-M6: 2 L1-L2: 5 L3: 2 L4: 5 L5-L6: 2	S1: 2 S2-S3: 5 S4-S5: 2 S6: 5 M1-M2: 2 M3-M6: 5 L1-L3: 2 L4: 5 L5: 2 L6: 5
Scaling factor (F)	-	S1-S3: 1.0 S4: 0.5 S5-S6: 1.0 M1: 0.5 M2-M3: 1.0 M4: 0.5 M5: 1.0 M6: 0.5 L1: 1.0 L2-L6: 0.5	-	S1: 0.5 S2: 1.0 S3: 1.0 S4: 0.5 S5: 1.0 S6-M4: 0.5 M5: 1.0 M6-L4: 0.5 L5: 1.0 L6: 1.0	S1: 0.5 S2: 1.0 S3: 1.0 S4: 0.5 S5: 1.0 S6-M4: 0.5 M5: 1.0 M6-L4: 0.5 L5: 1.0 L6: 1.0
Crossover rate (CR)	-	S1: 0.9 S2: 0.9 S3: 0.9 S4: 0.7 S5: 0.9 S6: 0.7 M1-L6: 0.9	-	S1-S5: 0.9 S6: 1.0 M1: 1.0 M2: 0.9 M3-M6: 1.0 L1-8.2: 0.9 L5: 1.0 L6: 1.0	S1-S5: 0.9 S6: 1.0 M1: 1.0 M2: 0.9 M3-M6: 1.0 L1-8.2: 0.9 L5: 1.0 L6: 1.0

6. Experimental results

The performances of AMOEA/D-DE and it alternative algorithms after implemented in various characteristics of MaCSP2SAL are reported based on six key metrics, i.e. generational distance (GD), inverted generational distance (IGD), ratio of non-dominated solution type 1 (R_{NDS1}) and type 2 (R_{NDS2}), distribution and spread (Δ^*) and CPU time. For the statistical reason, each algorithm is run for 3 replicates, and its corresponding mean and standard deviation are computed (Table 3). The details of the experimental results are elaborated as follows.

- GD: The GD metric indicates the level of overlap between the obtained PF of the given algorithm and the approximated true PF. The lower value is the better algorithm. The results are clearly shown that for medium- and large-sized problems, AMOEA/D-DE outperforms the other algorithms since its GD metric is lowest. Although AMOEA/D-DE is the best in 3 out of 6 problems (S1, S4, S5 and S6) for small-sized problems, it is ranked second best in S2, S3 and S4. Moreover, its GD metric is just marginally larger than the best.

- IGD: The IGD metric measures both convergence and diversity of the obtained PF of the algorithm by using a single gauge. The lower value is the better algorithm. Similar to the GD metric, AMOEA/D-DE performs best for medium- and large-sized problems. Apart from being ranked second best in 3 out of 6 of the small-sized problems (S1, S2 and S4), AMOEA/D-DE is the champion for the rest of them. In addition, the differences between the second best (AMOEA/D-DE) and the best one are trivial.

- R_{NDS1}: The R_{NDS1} metric is a type of the capacity metric which calculates the proportion between the numbers of non-dominated solutions on the obtained PF of the algorithm which are located on the approximated true PF and the number of its owned non-dominated solutions on the obtained PF. If this proportion is high, meaning that most of the obtained non-dominated solutions of the algorithm are on the approximated true PF. Therefore, the higher the value is the better algorithm. Obviously, for medium- and large-sized problems, the R_{NDS1} metric of AMOEA/D-DE is much higher than the others. For small-sized problems, 3 out of 6 problems (S2, S3 and S4), AMOEA/D-DE is the second best. However, the values of the R_{NDS1} metric obtained by the best in comparison with AMOEA/D-DE are insignificant. It is observed that in some problems COIN-E and MODE/D have R_{NDS1} equal to 0 which means no segment of their obtained PF is overlapped with the approximated true PF at all.

- R_{NDS2}: The R_{NDS2} metric is quite similar to R_{NDS1}. However, the difference is in the denominator of the proportion which is the number of non-dominated solutions on the approximated true PF. AMOEA/D-DE gives the best R_{NDS2} metric for medium- and large-sized problems. Only 3 out of 6 of the small-sized problems (S1, S2 and S4) that AMOEA/D-DE is the second best and again the values of the first and second best is more or less the same.

- Δ^*: The Δ^* metric indicates the diversity in terms of distribution and spread of the obtained non-dominated solutions of the algorithm. The lower value is the better algorithm. It is observed that COIN-E and MODE shows best Δ^* in most of the problem sizes. Meanwhile, AMOEA/D-DE gives best Δ^* in 1 out of 6 of the large-sized problems only. However, the Δ^* metric should be carefully interpreted. It is meaningless if an algorithm (e.g. MODE or COIN-E) demonstrates good spread metric but its non-dominated solutions are not located on the approximated true PF.

- CPU time: The time to solution and quality of solution are always a key trade-off aspect when comparing the performance of different algorithms. Of course, we would like to have an algorithm that could provide the best solution in a short time like in case of AMOEA/D-DE under the problem L6. However, most of the time, this expectation is unrealistic. It is obvious that MOEA/D often dominates the others since it consumes the least CPU time to reach final solutions.

DOI:10.4186/ej.2019.23.4.121

Table 3. Experimental results.

Algorithm	Problem Set																	
	S1	S2	S3	S4	S5	S6	M1	M2	M3	M4	M5	M6	L1	L2	L3	L4	L5	L6
Generational Distance																		
AMOEA/D-DE	**0.0047 (0.0040)**	0.0274 (0.0030)	0.0266 (0.0076)	0.0210 (0.0141)	**0.0352 (0.0166)**	**0.0235 (0.0065)**	**0.0226 (0.0302)**	**0.0214 (0.0306)**	**0.0078 (0.0046)**	**0.0062 (0.0024)**	**0.0072 (0.0043)**	**0.0123 (0.0084)**	**0.0085 (0.0037)**	**0.0118 (0.0082)**	**0.0397 (0.0284)**	**0.0183 (0.0078)**	**0.0219 (0.0122)**	**0.0155 (0.0081)**
MODE/D	0.0124 (0.0040)	**0.0210 (0.0145)**	0.0278 (0.0027)	0.0291 (0.0087)	0.0361 (0.0060)	0.0477 (0.0052)	0.0493 (0.0151)	0.0636 (0.0256)	0.0393 (0.0150)	0.0628 (0.0479)	0.0481 (0.0098)	0.0311 (0.0338)	0.0357 (0.0233)	0.0514 (0.0378)	0.1004 (0.0790)	0.0604 (0.0552)	0.0521 (0.0434)	0.1321 (0.1082)
MOEA/D	0.0204 (0.0168)	0.0893 (0.0689)	**0.0251 (0.0131)**	**0.0196 (0.0095)**	0.0399 (0.0430)	0.0487 (0.0404)	0.0985 (0.0212)	0.0448 (0.0088)	0.1182 (0.0252)	0.1710 (0.0544)	0.0611 (0.0441)	0.0611 (0.0511)	0.1282 (0.0412)	0.0614 (0.0204)	0.1854 (0.0661)	0.1109 (0.1283)	0.1219 (0.0355)	0.1082 (0.0923)
MODE	0.0488 (0.0107)	0.0777 (0.0224)	0.1118 (0.0174)	0.1490 (0.0201)	0.1407 (0.0285)	0.1311 (0.0219)	0.2151 (0.0966)	0.1816 (0.0244)	0.0677 (0.0711)	0.1494 (0.0637)	0.0946 (0.0520)	0.1179 (0.0216)	0.2449 (0.0361)	0.1881 (0.1442)	0.3203 (0.0267)	0.4521 (0.0912)	0.4089 (0.0446)	0.3041 (0.0226)
COIN-E	0.0773 (0.0196)	0.1208 (0.0177)	0.1325 (0.0131)	0.0999 (0.0154)	0.2109 (0.0339)	0.1718 (0.0154)	0.1596 (0.0149)	0.1080 (0.0091)	0.1327 (0.0074)	0.2086 (0.0471)	0.1029 (0.0138)	0.1658 (0.0166)	0.1866 (0.0227)	0.2146 (0.0093)	0.2703 (0.0267)	0.4021 (0.0912)	0.3589 (0.0446)	0.2541 (0.0226)
Inverted Generational Distance																		
AMOEA/D-DE	0.1477 (0.0056)	0.1586 (0.0040)	**0.1516 (0.0033)**	0.1828 (0.0047)	**0.1606 (0.0078)**	**0.1643 (0.0032)**	**0.1918 (0.0036)**	**0.2025 (0.0090)**	**0.1944 (0.0067)**	**0.1797 (0.0044)**	**0.1808 (0.0108)**	**0.1498 (0.0023)**	**0.2100 (0.0319)**	**0.1897 (0.0095)**	**0.1694 (0.0120)**	**0.1795 (0.0027)**	**0.1462 (0.0082)**	**0.1592 (0.0053)**
MODE/D	0.1528 (0.0020)	0.1672 (0.0064)	0.1568 (0.0050)	**0.1817 (0.0022)**	0.1683 (0.0057)	0.1800 (0.0168)	0.2251 (0.0297)	0.2197 (0.0177)	0.2055 (0.0153)	0.2204 (0.0071)	0.1885 (0.0126)	0.1717 (0.0039)	0.2881 (0.0181)	0.2508 (0.0200)	0.3409 (0.0114)	0.2877 (0.0139)	0.2575 (0.0124)	0.2682 (0.0631)
MOEA/D	0.1610 (0.0097)	0.2152 (0.0490)	0.1558 (0.0121)	0.1833 (0.0070)	0.1695 (0.0269)	0.1770 (0.0041)	0.2476 (0.0269)	0.2104 (0.0165)	0.2290 (0.0193)	0.2949 (0.0684)	0.2140 (0.0098)	0.1929 (0.0081)	0.2730 (0.0237)	0.2256 (0.0070)	0.2818 (0.0383)	0.2532 (0.0760)	0.2287 (0.0241)	0.2330 (0.0437)
MODE	**0.1141 (0.0075)**	**0.1438 (0.0041)**	0.1961 (0.0042)	0.2412 (0.0128)	0.1928 (0.0195)	0.2341 (0.0284)	0.2964 (0.0819)	0.2886 (0.0233)	0.2581 (0.0090)	0.2972 (0.0294)	0.2496 (0.0459)	0.2412 (0.0137)	0.4383 (0.0624)	0.3693 (0.1934)	0.5074 (0.0682)	0.7904 (0.2313)	0.5525 (0.0653)	0.5314 (0.0500)
COIN-E	0.1521 (0.0142)	0.1693 (0.0147)	0.1882 (0.0099)	0.1892 (0.0065)	0.2145 (0.0126)	0.2358 (0.0182)	0.3091 (0.0079)	0.2421 (0.0214)	0.3024 (0.0182)	0.2876 (0.0158)	0.2880 (0.0142)	0.3522 (0.0228)	0.3792 (0.0412)	0.3916 (0.0077)	0.4574 (0.0682)	0.7404 (0.2313)	0.5025 (0.0653)	0.4814 (0.0500)
R_{NDS1}																		
AMOEA/D-DE	**0.9713 (0.0294)**	0.7389 (0.0051)	0.6852 (0.0456)	0.7424 (0.1496)	**0.6565 (0.1447)**	**0.7547 (0.0758)**	**0.8059 (0.2478)**	**0.8104 (0.2560)**	**0.9183 (0.0289)**	**0.9423 (0.0290)**	**0.8811 (0.0671)**	**0.8428 (0.0959)**	**0.8858 (0.0558)**	**0.8842 (0.0717)**	**0.6749 (0.1861)**	**0.8401 (0.0509)**	**0.7647 (0.0952)**	**0.8230 (0.0729)**
MODE/D	0.8847 (0.0492)	**0.7900 (0.1238)**	0.6855 (0.0163)	0.6927 (0.0726)	0.6230 (0.0182)	0.5535 (0.0247)	0.5321 (0.1364)	0.4613 (0.2287)	0.6349 (0.1237)	0.4504 (0.3324)	0.3700 (0.1504)	0.6846 (0.2931)	0.4612 (0.3197)	0.4737 (0.3536)	0.2782 (0.4002)	0.4461 (0.4053)	0.3885 (0.4823)	0.2456 (0.2132)
MOEA/D	0.8142 (0.0872)	0.4160 (0.4482)	**0.7056 (0.1085)**	**0.7827 (0.0784)**	0.5799 (0.3664)	0.6388 (0.2755)	0.2632 (0.1304)	0.5987 (0.0438)	0.2356 (0.1116)	0.1604 (0.0768)	0.4135 (0.2751)	0.5363 (0.2557)	0.0277 (0.0397)	0.3301 (0.2390)	0.0513 (0.0888)	0.4417 (0.3832)	0.0224 (0.0194)	0.2639 (0.4057)
MODE	0.5714 (0.0796)	0.4236 (0.1391)	0.3133 (0.0533)	0.2231 (0.0891)	0.1679 (0.0981)	0.2281 (0.0838)	0.0506 (0.0876)	0.0029 (0.0050)	0.4642 (0.3726)	0.1074 (0.0931)	0.1340 (0.2166)	0.0455 (0.0710)	0.0000 (0.0000)	0.0547 (0.0806)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)
COIN-E	0.2281 (0.0730)	0.1619 (0.0830)	0.1128 (0.0615)	0.2206 (0.0715)	0.0577 (0.0228)	0.0501 (0.0371)	0.0877 (0.0339)	0.1579 (0.0199)	0.0852 (0.0087)	0.0627 (0.0695)	0.1429 (0.0725)	0.0640 (0.0434)	0.0827 (0.0615)	0.0827 (0.0199)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)

Note: The results are shown in terms of mean (standard deviation). The bold and underline element represents the best performance in each criterion and problem set.

Table 3. Experimental results (Continue).

Algorithm	Problem Set																	
	S1	S2	S3	S4	S5	S6	M1	M2	M3	M4	M5	M6	L1	L2	L3	L4	L5	L6
R_{NDS2}																		
AMOEA/D-DE	0.1382 (0.0081)	0.1254 (0.0026)	**0.0743 (0.0036)**	0.0548 (0.0108)	**0.0848 (0.0191)**	**0.0961 (0.0083)**	**0.1316 (0.0378)**	**0.1090 (0.0373)**	**0.1202 (0.0006)**	**0.1704 (0.0065)**	**0.1304 (0.0113)**	**0.1246 (0.0130)**	**0.1555 (0.0226)**	**0.1436 (0.0109)**	**0.0416 (0.0117)**	**0.0279 (0.0020)**	**0.0397 (0.0059)**	**0.0287 (0.0008)**
MODE/D	0.1133 (0.0076)	0.1126 (0.0162)	0.0692 (0.0035)	0.0509 (0.0076)	0.0808 (0.0049)	0.0647 (0.0049)	0.0856 (0.0242)	0.0532 (0.0234)	0.0737 (0.0194)	0.0713 (0.0541)	0.0476 (0.0172)	0.0929 (0.0405)	0.0954 (0.0661)	0.0848 (0.0633)	0.0180 (0.0260)	0.0168 (0.0153)	0.0218 (0.0271)	0.0091 (0.0079)
MOEA/D	0.1066 (0.0088)	0.0623 (0.0752)	0.0711 (0.0142)	**0.0571 (0.0082)**	0.0731 (0.0499)	0.0826 (0.0358)	0.0526 (0.0261)	0.0900 (0.0059)	0.0350 (0.0166)	0.0319 (0.0153)	0.0677 (0.0450)	0.0846 (0.0403)	0.0052 (0.0077)	0.0511 (0.0377)	0.0029 (0.0051)	0.0145 (0.0125)	0.0011 (0.0010)	0.0090 (0.0139)
MODE	**0.1529 (0.0213)**	**0.1269 (0.0417)**	0.0400 (0.0068)	0.0205 (0.0082)	0.0301 (0.0176)	0.0370 (0.0136)	0.0085 (0.0147)	0.0004 (0.0007)	0.0569 (0.0475)	0.0184 (0.0160)	0.0185 (0.0299)	0.0059 (0.0093)	0.0000 (0.0000)	0.0094 (0.0141)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)
COIN-E	0.0610 (0.0195)	0.0473 (0.0237)	0.0144 (0.0079)	0.0203 (0.0066)	0.0103 (0.0041)	0.0081 (0.0060)	0.0175 (0.0068)	0.0239 (0.0030)	0.0126 (0.0013)	0.0125 (0.0138)	0.0234 (0.0119)	0.0099 (0.0067)	0.0171 (0.0127)	0.0148 (0.0036)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)
Diversity (Distribution and Spread)																		
AMOEA/D-DE	0.7953 (0.0218)	0.5633 (0.0566)	0.5332 (0.0125)	0.4916 (0.0220)	0.7747 (0.0242)	0.6524 (0.0311)	0.5779 (0.0326)	0.5729 (0.0431)	0.6286 (0.0193)	0.4823 (0.0310)	0.7012 (0.0154)	0.6609 (0.0362)	0.7351 (0.0277)	0.6857 (0.0265)	**0.5932 (0.0195)**	0.6193 (0.0181)	0.6173 (0.0058)	0.5649 (0.0430)
MODE/D	0.7946 (0.0570)	0.5575 (0.0239)	0.5573 (0.0662)	0.4863 (0.0092)	0.7985 (0.0136)	0.6480 (0.0508)	0.5920 (0.0261)	0.6293 (0.0823)	0.6673 (0.0531)	0.5605 (0.0178)	0.7572 (0.0501)	0.7364 (0.0333)	0.7387 (0.0521)	0.7705 (0.1586)	0.6892 (0.0419)	0.6413 (0.0523)	0.7194 (0.0627)	0.6310 (0.0789)
MOEA/D	0.8149 (0.0161)	0.6055 (0.1142)	0.5716 (0.0343)	0.5144 (0.0100)	0.8041 (0.0290)	0.6628 (0.0512)	0.5196 (0.0288)	0.5910 (0.0462)	0.6473 (0.0360)	0.5294 (0.0283)	0.7329 (0.0175)	0.7588 (0.0329)	0.7098 (0.0413)	0.6625 (0.0219)	0.7030 (0.0114)	0.6457 (0.0254)	0.6490 (0.0934)	0.6117 (0.0214)
MODE	**0.5663 (0.0838)**	**0.4736 (0.0546)**	**0.3468 (0.0353)**	0.3644 (0.0281)	**0.6749 (0.0151)**	**0.5569 (0.0121)**	0.5337 (0.0436)	0.5279 (0.0285)	0.4541 (0.0261)	0.3883 (0.0646)	0.5994 (0.0283)	0.6154 (0.0368)	0.6861 (0.0119)	0.6078 (0.0248)	0.6198 (0.0415)	**0.5202 (0.0564)**	**0.5557 (0.0372)**	**0.4766 (0.0627)**
COIN-E	0.7068 (0.0323)	0.5248 (0.0658)	0.3664 (0.0088)	**0.3609 (0.0103)**	0.6890 (0.0144)	0.5577 (0.0112)	**0.2974 (0.0313)**	**0.3581 (0.0115)**	**0.3185 (0.0073)**	**0.3376 (0.0153)**	**0.3484 (0.0321)**	**0.3461 (0.0366)**	**0.3031 (0.0218)**	**0.3725 (0.0310)**	0.6392 (0.0419)	0.5913 (0.0523)	0.6694 (0.0627)	0.5810 (0.0789)
Computational time (sec.)																		
AMOEA/D-DE	683.85 (2.51)	696.30 (3.13)	771.55 (1.22)	763.60 (4.29)	1028.00 (1.60)	1023.70 (1.92)	2183.88 (3.58)	2175.40 (2.04)	2957.15 (1.36)	2974.05 (3.17)	4593.42 (2.76)	4485.66 (2.30)	10368.531 (9.400)	10281.198 (34.700)	28092.339 (530.000)	29219.948 (65.545)	38170.042 (1106.590)	**38728.380 (611.565)**
MODE/D	776.23 (10.66)	725.20 (13.63)	969.67 (4.06)	1006.13 (5.06)	1165.38 (8.00)	1194.82 (12.79)	2235.26 (11.60)	2229.61 (10.43)	2974.87 (5.20)	2993.06 (30.40)	4705.70 (9.86)	4945.76 (396.00)	10318.026 (15.200)	10322.021 (120.500)	**28078.323 (534.000)**	29190.661 (188.606)	37996.802 (1376.961)	39296.656 (1031.302)
MOEA/D	**470.00 (1.87)**	**462.26 (0.85)**	**548.65 (0.51)**	**550.93 (1.25)**	**821.55 (1.96)**	**820.95 (0.78)**	**1976.84 (15.33)**	**1968.98 (3.36)**	**2690.03 (2.20)**	**2684.44 (15.57)**	**4460.04 (13.16)**	**4540.11 (78.40)**	**10125.703 (23.100)**	**10042.813 (30.100)**	28234.167 (1534.000)	**28925.906 (229.296)**	**37409.370 (752.173)**	38877.620 (1236.215)
MODE	485.82 (5.83)	479.79 (7.38)	558.13 (1.78)	560.13 (0.93)	830.60 (2.30)	828.28 (1.34)	1980.41 (18.90)	1979.71 (1.03)	2709.65 (3.72)	2692.38 (7.73)	4482.41 (3.74)	5119.31 (785.00)	10157.042 (33.700)	10080.620 (43.900)	28252.922 (571.000)	29168.776 (156.315)	37537.464 (691.065)	39141.250 (1023.600)
COIN-E	861.96 (57.6)	804.79 (14.19)	1135.21 (2.08)	1283.90 (41.59)	1712.03 (189.07)	1452.55 (93.88)	3624.32 (2.28)	4026.99 (342.03)	5284.17 (7.54)	5277.26 (14.77)	8982.06 (932.34)	8839.20 (916.36)	17976.80 (244.38)	17978.60 (354.77)	48216.81 (528.07)	51262.03 (191.84)	64873.74 (437.87)	69877.96 (824.76)

Note: The results are shown in terms of mean (standard deviation). The bold and underline element represents the best performance in each criterion and problem set.

In order to demonstrate the overall performances of different algorithms based on three problem sizes, each metric of six instances of each problem size (S, M and L) is averaged and ranked. The results are shown in Table 4. It is obvious that AMOEA/D-DE ranks first for all convergence related metrics, i.e. GD, IGD, R_{NDS1} and R_{NDS2}. Hence, it could be concluded that AMOEA/D-DE should be considered as a viable hybrid algorithm for solving many-objective car sequencing problems. In contrast, the poor performance algorithms are MODE and COIN-E. For the diversity related metric, MODE and COIN-E seem to be the best, whereas AMOEA/D-DE comes third. The algorithms that show the poorest performance on diversity are MODE/D and MOEA/D.

Table 4. Overall ranking of different algorithms based on problem sizes.

Algorithm	Problem Set		
	Small	Medium	Large
GD			
AMOEA/D-DE	**1**	**1**	**1**
MODE/D	2	2	2
MOEA/D	3	3	3
MODE	4	4	5
COIN-E	5	5	4
IGD			
AMOEA/D-DE	**1**	**1**	**1**
MODE/D	2	2	3
MOEA/D	3	3	2
MODE	4	4	5
COIN-E	5	5	4
R_{NDS1}			
AMOEA/D-DE	**1**	**1**	**1**
MODE/D	2	2	2
MOEA/D	3	3	3
MODE	4	4	5
COIN-E	5	5	4
R_{NDS2}			
AMOEA/D-DE	**1**	**1**	**1**
MODE/D	2	2	2
MOEA/D	3	3	3
MODE	4	4	5
COIN-E	5	5	4
Diversity Δ^*			
AMOEA/D-DE	3	3	3
MODE/D	4	5	5
MOEA/D	5	4	4
MODE	**1**	2	2
COIN-E	2	**1**	**1**
CPU time (sec.)			
AMOEA/D-DE	3	3	3
MODE/D	**1**	**1**	**1**
MOEA/D	2	2	2
MODE	4	4	4
COIN-E	5	5	5

Note: Number represents the rank of each algorithm. Bold shows the best performing algorithm. Underline indicates no significant difference between algorithms at a 95% confidence interval of the mean.

To illustrate the evolutionary process of each algorithm on different problem sizes, its metrics are plotted against the evolving generations as shown in Figures 2 - 4 for small- to large-sized problems, respectively. It is observed that the progressive evolution of AMOEA/D-DE in terms of the convergence-related metrics seems not so much different from the others for the small-sized problem. However, for medium- and large-sized problems, the performance of AMOEA/D-DE becomes apparent. Moreover, the gaps in performance differences between AMOEA/D-DE and the others are significantly widened in the large-sized problem. For the spread-related metric, MODE and COIN-E clearly outperform the others. However, this metric should be considered as a second priority when comparing to the convergence-related metrics since the usefulness of the algorithm with a very good spread will be devalued unless it could achieve very good convergence at the same time.

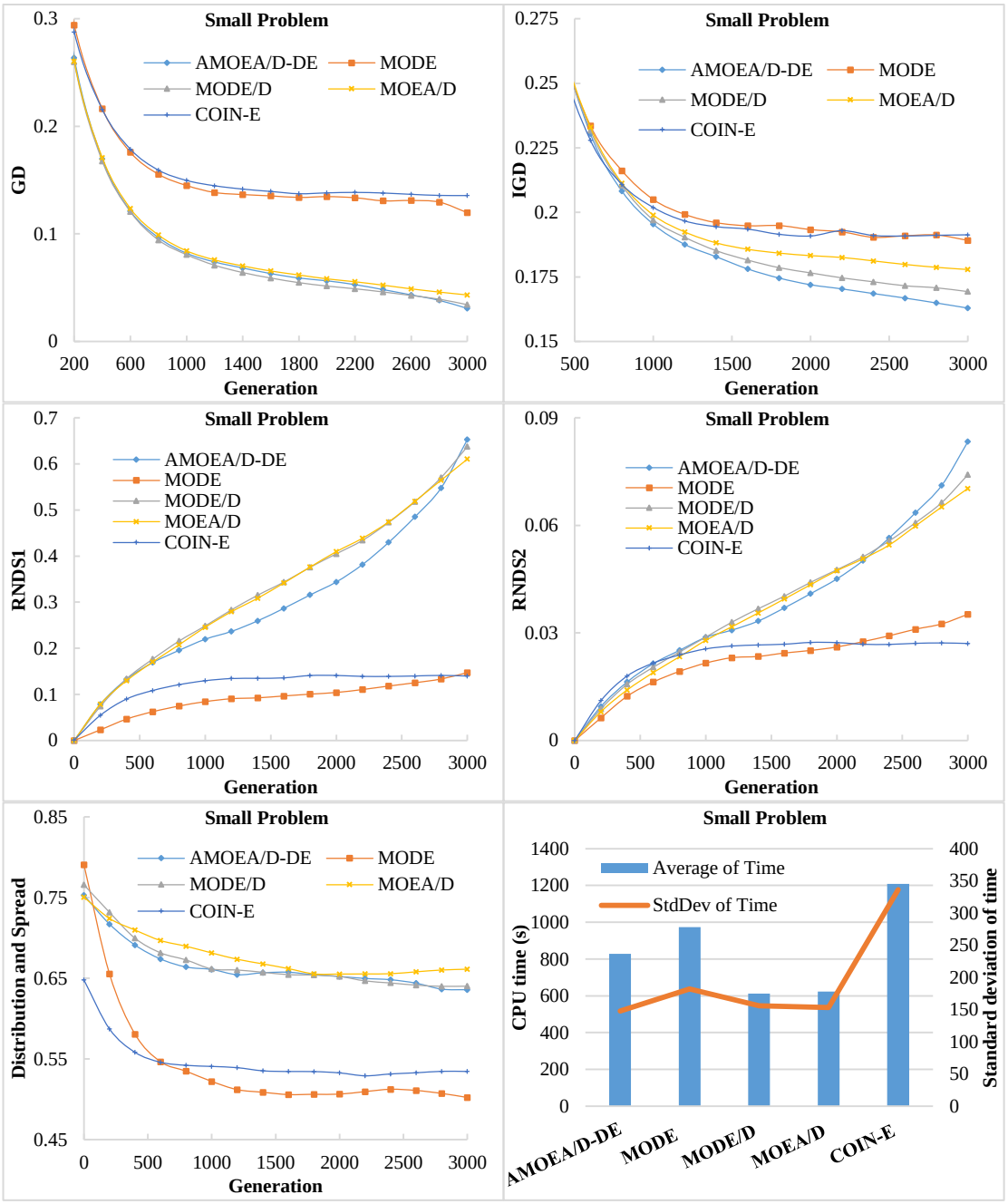

Fig. 2. Average results of different algorithms on small-sized problems.

DOI:10.4186/ej.2019.23.4.121

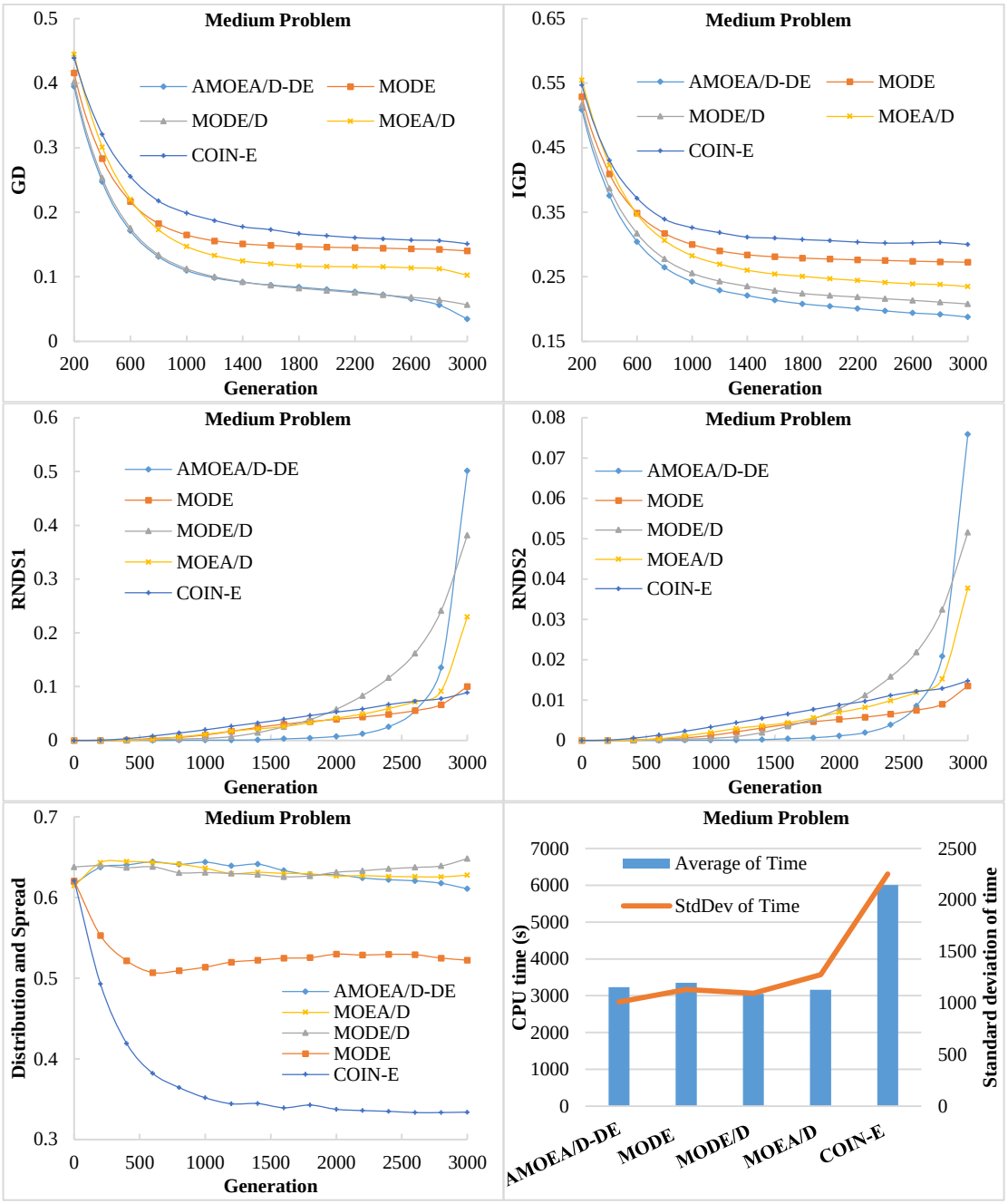

Fig. 3. Average results of different algorithms on medium-sized problems.

DOI:10.4186/ej.2019.23.4.121

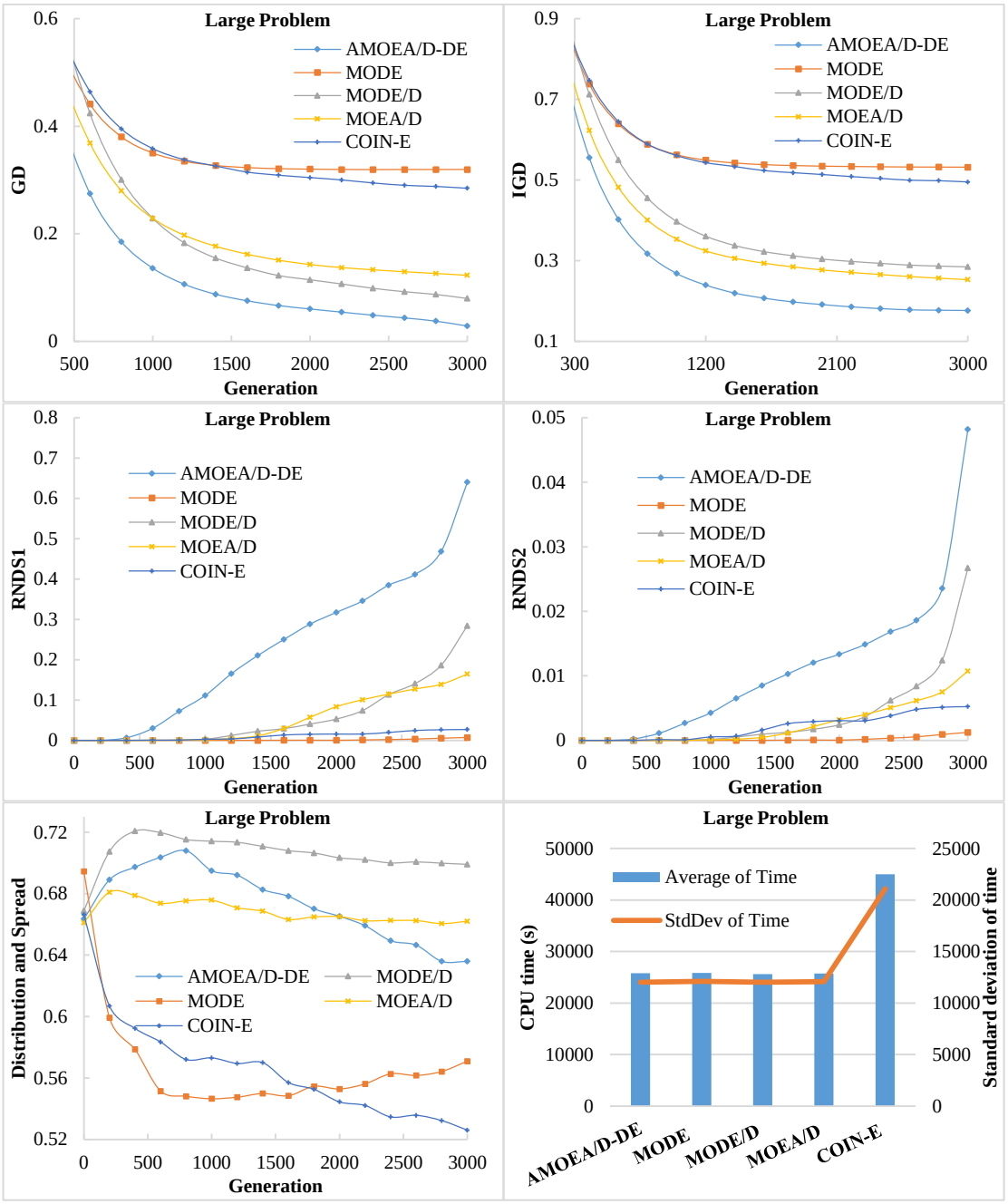

Fig. 4. Average results of different algorithms on large-sized problems.

To visualise the distribution of non-dominated solutions on many objectives, the 3D-RadVis Antenna recently developed by Ibrahim et al. [25] is employed. This visualised technique is very useful in observing relative locations of non-dominated solutions and trade-offs among various objectives due to its ability to map M-dimensional objective space to a 3D radial coordinate plot. Figures 5 - 7 depict the 3D-RadVis Antenna plots of four tested algorithms on different sizes of problems (S, M and L). It is clear that AMOEA/D-DE demonstrates the best convergence, particularly on medium- and large-sized problems since most of its non-dominated solutions are located at the bottom of the graphs.

DOI:10.4186/ej.2019.23.4.121

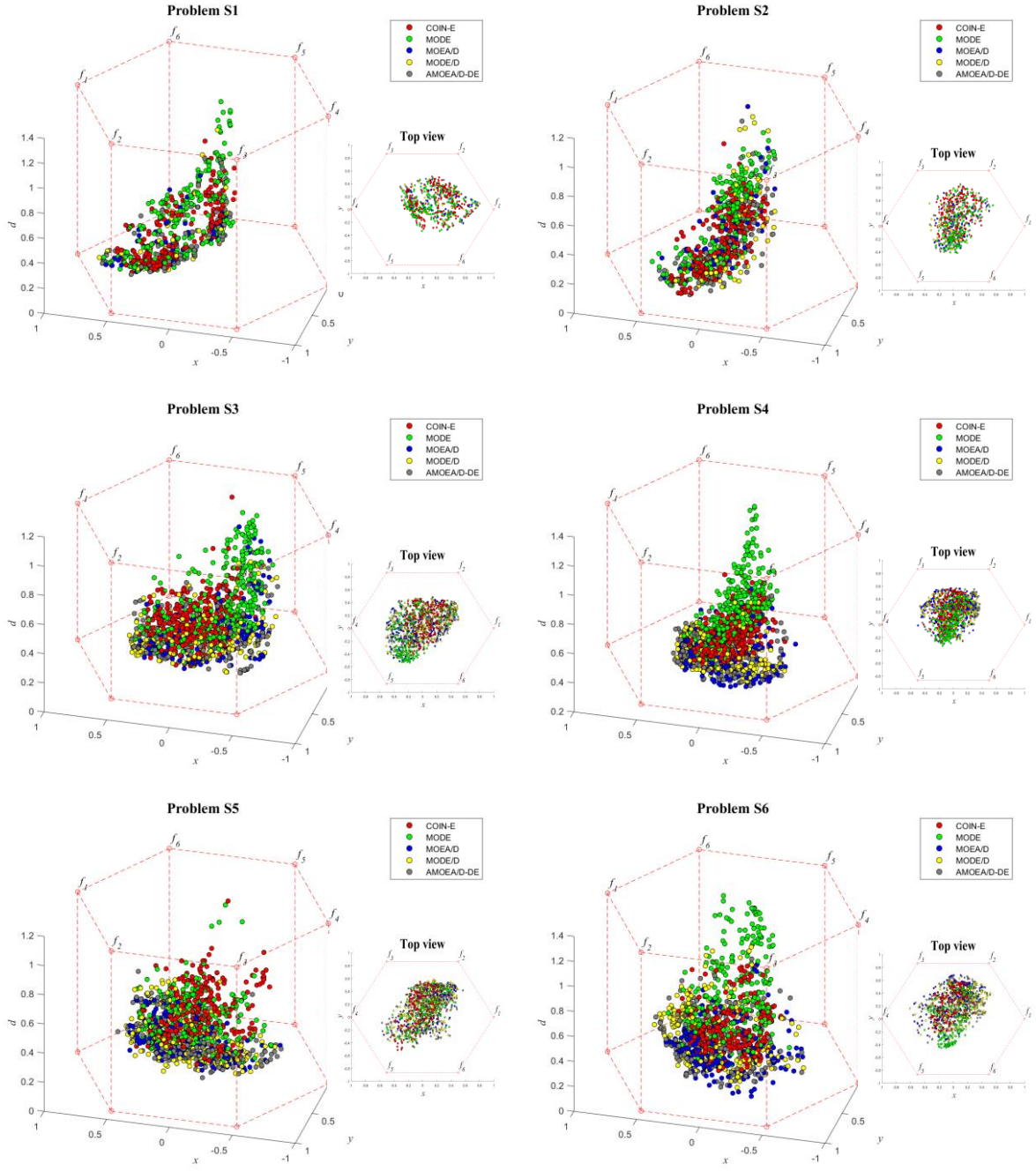

Fig. 5. 3D-RadVis antenna plots for problems S1-S6.

DOI:10.4186/ej.2019.23.4.121

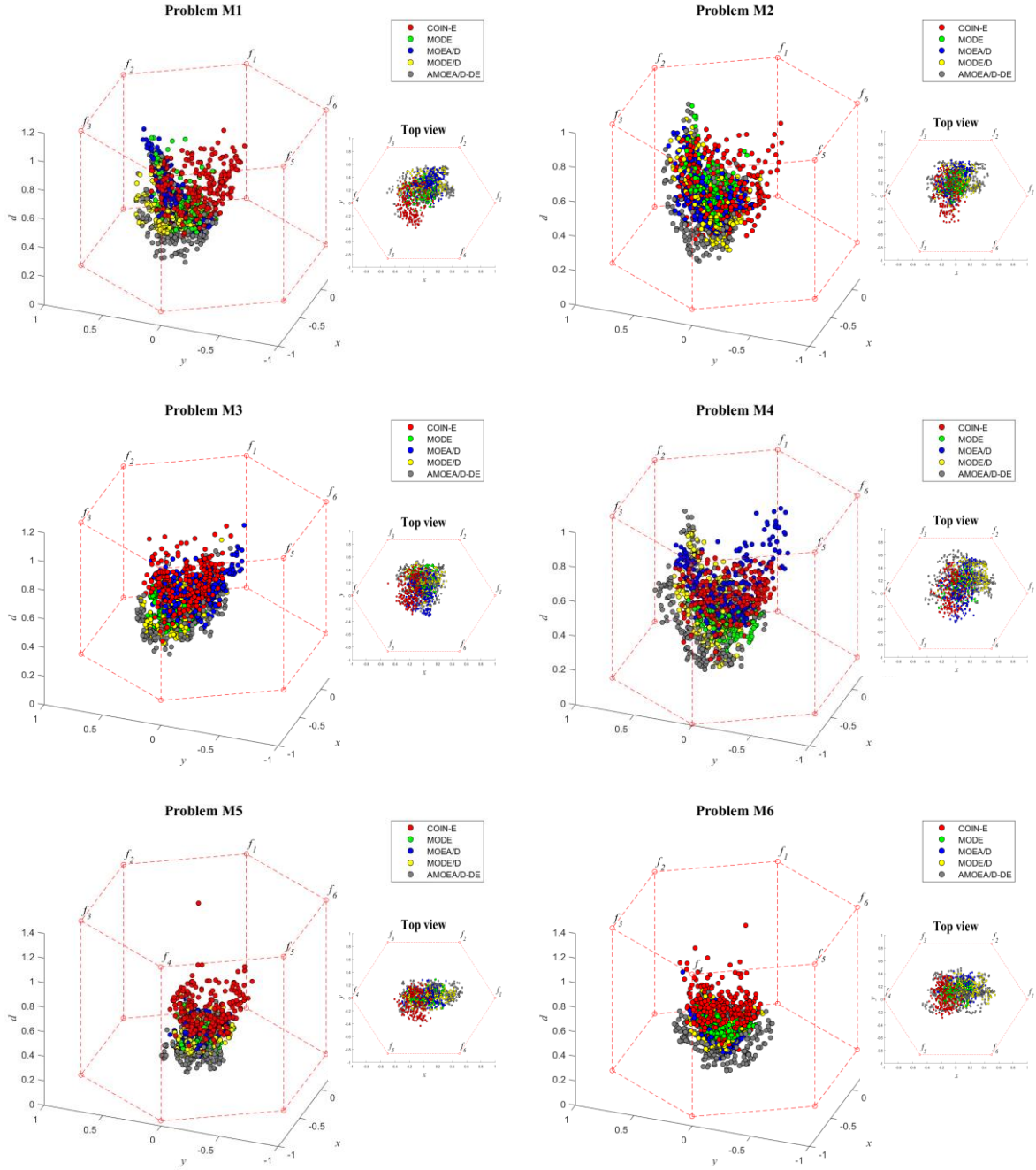

Fig. 6. 3D-RadVis antenna plots for problems M1-M6.

DOI:10.4186/ej.2019.23.4.121

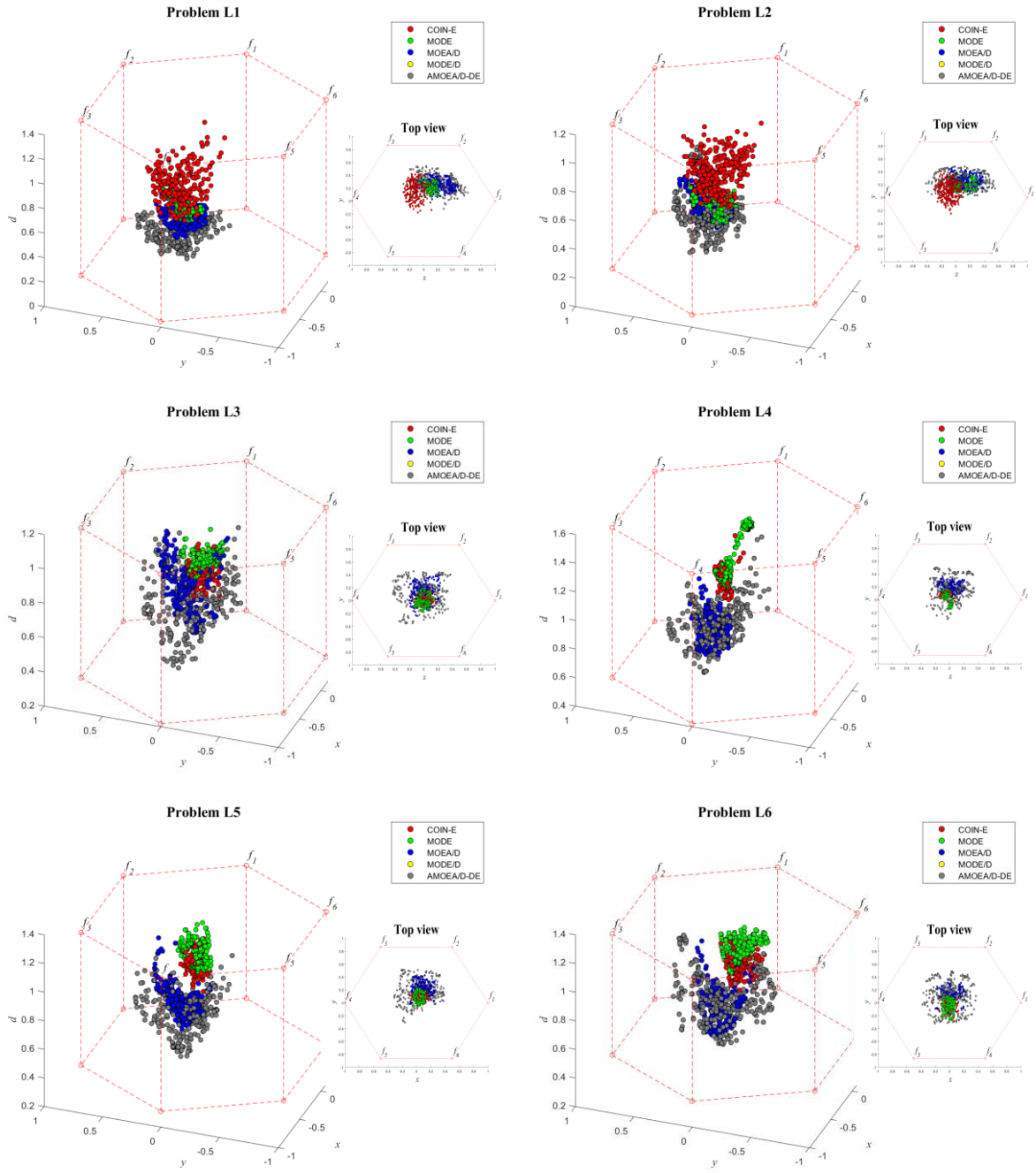

Fig. 7. 3D-RadVis antenna plots for problems L1-L6.

7. Concluding Remarks

The CSP in this research is the extension of its original version. 2SAL is assumingly installed in the assembly shop rather than 1SAL which more closely resembles the actual operations in practice. However, this new environment poses a new challenge to research scholars since a number of additional complex constraints needs to be considered. Moreover, many conflicting objectives have to be optimized simultaneously to satisfy customer demands, operational requirements, just-in-time production philosophy and workload requirements. Most of the objectives are soft which can be violated at a cost. To solve the MaCSP2SAL, the AMOEA/D-DE is developed to tackle many-objective combinatorial optimisation problems. The performance of AMOEA/D-DE is compared to COIN-E, MODE, MODE/D and MOEA/D under various problem characteristics and sizes. It is obvious that AMOEA/D-DE outperforms other algorithms in terms of GD, IGD, R_{NDS1} and R_{NDS2}. The implication of the results is that the AMOEA/D-DE not only could produce a higher number of non-dominated solutions when comparing to the others but its PF is mostly overlapped with the approximated true PF. This demonstrates the robustness of AMOEA/D-DE where its embedded adaptive mechanism is triggered according to the search trajectory requirement. Although AMOEA/D-DE yields the diversity metric poorer than MODE and COIN, it still ranks third best.

References

[1] B. D. Parrello, W. C. Kabat, and L. Wos, "Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem," *Journal of Automated Reasoning*, vol. 2, no. 1, pp. 1-42, 1986.

[2] M. Prandtstetter and G. R. Raidl, "An integer linear programming approach and a hybrid variable neighbourhood search for the car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1004-1022, 2008.

[3] T. Kis, "On the complexity of the car sequencing problem," *Operations Research Letters*, vol. 32, no. 4, pp. 331-335, 2004.

[4] B. Estellon, F. Gardi, and K. Nouioua, "Two local search approaches for solving real-life car sequencing problems," *European Journal of Operational Research*, vol. 191, no. 3, pp. 928-944, 2008.

[5] C. Gagné, M. Gravel, and W. L. Price, "Solving real car sequencing problems with ant colony optimization," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1427-1448, 2006.

[6] J. Gottlieb, M. Puchta, and C. Solnon, "A study of greedy, local search, and ant colony optimization approaches for car sequencing problems," in *Workshops on Applications of Evolutionary Computation*, 2003, pp. 246-257.

[7] P. Chutima and S. Olarnviwatchai, "A multi-objective car sequencing problem on two-sided assembly lines," Journal of Intelligent Manufacturing, vol. 29, no. 7, pp. 1617-1636, 2018.

[8] Z. He and G. G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 145-160, 2016.

[9] T. Warwick and E. P. Tsang, "Tackling car sequencing problems using a generic genetic algorithm," *Evolutionary Computation*, vol. 3, no. 3, pp. 267-298, 1995.

[10] I. P. Gent, "Two results on car-sequencing problems," Report University of Strathclyde, APES-02-98, vol. 7, 1998.

[11] C. Solnon, V. D. Cung, A. Nguyen, and C. Artigues, "The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 912-927, 2008.

[12] A. Jaszkiewicz, P. Kominek, and M. Kubiak, "Adaptation of the genetic local search algorithm to a car sequencing problem," in *7th National Conference on Evolutionary Algorithms and Global Optimization*, Kazimierz Dolny, Poland, 2004, pp. 67-74.

[13] K. Smith, M. Palaniswami, and M. Krishnamoorthy, "Traditional heuristic versus Hopfield neural network approaches to a car sequencing problem," *European Journal of Operational Research*, vol. 93, no. 2, pp. 300-316, 1996.

[14] O. Briant, D. Naddef, and G. Mounié, "Greedy approach and multi-criteria simulated annealing for the car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 993-1003, 2008.

[15] M. Puchta and J. Gottlieb, "Solving car sequencing problems by local optimization," in *Workshops on Applications of Evolutionary Computation*, 2002, pp. 132-142.

[16] C. C. Ribeiro, D. Aloise, T. F. Noronha, C. Rocha, and S. Urrutia, "An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 596-611, 2008.

[17] C. C. Ribeiro, D. Aloise, T. F. Noronha, C. Rocha, and S. Urrutia, "A hybrid heuristic for a multi-objective real-life car sequencing problem with painting and assembly line constraints," *European Journal of Operational Research*, vol. 191, no. 3, pp. 981-992, 2008.

[18] C. Solnon, "Combining two pheromone structures for solving the car sequencing problem with Ant Colony Optimization," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1043-1055, 2008.

[19] H. Gavranović, "Local search and suffix tree for car-sequencing problem wivh colours," *European Journal of Operational Research*, vol. 191, no. 3, pp. 972-980, 2008.

[20] J.-F. Cordeau, G. Laporte, and F. Pasin, "Iterated tabu search for the car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 945-956, 2008.

[21] A. Zinflou, C. Gagné, and M. Gravel, "Crossover operators for the car sequencing problem," in *European Conference on Evolutionary Computation in Combinatorial Optimization*, 2007, pp. 229-239.

[22] A. Zinflou and C. Gagné, "Tackling the industrial car sequencing problem using GISMOO algorithm," in *Assembly Line-Theory and Practice*. InTech, 2011.

[23] E. G. K. Atiker, F. B. Yeni, P. A. Sarvari, and E. Çevikcan, "A heuristic approach for car sequencing problem including assembly ratio and color constraints," in *Handbook of Research on Applied Optimization Methodologies in Manufacturing Systems*, 2018, ch. 4, pp. 57-76.

[24] M. R. Bahubalendruni and B. B. Biswal, "A review on assembly sequence generation and its automation," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 230, no. 5, pp. 824-838, 2016.

[25] M. R. Bahubalendruni, B. B. Biswal, M. Kumar, and R. Nayak, "Influence of assembly predicate consideration on optimal assembly sequence generation," *Assembly Automation*, vol. 35, no. 4, pp. 309-316, 2015.

[26] N. Boysen, M. Fliedner, and A. Scholl, "Sequencing mixed-model assembly lines: Survey, classification and model critique," *European Journal of Operational Research*, vol. 192, no. 2, pp. 349-373, 2009.

[27] J. Miltenburg, G. Steiner, and S. Yeomans, "A dynamic programming algorithm for scheduling mixed-model, just-in-time production systems," *Mathematical and Computer Modelling*, vol. 13, no. 3, pp. 57-66, 1990.

[28] G. Stiener and S. Yeomans, "Level schedules for mixed-model, just-in-time assembly process," *Management Science*, vol. 39, no. 6, pp. 728-735, 1993.

[29] P. Chutima and N. Kampirom, "A multi-objective coincidence memetic algorithm for a mixed-model U-line sequencing problem," *International Journal of Advanced Operations Management*, vol. 2, no. 3-4, pp. 201-248, 2010.

[30] P. R. McMullen, "JIT sequencing for mixed-model assembly lines with setups using tabu search," *Production Planning & Control*, vol. 9, no. 5, pp. 504-510, 1998.

[31] C. J. Hyun, Y. Kim, and Y. K. Kim, "A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines," *Computers & Operations Research*, vol. 25, no. 7-8, pp. 675-690, 1998.

[32] P. R. McMullen and G. V. Frazier, "A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line," *IIE Transactions*, vol. 32, no. 8, pp. 679-686, 2000.

[33] S. A. Mansouri, "A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines," *European Journal of Operational Research*, vol. 167, no. 3, pp. 696-716, 2005.

[34] P. R. McMullen, "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives," *Artificial Intelligence in Engineering*, vol. 15, no. 3, pp. 309-317, 2001.

[35] R. Tavakkoli-Moghaddam and A. Rahimi-Vahed, "Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 1471-1481, 2006.

[36] A. R. Rahimi-Vahed, M. Rabbani, R. Tavakkoli-Moghaddam, S. A. Torabi, and F. Jolai, "A multi-objective scatter search for a mixed-model assembly line sequencing problem," *Advanced Engineering Informatics*, vol. 21, no. 1, pp. 85-99, 2007.

[37] M. Fliedner and N. Boysen, "Solving the car sequencing problem via branch & bound," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1023-1042, 2008.

[38] P. Chutima and K. Jitmetta, "Adaptive biogeography-based optimisation for two-sided mixed-model assembly line sequencing problems," *International Journal of Operational Research*, vol. 16, no. 4, pp. 390-420, 2013.

[39] N. Manavizadeh, L. Tavakoli, M. Rabbani, and F. Jolai, "A multi-objective mixed-model assembly line sequencing problem in order to minimize total costs in a Make-To-Order environment, considering order priority," *Journal of Manufacturing Systems*, vol. 32, no. 1, pp. 124-137, 2013.

[40] M. Alaghebandha, V. Hajipour, and M. Hemmati, "Optimizing multi-objective sequencing problem in mixed-model assembly line on just-in-time: Particle swarm optimization algorithm," *International Journal of Management Science and Engineering Management*, vol. 12, no. 4, pp. 288-298, 2017.

[41] C. C. Coello and C. S. P. Zacatenco, "20 years of evolutionary multi-objective optimization: What has been done and what remains to be done," in *Computational Intelligence: Principles and Practice*. 2006, pp. 73-88.

[42] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple travelling salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 5, pp. 682-691, 2012.

[43] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712-731, 2007.

[44] B. Wang, Z. Wu, and Z. Zhao, "Performance comparison of GA, PSO, and DE approaches in estimating low atmospheric refractivity profiles," *Wuhan University Journal of Natural Sciences*, vol. 15, no. 5, pp. 433-439, 2010.

[45] K. Chandrasekar and N. Ramana, "Performance comparison of GA, DE, PSO and SA approaches in enhancement of total transfer capability using FACTS devices," *Journal of Electrical Engineering and Technology*, vol. 7, no. 4, pp. 493-500, 2012.

[46] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," *in Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 583-592.

[47] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284-302, 2009.

[48] R. Storn, "Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report, International Computer Science Institute, vol. 11, 1995.

[49] S. M. Venske, R. A. Gonçalves, and M. R. Delgado, "ADEMO/D: Multiobjective optimization by an adaptive differential evolution algorithm," *Neurocomputing*, vol. 127, pp. 65-77, 2014.

[50] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer Science & Business Media, 2006.

[51] A. P. Engelbrecht, "Particle swarm optimization," in *Computational Intelligence: An Introduction*, 2nd ed., pp. 289-358, 2007.

[52] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation*, vol. 13, no. 5, pp. 945-958, 2009.

[53] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2005, vol. 2, pp. 1785-1791.

[54] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing*, vol. 11, no. 7, pp. 617-629, 2007.

[55] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646-657, 2006.

[56] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, vol. 10, no. 10, pp. 293-298, 2002.

[57] T. Kirdphoksap and P. Chutima, "Many-objective car sequencing problem on mixed-model two-sided assembly lines," *The Journal of KMUTNB*, vol. 28, no. 4, pp. 717-735, 2018.

[58] Q. Lin, Q. Zhu, P. Huang, J. Chen, Z. Ming, and J. Yu, "A novel hybrid multi-objective immune algorithm with adaptive differential evolution," *Computers & Operations Research*, vol. 62, pp. 95-111, 2015.

[59] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2391-2404, 2014.

[60] J. Bartholdi, "Balancing two-sided assembly lines: A case study," *International Journal of Production Research*, vol. 31, no. 10, pp. 2447-2461, 1993.

[61] Y. K. Kim, Y. Kim, and Y. J. Kim, "Two-sided assembly line balancing: a genetic algorithm approach," *Production Planning & Control*, vol. 11, no. 1, pp. 44-53, 2000.

[62] U. Özcan and B. Toklu, "Multiple-criteria decision-making in two-sided assembly line balancing: A goal programming and a fuzzy goal programming models," *Computers & Operations Research*, vol. 36, no. 6, pp. 1955-1965, 2009.

[63] P. Chutima and P. Chimklai, "Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 39-55, 2012.