European Journal of Operational Research 205 (2010) 346–360

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

A water flow-like algorithm for manufacturing cell formation problems

Tai-Hsi Wu [a,*], Shu-Hsing Chung [b], Chin-Chih Chang [b]

[a] Department of Business Administration, National Taipei University, 151, University Road, San Shia, Taipei 407, Taiwan
[b] Department of Industrial Engineering and Management, National Chiao Tung University, 1001, Ta Hsueh Road, Hsinchu 300, Taiwan

ARTICLE INFO

Article history:
Received 29 June 2008
Accepted 13 January 2010
Available online 18 January 2010

Keywords:
Manufacturing
Heuristics
Cell formation
Meta-heuristics

ABSTRACT

Available research on the manufacturing cell formation problem shows that most solution approaches are either single- or multiple-solution-agent-based, with a fixed size of solution agents. Frequent problems encountered during the process of solving the cell formation problem include solutions being easily trapped in local optima and bad solution efficiency. Yang and Wang [Yang, F.-C., Wang, Y.-P., 2007. Water flow-like algorithm for object grouping problems. Journal of the Chinese Institute of Industrial Engineers, 24 (6), 475–488] proposed the water flow-like algorithm (WFA) to overcome the shortcomings of single- and multiple-solution -agent-based algorithms. WFA has the features of multiple and dynamic numbers of solution agents, and its mimicking of the natural behavior of water flowing from higher to lower levels coincides exactly with the process of searching for optimal solutions. This paper therefore adopts the WFA logic and designs a heuristic algorithm for solving the cell formation problem. Computational results obtained from running a set of 37 test instances from the literature and newly created have shown that the proposed algorithm has performed better than other benchmarking approaches both in solution effectiveness and efficiency, especially in large-sized problems. The superiority of the proposed WFACF over other approaches from the literature should be attributed to the collaboration of the WFA logic, the proposed prior estimation of the cell size, and the insertion-move. The WFA is a novel heuristic approach that deserves more attention. More attempts on adopting the WFA logic to solve many other combinatorial optimization problems are highly recommended.

1. Introduction

Manufacturing systems have become more efficient and productive with the application of group technology (GT) within manufacturing environments. GT groups parts with similar design characteristics or manufacturing characteristics into part families. One application of GT is cellular manufacturing (CM). By adopting GT in CM, benefits such as reduced inventory, reduced capacity, reduced labor and overtime costs, shorter manufacturing lead times, and faster response to internal and external changes including machine failures, product mix, and demand changes are realized. The entire production system is decomposed into production cells through CM. Machines are then assigned to these cells to process one or more part families so that each cell operates independently and intercellular movements are minimized or the number of part flows processed within cells is maximized.

Cell formation (CF) is one of the most important steps in CM. It becomes difficult to obtain optimal solutions in an acceptable amount of time, especially for problems with large sizes. Extensive research has been devoted to CF problems, and various methods have been proposed to identify machine cells and part families. These methods fall into five categories: array-based, hierarchical/non-hierarchical clustering, graph-based, mathematical programming, and heuristics/meta-heuristics.

Array-based methods obtain visible groupings of machines and parts until a satisfactory solution is found by repeatedly rearranging the rows and columns of the machine–part matrix. These methods include the bond energy algorithm (BEA) by McCormick et al. (1972), the rank order clustering (ROC) algorithm by King (1980), the ROC2 by King and Nakornchai (1982), the direct clustering algorithm (DCA) by Chan and Milner (1982), and the close neighbor algorithm (CAN) by Boe and Cheng (1991). Hierarchical/Non-hierarchical clustering methods in studies by McAuley (1972), Mosier and Taube (1985b), Seifoddini and Wolfe (1987), and Yasuda and Yin (2001) use a measure of similarity or dissimilarity for the grouping of machines or parts. Yin and Yasuda (2006) discussed the similarity coefficients developed to date for use in solving the CF problem. In graph-based approaches, the process of forming manufacturing cells starts by collecting the problem data and then converting them into a weighted graph representation. In these approaches, nodes represent machines and arcs represent their relationships, defined as the value of total part flow

* Corresponding author. Tel.: +886 2 86746574; fax: +886 2 86715912.
E-mail addresses: taiwu@mail.ntpu.edu.tw (T.-H. Wu), shchung@mail.nctu.edu.tw (S.-H. Chung), chinju.chang@gmail.com (C.-C. Chang).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.01.020

between machines. Complicated features of the CF problems have been incorporated in models using mathematical programming approaches, but these usually become computationally intractable especially for large-sized problems. Kusiak (1987) presented a *p*-median model, while Boctor (1991) suggested a linear 0–1 formulation for the CF problem. However, the classical *p*-median model is limited to small-sized CF problems since it requires many binary variables. Won and Lee (2004) hence proposed two modified *p*-median formulations to resolve this difficulty. Chen and Heragu (1999) proposed two stepwise decomposition approaches to solve large-sized CF problems. Both approaches analyze the part–machine relation, decompose the original problem into several large subsystems, and solve each subsystem using optimal solution technique.

Meta-heuristic algorithms such as tabu search, genetic algorithm (GA), simulated annealing (SA), and neural networks all comprise another class of search methods that have been adopted to solve the CF problem and its variants efficiently due to their excellent performance in solving combinatorial optimization problems. Sun et al. (1995) presented a short-term tabu search-based algorithm for solving the CF problem which aimed to minimize the intercellular parts flows. Wu et al. (2004), on the other hand, maximized the parts flow within cells using a long-term tabu search-based algorithm. Cao and Chen (2004) developed a tabu search considering cell setup cost to solve the CF problem. Cheng et al. (1998) formulated a traveling salesman problem (TSP) to address the CF problem and proposed a solution methodology based on GA, while a hierarchical clustering approach based on genetic programming was presented by Dimopoulos and Mort (2001). Onwubulo and Mutingi (2001) developed a GA that accounts for intercellular movements and cell-load variation. Gonçalves and Resende (2004) reported promising results with a hybrid algorithm that combined a local search and a GA. Instead of the generally used simple machine encoding, Filho and Tiberti (2006) proposed a GA based on group encoding. The CF problem was addressed by James et al. (2007) using a hybrid grouping GA that combines a local search with a standard grouping GA to form machine–part cells. Wu et al. (2008) proposed a simple yet effective simulated annealing-based algorithm (SACF), for solving the CF problems where singletons (cells having less than two parts or two machines) are allowed. Yang and Yang (2008) proposed a modified ART1 neural learning algorithm, in which the vigilance parameter can be simply estimated by the data so that it is more efficient and reliable compared with previous neural network approaches.

Among the aforementioned heuristic algorithms, the SA and the tabu search are single-solution-agent-based algorithms. A single-agent-based heuristic algorithm searches the solution space step by step through the usage of systematic or random neighborhood exploration. Some of them employ adaptive memory, while some of them are memoryless. The GA, however, belongs to the group of multiple-solution-agent-based algorithms, which starts the optimization with a set of possible solutions, not only one possible solution. The behavior of GAs is characterized by a balance between exploitation and exploration in the search space. The balance is strongly affected by strategy parameters such as maximum generation and population size, and GA's performance depends very much on details of the settings of these parameters. Fixed parameters are usually used in most of the GA applications. Since the GA is an intrinsically dynamic and adaptive process, the use of constant parameters is thus in contrast to the general evolutionary spirit. Therefore, it is natural to try to modify the values of parameters during the run of the algorithm (Gen and Cheng, 2000). Yang and Wang (2007) stated that neither the single nor the multiple agent method is agile enough to conduct an efficient and effective solution search. They hence proposed a methodology using a dynamic size of solution agents, the water flow-like algorithm (WFA).

The design of the WFA method was inspired by water flowing from higher to lower levels, where a flow will split into multiple subflows when it moves through uneven terrains. Conversely, subflows merge when they meet at the same level. Water flow ceases and stagnates at the lowest depressions, when momentum cannot expel water out of the depressions. Water flowing is analogous to problem solving. A flow is regarded as a solution agent, the solution space of a problem is the geographical terrain, and the altitude of a flow represents the objective function value. Since the number of flows dynamically changes in this method, it is an agent population-varying method.

To our knowledge, the WFA has not been applied to solve any other combinatorial optimization problems aside from the bin packing problems studied in Yang and Wang (2007). Besides, the solution process of searching for the optima is so analogous to water moving to a lower position. The above two points motivate our using WFA for solving the CF problem.

In this paper, we adopt the WFA logic and develop a heuristic algorithm (WFACF) for the CF problem. This approach combines a WFA using specifically tailored operations with a similarity coefficient method constructed for generating quick initial solutions for later improvement, and two solution improving strategies for finding the best neighborhood solution. We test 37 problems from the literature and newly created, and provide comparisons against several algorithms from the literature. The WFACF is shown to perform better than other benchmarking approaches both in solution effectiveness and efficiency, especially in large-sized problems. According to our knowledge, this is the first time WFA has been used to successfully solve combinatorial optimization problems aside from the bin packing problems. Apart from the above contribution, we further verify the effects of the two primary operations of the WFA, namely, the evaporation and the precipitation, and conclude that they may not be as decisive as they were claimed in the original WFA.

The remainder of the paper is organized as follows. Section 2 describes the problem definition while Section 3 details the proposed solution algorithm for the CF problem. Section 4 shows the computational results on test problems adopted from the literature and includes thorough analyses and discussions. The conclusions are laid out in Section 5.

2. Cell formation problem

CF in a given 0–1 machine–part incidence matrix involves a rearrangement of rows and columns of the matrix to create part families and machine cells. In this research, we attempt to determine a rearrangement minimizing intercellular movement and maximizing the utilization of the machines within a cell. Fig. 1 gives a sample solution matrix for a CF problem, in which two blocks can be observed along the diagonal of the solution matrix.

There have been several measures of goodness of machine–part groups in CM in the literature. Two measures frequently used are grouping efficiency (Chandrashekharan and Rajagopalan, 1986a) and grouping efficacy (Kumar and Chandrasekharan, 1990). Grouping efficiency η is defined as follows:

		P2	P3	P5	P1	P4
Cell 1	M2	1	1	1	0	0
	M4	1	1	1	0	0
Cell 2	M1	0	0	0	1	1
	M3	0	1	0	1	1
	M5	0	0	0	1	0

Fig. 1. Sample solution matrix.

$$\eta = q\eta_1 + (1-q)\eta_2,$$

where η_1 is the ratio of the number of 1s in the diagonal blocks to the total number of elements in the diagonal blocks of the final matrix, η_2 is the number of 0s in the off-diagonal blocks to the total number of elements in the off-diagonal blocks of the final matrix, and q is a weight factor. Any 1s outside the diagonal blocks are called "exceptional elements," and any 0s inside the diagonal blocks are called "voids."

Although grouping efficiency has been widely used, critics argue that in some cases, the size of the matrix impairs its discrimination ability. To overcome this problem, Kumar and Chandrasekharan (1990) proposed another measure, grouping efficacy Γ, which is defined as:

$$\Gamma = \frac{e - e_0}{e + e_v},$$

where e is the total number of 1s in the matrix, e_0 is the total number of exceptional elements, and e_v is the total number of voids. Grouping efficacy ranges from 1 to 0, with 1 as the perfect grouping. As grouping efficacy has been widely accepted in recent studies regarding the CF problem, it is used as the performance measure for the proposed algorithm in this study.

3. Water flow-like approach for the CF problem

This paper adopts the WFA logic and designs a heuristic algorithm for solving the CF problem because the WFA has the features of multiple and dynamic numbers of solution agents. Adopted behaviors from fluid flows of the WFA are introduced in Section 3.1, followed by the proposed heuristic algorithm. Construction of the initial solutions is given in Section 3.2, strategies and procedures for improving the solutions through neighborhood searching are presented in Section 3.3, while the complete algorithm of the proposed WFACF is described in detail in Section 3.4.

3.1. Water flow-like algorithm

The design of the WFA method (Yang and Wang, 2007) was inspired by the natural behavior of water flowing from higher to lower levels. On the earth's surface, a flow will split into multiple subflows when rugged terrains are traversed. Subflows, however, will merge when they arrive at the same location. Governed by gravity and driven by fluid momentum, flows can run to higher levels or run over bumps to navigate various terrains. Water flowing will cease and stagnate at locally or globally lowest depressions, when the momentum left cannot expel the water out of the depressions. As the solution space of a problem can be mapped to the geographical terrain, and the objective value is mapped to the altitude, each flow can then be regarded as a solution agent. Water moving to a lower position can be considered as a solution searching for the optima. The solution search process is thus modeled as water flowing.

Yang and Wang (2007) have adopted several natural behaviors of water flow in presenting the WFA (Dougherty and Marryott, 1991). Their design ideas are summarized as follows:

1. Driven by gravity and governed by the energy conservation law, water will constantly flow to lower altitudes. Therefore, the solution search will recursively move from inferior to superior solutions.
2. Fluid momentum drives water moving forward through rough terrains. A flow will split into subflows when it encounters rugged terrain and when its momentum exceeds a base amount for splitting. WFA simulates this behavior as an agent forking operation; that is, more than two agents are derived from a single agent. A flow with larger momentum will generate more streams of subflows than one with less momentum. A flow with limited momentum will yield to the landform and maintain a single flow. Therefore, the fluid momentum of a flow is recalculated to determine the number of subflows that can be forked after each move.
3. Water flows to lower altitudes and occasionally swells to higher altitudes as long as the kinetic energy is larger than the required potential energy. To avoid being trapped within a local minimum, WFA allows the water to flow to a worse location to broaden the exploration area, provided it has enough kinetic energy.
4. A number of flows merge into a single flow when they meet at the same location. WFA reduces the number of solution agents when multiple agents result in the same objective value to avoid redundant searches.
5. Water flows are subject to water evaporation in the atmosphere. The evaporated water will return to the ground during rainfall. In WFA, a part of the water flow is manually removed to mimic water evaporation. After evaporation, a precipitation operation is implemented in WFA to simulate natural rainfall and explore a wider solution area.

On the basis of the above idea, the computational flow of the WFA consists of four primary operations: (1) flow splitting and moving, (2) flow merging, (3) water evaporation, and (4) precipitation. Before proceeding to the descriptions of these four operations, we introduce some notations.

G	iteration limit
W_0	initial mass of original flow
W_i	mass of flow i
V_0	initial velocity of original flow
V_i	velocity of flow i
T	base momentum
$\bar{n}$	upper limit on the number of subflows split from a flow
n_i	number of subflows forked from flow i
N	total number of water flows in the current iteration
X_i	solution corresponding to flow i
U_{ik}	solution corresponding to subflow k split from flow i
w_{ik}	mass of subflow k split from flow i
μ_{ik}	velocity of subflow k split from flow i
δ_{ik}	altitude drop from flow i to its subflow k; equivalently, changes in objective value from solution i to its neighborhood solution k
g	gravitational acceleration
t	a prescribed number of iterations a flow will be removed completely by evaporation

3.1.1. Flow splitting and moving operation

It is assumed that there is only one water flow in the beginning of the WFA, and that its location is randomly generated. Driven by fluid momentum and potential energy, the flow starts to move to new locations to explore the solution space for new and better solutions. Yang and Wang (2007) used constant-step movement to the best neighborhood solution when solving the object grouping problem in their paper. However, various flow-moving strategies can be designed and applied depending on the characteristics of different optimization problems.

In the WFA, flow splitting results from capable momentum, and a flow with higher momentum generates more subflows than a lower one. The locations of the split subflows are derived from the neighboring locations of the original flow. When a flow does not split, it goes on as a single stream to the best feasible neighboring location. Let N be the number of water flows in the current iteration, the number of subflows n_i forked from flow i $(i = 1, 2, \ldots, N)$ is determined by its momentum, W_iV_i. A flow with zero momentum stays where it is and is considered a stagnant solution. A flow can split into subflows only when its momentum exceeds a predefined base momentum *T*. The number of subflows is determined by dividing its momentum by the base momentum *T*. If the momentum of a flow is between 0 and *T*, it is treated as a single stream moving to a new location without splitting. As the process of the WFA proceeds, it is possible that the number of subflows grows exponentially and exhausts the computational resource. Yang and Wang (2007) suggests imposing an upper limit, $\bar{n}$, on the number of subflows forked from a flow at each iteration. The number of subflows split from a flow can thus be obtained through the formula below:

$$n_i = \min\left\{\max\left\{1,\ \mathrm{int}\left(\frac{W_iV_i}{T}\right)\right\}, \bar{n}\right\}. \tag{1}$$

When the flow is split into subflows, its original mass has to be accordingly distributed to subflows based on the rule designed. Yang and Wang (2007) distribute the mass based on the ranks of the subflows, as shown in Eq. (2)

$$w_{ik} = \left(\frac{n_i + 1 - k}{\sum_{r=1}^{n_i} r}\right) W_i, \quad k = 1, 2, \ldots, n_i. \tag{2}$$

For instance, if $W_i = 5$ and $n_i = 3$, then

$$w_{i1} = \left(\frac{3}{1+2+3}\right)5, \quad w_{i2} = \left(\frac{2}{1+2+3}\right)5, \quad w_{i3} = \left(\frac{1}{1+2+3}\right)5.$$

The velocity of each subflow is computed from the equation of energy conservation. μ_{ik}, the velocity of subflow *k* split from flow *i*, is:

$$\mu_{ik} = \begin{cases} \sqrt{V_i^2 + 2g\delta_{ik}}, & \text{if } V_i^2 + 2g\delta_{ik} > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where *g* is the gravitational acceleration, and δ_{ik} is the altitude drop from flow *i* to its subflow *k*, that is, the improvement of objective value moving from current solution *i* to its neighborhood solution *k*. When $V_i^2 + 2g\delta_{ik} < 0$, the momentum delivered to subflow *k* has been used up, implying that this subflow will stagnate in its current location (solutions trapped in local optima) with no splitting and movement. Such stagnant flow will gradually evaporate into the atmosphere, returning to the ground by precipitation later on.

At the end of the splitting and moving operation, the original flow is hence discarded because subflows have been generated. Information regarding the current number of subflows and solutions sets will then be recorded.

3.1.2. Flow-merging operation

When more than two flows move to the same location, they will merge into one flow with a bigger mass and momentum. Whether a flow shares the same location with others in the WFA is thus systematically examined. If a flow does share the same location, the latter flow is then merged into the former one. Assuming it is found that flows *i* and *j* share the same location, then flow *j* will be deleted and the mass and velocity of flow *i* will be updated as follows:

$$W_i = W_i + W_j, \tag{4}$$

$$V_i = \frac{W_iV_i + W_jV_j}{W_i + W_j}. \tag{5}$$

Using the flow-merging operation, the WFA reduces the number of solution agents when multiple agents result in the same objective value to avoid redundant searches.

3.1.3. Water evaporation operation

It is natural that water evaporates and returns to the ground through precipitation after possible movement from its original location. Water evaporation and precipitation coincide with the "escaping from local optima" mechanism many heuristic algorithms nowadays use to avoid being trapped and to explore more solution spaces.

Each flow in the WFA is subject to water evaporation, where part of the water evaporates into the atmosphere. It is determined that a flow will be completely removed after a prescribed number of iterations, *t*; that is, the masses of all flows are decreased by the ratio of *1/t* as shown in Eq. (6), every time evaporation occurs.

$$W_i = \left(1 - \frac{1}{t}\right)W_i, \quad i = 1, 2, \ldots, N. \tag{6}$$

3.1.4. Precipitation operation

When water vapor accumulates to a certain volume, it will return to the ground in some form such as rain. In the original WFA, two types of precipitation are performed to simulate the natural cycle of water, enforced and regular precipitation.

Enforced precipitation is applied when all flows are grounded with zero velocities. Under this circumstance, all flows are forced to evaporate into the atmosphere and then returned to the ground without changing the number of current flows. However, the locations of these returned flows are deviated stochastically from the original ones. Mass of W_0 is proportionally distributed to flows based on their original mass with the same initial velocity. Consequently, the mass assigned to flow *i*, W_i', can be determined using Eq. (7)

$$W_i' = \left(\frac{W_i}{\sum_{k=1}^{N} W_k}\right)W_0. \tag{7}$$

Regular precipitation is applied periodically in balance with water evaporation. The regular precipitation operation is performed every *t* (same *t* value as in evaporation) iterations to pour down the evaporated water. Note that the cumulated mass of the evaporated water is $W_0 - \sum_{k=1}^{N} W_k$. Thus, instead of using Eq. (7), the mass assigned to flow *i*, W_i', is determined using Eq. (8) when applying regular precipitation. The newly poured flow joins the current solution set, thus increasing the number of current solutions. In addition, both the enforced and regular precipitation might generate several new flows on the same locations. A flow-merging operation will be executed to eliminate possible redundant flows

$$W_i' = \frac{W_i}{\sum_{k=1}^{N} W_k}\left(W_0 - \sum_{k=1}^{N} W_k\right). \tag{8}$$

WFA has the features of multiple and dynamic numbers of solution agents, and its mimicking of the natural behavior of water flowing from higher to lower levels coincides exactly with our process of searching for optimal solutions. This paper therefore adopts the WFA logic and designs a heuristic algorithm for solving the CF problem.

3.2. Construction of initial solutions

A large number of similarity coefficients methods (SCM) have been proposed for the CF problem. In this study, due to their simplicity and easy implementation, the SCM approach is used to generate quick initial solutions for later improvement by the proposed algorithm. It is well known that decomposing an originally difficult problem into several subproblems usually increases problem-solving efficiency. An intuitive solution approach is to decompose the entire problem into two subproblems that deal with the assignment of machines and parts, respectively, since the CF problem considers the grouping of machines and parts. In our construction of the initial solution, machines assignment is determined in the first stage, while the assignment of parts follows in the second stage.

Our approach for generating initial solutions consists of three steps: (1) compute similarity values between machine pairs and construct a similarity matrix; (2) use a clustering rule to process the values in the similarity matrix and form machine cells; and (3) assign parts to machine cells using a parts assignment procedure.

3.2.1. Machines assignment

As in McAuley's research (McAuley, 1972), Jaccard's similarity measure is used to evaluate the similarity between machines as an important index for assigning machines to cells in this subproblem. The similarity measure, denoted S_{ij}, is defined as $S_{ij} = \frac{a_{ij}}{a_{ij}+b_{ij}+c_{ij}}$, where a_{ij} represents the number of parts processed by both machines i and j; while b_{ij} is the number of parts processed by machine i but not by machine j, and c_{ij} is the number of parts processed by machine j but not by machine i. After calculating the similarity matrix for each pair of machines, we are able to generate the initial machines assignment by using the following greedy rule: the higher the similarity measure a pair of machines has, the higher the priority its machines have for placement in the same cell. This process is repeated until all machines have been assigned to cells.

Considering the sample machine–part matrix in Fig. 2a, the corresponding similarity matrix for machines is displayed in Fig. 2b. Assuming that two cells are to be formed, the largest coefficient in the matrix of Fig. 2b is 0.67, indicating that machines 2 and 4 must be assigned to the same cell, for instance cell 1. We proceed with the second-largest coefficient in the matrix, 0.5, appearing in pairs (1,3) and (1,5). As these three machines do not have any relationship with any machines in cell 1, together they should be assigned to the next cell, cell 2. Fig. 3 shows the machines assignment using the proposed greedy rule.

3.2.2. Parts assignment

In this procedure, the parts are assigned to machine cells so that the number of voids and exceptional elements – major components comprising the formula of grouping efficacy – are explicitly considered. It is summarized as follows:

Step 1. *Read the results of machines assignment.*
Step 2. *For each part, find the cell to which a part assignment will result in the least sum of number of exceptional elements and the number of voids. If a tie happens, assign the part to a cell with the least number of voids.*
Step 3. *Repeat Step 2 until all parts have been assigned to cells.*

Results of machines assignment shown in Fig. 3 are used to demonstrate the above procedure. After calculating the sum of voids and exceptional elements for each part-cell combination, it can be observed in Fig. 4 that parts 2, 3 and 5 are assigned to cell 1, while parts 1 and 4 are assigned to cell 2. The initial solution matrix for this CF problem can thus be obtained.

		P1	P2	P3	P4	P5
Cell 1	M2	0	1	1	0	1
	M4	0	1	1	0	0
Cell 2	M1	1	0	0	1	0
	M3	1	0	0	0	0
	M5	0	0	0	1	0

Fig. 3. Assignment of machines.

3.3. Searching neighborhood solutions (flow splitting and moving)

The initial solution just obtained can be considered as the starting water flow, ready to split iteratively into subflows, traverse the solution space, and move toward the optimal solution through the WFA logic.

Among the four operations in the WFA, flow splitting and moving operation is endowed with the mission of searching for better neighborhood solutions and ultimately, the optimal solution. In solving the CF problem, we have designed two strategies, namely, machine shifting and insertion-move, for finding the best neighborhood solution of the current solution.

3.3.1. Machine shifting

In the WFA, the split subflows' locations are derived from the neighboring locations of the original flow. In terms of the CF problem, the initial solution matrix obtained in Section 3.2 is the original water flow. Before designing a strategy for finding a good neighborhood solution, it is crucial to identify and analyze factors influencing solution efficiency and quality. In Section 3.2, after machines assignment has been determined, a simple procedure for assigning parts follows. The procedure for assigning machines actually leads the procedure for assigning parts in this study. The solution quality of machines assignment thus plays a very critical role in the success of the entire solution quality. Hence, a "machine shifting" strategy is proposed to find a rough direction for the neighborhood solutions in the first stage promptly; the exact location for the best neighborhood solution is then obtained through the "insertion-move" strategy in the second stage.

The machine-shifting strategy is implemented by reassigning a machine to any cells other than the current one based on a prescribed probability r. For each machine in the current solution, a random number from (0,1) is first drawn. If the value is greater than r, then the machine is assigned to other cells; otherwise, it stays in the current cell. If singletons are not allowed, an additional check is performed after the machine shifting. The procedure of machine-shifting strategy in pseudo-code format is described in Fig. 5.

3.3.2. Insertion-move

The machine-shifting strategy only gives a quick and rough direction of the location of the neighborhood solutions. To assure that high-quality neighborhood solutions can be found at each iteration of the algorithm, a deliberate strategy has to be developed to

	P1	P2	P3	P4	P5
M1	1	0	0	1	0
M2	0	1	1	0	1
M3	1	0	0	0	0
M4	0	1	1	0	0
M5	0	0	0	1	0

(a) machine-part matrix

	M1	M2	M3	M4	M5
M1	-	0	0.5	0	0.5
M2		-	0	0.67	0
M3			-	0	0
M4				-	0
M5					-

(b) similarity matrix for machines

Fig. 2. Machine–part matrix and corresponding similarity matrix for machines.

		P2	P3	P5	P1	P4
Cell 1	M2	1	1	1	0	0
	M4	1	1	0	0	0
Cell 2	M1	0	0	0	1	1
	M3	0	0	0	1	0
	M5	0	0	0	0	1

Fig. 4. Initial solution matrix obtained.

```
// Machine shifting procedure

// $x_i$ : cell number to which machine $i$ is assigned
// $CS(d)$ : number of machines in cell $d$
// $C^*$: cell size
// $U$: uniform distribution
// $M$: number of machines considered in the CF problem

FOR each machine $i$
{
        Let $x_i = x_i^*$
        Generate a random number $u \in U(0, 1)$
        IF $(u > r)$
            $d = U(1, C^*)$ and $d \neq x_i$
            Let $x_i = d$
}

//Checking for singletons

FOR each cell $d$
{
        Calculate $CS(d)$.
        WHILE $CS(d) < 2$
        {
            Find machine $i$ ($i = U(1, M)$ and $x_i \neq d$ and $CS(x_i) > 2$)
            $CS(x_i) = CS(x_i) - 1$
            $x_i = d$
            $CS(d) = CS(d) + 1$
        }
}
```

Fig. 5. Machine shifting strategy.

find the best neighborhood solution. The neighborhood of a given solution is defined as the set of all feasible solutions reachable by a single move. This study implements an insertion-move, which is an operation that moves a machine i from its current cell k (source cell) to a new cell k' (destination cell). The new move is denoted (k', i). For the insertion-move, a move that results in the most improvement in the objective function value from the current solution is selected, that is:

$$Z(k', i) = \max\{obj^{(k',i)} - obj^{(k,i)}, \forall k, k' \in C, k' \neq k, \ \forall i \in M\},$$

where C and M are the sets for cells and machines, respectively.

Fig. 6 demonstrates the splitting and moving operation we proposed for searching neighborhood solutions for the CF problem. As flow i splits into subflows, and the number of subflows n_i is determined by its momentum, say n_i equals k. The machine-shifting strategy is implemented to determine the rough directions for the k subflows, that is, we now have the locations of $X_{i1}, X_{i2}, \ldots, X_{ik}$. The insertion-move is then performed to find the best neighborhood solution around X_{i1}, that is, the X^*_{i1}. This is repeated until the best neighborhood solution for each of the subflows has been found. Iteration by iteration, these newly generated subflows may merge with others sharing the same location, proceed in a single stream, further split into more subflows at later iterations, or stagnate in the current location until the stopping criteria of the algorithm is met.

The WFA procedure designed is described in detail as follows:

Procedure WFA
Step 1. Initial WFA parameter settings: G, N, W_0, V_0, T.
Step 2. For each flow $i(i = 1, 2, \ldots, N)$, repeats Steps 3–6.
Step 3. Calculate number of subflows n_i for each flow i based on Eq. (1).
Step 4. For each subflow k of flow i, find the best neighborhood solution through the machine-shifting and insertion-move strategies described in Section 3.3.
Step 5. Distribute the mass of flow i to its subflows based on Eq. (2).
Step 6. Calculate the improvement in objective values and update the resulting velocity of subflow k split from flow i based on Eq. (3).
Step 7. Merge subflows with the same objective values and update the resulting mass and velocity based on Eqs. (4) and (5).
Step 8. Update the number of subflows for each flow i.
Step 9. Update the total number of water flow:$N \leftarrow \sum_{i=1}^{N} n_i$.
Step 10. Perform evaporation operation and update the resulting mass and velocity for each water flow based on Eq. (6).
Step 11. Check whether precipitation condition(s) is/are met. If yes, perform Steps 12, 13, and 14; otherwise, go to Step 15.
Step 12. Perform machine-shifting strategy to the current best solution to generate new solutions deviated from the current ones.
Step 13. Distribute the mass to flows poured based on Eq. (7) or (8) depending on the type of precipitation.
Step 14. Check whether the new solution has the same objective value as the other solutions. If yes, merge it and update the resulting mass and velocity based on Eqs. (4) and (5), then update the total number of water flow N.
Step 15. Check whether iteration limit G has been reached. If yes, stop the algorithm; otherwise, increment the iteration counter by 1 and return to Step 2.

Note that in the machine-shifting strategy, if the threshold probability value is set at 0.7, it implies that each machine has a 30% probability of being assigned to other cells. In the above WFA procedure, the machine-shifting strategy is used in Steps 4 and 12, respectively, with different threshold probability values: 0.7 in Step 4 and 0.5 in Step 12. This is because the main purpose of Step 4 is to find some neighborhoods of the current solution, thus the probability of being assigned to other cells is set at a comparatively low value. The purpose of Step 12, on the other hand, is to explore solutions of unvisited regions through the precipitation operation; thus, it becomes necessary to increase the probability of being assigned to other cells to find solutions more deviated from the current best.

When implementing the four operations in the WFA procedure, we made several changes based on our trials, experiences, and observations. First, in the flow splitting and moving operation, there is no upper limit on the number of subflows forked from a flow at each iteration since these numbers are always within a controllable size during the problem-solving process.

Second, the mass of the subflows is determined based solely on the ranks of the subflows (as shown in the example of Section 3.1.1) without considering their respective performances in the original WFA. We hence design a new formula shown in Eq. (9) for assigning mass to each subflows based on the idea that subflows should compete for their masses. That is, subflows with bet-

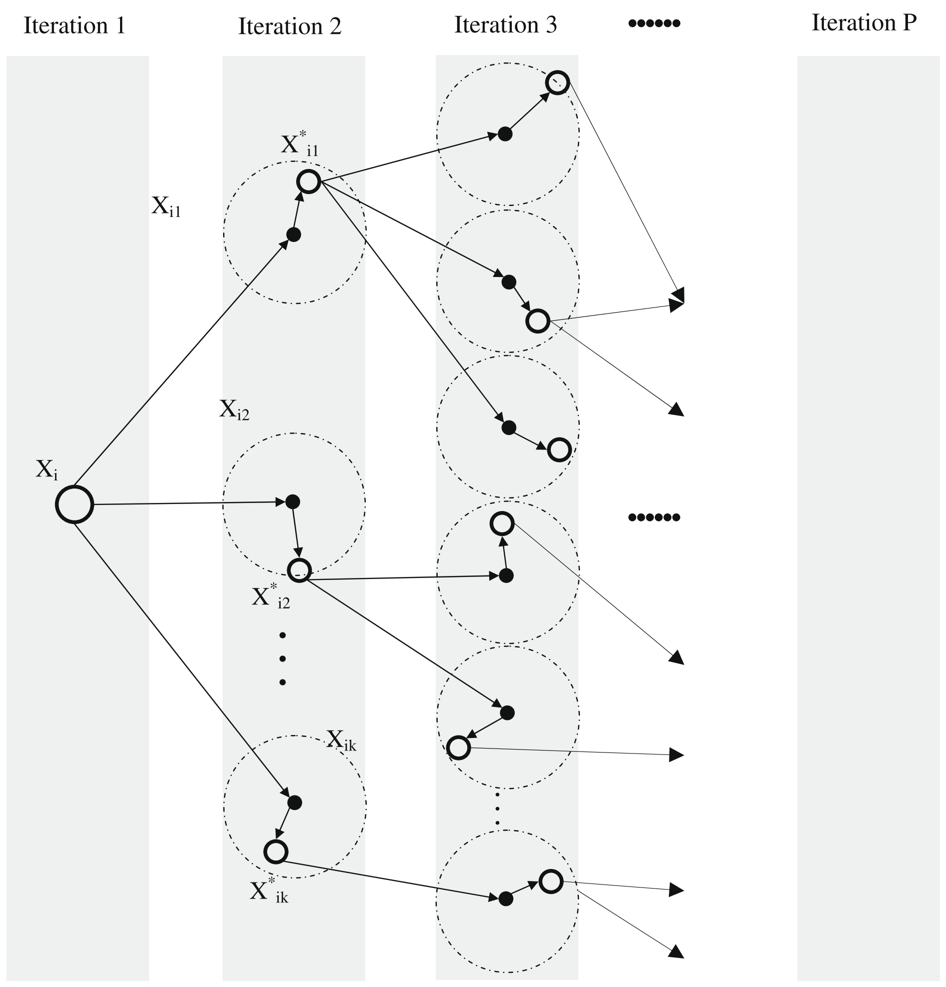

Fig. 6. Proposed flow splitting and moving operation for searching neighborhood solutions.

ter objective values should possess more masses and should stay longer in the water-flowing process.

$$w_{ik} = \left(\frac{E(U_{ik})}{\sum_{k=1}^{n_i} E(U_{ik})}\right) W_i, \tag{9}$$

where $E(U_{ik})$ is the objective value (the grouping efficacy values in the CF problem) of solution U_{ik}.

Third, in addition to the fixed-ratio evaporation presented in the original WFA, another way of evaporation is presented and added in the procedure, the velocity-based evaporation. It can be observed from Eq. (3) that the higher the altitude drop (i.e., the larger improvement in objective value) of a subflow, the larger the velocity it will be rendered to. We define an evaporation ratio conversely related to improvement in velocity, where flows with smaller velocities should be evaporated more quickly than those with larger velocities. The formula is listed below:

$$W_i = (1 - \rho_i) W_i, \tag{10}$$

$$\text{where } \rho_i = \begin{cases} 1, & \text{if } \mu_{ik} = 0, \\ 0, & \text{if } \frac{\mu_{ik}}{V_i} \geqslant 1, \\ 1 - \frac{\mu_{ik}}{V_i}, & \text{if } 0 < \frac{\mu_{ik}}{V_i} < 1. \end{cases}$$

Lastly, two types of precipitation are performed in the original WFA, namely, enforced and regular precipitation. Enforced precipitation is applied when all flows are grounded with zero velocities. However, this has never happened in our solution process of the CF problem. Another precipitation is hence presented and added to the procedure, that is, the moist precipitation. Moist precipitation is used when the mass of the evaporated water flow reaches half of its original total mass, W_0.

The proposed procedure WFA in pseudo-code format is shown in Fig. 7.

3.4. Proposed algorithm WFACF for CF problem

Most of the algorithms designed to solve the CF problem attempt to obtain the machine–part groupings so that some decision objectives, such as the grouping efficiency or the grouping efficacy, can be maximized. However, without the prior determination of the "cell size," the above objectives can hardly be achieved. It is given beforehand in a few cases, but is left to be determined as part of the decision in most. Usually, in the iterative solution process, the initial cell size is set at two and is gradually increased by one. These algorithms are then repeatedly applied until a cell size resulting in the best grouping efficiency/efficacy value has been found. Thus, many computational efforts have to be exerted in order to obtain the optimal cell size. Instead of starting from a beginning number, identifying a good intermediate point for the cell size at the very beginning should save plenty of run time when designing an algorithm to search for the optimal cell size.

Take a test problem from the literature (Carrie, 1973) as an example. The relationship between the cell size and the resulting grouping efficacy is shown in Fig. 8. It is observed that the grouping efficacy value increases as the cell size increases, and the optima is found when cell size equals nine. After that, it starts to decrease as the cell size increases. Similar observations can be found in other test problems.

Hence, we propose a two-stage algorithm, WFACF, for solving the CF problem in this study. In the first stage, feasible solutions without an elaborate solution improvement are generated to derive a cell size quickly, which is then used as input to the second stage to search for the optimal/near-optimal solution through the proposed WFA. We anticipate that the cell size obtained in stage 1 can serve as a good lower bound to start the solution process in stage 2. Hence, a considerable amount of computational efforts can be saved, especially when large-sized problems are solved. Before we explain the solution procedure, additional notations are introduced.

NC	number of cells (cell size)
p_0	initial solution of parts assignment
p^*	incumbent solution of parts assignment of current cell size
p^{**}	best solution of parts assignment so far
m_0	initial solution of machines assignment
m^*	incumbent solution of machines assignment of current cell size
m^{**}	best solution of machines assignment so far

```
//WFA procedure:
WFA parameter settings: the iteration limit $G$, the initial mass $W_0$, the initial velocity $V_0$, the base
momentum T and the initial branch number $N$=1, iteration counter $Q = 0$ .
DO
{
    FOR each flow $i \in \{1,2,...,N\}$
    {
            Calculate the number of sub flow $n_i$ based on equation (1)
            FOR each subflow $k \in \{1,2,...,n_i\}$
            {
                Set machine shifting threshold $r = 0.7$.
                Generate a neighborhood solution using machine shifting procedure.
                Assign parts to machine cells using parts assignment procedure.
                Calculate the grouping efficacy value of the resulting cell configuration.
                DO
                {
                    Find the best neighborhood solution by performing the insertion-move.
                    Assign parts to machine cells using parts assignment procedure.
                    Calculate the grouping efficacy value of the resulting cell configuration.
                    IF current solution is better than the best solution recorded so far,
                        Best = current solution.
                }
                WHILE the grouping efficacy value can still be improved
                Calculate the velocity of subflow $\mu_{ik}$ split from flow $i$ by using equation (3).
            }
            FOR each subflow $k \in \{1,2,...,n_i\}$
                Calculate the mass of sub flow $w_{ik}$ split from flow $i$ using equation (9).
    }
    IF flows have the same solutions
    THEN run flow merging operation and update $W_i$ and $V_i$ using equations (4) and (5).
    Run water evaporation and update the mass of flow $W_i$ by equation (6) or equation (10) .
    Update the total number of flow $N$ by equation (1).
    IF precipitation condition is met
    {
        FOR each flow $i$
        {
            Calculate the masses of the pour-downed flows $W_i$' using equation (8) and let $V_i$'= $V_0$.
            Set machine shifting threshold $r = 0.5$.
            Generate a neighborhood solution using machine shifting procedure.
            Assign parts to machine cells using parts assignment procedure.
            Calculate the grouping efficacy value of the resulting cell configuration.
            IF flows have the same solutions
            THEN run flow merging operation and update $W_i$ and $V_i$ using equations (4) and (5).
            Update the total number of flow $N$.
        }
    }
    Update the iteration counter $Q = Q + 1$
} WHILE $Q<=G$
```

Fig. 7. Pseudocode of proposed WFA procedure.

E_0	grouping efficacy value of cell configuration (m_0, p_0)
E^*	grouping efficacy value of cell configuration (m^*, p^*)
E^{**}	grouping efficacy value of cell configuration (m^{**}, p^{**})

Procedures of both stages are described below.
Stage 1 of WFACF:

Step 1. Read machine–part incidence matrix of test problems, and set $NC = 2, E_0 = E^* = 0$.
Step 2. Compute the similarity values between machine pairs and construct a similarity matrix.
Step 3. Generate initial machines assignment, m_0, by performing the machines assignment procedure in Section 3.2.1.
Step 4. Generate initial parts assignment, p_0, based on m_0 in Step 3, by performing the parts assignment procedure in Section 3.2.2.
Step 5. Calculate the grouping efficacy value E_0 of the resulting cell configuration (m_0, p_0).
Step 6. If $E_0 > E^*$, then set $(m^*, p^*) = (m_0, p_0), E^* = E_0, NC = NC+1$, go to Step 3; otherwise, report current cell configuration (m^*, p^*) and E^* and terminate stage 1.

The solution obtained at the end of stage 1, including the suggested number of cells and cell configurations (m^*, p^*), is then used as the input to stage 2 to search for the optimal/near-optimal solution through the proposed WFA procedure in Section 3.3.

Stage 2 of WFACF:

Step 1. Read solutions from stage 1, including number of cells, (m^*, p^*), and E^*. Set $(m^{**}, p^{**}) = (m^*, p^*), E^{**} = E^*$.
Step 2. Perform the proposed WFA procedure in Section 3.3, obtain solution (m^*, p^*) and the resulting grouping efficacy value E^*.
Step 3. If $E^* > E^{**}$, then set $E^{**} = E^*, (m^{**}, p^{**}) = (m^*, p^*)$, $NC = NC + 1$, go to Step 2; otherwise, report the current best cell configuration (m^{**}, p^{**}) and E^{**} and terminate stage 2.

The procedures of stages 1 and 2 of the proposed WFACF in pseudo-code format are given in Figs. 9 and 10, respectively. After

```
//Stage 2 of WFACF
SET NC=C*
READ results from Stage 1 of WFACF
DO
{
        Run Procedure WFA
        IF current solution is better then the best recorded so far,
            best solution = current solution
            C* = NC
        Increment cell number by 1: NC = NC+1
} WHILE the grouping efficacy value can be improved
SHOW best solution
```

Fig. 10. Procedure of stage 2 of WFACF.

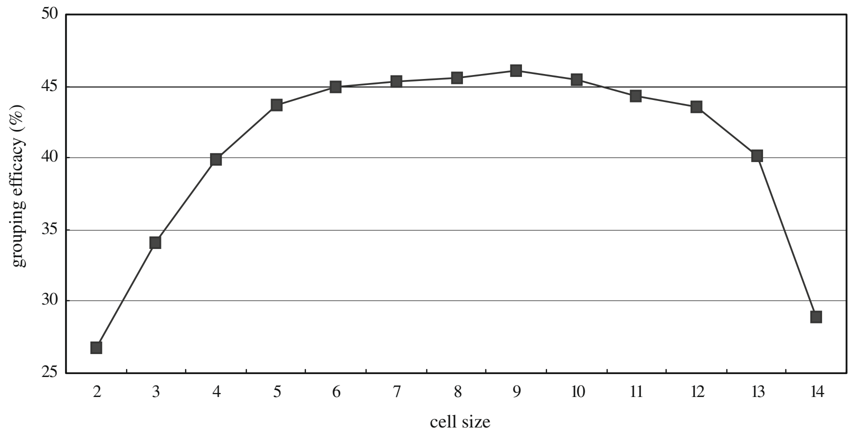

Fig. 8. Relationship between grouping efficacy and cell size.

```
//Stage 1 of WFACF
READ machine-part incidence matrix and let cell number C*=NC=2
Calculate machine similarity matrix
DO
{
        Create machine cells using machines assignment procedure
        Assign parts to machine cells using parts assignment procedure
        Calculate the grouping efficacy value of the cell configuration
        If current solution is better than the best recorded so far, update C* = NC
        Increment number of cell by 1: NC = NC+1
} WHILE the grouping efficacy value can be improved
```

Fig. 9. Procedure of stage 1 of WFACF.

intensive testing, parameters of the WFA, iteration limit G, initial mass W_0, initial velocity V_0, base momentum T and the number of iterations for precipitation t are set at 100, 40, 15, 100, 20 respectively.

4. Computational results and discussion

Thirty-seven test instances from the literature and newly created are used to evaluate the computational characteristics of the proposed heuristic WFACF, the results of which are compared with those of algorithms reported in the literature. Some studies allow the existence of singletons in the solutions and some do not. We follow the studies that do not allow singletons in this research. The matrices of the test problems range from 5×7 to 50×100; they comprise well-structured and unstructured matrices. The first 35 test problems are widely adopted as the test instances in many studies (Gonçalves and Resende, 2004; James et al., 2007; Wu et al., 2008), while problems #36 and #37 are generated by this study. The proposed algorithm WFACF was coded in C and implemented on a Pentium III 933 MHz personal computer with 256 MB random access memory (RAM).

4.1. Computational results

Thirty-five test problems from the literature and two test problems newly created by this study are used to examine the performance of the proposed WFACF algorithm. The results were then compared with the best results found in the literature, that is, the Hybrid-GA of Gonçalves and Resende (2004), where singletons are not allowed. Wu et al. (2008) proposed a simple yet effective simulated annealing-based algorithm (SACF), for solving the CF problems where singletons are allowed. The corresponding results are compared with several well-known algorithms published. Their comparative study shows that the SACF algorithm improves the grouping efficacy for 72% of the test problems. We adopted the SACF as a target for comparison with minor modifications to the program code of SACF to cope with the singletons restriction in this study. Table 1 gives the computational results of these methods. Ten independent runs were performed for each test instance due to the stochastic features that SACF, Hybrid-GA, and WFACF might have.

For the 35 test problems directly adopted from the literature, Tables 1 and 2 show that WFACF produced the best solutions in 34 out of 35 problems, while Hybrid-GA found 26 and SACF found 20. Solution qualities of the initial solutions (solutions obtained at the end of stage 1 of WFACF) are generally good. In 14 of 35 test problems, the initial solutions are exactly equal to the best solutions. In 25 out of 35 test problems, both the WFACF and the Hybrid-GA obtain the same results, while WFACF produced better results than those of Hybrid-GA in nine problems (#16, #20, #21, #26, #29, #30, #31, #33, and #34) and Hybrid-GA has better results than those of WFACF only in problem #19. It can thus be concluded that WFACF performs better than Hybrid-GA and SACF, especially in test problems with larger sizes. Similar conclusion can be made for test problems #36 and #37. All three methods give cell sizes resulting in the best grouping efficacy for each test problem. Cell sizes from the three methods are the same in most test instances, except in problems with larger sizes where minor differences can be seen. Fig. 11 displays the error percentage to the best solutions of the SACF, Hybrid-GA, and WFACF methods, respectively, on 17 test instances, in which there are different grouping efficacies. The result of Fig. 11 matches the observations in Tables 1 and 2.

As for the comparison of run time, no significant difference is observed between the SACF and the WFACF when small or medium-sized problems are solved. The WFACF, however, performs more efficiently than the SACF in large-sized test problems (#31, #32, #33, #35, and #37). Although this study has a lower computational environment (933 MHz) compared with the Hybrid-GA (1.333 GHz), both SACF and WFACF take only a fraction of the run time of Hybrid-GA to obtain comparatively favorable solutions. The superiority in computational efficiency can be easily observed.

4.2. Analyses and discussion

One of the main objectives of this study is to examine in detail the various mechanisms of the WFA and their corresponding effects to the overall solution efficiency and efficacy. Effects of several strategies proposed in this study, together with the evaporation and precipitation operations, are therefore further analyzed in this section.

4.2.1. Effect of prior estimation of cell size

In the first stage of the proposed algorithm WFACF, feasible solutions without an elaborate solution improvement are generated to derive a cell size quickly, which is then used as input to the second stage to search for the optimal/near-optimal solution. It is anticipated that the obtained cell size can serve as a good lower bound to start the solution process in stage 2, saving a considerable amount of computational efforts. An experiment without prior estimation of cell size (skipping stage 1) in the algorithm is conducted and compared with the WFACF. A comparison of results is shown in Fig. 12. On average, the computer run time of WFACF (with prior estimation of the cell size) is 44% less than the case without prior estimation of the cell size. Additionally, the savings are even more significant in large-sized test problems. This finding justifies the effects of stage 1 of the proposed WFACF.

4.2.2. Effects of evaporation, precipitation, and insertion-move

The water evaporation and precipitation operations coincide with the "escaping from local optima" mechanism many heuristic algorithms nowadays possess to avoid being trapped and to explore more solution spaces. Additionally, the insertion-move presented assures that high-quality neighborhood solutions can be found at each iteration of the algorithm. These three factors and their effect on solution qualities are thus considered in an experiment. The evaporation factor has three settings: no evaporation, fixed-ratio evaporation, or velocity-based evaporation; the precipitation factor has three settings: no precipitation, regular precipitation, or moist precipitation; while the insertion-move factor has two options: with or without insertion-move. Legitimate combinations of the above three factors comprise 10 testing scenarios listed in Table 3. In each scenario, 35 test instances of Section 4.1 are computed and the average grouping efficacies over the 35 instances are recorded and analyzed in Fig. 13. The 10 scenarios have been separated into two groups. Scenarios adopting the insertion-move (scenarios #1, #3, #5, #7, and #9, same color in different line types in Fig. 13) result in better objective values and obviously surpass the other group (scenarios #2, #4, #6, #8, and #10, same color in different line types in Fig. 13), which does not include the insertion-move in the algorithm. As expected, Fig. 14 shows that the group with the insertion-move spends more run time searching for the best solutions, especially in large-sized problems (problems #25–35). Insertion-move beats the other two factors and becomes the most significant and dominant factor in terms of solution quality. This implies that the water evaporation and precipitation operations may not be as significant as they were supposed to be in the solution process of CF problem, though considerable efforts have been spent in designing the contents of both operations. The importance of a good neighborhood-searching method, such as

Table 1
Performance of WFACF compared to SACF and Hybrid-GA.

Test instances			SACF (10 replicates)			Hybrid-GA (10 replicates)			WFACF (10 replicates)					
No.	Source	Size	Max	Cell size	CPU time (s)	Max	Cell size	CPU time (s)	Initial	Max	Initial cell size	Cell size	CPU time (s)	St. dev.
1	King and Nakornchai (1982)	5 × 7	73.68	2	0.00	73.68	2	0.53	73.68	73.68	2	2	0.01	0
2	Waghodekar and Sahu (1984)	5 × 7	62.50	2	0.00	62.50	2	0.47	52.00	62.50	2	2	0.00	0
3	Seifoddini (1989)	5 × 18	79.59	2	0.01	79.59	2	0.85	79.59	79.59	2	2	0.01	0
4	Kusiak and Cho (1992)	6 × 8	76.92	2	0.00	76.92	2	0.66	76.92	76.92	2	2	0.01	0
5	Kusiak and Chow (1987)	7 × 11	53.13	3	0.00	53.13	3	1.09	51.52	53.13	3	3	0.01	0
6	Boctor (1991)	7 × 11	70.37	3	0.01	70.37	3	1.35	70.37	70.37	3	3	0.01	0
7	Seifoddini and Wolfe (1986)	8 × 12	68.29	3	0.01	68.29	3	1.44	68.29	68.29	3	3	0.03	0
8	Chandrashekharan and Rajagopalan (1986a)	8 × 20	85.25	3	0.02	85.25	3	1.68	85.25	85.25	3	3	0.04	0
9	Chandrashekharan and Rajagopalan (1986b)	8 × 20	58.41	2	0.02	58.72	2	1.68	58.72	58.72	2	2	0.05	0
10	Mosier and Taube (1985a)	10 × 10	70.59	3	0.01	70.59	3	1.82	70.59	70.59	3	3	0.07	0
11	Chan and Milner (1982)	10 × 15	92.00	3	0.02	92.00	3	2.19	92.00	92.00	3	3	0.08	0
12	Askin and Subramanian (1987)	14 × 24	67.61	6	0.29	69.86	5	6.05	67.12	69.86	5	5	0.25	0
13	Stanfel (1985)	14 × 24	69.33	5	0.19	69.33	5	6.24	69.33	69.33	5	5	0.26	0
14	McCormick et al. (1972)	16 × 24	51.96	6	0.28	51.96	6	7.85	40.94	51.96	3	6	0.95	0
15	Srinivasan et al. (1990)	16 × 30	67.83	4	0.23	67.83	4	10.24	61.33	67.83	4	4	0.69	0
16	King (1980)	16 × 43	55.76	6	1.63	54.86	5	13.92	53.93	55.90	6	6	1.01	0.21
17	Carrie (1973)	18 × 24	53.64	6	0.56	54.46	6	11.34	44.53	54.46	5	6	1.17	0.15
18	Mosier and Taube (1985b)	20 × 20	42.34	5	0.24	42.96	5	10.89	35.19	42.96	2	5	1.53	0.14
19	Kumar et al. (1986)	20 × 23	49.65	5	0.40	49.65	5	12.03	44.27	49.61	6	6	0.85	0.11
20	Carrie (1973)	20 × 35	76.14	4	1.01	76.22	4	15.61	76.14	76.54	4	5	1.59	0.06
21	Boe and Cheng (1991)	20 × 35	56.04	4	0.94	58.07	5	16.38	55.06	58.15	5	5	1.10	0.09
22	Chandrasekharan and Rajagopalan (1989)	24 × 40	100.0	7	1.76	100.0	7	19.26	100.0	100.0	7	7	1.33	0
23	Chandrasekharan and Rajagopalan (1989)	24 × 40	85.11	7	1.91	85.11	7	25.28	85.11	85.11	7	7	2.18	0
24	Chandrasekharan and Rajagopalan (1989)	24 × 40	73.51	7	1.55	73.51	7	26.82	73.51	73.51	7	7	2.11	0
25	Chandrasekharan and Rajagopalan (1989)	24 × 40	51.88	9	4.76	51.97	9	26.48	44.10	51.97	6	10	4.58	0.1
26	Chandrasekharan and Rajagopalan (1989)	24 × 40	44.44	10	6.28	47.06	9	25.97	42.52	47.37	9	10	5.90	0.1
27	Chandrasekharan and Rajagopalan (1989)	24 × 40	44.24	9	5.10	44.87	9	26.06	36.27	44.87	8	10	3.05	0.05
28	McCormick et al. (1972)	27 × 27	53.19	3	0.31	54.27	4	25.90	32.38	54.27	2	4	2.56	0
29	Carrie (1973)	28 × 46	40.97	6	4.05	44.62	9	43.78	39.34	46.06	6	9	23.36	0.11
30	Kumar and Vannelli (1987)	30 × 41	58.58	10	8.08	58.48	11	43.00	53.67	59.52	10	10	5.41	0.12
31	Stanfel (1985)	30 × 50	60.00	12	14.95	59.66	12	52.45	54.55	60.00	9	12	10.26	0
32	Stanfel (1985)	30 × 50	50.51	11	20.42	50.51	11	48.97	43.86	50.51	8	11	10.17	0
33	King and Nakornchai (1982)	36 × 90	39.95	11	185.7	42.64	9	81.46	33.70	46.15	6	12	31.80	0.1
34	McCormick et al. (1972)	37 × 53	58.23	2	1.15	56.42	2	87.66	35.06	59.85	2	3	17.35	0.02
35	Chandrasekharan and Rajagopalan (1987)	40 × 100	84.03	10	161.0	84.03	10	152.1	84.03	84.03	10	10	19.20	0
36	This study	40 × 100	41.67	4	9.96	–	–	–	8.05	46.96	6	6	12.94	3.28
37	This study	50 × 100	43.89	7	65.07	–	–	–	5.08	44.20	7	7	22.58	0.53

Table 2
Comparisons of solution ranking of three methods (for the first 35 test problems).

Solutions ranking	Ranked 1st	Ranked 2nd	Ranked 3rd
SACF	20	10	5
Hybrid-GA	26	6	3
WFACF	34	1	0

the insertion-move we proposed, can never be overemphasized. This conclusion is applicable to any meta-heuristic algorithms. Although the water evaporation and precipitation operations may not be critical in solving the CF problem in this study, we still believe that the water-flow-like logic, with proper design and collaboration in its internal operations, can be applied to solve other combinatorial optimization problems.

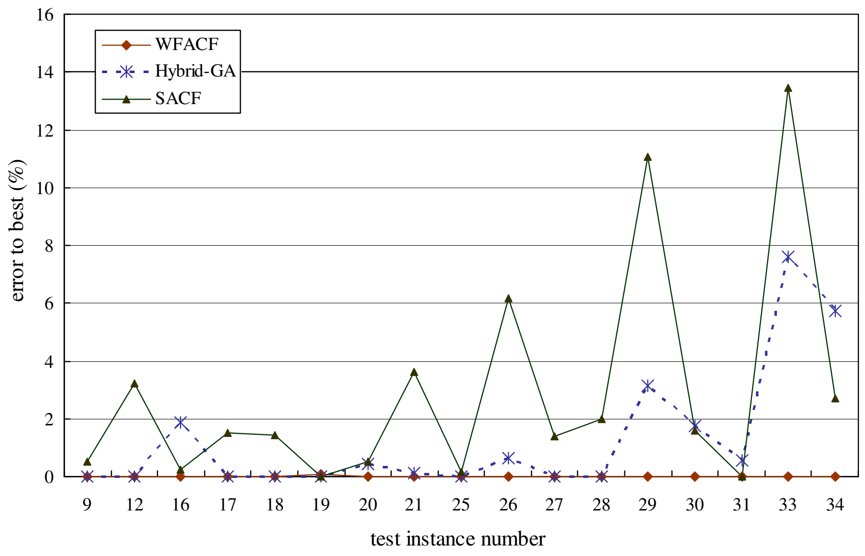

Fig. 11. Comparisons of error percentage to best solutions of SACF, Hybrid-GA, and WFACF.

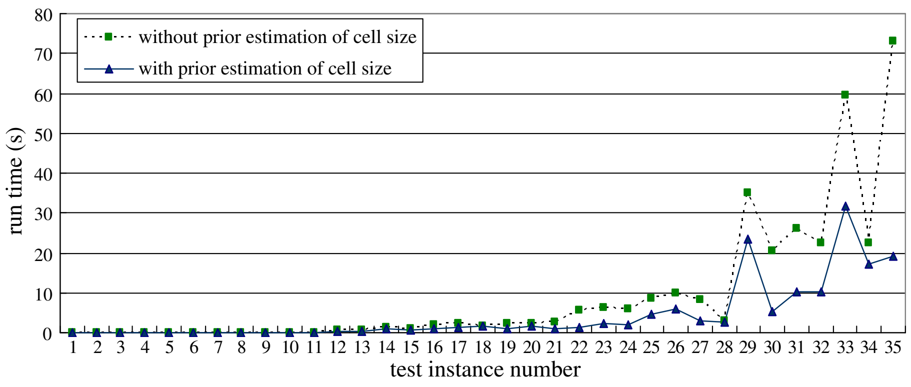

Fig. 12. Run time comparisons of with and without prior estimation of cell size.

Table 3
Experimental testing scenarios.

Scenario #	Evaporation setting	Precipitation setting	Insertion-move option	Mean grouping efficacy (%)
1	Velocity based	Moist	With	64.2832
2	Velocity based	Moist	Without	59.1495
3	Velocity based	Regular ($t = 20$)	With	64.2816
4	Velocity based	Regular ($t = 20$)	Without	59.3005
5	Fixed ratio (0.05)	Moist	With	64.2954
6	Fixed ratio (0.05)	Moist	Without	59.3558
7	Fixed ratio (0.05)	Regular ($t = 20$)	With	64.2625
8	Fixed ratio (0.05)	Regular ($t = 20$)	Without	59.2771
9	No evaporation	No precipitation	With	64.2664
10	No evaporation	No precipitation	Without	59.4040

5. Conclusions

The WFA uses a dynamic size of solution agents and is able to overcome the drawbacks of both single- and multiple-solution-agent-based algorithms. The WFA is a very new meta-heuristic algorithm, and to our knowledge, this is the first time the WFA is applied to solve a combinatorial optimization problem aside from the bin packing problems studied in the original paper of Yang and Wang (2007). This research has adopted the WFA logic and proposed a heuristic algorithm, WFACF, for solving the CF problem to assure good solution quality while avoiding a heavy computational burden.

The similarity coefficients method, together with the proposed machines assignment and parts assignment procedures, has been used to generate quick initial solutions for later improvement by the WFACF. Two strategies, the machine shifting and the

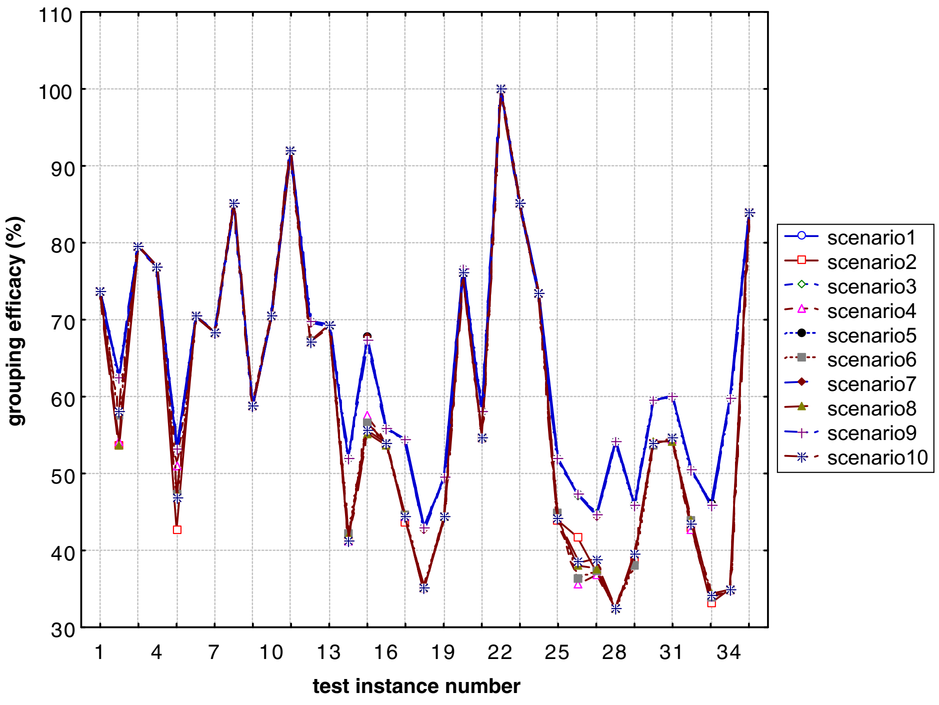

Fig. 13. Comparison of mean grouping efficacy for 10 testing scenarios.

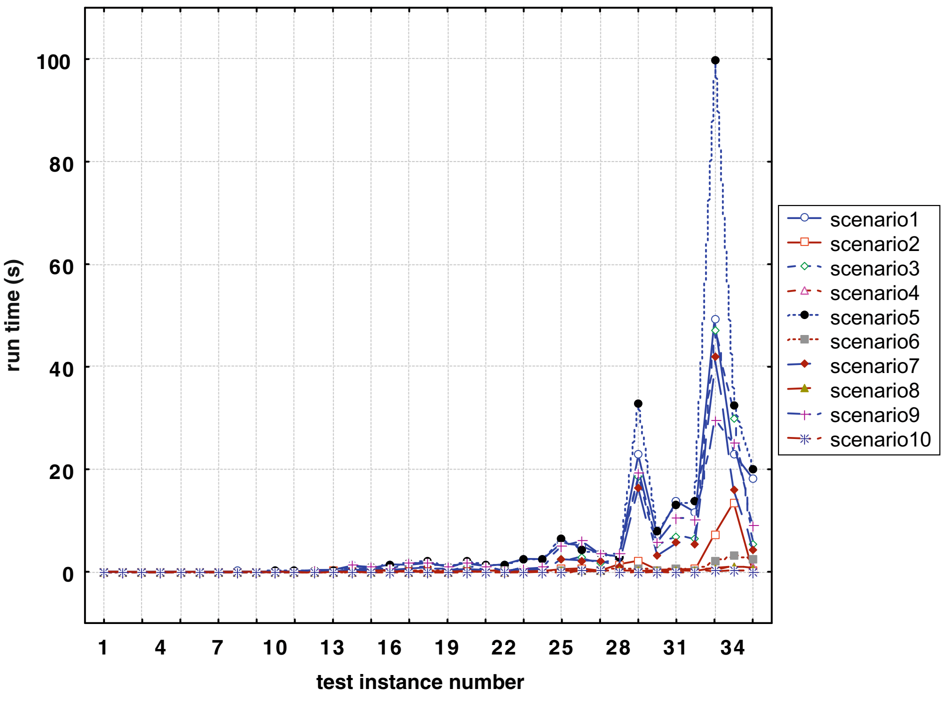

Fig. 14. Comparison of mean run time for 10 testing scenarios.

insertion-move, have been presented for finding the best neighborhood solution around the current solution in the flow splitting and moving operations in the design of WFACF. In addition to the original evaporation and precipitation operation of the WFA, velocity-based evaporation and moist precipitation have been offered in this study.

The proposed WFACF is composed of two stages. In the first stage, feasible solutions without an elaborate solution improvement are generated to derive a cell size quickly, which is then used as input to the second stage in searching for the optimal/near-optimal solution. It is anticipated that the cell size obtained in stage 1 can serve as a good lower bound to start the solution process in stage 2, thus saving a considerable amount of computational efforts especially when large-sized problems are solved. The effect of this ingenious design has been justified through further experiment and analysis.

Computational results obtained from running a set of 35 test instances from the literature have shown that WFACF has produced best solutions in 34 out of 35 problems, while Hybrid-GA has found 26, and SACF has found 20. The proposed WFACF has performed better than the Hybrid-GA and the SACF, especially in test problems with larger sizes. As for the comparison of run time, WFACF has taken only a small fraction of the run time of Hybrid-GA to obtain comparatively favorable solutions. Apart from the performance in solution effectiveness, the superiority of the proposed WFACF in solution efficiency over other approaches from the literature can be easily observed. Further analysis has been conducted to justify the effect of prior estimation of cell size in stage 1 of the WFACF. Computational result has demonstrated that the average run time of WFACF (case with prior estimation of the cell size) is 44% less than the case without prior estimation of the cell size. In addition, the savings are even more significant in large-sized test problems.

We have further verified the effects of the evaporation, precipitation operation, and insertion-move in the WFACF. Legitimate combinations of the above three factors have comprised 10 testing scenarios. In each scenario, 35 test instances were computed and the average grouping efficacies over the 35 instances were recorded and analyzed. The insertion-move has beaten the other two factors and has become the most significant and dominant factor in terms of solution quality. This implies that the water evaporation and precipitation operations may not be as decisive as they were supposed to be in the WFACF solution process of CF problem, though considerable efforts have been spent in designing the contents of both operations.

The contributions of this paper are summarized as follows:

1. We have applied the WFA for solving the CF problem with both solution efficiency and effectiveness outperforming other benchmarking algorithms in the literature. This is the first time WFA has been used to solve combinatorial optimization problems aside from the bin packing problems studied in the original paper.
2. Although the logic of WFA is adopted in this paper, we have specifically designed tailor-made WFA operations for solving the CF problems. For example, in the flow splitting and moving operation, both machine shifting and insertion-move strategies are proposed for finding the best neighborhood solution. In addition, a new formula is presented for making subflows with better objective values possess more masses and stay longer in the water-flowing process. Moreover, in the evaporation operation, in addition to the fixed-ratio evaporation of the original WFA, the velocity-based evaporation is presented and added in the procedure. As for the precipitation operation, the moist precipitation is presented and added to the procedure when the mass of the evaporated water flow reaches half of its original total mass.
3. We have designed a very efficient solution algorithm. In order to find the best solution, the solution algorithms have to be repeatedly applied until a cell size resulting in the best grouping efficiency/efficacy value has been found. Thus, many computational efforts have to be exerted in order to obtain the optimal cell size. In WFACF, prior estimation of cell size is implemented in stage 1 to serve as a good lower bound to start the solution process in stage 2, so that a considerable amount of computational efforts can be saved. Computational result has demonstrated that the average run time of WFACF (case with prior estimation of the cell size) is 44% less than the case without prior estimation of the cell size, and the savings are even more significant in large-sized test problems.

In conclusion, the superiority of the proposed WFACF both in solution effectiveness and efficiency over other approaches from the literature should be attributed to the collaboration of the WFA logic, the proposed prior estimation of the cell size, and the insertion-move. The WFA is a novel heuristic approach that deserves more attention. More attempts on adopting the WFA logic to solve many other combinatorial optimization problems should be a good direction of future research.

References

Askin, R.G., Subramanian, S., 1987. A cost-based heuristic for group technology configuration. International Journal of Production Research 25, 101–113.

Boctor, F., 1991. A linear formulation of the machine–part cell formation problem. International Journal of Production Research 29 (2), 343–356.

Boe, W., Cheng, C.H., 1991. A close neighbor algorithm for designing cellular manufacturing systems. International Journal of Production Research 29 (10), 2097–2116.

Cao, D., Chen, M., 2004. Using penalty function and Tabu search to solve cell formation problems with fixed cell cost. Computers and Operations Research 31, 21–37.

Carrie, S., 1973. Numerical taxonomy applied to group technology and plant layout. International Journal of Production Research 11, 399–416.

Chan, H.M., Milner, D.A., 1982. Direct clustering algorithm for group formation in cellular manufacture. Journal of Manufacturing System 1, 65–75.

Chandrashekharan, M.P., Rajagopalan, R., 1986a. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. International Journal of Production Research 24 (2), 451–464.

Chandrashekharan, M.P., Rajagopalan, R., 1986b. MODROC: An extension of rank order clustering for group technology. International Journal of Production Research 24 (5), 1221–1233.

Chandrasekharan, M.P., Rajagopalan, R., 1987. ZODIAC - An algorithm for concurrent formation of part families and machine cells. International Journal of Production Research. 25 (6), 835–850.

Chandrasekharan, M.P., Rajagopalan, R., 1989. Groupability: An analysis of the properties of binary data matrices for group technology. International Journal of Production Research 27 (6), 1035–1052.

Chen, J.-S., Heragu, S.S., 1999. Stepwise decomposition approaches for large scale cell formation problems. European Journal of Operational Research 113, 64–79.

Cheng, C.H., Gupta, Y.P., Lee, W.H., Wong, K.F., 1998. A TSP-based heuristic for forming machine groups and part families. International Journal of Production Research 36, 1325–1337.

Dimopoulos, C., Mort, N., 2001. A hierarchical clustering methodology based on genetic programming for the solution of simple cell-formation problems. International Journal of Production Research 39, 1–19.

Dougherty, D.E., Marryott, R.A., 1991. Optimal groundwater management. Water Resources Research 27, 2493–2508.

Filho, E.V.G., Tiberti, A.J., 2006. A group genetic algorithm for the machine cell formation problem. International Journal of Production Economics 102, 1–21.

Gen, M., Cheng, R., 2000. Genetic Algorithm and Engineering Optimization. John Wiley and Sons. pp. 8–16.

Gonçalves, J.F., Resende, M.G.C., 2004. An evolutionary algorithm for manufacturing cell formation. Computers and Industrial Engineering 47, 247–273.

James, T.L., Brown, E.C., Keeling, K.B., 2007. A hybrid grouping genetic algorithm for the cell formation problem. Computers and Operations Research 34, 2059–2079.

King, J.R., 1980. Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm. International Journal of Production Research 18 (2), 213–232.

King, J.R., Nakornchai, V., 1982. Machine-component group formation in group technology: Review and extension. International Journal of Production Research 20 (2), 117–133.

Kumar, K.R., Vannelli, A., 1987. Strategic subcontracting for efficient disaggregated manufacturing. International Journal of Production Research 25 (12), 1715–1728.

Kumar, C.S., Chandrasekharan, M.P., 1990. Grouping efficacy: A quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. International Journal of Production Research 28, 233–243.

Kumar, K.R., Kusiak, A., Vannelli, A., 1986. Grouping of parts and components in flexible manufacturing systems. European Journal of Operations Research 24, 387–397.

Kusiak, A., 1987. The generalized group technology concept. International Journal of Production Research 25 (4), 561–569.

Kusiak, A., Cho, M., 1992. Similarity coefficient algorithm for solving the group technology problem. International Journal of Production Research 30, 2633–2646.

Kusiak, A., Chow, W., 1987. Efficient solving of the group technology problem. Journal of Manufacturing Systems 6 (2), 117–124.

McAuley, J., 1972. Machine grouping for efficient production. Production Engineering 51, 53–57.

McCormick, W.T., Schweitzer, P.J., White, T.W., 1972. Problem decomposition and data reorganization by a clustering technique. Operations Research 20, 993–1009.

Mosier, C.T., Taube, L., 1985a. The facets of group technology and their impact on implementation. Omega 13 (6), 381–391.

Mosier, C.T., Taube, L., 1985b. Weighted similarity measure heuristics for the group technology machine clustering problem. Omega 13 (6), 577–583.

Onwubulo, G.C., Mutingi, M., 2001. A genetic algorithm approach to cellular manufacturing systems. Computers and Industrial Engineering 39, 125–144.

Seifoddini, H., 1989. Single linkage versus average linkage clustering in machine cells formation applications. Computers and Industrial Engineering 16 (3), 419–426.

Seifoddini, H., Wolfe, P.M., 1986. Application of the similarity coefficient method in group technology. IIE Transactions 18 (3), 266–270.

Seifoddini, H., Wolfe, P.M., 1987. Selection of a threshold value based on material handling cost in the machine-component grouping. IIE Transactions 19, 266–270.

Srinivasan, G., Narendran, T., Mahadevan, B., 1990. An assignment model for part-families problem in group technology. International Journal of Production Research 28, 145–152.

Stanfel, L., 1985. Machine clustering for economic production. Engineering Costs and Production Economics 9, 73–81.

Sun, D., Lin, L., Batta, R., 1995. Cell formation using tabu search. Computers and Industrial Engineering 28, 485–494.

Waghodekar, P.H., Sahu, S., 1984. Machine-component cell formation in group technology MACE. International Journal of Production Research 22, 937–948.

Won, Y., Lee, K.C., 2004. Modified *p*-median approach for efficient GT cell formation. Computers and Industrial Engineering 46, 495–510.

Wu, T.-H., Low, C., Wu, W.-T., 2004. A tabu search approach to the cell formation problem. International Journal of Advanced Manufacturing Technology 23, 916–924.

Wu, T.-H., Chang, C.-C., Chung, S.-H., 2008. A simulated annealing algorithm to manufacturing cell formation problems. Expert Systems with Applications 34, 1609–1617.

Yang, F.-C., Wang, Y.-P., 2007. Water flow-like algorithm for object grouping problems. Journal of the Chinese Institute of Industrial Engineers 24 (6), 475–488.

Yang, M.-S., Yang, J.-H., 2008. Machine–part cell formation in group technology using a modified ART1 method. European Journal of Operational Research 188, 140–152.

Yasuda, K., Yin, Y., 2001. A dissimilarity measure for solving the cell formation problem in cellular manufacturing. Computers and Industrial Engineering 39, 1–17.

Yin, Y., Yasuda, K., 2006. Similarity coefficient methods applied to the cell formation problem: A taxonomy and review. International Journal of Production Economics 101, 329–352.