Computers & Industrial Engineering 90 (2015) 292–304

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

# An iterated local search heuristic for cell formation

Michael J. Brusco

*College of Business, Florida State University, 821 Academic Way, Tallahassee, FL 32306-1110, United States*

ARTICLE INFO

*Article history:*
Received 16 June 2015
Received in revised form 1 September 2015
Accepted 21 September 2015
Available online 8 October 2015

*Keywords:*
Cell formation
Part-machine clustering
Cellular manufacturing
Grouping efficacy index
Iterated local search
Heuristics

ABSTRACT

The grouping efficacy index (GEI) has emerged as the most popular objective criterion for part-machine clustering problems associated with manufacturing cell formation. A variety of metaheuristics have been proposed for cell formation based on the GEI, including methods such as simulated annealing, tabu search, genetic algorithms, variable neighborhood search, and water flow-like algorithms. In this paper, we develop and implement an iterated local search (ILS) heuristic that has proved effective for a variety of different combinatorial optimization problems. Computational results revealed that the ILS generally matches the optimal (or best known) solutions for 37 test problems from the literature. An inherent advantage of the ILS is its simplicity. All test problems, along with the Fortran source codes and executables for the ILS heuristics under the assumptions of forbidden and permitted residual cells, are available from an internet website associated with the manuscript.



## 1. Introduction

The part-machine clustering problem in manufacturing cell formation requires the partitioning of $N$ parts ($1 \leqslant n \leqslant N$) and $M$ machines ($1 \leqslant m \leqslant M$) with the goal of establishing cells (clusters) that have many intra-cell operations but few inter-cell operations. More specifically, cells should consist of parts that require processing operations on the machines assigned to their cell, but not on machines assigned to other cells. If part $n$ and machine $m$ are assigned to the same cell, but part $n$ does not require an operation on machine $m$, then this is known as a 'void'. Likewise, if part $n$ and machine $m$ are assigned to different cells, but part $n$ does require an operation on machine $m$, then this is known as an 'exceptional element'. Most partitioning methods for the part-machine clustering problem seek to optimize some function of the total number of voids and exceptional elements. Throughout the remainder of this paper, we focus on a particular objective criterion known as the grouping efficacy index (GEI: Kumar & Chandrasekharan, 1990). The GEI is arguably one of the most popular indices used in the cell formation literature (Brusco, 2015; Brusco & Steinley, 2007; Bychkov, Batsyn, & Pardalos, 2014; James, Brown, & Keeling, 2007; Li, Baki, & Aneja, 2010; Noktehdan, Karimi, & Kashan, 2010; Paydar & Saidi-Mehrabad, 2013; Tariq, Hussain, & Ghafoor, 2009; Tunnukij & Hicks, 2009; Wu, Chang, & Chung, 2008; Wu, Chung, & Chang, 2010).

The GEI is a nonlinear (fractional) objective function, which it makes it a somewhat formidable criterion from a mathematical programming standpoint. Nevertheless, Crama and Oosten (1996) observed that part-machine clustering based on the GEI could be accomplished by solving a series of integer programs using a method originally developed by Dinkelbach (1967). Considerable success with this approach was reported by Elbenani and Ferland (2012) when the number of cells was prespecified in advance, whereas Bychkov et al. (2014) reported somewhat less promising findings when the number of cells was allowed to vary. In addition to mathematical programming approaches, Brusco (2015) has recently designed a standalone branch-and-bound algorithm that is computationally efficient and effective for smaller problems from the literature. Despite the recent progress in the design of exact methods, heuristic methods remain necessary for larger instances of part-machine clustering problems, particularly for ill-structured matrices (e.g., those problems with optimal or best-known GEI values less than 0.5).

During the past few decades, numerous metaheuristic procedures have been proposed for maximizing the GEI. These methods include, but are not necessarily limited to, genetic algorithms (Tariq et al., 2009; Tunnukij & Hicks, 2009), simulated annealing (Wu et al., 2008), ant colony methods (Li et al., 2010), water flow-like algorithms (Wu et al., 2010), and evolutionary algorithms with embedded variable neighborhood search (Paydar & Saidi-Mehrabad, 2013). Reviews of implementations of different metaheuristics are provided by Papaioannou and Wilson (2010) and Ghosh, Sengupta, Chattopadhyay, and Dan (2011).

*E-mail address:* mbrusco@business.fsu.edu

http://dx.doi.org/10.1016/j.cie.2015.09.010
0360-8352/

In this paper, we develop and evaluate an iterated local search (ILS) heuristic for cell formation. Essentially, as noted by Lourenço, Martin, and Stützle (2003, 2010), ILS involves iteration over standard local search operations such as insertions, transfers, or exchanges. The basic principle of the ILS is that a solution to a combinatorial optimization problem is systematically perturbed in some manner and then subsequently improved via the application of an appropriate type of local-search heuristic. Moreover, the process selected for perturbation can control the size of the neighborhood explored in the search process. A review of applications of ILS was reported by Lourenço, Martin, and Stützle (2010) and, more recently, there have been applications pertaining to permutation flowshop scheduling (Dong, Nowak, Chen, & Lin, 2015), vehicle routing (Silva, Subramanian, & Ochi, 2015), and *p*-center location (Quevedo-Orozco & Rios-Mercado, 2015). Despite the gainful implementations of ILS in other contexts, to the best of our knowledge, the method has not been implemented within the framework of cell formation.

The motivation for the design of an ILS heuristic for cell formation is based on two factors: (i) it is posited that ILS is preferable to running multiple restarts of a local-search heuristic for cell formation problems using different randomly-generated initial partitions for each restart, and (ii) it is proposed that ILS generally requires less parameterization and algorithm fine tuning than sophisticated metaheuristics. Local-search heuristics can exhaust a considerable amount of computation time when refining a random initial partition. Moreover, the propensity of the random restart approach to produce solutions with serious departures from global optima as problem size increases has been documented (Lourenço, Martin, & Stützle, 2003). Contrastingly, ILS can allow for a vigorous search of the neighborhood of a locally-optimal solution and requires minimal time to refine a modestly perturbed solution back to a local optimum.

It is not uncommon for ILS to use local search heuristics similar to those that have been embedded in sophisticated metaheuristics; however, ILS typically will require less parameterization and fine tuning. Consider, for example, the following metaheuristics and their commonly required parameters and implementation decisions: (i) simulated annealing (maximum temperature, minimum temperature, trial solution generation mechanism, temperature length, cooling factor), (ii) tabu search (trial solution generation mechanism, tabu list length, maximum number of trial solutions, and other intensification and diversification parameters), and (iii) genetic algorithms (population size, type of crossover operations, mutation probabilities, and termination criterion). By contrast, our implementation of ILS requires only two parameters that correspond to limits on the number of trial solutions generated and increments of the number of clusters with no GEI improvement. If the simpler ILS approach can produce solutions that are superior to (or even competitive with) those of the more sophisticated metaheuristics, then this would afford a valuable contribution.

Section 2 briefly describes the GEI and the underlying partitioning problem. Section 3 presents the details of the ILS. A suite of 37 test problems are described in Section 4. Computational results for the ILS under the assumption that residual cells (i.e., cells with only parts or only machines) are forbidden are provided in Section 5. Results obtained when residual cells are permitted are reported in Section 6. The paper concludes with a brief summary in Section 7.

## 2. The grouping efficacy index (GEI)

The formulation of a part-machine clustering cell formation problem based on the GEI begins with the definition of $S$ as the set of all $N+M$ objects (parts and machines) to be considered for partitioning. Moreover, it is assumed that objects $1 \leqslant j \leqslant N$ in $S$ are the parts and objects $N+1 \leqslant j \leqslant N+M$ are the machines. We denote $P = \{C_1, \ldots, C_K\}$ as a $K$-cell partition of the objects in $S$, where $C_k$ is the subset of objects assigned to cell $k$, for all $1 \leqslant k \leqslant K$. The conditions necessary for a partition of the objects are:

Condition 1 (Non-empty): $C_k \neq \varnothing$ for all $1 \leqslant k \leqslant K$.
Condition 2 (Mutually exclusive): $C_k \cap C_l = \varnothing$ for all $1 \leqslant k < l \leqslant K$.
Condition 3 (Exhaustive): $C_1 \cup \ldots \cup C_K = S$.

The three conditions for a partition of the objects in $S$ into $K$ cells do not preclude the possibility that a cell might contain only parts (or only machines). When they occur, these are known as *residual cells*, and their members can be viewed as outliers. However, because residual cells do not satisfy the pragmatic property of having both parts and machines in a cell, most studies have imposed constraints to prevent residual cells from occurring. Expanding the conditions above to forbid residual cells is accomplished as follows:

Condition 4 (At least one part): $C_k \cap \{1, \ldots, N\} \neq \varnothing$ for all $1 \leqslant k \leqslant K$.
Condition 5 (At least one machine): $C_k \cap \{N+1, \ldots, N+M\} \neq \varnothing$ for all $1 \leqslant k \leqslant K$.

In this paper, we provide two sets of analyses: (a) one analysis forbidding residual cells, and (b) one analysis permitting residual cells.

The $N \times M$ part-machine clustering matrix, $\mathbf{A} = [a_{nm}]$, consists of elements $a_{nm} = 1$ if part $n$ requires processing on machine $m$ or $a_{nm} = 0$ otherwise; for all $1 \leqslant n \leqslant N$ and $1 \leqslant m \leqslant M$. The total number of processing operations in $\mathbf{A}$ is:

$$e = \sum_{n=1}^{N} \sum_{m=1}^{M} a_{nm}. \tag{1}$$

For any particular partition, $P$, the number of voids, which is the number of intra-cell zeros, is computed as follows:

$$e_V(P) = \sum_{k=1}^{K} \left( \sum_{(i \in C_k : i \leqslant N)} \sum_{(j \in C_k : j > N)} (1 - a_{i(j-N)}) \right), \tag{2}$$

and the number of exceptional elements (i.e., the number of inter-cell ones) is:

$$e_O(P) = \sum_{k=1}^{K} \sum_{l \neq k} \left( \sum_{(i \in C_k : i \leqslant N)} \sum_{(j \in C_l : j > N)} a_{i(j-N)} \right). \tag{3}$$

The GEI for partition $P$ is then computed as a function of the number of voids, exceptional elements, and total number of operations as:

$$\Gamma(P) = \frac{e - e_O(P)}{e + e_V(P)}. \tag{4}$$

Defining $\Pi$ as the set of all partitions of $N+M$ objects into $K$ cells, the cell formation problem based on the GEI is:

$$\max_{P \in \Pi} : \{\Gamma(P)\}. \tag{5}$$

The solution space (number of partitions) for the optimization problem posed by (5) is a Stirling number of the second kind (Clapham, 1996), and can be computed using the following formula (Brusco, 2015, p. 655):

$$\theta(N, M, K) = \frac{1}{K!} \sum_{k=0}^{K} (-1)^k \binom{K}{k} (K-k)^{(N+M)}. \tag{6}$$

## 3. Iterated local search heuristic

### 3.1. A general explanation of the ILS heuristic

Succinctly, the ILS heuristic operates by repetition of a two-step process: (i) perturbing an incumbent locally-optimal solution, and (ii) applying a local-search heuristic to return the perturbed solution to a local optimum. Typically, the solution obtained after the second step only replaces the incumbent solution if it produces a superior value of the criterion function of interest. As noted previously, the concept of ILS is not novel and it has been applied in a variety of other contexts (see Lourenço et al., 2010 for an overview of applications). Moreover, the key principles of perturbing a solution, re-optimizing, and accepting better solutions as they are generated can also be implemented within the framework of other metaheuristics.

In our implementation, the process begins with the generation of a random initial partition of parts and machines. A relocation heuristic is then used to refine this partition to local optimality with respect to the GEI index. From here, the algorithm simply proceeds with the repetition of perturbation and re-optimization until a limit on the number of trial partitions generated is reached. Perturbation occurs by reassigning a number of randomly selected parts and machines to (randomly selected) different clusters. The number of parts/machines to be reassigned is also randomly selected, which allows for a search of various neighborhood sizes around the local optimum. The re-optimization process uses the relocation heuristic to return the perturbed solution to local optimality and, if a better solution (i.e., one with a larger GEI) is identified via this process, then it becomes the incumbent local optimum. Detailed descriptions of initialization, perturbation, and the relocation heuristic are provided later in this section

Fig. 1 provides the pseudo code of the main program for the ILS heuristic that we designed for cell formation. The main program calls three subroutines: (i) INITIALIZE, which generates an initial partition of the objects into $K$ cells, (ii) RELOCATION, which is a heuristic that produces a partition that is locally-optimal with respect to all possible reassignments of objects from their current cell to one of the other cells, and (iii) PERTURBATION, which destroys locally-optimal solutions such that they necessitate repair. The pseudo codes for subroutines (i), (ii), and (iii) are displayed in Figs. 2–4, respectively.

### 3.2. The main program of the ILS heuristic

The main program requires the selection of two input parameters. The first, *maxit*, is an upper limit on the number of iterations of the local search that are available for each number of cells ($K$) evaluated. For our comparison to existing methods, we set *maxit* = 100,000; however, we also explored the use of *maxit* = 1000 and *maxit* = 10,000 on solution quality and computation time. The second parameter, *maxnoimp*, is an upper limit on the number of increments of $K$ with no improvement in the GEI that will be permitted. In our experiments, we used *maxnoimp* = 2.

The ILS heuristic begins with the specification of *maxit* = 100,000 and *maxnoimp* = 2. The best-found GEI index value is initialized to $\Gamma^* = 0$, the counter for the number of increments of $K$ with no improvement in the GEI is set equal to *noimp* = 1, the

```
Set maxit = 100,000; maxnoimp = 2; Γ* = 0; noimp = 1; K = 1.
Establish the Outer While Loop that Evaluates Different K
While noimp ≤ maxnoimp
    Set K = K + 1
    Call INITIALIZE subroutine to obtain initial partition, P
    Call RELOCATION subroutine to refine P
    Set P_K = P,
    Set Γ*_K = Γ(P)
    Set iter = 0
Establish the Inner While Loop for ILS at a given K
    While iter ≤ maxit
        Set iter = iter + 1
        Call PERTURBATION subroutine to destroy P
        Call RELOCATION subroutine to repair P
        if Γ(P) > Γ*_K then
            Set Γ*_K = Γ(P)
            Set P_K = P
        end if
    End While
Evaluate the K-cluster Partition Against the Best-Found Partition
    if Γ*_K > Γ* then
        Set Γ* = Γ*_K
        Set P* = P_K
        Set K* = K
        Set noimp = 1
    else
        Set noimp = noimp + 1
    end if
End While
Return K*, P*, Γ*
```

**Fig. 1.** Pseudo code for the ILS MAIN PROGRAM.

Set $C_k = \varnothing \ \forall\ 1 \le k \le K$
Set $\xi(j, k) = 0 \ \forall\ 1 \le j \le M$ and $\forall\ 1 \le k \le K$, and set $\gamma(i, l) = 0 \ \forall\ 1 \le i \le N$ and $\forall\ 1 \le l \le K$
**Randomly choose *K* parts as centers**
For $k = 1$ to $K$
    randomly select a part, $h$ : $\{a(h, j) \ne \xi(j, q) \ \forall\ 1 \le j \le M\}$ for any $q$ $(1 \le q \le k\text{-}1)$.
    For $j = 1$ to $M$
        $\xi(j, k) = a(h, j)$
    Next $j$
Next $k$
**Randomly choose *K* machines as centers**
For $k = 1$ to $K$
    randomly select a machine, $h$ : $\{a(j, h) \ne \gamma(i, q) \ \forall\ 1 \le i \le N\}$ for any $q$ $(1 \le q \le k\text{-}1)$.
    For $i = 1$ to $N$
        $\gamma(i, k) = a(i, h)$
    Next $j$
Next $k$
**Assign each part to its nearest part center**
For $i = 1$ to $N$
    $mindist = \infty$
    For $k = 1$ to $K$
        $dist = 0$
        For $j = 1$ to $M$
            $dist = dist + \left| a(i, j) - \xi(k, j) \right|$
        Next $j$
        if $dist < mindist$
            $mindist = dist$
            $k' = k$
        end if
    next $k$
    $C_k = C_{k'} \cup \{i\}$
next $i$
**Assign each machine to its nearest machine center**
For $j = 1$ to $M$
    $mindist = \infty$
    For $l = 1$ to $K$
        $dist = 0$
        for $i = 1$ to $N$
            $dist = dist + \left| a(i, j) - \gamma(i, l) \right|$
        next $i$
        if $dist < mindist$
            $mindist = dist$
            $l' = l$
        end if
    next $l$
    $C_l = C_l \cup \{j+N\}$
next $i$
Set $P = (C_1, \ldots, C_K)$
Return $P$

**Fig. 2.** Pseudo code for the INITIALIZE subroutine.

number of cells is initialized to $K = 1$. The outer *While* loop in the pseudo code is active until $noimp > maxnoimp$, and this condition results in the termination of the ILS heuristic and the return of the best-found partition, $P^*$, along with its corresponding number of cells ($K^*$) and GEI index value ($\Gamma^*$).

In the first step within the outer *While* loop, $K$ is incremented. Next, the INITIALIZE subroutine is called to produce a random $K$-cell partition, and this partition is passed to the RELOCATION subroutine that produces a partition $P$ as output. This partition is stored as the best-found partition for $K$ cells, $P_K$, and its corresponding GEI is the best-found index value for $K$ cells, $\Gamma^*_K$. The iteration counter ($iter = 0$) is initialized immediately before proceeding into the inner *While* loop of the algorithm. Within the inner *While* loop, the iterated local search is performed for $K$ cells. Upon entering the inner loop, *iter*, is incremented and then the PERTURBATION and RELOCATION subroutines are called,

```
Input P
Relocation Heuristic Process
While τ = 1
        τ = 0
        % Check All Relocations of Parts to Different Cells
        For i = 1 to N
                h = h : i ∈ C_h
                Apply One of the Two If Statements Depending on Whether Residual Cells are
                Permitted or Forbidden
                if |C_h| > 1                          If Residual Cells Permitted
                if |C_h ∩ {1,…,N}| > 1                If Residual Cells Forbidden
                        for k = 1 to K
                                if k ≠ h then
                                        P′ = {P \ {C_k, C_h}} ∪ {C_k ∪ {i}} ∪ {C_h \ {i}}
                                        if Γ(P′) > Γ(P)
                                                P = P′
                                                τ = 1
                                        end if
                                end if
                        next k
                end if
        next i
        % Check All Relocations of Machines to Different Cells
        For i = N+1 to N+M
                h = h : i ∈ C_h
                Apply One of the Two If Statements Depending on Whether Residual Cells are
                Permitted or Forbidden
                if |C_h| > 1                          If Residual Cells Permitted
                if |C_h ∩ {N+1,…,N+M}| > 1            If Residual Cells Forbidden
                        for k = 1 to K
                                if k ≠ h then
                                        P′ = {P \ {C_k, C_h}} ∪ {C_k ∪ {i}} ∪ {C_h \ {i}}
                                        if Γ(P′) > Γ(P)
                                                P = P′
                                                τ = 1
                                        end if
                                end if
                        next k
                end if
        next i
End While
Return P
```

**Fig. 3.** Pseudo code for the RELOCATION subroutine.

respectively. If a partition with a new best-found index value for $K$ cells emerges from the RELOCATION routine, then $P_K$ and $\Gamma_K^*$ are updated accordingly. The inner *While* loop is active until *iter* > *maxit*.

Upon completion of the inner *While* loop, the best-found GEI index value for $K$ cells is compared to the best-found GEI index value across all numbers of cells. If $\Gamma_K^* > \Gamma^*$, then the $K$-cell partition is installed as the new best-found partition ($P^* = P_K$), the best-found GEI index is updated ($\Gamma^* = \Gamma_K^*$), the optimal number of cells is set as $K^* = K$, and *noimp* is reset to one. However, if $\Gamma_K^* \leqslant \Gamma^*$, then increasing to $K$ cells provided no improvement in the best-found GEI index, and *noimp* is incremented. A setting of *maxnoimp* = 1 would result in termination of the algorithm for the first value of $K$ whereby no improvement in the GEI was found. Although this setting is typically sufficient, we opted for *maxnoimp* = 2 to provide a bit more leeway.

### 3.3. The INITIALIZE subroutine

A simple approach to constructing the initial partition is to randomly choose the cell membership for each of the $N + M$ objects as an integer on the interval [1,$K$], with equal probability for each cell. This approach has two limitations. First, when $K$ is large and $N + M$ is small, it is possible that some cells will not be assigned any objects and, therefore, corrective action is necessary to remedy this problem. Second, and more importantly, when $N + M$ is large relative to $K$, this generation process has a propensity to produce cells of roughly equal size. If the sizes of the cells in the optimal partition are grossly unequal (as is commonly the case), then this initialization process can make it more difficult to find the optimal partition. For this reason, we recommend the quasi-random procedure used by Brusco (2015), as shown in Fig. 2.

The initialization procedure begins with the random selection (without replacement) of $K$ parts and $K$ machines to serve as 'centers' for parts and machines. Accordingly, randomness in the initialization procedure is afforded by this process. The remainder of this initialization procedure, however, uses greed to assign parts and machines to cells. Specifically, for each part, the Hamming distance between that part and each of the $K$ part centers is computed and the part is assigned to the cell associated with its nearest center. A similar process is used to assign machines to cells based on

```
Input P
Randomly Select the Number of Objects to Reassign
Select Φ from integers on the interval [2, ⌊(N+M)/2⌋] based on a uniform distribution,
Reassign Φ Objects
For l = 1 to Φ
Make Sure Reassignment of Randomly Selected Object is Feasible
    Set select = 0
    While select = 0
        randomly choose an object j from the interval [1, N+M]
        Define h : j ∈ C_h
        Set card_parts = | C_h ∩ { 1,…,N } |
        Set card_machines = | C_h ∩ {N+1,…,N+M} |
        Apply One of the Two If Statements Depending on Whether Residual Cells
        are Permitted or Forbidden
        if card_parts + card_machines > 1 then                          Permitted
        if (j ≤ N and card_parts > 1) ∨ (j > N and card_machines > 1) then   Forbidden
            select = 1
        end if
    End While
    Move object j from C_h to cell C_k, where k ≠ h is randomly selected from [1, K]
Next l
Return P
```

**Fig. 4.** Pseudo code for the PERTURBATION subroutine.

Hamming distance. The result of this process is an initial partition, $P$, of parts and machines to $K$ cells.

### 3.4. The RELOCATION subroutine

The RELOCATION subroutine is a relocation heuristic similar in design to those used for general clustering problems (Banfield & Bassill, 1977; Hartigan, 1975). The pseudo code in Fig. 3 provides the precise steps for the algorithm used herein. The input to the subroutine is an initial partition, $P$, that is typically not locally-optimal with respect to all possible relocations of an object from its current cell to one of the other cells. The initial partition is refined by a process of considering each of the $N + M$ objects (i.e., parts and machines) in turn and evaluating the effect on the GEI of their relocation from their current cell, $h$, to each of the other cells, $1 \leqslant k \neq h \leqslant K$. Each time an improvement in the GEI is realized from moving an object $i$ from its current cell $h$ to a different cell $k$, such a relocation is accepted and the incumbent solution is updated. The algorithm cycles through all objects and cell reassignments until convergence. That is, the algorithm terminates when no relocation of an object will further improve the GEI.

The pseudo code in Fig. 3 indicates that, in the case where residual cells are permitted, the algorithm will not permit relocation of an object if it is the only object in the cell (i.e., $|C_h| = 1$), as this would create an empty cell (thus violating the requirement of a partition). However, when residual cells are forbidden, this condition must be strengthened to ensure that relocation does not occur if the object in question is the only part (or the only machine) in that cell, as its removal would create a residual cell.

### 3.5. The PERTURBATION subroutine

The PERTURBATION subroutine takes, as input, a partition ($P$) that has been refined by the RELOCATION subroutine and is, therefore, locally-optimal with respect to all possible relocations of objects to a different cell. The PERTURBATION subroutine destroys this local optimality by reassigning a randomly-selected number of objects from their current cell to a different (randomly-selected) cell. The number of objects for reassignment, $\Phi$, at any given iteration of the ILS is randomly selected based on a uniform distribution from the set of integers on the interval $[2, \lfloor (N + M)/2 \rfloor]$, where $\lfloor x \rfloor$ is the greatest integer equal to or less than $x$. Given a randomly-selected value of $\Phi$ at a given iteration, it is necessary to randomly select $\Phi$ objects for relocation. However, it is important to ensure that reassignment of the randomly-selected objects is feasible, so as not to produce empty cells or residual cells if they are forbidden. When residual cells are permitted, a randomly-selected object cannot be relocated if it is the only object in the cell. Likewise, in the instances where residual cells are not permitted, a randomly-selected object cannot be reassigned if it is the only part (or only machine) in the cell.

The process used for PERTURBATION is conceptually straightforward. Nevertheless, it has the distinct advantage of ensuring a good mix of both local and distant neighborhood explorations. Reassignment of only small number of objects (e.g., $2 \leqslant \Phi \leqslant 10$) will allow for exploration of a rather localized neighborhood of the current partition. Contrastingly, larger values of $\Phi$ near $\lfloor (N + M)/2 \rfloor$ facilitate investigation of more distant neighborhoods, and such investigation is known to be essential for escaping some local optima (Brusco, Jacobs, Bongiorno, Lyons, & Tang, 1995; Ruiz & Stützle, 2007).

### 3.6. Implementation and availability details

The ILS heuristic was programmed in Fortran 90. Two versions of the heuristic were prepared. One version permits residual cells, whereas the other forbids them. The programs were implemented on a 3.4 GHz Intel Core i7-2600 processor with 8.0 GB of RAM. The test problems, Fortran source codes, and executable files (for both permitted and forbidden residual cells) are available from the website https://myweb.fsu.edu/mbrusco/cgi-bin/ILS/.

## 4. Test problems

The part-machine clustering problems used in our computational analyses were collected for a study by Gonçalves and

**Table 1**
Test problems: sources and sizes.

| Number* | Source for the test problem matrix | Location within the source | Size |
|---|---|---|---|
| 1 | King and Nakornchai (1982) | Fig. 1, p. 122 | 5 × 7 |
| 2 | Waghodekar and Sahu (1984) | Fig. 3a, p. 943 | 5 × 7 |
| 3 | Seifoddini (1989) | Table 1, p. 1162 | 5 × 18 |
| 4 | Kusiak and Cho (1992) | Eq. (12), p. 2641 | 6 × 8 |
| 5 | Kusiak and Chow (1987) | Example 3, Equation 15, p. 122 | 7 × 11 |
| 6 | Boctor (1991) | Fig. 1, p. 345 | 7 × 11 |
| 7 | Seifoddini and Wolfe (1986) | Fig. 1, p. 272 | 8 × 12 |
| 8 | Chandrasekharan and Rajagopalan (1986a) | Table 1, p. 458 | 8 × 20 |
| 9 | Chandrasekharan and Rajagopalan (1986b) | Fig. 3a, p. 1225 | 8 × 20 |
| 10 | Mosier and Taube (1985a) | Fig. 1, p. 384 | 10 × 10 |
| 11 | Chan and Milner (1982) | Fig. 2a, p. 68 | 10 × 15 |
| 12a | Askin and Subramanian (1987) | Table 2, p. 111 | 14 × 23 |
| 12b | Askin and Subramanian (1987) | Table 2, p. 111 | 14 × 24 |
| 13 | Stanfel (1985) | Fig. 2, p. 77 | 14 × 24 |
| 14 | McCormick, Schweitzer, and White (1972) | Fig. 8, p. 1006 | 16 × 24 |
| 15 | Srinivasan, Narendran, and Mahadevan (1990) | Fig. 3, p. 149 | 16 × 30 |
| 16 | King (1980) | Fig. 19, p. 226 | 16 × 43 |
| 17 | Carrie (1973) | Fig. 4, p. 403 | 18 × 24 |
| 18 | Mosier and Taube (1985b) | Matrix 2, p. 577 | 20 × 20 |
| 19 | Kumar, Kusiak, and Vannelli (1986) | Fig. 7, p. 395 | 20 × 23 |
| 20 | Carrie (1973) | Fig. 1, p. 401 | 20 × 35 |
| 21 | Boe and Cheng (1991) | Fig. 5, p. 2101 | 20 × 35 |
| 22 | Chandrasekharan and Rajagopalan (1989) | Fig. 1, p. 1039 – matrix 1 | 24 × 40 |
| 23 | Chandrasekharan and Rajagopalan (1989) | Fig. 2, p. 1041 – matrix 2 | 24 × 40 |
| 24 | Chandrasekharan and Rajagopalan (1989) | Fig. 2, p. 1042 – matrix 3 | 24 × 40 |
| 25 | Chandrasekharan and Rajagopalan (1989) | Fig. 2, p. 1044 – matrix 5 | 24 × 40 |
| 26 | Chandrasekharan and Rajagopalan (1989) | Fig. 2, p. 1045 – matrix 6 | 24 × 40 |
| 27 | Chandrasekharan and Rajagopalan (1989) | Fig. 2, p. 1046 – matrix 7 | 24 × 40 |
| 28 | McCormick et al. (1972) | Fig. 3, p. 999 | 27 × 27 |
| 29 | Carrie (1973) | Fig. 5, p. 403 | 28 × 46 |
| 30 | Kumar and Vannelli (1987) | Fig. 8, p. 1724 | 30 × 41 |
| 31 | Stanfel (1985) | Fig. 5, p. 79 | 30 × 50 |
| 32 | Stanfel (1985) | Fig. 6, p. 80 | 30 × 50 |
| 33a | King and Nakornchai (1982) | Fig. 5a, p. 130 | 30 × 90 |
| 33b | King and Nakornchai (1982) | Fig. 5a, p. 130 | 36 × 90 |
| 34 | McCormick et al. (1972) | Fig. 6, p. 1003 | 37 × 53 |
| 35 | Chandrasekharan and Rajagopalan (1987) | Fig. 5, p. 846 | 40 × 100 |

* Problems 12 and 33 are source matrices that have empty rows or columns. Therefore, two versions (a and b) of these source matrices are considered: one version (a) excludes the empty rows/columns, whereas the other version (b) includes them.

Resende (2004), and been have used (either all or in part) for testing purposes by several other authors (James et al., 2007; Li et al., 2010; Noktehdan et al., 2010; Paydar & Saidi-Mehrabad, 2013; Tunnukij & Hicks, 2009). Although there were originally 35 test problems in this suite, we consider two versions of two of the problems, which yields a total of 37. The original sources and sizes of the test problems are provided in Table 1.

In the development of our comparative analyses, there are two issues that complicate the identification of benchmark solutions. One issue is the permission or forbiddance of residual cells. To rectify this problem, results obtained using the more common assumption that residual cells are not allowed are provided in Section 5, whereas results obtained under the assumption that residual cells are viable are reported in Section 6.

A second and more formidable issue pertains to the accuracy and consistency of the part-machine clustering problems used across different studies. As has been noted in previous comparative analyses (James et al., 2007; Tunnukij & Hicks, 2009), there is the propensity for inconsistency between the original published source for test problems and data that have been used and/or published in other sources. To the greatest extent possible, we relied on the original sources of data in our analyses, and commentary is provided where our results differ from those reported by other authors.

In addition to the possibility of errors occurring in the data transcription process, there is also the potential for alternative interpretations of the data in some instances. For example, test problem #12 from Askin and Subramanian (1987, p. 111) presents an opportunity for conflicting results. This test problem is usually presented as a 14 × 24 matrix; however, in the original source, the data are presented in a 23 × 14 matrix (row 22 is missing). Some authors treat this matrix as 14 × 23 or, equivalently, 23 × 14 (Brusco, 2015; Tariq et al., 2009). This version of the problem is labeled #12a. However, it seems that most authors include the empty 24th row (or column) in the data (we refer to this version as problem #12b). Test problem #33 from King and Nakornchai (1982, Fig. 5a, p. 130) exhibits the same issue as test problem 12. James et al. (2007, p. 2070) have observed that this matrix is typically reported as 36 × 90 in the literature, yet should actually be perceived as 30 × 90 because of empty rows. Indeed, there are six empty rows in this source matrix, and the inclusion or exclusion of these rows can have an impact on the GEI. We report results for this test problem as both a 30 × 90 matrix (problem #33a, empty rows excluded) and a 36 × 90 matrix (problem #33b, empty rows included).

Although considerable care was taken to be as accurate as possible, the potential for data inconsistency remains. To facilitate data comparisons and solution verification for researchers, we have prepared a supplement associated with this manuscript that contains the precise data used for each test problem, as well as Fortran source codes, executables, and batch files for the ILS heuristic that can be used to replicate the results reported in the paper.

## 5. Computational results and discussion – Residual cells forbidden

### 5.1. Comparison of ILS to other methods in the literature

Table 2 presents a comparison of the GEI objective function values obtained by ILS to those of other methods from the cell-formation literature under the assumption that residual cells are not permissible. The first three columns (immediately after the 'problem' column) correspond to exact procedures for the cell-formation problem, whereas the remaining columns are heuristic methods. The methods in Table 2 include:

(1) ILP1 – an integer programming approach developed by Bychkov et al. (2014).
(2) ILP2 – an integer programming approach developed by Elbenani and Ferland (2012).
(3) BB – a branch-and-bound approach developed by Brusco (2015).
(4) SACF – a simulated annealing heuristic (Wu et al., 2008).
(5) WFACF – a water flow-like algorithm (Wu et al., 2010).
(6) EnGGA – an enhanced grouping genetic algorithm (Tunnukij & Hicks, 2009).
(7) HGGA – a hybrid grouping genetic algorithm (James et al., 2007).
(8) HGDE – a hybridized grouping differential evolution algorithm (Noktehdan et al., 2010).
(9) ACO – an ant colony optimization heuristic (Li et al., 2010).
(10) ILS – iterated local search (proposed method).

For test problems #1 through #13, there is strong concordance among the exact procedures and most of the heuristic methods. Two notable exceptions are the water flow-like and simulated annealing algorithms. The WFACF is often seriously underperforming with respect to GEI, failing to match global optimum for even small problems such as #1, #2, #5, #6, #7, and #10. Although typically superior to WFACF, the SACF algorithm also failed to match the optimal solution for smaller problems such as #9 and #13.

Comparison of methods also requires some clarifications regarding discrepancies for test problems. For example, Elbenani and Ferland (2012) report a globally-optimal GEI of .5326 for $K = 8$ cells for test problem #14. This is matched by EnGGA and ILS, but none of the other methods. Most of the GEI values reported for the other heuristics for this problem are smaller; however, Noktehdan et al. (2010) report larger values of .5341 and .5385 for their HGDE and GDE heuristics, respectively. In fact, even the average GEI values they report are larger than .5326. Although the explanation for the disparity among GEI values reported for this problem is unclear, it has been noted by Tunnukij and Hicks (2009, p. 2002) that data published for this problem by

**Table 2**
Comparison of the GEI values obtained by ILS to those reported for other methods from the literature for the same test problems – residual cells forbidden.

| Problem | ILP1 | ILP2 | BB | SACF | WFACF | EwGGA | HGGA | HGDE | ACO | ILS |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | .8235 | .8235(2) | .8235(2) | – | .7368 | .8235 | .8235 | .8235 | .8235 | .8235 |
| 2 | .6957 | .6957(2) | .6957(2) | .6957 | .6250 | .6957 | .6957 | .6957 | .6957 | .6957 |
| 3 | .7959 | .7959(2) | .7959(2) | .7959 | .7959 | .7959 | .7959 | .7959 | .7959 | .7959 |
| 4 | .7692 | .7692(2) | .7692(2) | .7692 | .7692 | .7692 | .7692 | .7692 | .7692 | .7692 |
| 5 | .6087 | .6087(5) | .6087(5) | .6087 | .5313 | .6087 | .6087 | .6087 | .6087 | .6087 |
| 6 | .7083 | .7083(4) | .7083(4) | .7083 | .7037 | .7083 | .7083 | .7083 | .7083 | .7083 |
| 7 | .6944 | .6944(4) | .6944(4) | – | .6829 | .6944 | .6944 | .6944 | .6944 | .6944 |
| 8 | .8525 | .8525(3) | .8525(3) | .8525 | .8525 | .8525 | .8525 | .8525 | .8525 | .8525 |
| 9 | .5872 | .5872(2) | .5872(2) | .5841 | .5872 | .5872 | .5872 | .5872 | .5872 | .5872 |
| 10 | .7500 | .7500(5) | .7500(5) | .7500 | .7059 | – | .7500 | .7500 | .7500 | .7500 |
| 11 | .9200 | .9200(3) | .9200(3) | .9200 | .9200 | – | .9200 | .9200 | .9200 | .9200 |
| 12a | – | – | .7313(7) | – | – | – | – | – | – | .7313 |
| 12b | .7206 | .7206(7) | – | – | .6986 | – | .7206 | .7206 | .7206 | .7206 |
| 13 | .7183 | .7183(7) | .7183(7) | .7121 | .6933 | – | .7183 | .7183 | .7183 | .7183 |
| 14 | – | .5326(8) | – | – | .5196 | .5326 | .5275 | .5341 | .5275 | .5326 |
| 15 | – | .6953(6) | .6899(6) | – | .6783 | .6899 | .6899 | .6899 | .6899 | .6899 |
| 16 | – | .5753(8) | – | .5244 | .5590 | .5753 | .5753 | .5753 | .5753 | .5753 |
| 17 | – | .5773(9) | – | – | .5446 | .5773 | .5773 | .5773 | .5773 | .5773 |
| 18 | – | – | – | .4104 | .4296 | – | .4318 | .4345 | .4345 | .4345 |
| 19 | – | .5081(7) | – | .5081 | .4961 | – | .5081 | .5081 | .5081 | .5081 |
| 20 | – | .7791(5) | .7791(5) | .7840 | .7654 | .7791 | .7791 | .7791 | .7791 | .7791 |
| 21 | – | .5798(5) | – | .5604 | .5815 | .5798 | .5798 | .5798 | .5798 | .5798 |
| 22 | 1.0000 | 1.0000(7) | – | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 23 | – | .8511(7) | – | .8511 | .8511 | .8511 | .8511 | .8511 | .8511 | .8511 |
| 24 | – | .7351(7) | – | .7351 | .7351 | .7351 | .7351 | .7351 | .7351 | .7351 |
| 25 | – | – | – | .5244 | .5197 | .5329 | .5329 | .5329 | .5329 | .5329 |
| 26 | – | – | – | .4713 | .4737 | .4895 | .4895 | .4895 | .4895 | .4895 |
| 27 | – | – | – | .4464 | .4487 | .4658 | .4726 | .4726 | .4726 | .4658 |
| 28 | – | .5482(5) | – | – | .5427 | .5482 | .5402 | .5482 | .5482 | .5482 |
| 29 | – | – | – | – | .4606 | – | .4691 | – | .4708 | .4723 |
| 30 | – | .6331(14) | – | .6242 | .5952 | .6331 | .6331 | .6331 | .6331 | .6331 |
| 31 | – | – | – | .6012 | .6000 | – | .5977 | .5977 | .5977 | .5977 |
| 32 | – | – | – | .5051 | .5051 | – | .5083 | – | .5083 | .5083 |
| 33a | – | – | – | – | – | – | – | – | – | .4801 |
| 33b | – | – | – | – | .4615 | – | .4635 | – | .4711 | .4721 |
| 34 | – | .6064(3) | – | – | .5985 | .6064 | .6064 | .6064 | .6064 | .6064 |
| 35 | – | .8403(10) | | .8043 | .8403 | – | .8403 | – | .8403 | .8403 |

ILP1 corresponds to the method of Bychkov et al. (2014), and the solution values reported are purported to be globally-optimal across all values of $K$. ILP2 corresponds to the method of Elbenani and Ferland (2012), and the solution values reported are purported to be optimal only for the value of $K$ reported in parentheses. Finally, BB corresponds to results obtained using the branch-and-bound algorithm of Brusco (2015) and these results are also purported to be optimal only for the value of $K$ shown in parentheses. It is not known whether the results for WFACF, HGGA, and ACO in row 33b actually correspond to 33a or 33b.

Gonçalves and Resende (2004) are inconsistent with the original source.

Elbenani and Ferland (2012) report a globally-optimal GEI of .6953 for test problem #15 at $K = 6$ cells. However, Bychkov et al. (2014) report a global optimum of .6900 for this problem (across all values of $K$). Similarly, six studies using heuristic methods report a best-found objective function value of .6899 for problem #15. We applied the exact branch-and-bound procedure developed by Brusco (2015) to problem #15 for $2 \leqslant K \leqslant 7$ cells and found a maximum of .68992 at $K = 6$. Presumably, this is consistent with the results obtained from most heuristics, as well as the Bychkov et al. (2014) result after rounding up the GEI value. The reason for the discrepancy with the Elbenani and Ferland (2012) result is unknown.

Most of the heuristic methods, as well ILP2, yield consistent results for problems #16, #17, and #19. Problem #18 is ill-structured and presents a more formidable challenge for some of the heuristics. The HGDE, ACO, and ILS methods produce the best-known GEI of .4345 for this problem. The reported GEI values for most methods are consistent across problems #20 through #24; however, there are two notable exceptions. The reported GEI for SACF for problem #20 is .7840 for $K = 5$ cells; however, both Elbenani and Ferland (2012) and implementation of branch-and-bound have confirmed a globally optimal GEI value of .7791 at $K = 5$ cells for this problem. Likewise, the reported GEI for WFACF for problem #21 is .5815 for $K = 5$ cells; however, Elbenani and Ferland (2012) confirmed a globally optimal GEI value of .5798 at $K = 5$ cells for this problem, which matches the result obtained by most heuristics.

Like Tunnukij and Hicks (2009, p. 2003), we had difficulty achieving the best-known objective function value for test problem #27. Those authors report that the GEI associated with the partition for problem #27 provided by James et al. (2007, p. 2075) is .4527 and not .4726. There is a modest degree of confusion because the matrix in the original source (Chandrasekharan & Rajagopalan, 1989, p. 1046) is in $24 \times 40$ form rather than the $40 \times 24$ form displayed in James et al. (2007, p. 2075). Nevertheless, inspection of these two sources leads to the discovery of the basis of the discrepancy. The $40 \times 24$ partition provided by James et al. (2007, p. 2075) has an entry of '1' in column 18, row 10. However, a careful scan of both the original and permuted version of the $24 \times 40$ matrix in Chandrasekharan and Rajagopalan (1989) shows that the there is a '1' in row 18, column 11 and not row 18 column 10. Therefore, in the transposed James et al. (2007, p. 2075) matrix, the '1' belongs in row 11 column 18 and not row 10 column 18.

Comparisons are a bit more challenging for problems #28 through #35 because of the scarcity of data available. However, there is significant agreement among heuristics for problems #28, #30, #31, #32, #34, and #35. The ILS algorithm produced a better GEI than the other heuristics for problem #29. Comparisons for problems #33a and #33b are exceptionally difficult because, in most cases, it in unclear whether the problem is being treated as

**Table 3**
Comparison of the GEI values, number of clusters, and computation times for ILS at different values of *maxit* – residual cells forbidden.

| Problem | Best-found GEI values | | | Number of cells/clusters ($K$) | | | Computation time (seconds) | | |
|---|---|---|---|---|---|---|---|---|---|
| | 100,000 | 10,000 | 1000 | 100,000 | 10,000 | 1000 | 100,000 | 10,000 | 1000 |
| 1 | .8235 | | | 2 | | | .43 | .06 | .02 |
| 2 | .6957 | | | 2 | | | .45 | .05 | .00 |
| 3 | .7959 | | | 2 | | | .89 | .09 | .01 |
| 4 | .7692 | | | 2 | | | .98 | .11 | .02 |
| 5 | .6087 | | | 5 | | | 2.79 | .29 | .03 |
| 6 | .7083 | | | 4 | | | 2.42 | .25 | .03 |
| 7 | .6944 | | | 4 | | | 3.81 | .39 | .03 |
| 8 | .8525 | | | 3 | | | 3.81 | .38 | .05 |
| 9 | .5872 | | | 2 | | | 3.69 | .37 | .04 |
| 10 | .7500 | | | 5 | | | 5.75 | .58 | .05 |
| 11 | .9200 | | | 3 | | | 3.61 | .36 | .05 |
| 12a | .7313 | | | 7 | | | 27.22 | 2.72 | .26 |
| 12b | .7206 | | | 7 | | | 28.71 | 2.85 | .29 |
| 13 | .7183 | | | 7 | | | 28.25 | 2.84 | .28 |
| 14 | .5326 | | | 8 | | | 55.16 | 5.52 | .54 |
| 15 | .6899 | | | 6 | | | 34.99 | 3.51 | .36 |
| 16 | .5753 | | | 8 | | | 95.82 | 9.55 | .95 |
| 17 | .5773 | | | 9 | | | 71.85 | 7.17 | .71 |
| 18 | .4345 | | | 5 | | | 42.47 | 4.29 | .43 |
| 19 | .5081 | | | 7 | | | 63.49 | 6.38 | .64 |
| 20 | .7791 | | | 5 | | | 34.85 | 3.48 | .34 |
| 21 | .5798 | | | 5 | | | 49.75 | 4.99 | .50 |
| 22 | 1.0000 | | | 7 | | | 62.51 | 6.26 | .63 |
| 23 | .8511 | | | 7 | | | 80.78 | 8.07 | .81 |
| 24 | .7351 | | | 7 | | | 97.55 | 9.70 | .99 |
| 25 | .5329 | | | 11 | | | 232.08 | 23.18 | 2.34 |
| 26 | .4895 | | | 12 | | | 276.25 | 27.69 | 2.79 |
| 27 | .4658 | | −.0027 | 12 | | 11 | 273.82 | 27.36 | 2.45 |
| 28 | .5482 | | | 5 | | | 60.27 | 6.02 | .61 |
| 29 | .4723 | | | 11 | | | 315.44 | 31.61 | 3.15 |
| 30 | .6331 | | −.0055 | 14 | | 14 | 420.74 | 42.09 | 4.21 |
| 31 | .5977 | | | 13 | | | 407.07 | 40.70 | 4.07 |
| 32 | .5083 | | | 14 | | | 529.34 | 52.94 | 5.28 |
| 33a | .4801 | | −.0012 | 16 | | 16 | 1077.12 | 107.53 | 10.89 |
| 33b | .4721 | −.0007 | −.0013 | 16 | 17 | 15 | 1247.42 | 124.91 | 11.43 |
| 34 | .6064 | | | 3 | | | 68.65 | 6.88 | .69 |
| 35 | .8403 | | | 10 | | | 550.46 | 55.00 | 5.49 |

All GEI values for *maxit* = 100,000 are reported. Blank entries for GEI in the 10,000 and 1000 columns indicate that these levels of *maxit* produced the same GEI value as 100,000 restarts. All values of $K$ for *maxit* = 100,000 are reported. Blank entries for $K$ occur in the columns for 10,000 and 1000 if these values of *maxit* matched the solution for *maxit* = 100,000. Computation times for all levels of *maxit* are provided to show that they decline by an order of magnitude as *maxit* declines by an order of magnitude.

**Table 4**
Comparison of the GEI values obtained by ILS to those reported for other methods from the literature for the same test problems – residual cells permitted.

| Problem | BB | ACO | GA-VNS | ILS |
|---|---|---|---|---|
| 1 | .8235(2) | .8235 | .8235 | .8235 |
| 2 | .6957(2) | .6957 | .6957 | .6957 |
| 3 | .8085(3) | .8085 | .8085 | .8085 |
| 4 | .7917(3) | .7917 | .7917 | .7917 |
| 5 | .6087(5) | .6087 | .6087 | .6087 |
| 6 | .7083(4) | .7083 | .7083 | .7083 |
| 7 | .6944(4) | .6944 | .6944 | .6944 |
| 8 | .8525(3) | .8525 | .8525 | .8525 |
| 9 | .5872(2) | .5872 | .5872 | .5872 |
| 10 | .7500(5) | .7500 | .7500 | .7500 |
| 11 | .9200(3) | .9200 | .9200 | .9200 |
| 12a | .7424(7) | – | – | .7424 |
| 12b | .7424(7) | .7313 | .7424 | .7424 |
| 13 | .7286(7) | .7286 | .7286 | .7286 |
| 14 | .5385(8) | .5326 | .5385 | .5385 |
| 15 | .6992(6) | .6992 | .7076 | .6993 |
| 16 | – | .5804 | .5764 | .5804 |
| 17 | .5773(9) | .5773 | .5773 | .5773 |
| 18 | – | .4397 | .4326 | .4397 |
| 19 | .5081(7) | .5081 | .5081 | .5081 |
| 20 | .7888(6) | .7888 | .7840 | .7888 |
| 21 | – | .5860 | .5815 | .5860 |
| 22 | 1.0000(7) | 1.0000 | 1.0000 | 1.0000 |
| 23 | .8511(7) | .8511 | .8511 | .8511 |
| 24 | .7351(7) | .7351 | .7351 | .7351 |
| 25 | – | .5329 | .5329 | .5329 |
| 26 | – | .4895 | .4895 | .4895 |
| 27 | – | .4726 | .4726 | .4658 |
| 28 | – | .5482 | .5482 | .5482 |
| 29 | – | .4772 | .4691 | .4785 |
| 30 | – | .6331 | .6331 | .6331 |
| 31 | – | .5977 | .6012 | .5977 |
| 32 | – | .5083 | .5083 | .5084 |
| 33a | – | – | – | .4829 |
| 33b | – | .4753 | .4635 | .4829 |
| 34 | – | .6131 | .6064 | .6131 |
| 35 | – | .8403 | .8403 | .8403 |

$30 \times 90$ or $36 \times 90$. However, problem #33b is recognized as particularly difficult because of its size, ill-structuredness, and the number of cells it requires.

Despite the discrepancies in solution values described above, it is reasonable to conclude that ILS is competitive with the most sophisticated heuristic procedures in the literature. The ILS heuristic can be compared, head-to-head, with WFACF, HGGA, and ACO for 35 problems. The ILS heuristic provided a better GEI than WFACF for 24 problems, the same GEI for nine problems, and a worse GEI for two problems. For at least one of the two problems (problem #21) where WFACF produced a better result, it is clear that this was attributable to a data discrepancy because ILS matched the known global optimum. The ILS heuristic obtained a better GEI than HGGA for five problems (#14, #18, #28, #29, #33b), a worse GEI for one problem (#27), and the same GEI for the remaining problems. Likewise, the ILS heuristic obtained a better GEI than ACO for three problems (#14, #29, #33b), a worse GEI for one problem (#27), and the same GEI for the remaining problems. The one instance (problem #27) where HGGA and ACO provided a better GEI than the ILS heuristic is due to the error in this dataset described above. The HGDE heuristic is also competitive, matching the GEI obtained by the ILS heuristic for 29 problems, and yielding a better GEI for two problems (#14 and #27). However, the GEI reported for HGDE on problem #14 exceeds the known global optimum value, and the result for problem #27 is due to the aforementioned data error. Unfortunately, results for HGDE are not available for some of the most difficult problems (e.g., #29 and #33b).

### 5.2. The sensitivity of ILS to the maxit parameter

Table 3 presents an evaluation of the sensitivity of the ILS heuristic to the selection of the *maxit* parameter with respect to GEI, the selected number of cells, and computation time. The GEI values and number of cells ($K$) for all 37 problems are reported for *maxit* = 100,000. However, to make the table easier to read, the GEI and $K$ results for *maxit* = 10,000 and *maxit* = 1000 are only displayed if these parameter values resulted in a different solution than *maxit* = 100,000. It is immediately clear that reducing *maxit* from 100,000 to 10,000 provides no degradation in solution quality for 36 of the 37 problems. For problem #33b, *maxit* = 10,000 produced a $K$ = 17-cell solution with GEI of .4714, which is a difference of −.0007 from the slightly better $K$ = 16-cell solution achieved at *maxit* = 100,000. A further reduction to *maxit* = 1000 resulted in slightly greater degradation in performance. Nevertheless, even at this modest parameter setting, the ILS heuristic matched the *maxit* = 100,000 results for 33 of the 37 problems, with a maximum difference of −.0055 for problem #30.

The three rightmost columns in Table 3 indicate that computation time for the ILS heuristic increases by roughly an order of magnitude each time the *maxit* parameter increases by an order of magnitude. Average computation times are meaningless given the disparity in the sizes of the test problems and, accordingly, we will use the most challenging problem (problem #33b) as an example because it required the largest computation time. At *maxit* = 1000, a solution with GEI of .4708 was obtained in less than 12 s. An investment of roughly two minutes at *maxit* = 10,000 yielded a GEI of .4714, and 20 min of computation time at *maxit* = 100,000 produced a GEI of .4721. It is noteworthy, however, that even the *maxit* = 1000 setting produced a better result than both HGGA and WCACF, and the *maxit* = 10,000 result was better than all three of the competing methods for this problem (see HGGA, WCACF, and ACO results in Table 2).

## 6. Computational results and discussion – Residual cells allowed

### 6.1. Comparison of ILS to other methods in the literature

Table 4 presents a comparison of the ILS heuristic to other methods from the cell-formation literature under the assumption that residual cells are permissible. The only results for an exact procedure under the condition where residual cells are permitted are those provided by Brusco (2015). Moreover, heuristic results were available from only two additional sources (Li et al., 2010; Paydar & Saidi-Mehrabad, 2013). The methods in Table 4 include:

(1) BB – a branch-and-bound approach developed by Brusco (2015).
(2) ACO – an ant colony optimization heuristic (Li et al., 2010).
(3) GA-VNS – genetic algorithm/variable neighborhood search (Paydar & Saidi-Mehrabad, 2013).
(4) ILS – iterated local search (proposed method).

The exact algorithm developed by Brusco (2015) produced a globally-optimal GEI value for 22 of the test problems; however, global optima for the remaining 15 problems were not verified. The comparison of the three heuristic methods (ACO, GA-VNS, ILS) is based on 35 of the 37 problems for which all three methods were evaluated. Although it is impossible to fully explain all of the differences in performance among competing methods, it does seem reasonable to conclude that ILS is at least competitive with the more sophisticated heuristic procedures.

The ILS heuristic obtained a better GEI value than GA-VNS for seven problems (#16, #18, #20, #21, #29, #33b, #34), a worse GEI for three problems (#15, #27, #31), and the same GEI for the remaining problems. At least two (problems #15 and #27) of the three problems for which the GA-VNS algorithm obtained a better result than ILS are explained by data discrepancies. The GEI reported for GA-VNS on problem #15 is super-optimal, as the known global for this problem was obtained by ILS. In light of the inconsistencies with problem #15 observed in Section 5.1, it seems likely that there is a data discrepancy among some versions of this problem used in the literature.

The ILS heuristic obtained a better GEI value than ACO for four problems (#12b, #14, #29, #33b), a worse GEI for one problem (#27), and the same GEI for the remaining problems. Once again, the smaller GEI obtained by ILS for problem #27 is almost certainly attributable to the data error originally noted by Tunnukij and Hicks (2009) and described in Section 5.1.

### 6.2. The sensitivity of ILS to the maxit parameter

Table 5 presents an evaluation of the sensitivity of the ILS heuristic (when used under the condition that residual cells are permitted) to the selection of the *maxit* parameter with respect to GEI, the selected number of cells, and computation time. Like Table 3, Table 5 displays the GEI values and number of cells ($K$) for all 37 problems for *maxit* = 100,000, whereas the GEI and $K$ results are displayed for *maxit* = 10,000 and *maxit* = 1000 only if these parameter values resulted in a different solution than *maxit* = 100,000. Remarkably, Table 5 reveals that reducing *maxit* from 100,000 to 10,000 provides no degradation in solution quality for any of the 37 problems. At the more modest parameter setting of *maxit* = 1000, the ILS heuristic matched the *maxit* = 100,000 results for 31 of the 37 problems. Across the six problems where the maximum GEI was not achieved using *maxit* = 1000 (#27, #29, #30, #32, #33a, #33b), the largest difference was −.0013 for problem #29.

The three rightmost columns in Table 5 once again indicate that computation time for the ILS heuristic increases approximately by an order of magnitude each time the *maxit* parameter increases by an order of magnitude. The computation times for each test problem are extremely similar to those in Table 3, which suggests that the permission or forbiddance of residual cells has minimal impact on computation time.

## 7. Conclusions

### 7.1. Summary of findings

Numerous metaheuristics have been proposed for the formation of manufacturing cells based on maximization of the grouping efficacy index. Genetic (evolutionary) algorithms have been especially popular (James et al., 2007; Tariq et al., 2009;

**Table 5**
Comparison of the GEI values, number of clusters, and computation times for ILS at different values of *maxit* – residual cells permitted.

| Problem | Best-found GEI values | | | Number of cells/clusters ($K$) | | | Computation time (seconds) | | |
|---|---|---|---|---|---|---|---|---|---|
| | 100,000 | 10,000 | 1000 | 100,000 | 10,000 | 1000 | 100,000 | 10,000 | 1000 |
| 1 | .8235 | | | 2 | | | .45 | .06 | .01 |
| 2 | .6957 | | | 2 | | | .50 | .05 | .00 |
| 3 | .8085 | | | 3 | | | .91 | .09 | .02 |
| 4 | .7917 | | | 3 | | | 1.11 | .10 | .02 |
| 5 | .6087 | | | 5 | | | 3.17 | .33 | .05 |
| 6 | .7083 | | | 4 | | | 2.81 | .28 | .04 |
| 7 | .6944 | | | 4 | | | 4.38 | .44 | .04 |
| 8 | .8525 | | | 3 | | | 3.96 | .39 | .05 |
| 9 | .5872 | | | 2 | | | 3.88 | .39 | .03 |
| 10 | .7500 | | | 5 | | | 7.22 | .72 | .07 |
| 11 | .9200 | | | 3 | | | 3.64 | .36 | .04 |
| 12a | .7424 | | | 7 | | | 31.94 | 3.18 | .31 |
| 12b | .7424 | | | 7 | | | 31.96 | 3.20 | .33 |
| 13 | .7286 | | | 7 | | | 32.26 | 3.22 | .31 |
| 14 | .5385 | | | 8 | | | 65.26 | 6.50 | .66 |
| 15 | .6993 | | | 6 | | | 35.80 | 3.61 | .36 |
| 16 | .5804 | | | 9 | | | 124.30 | 12.51 | 1.25 |
| 17 | .5773 | | | 9 | | | 84.32 | 8.43 | .84 |
| 18 | .4397 | | | 6 | | | 56.07 | 5.60 | .56 |
| 19 | .5081 | | | 7 | | | 67.77 | 6.77 | .67 |
| 20 | .7888 | | | 6 | | | 47.91 | 4.77 | .48 |
| 21 | .5860 | | | 6 | | | 62.30 | 6.23 | .62 |
| 22 | 1.0000 | | | 7 | | | 63.50 | 6.35 | .64 |
| 23 | .8511 | | | 7 | | | 81.61 | 8.16 | .81 |
| 24 | .7351 | | | 7 | | | 97.23 | 9.66 | .97 |
| 25 | .5329 | | | 11 | | | 242.49 | 24.26 | 2.43 |
| 26 | .4895 | | | 12 | | | 287.87 | 28.73 | 2.87 |
| 27 | .4658 | | −.0010 | 12 | | 12 | 285.14 | 28.55 | 2.84 |
| 28 | .5482 | | | 5 | | | 61.73 | 6.16 | .63 |
| 29 | .4785 | | −.0013 | 12 | | 11 | 373.30 | 37.20 | 3.26 |
| 30 | .6331 | | −.0004 | 14 | | 14 | 462.38 | 46.30 | 4.62 |
| 31 | .5977 | | | 13 | | | 424.15 | 42.44 | 4.24 |
| 32 | .5084 | | −.0002 | 15 | | 14 | 603.30 | 60.34 | 5.45 |
| 33a | .4829 | | −.0010 | 15 | | 14 | 1083.38 | 108.09 | 9.82 |
| 33b | .4829 | | −.0010 | 15 | | 14 | 1228.91 | 123.50 | 11.04 |
| 34 | .6131 | | | 3 | | | 65.29 | 6.42 | .67 |
| 35 | .8403 | | | 10 | | | 538.49 | 54.32 | 5.44 |

All GEI values for *maxit* = 100,000 are reported. Blank entries for GEI in the 10,000 and 1000 columns indicate that these levels of *maxit* produced the same GEI value as 100,000 restarts. All values of $K$ for *maxit* = 100,000 are reported. Blank entries for $K$ occur in the columns for 10,000 and 1000 if these values of *maxit* matched the solution for *maxit* = 100,000. Computation times for all levels of *maxit* are provided to show that they decline by an order of magnitude as *maxit* declines by an order of magnitude.

Tunnukij & Hicks, 2009), but ant colony (Li et al., 2010), simulated annealing (Wu et al., 2008), variable neighborhood search (Paydar & Saidi-Mehrabad, 2013) and water flow-like (Wu et al., 2010) methods have also been applied. The results reported herein have shown that iterated local search is competitive with the more sophisticated metaheuristics; producing comparable and, in some cases, superior partitions. This finding is consistent with the exceptionally good performance of iterated local search heuristics in other discrete optimization contexts (Lourenço et al., 2003, 2010).

An inherent advantage of the ILS heuristic is that the parameter selection process is generally less daunting than it is for simulated annealing (e.g., initial temperature, temperature length, cooling parameter, etc.), genetic algorithms (e.g., population size, mutation probability, termination criterion, etc.), and other metaheuristics. Nevertheless, some parameterization is still necessary for the ILS heuristic. In our implementation, two parameters were required: (i) an upper limit on the number of successive values of $K$ evaluated with no improvement in the best-found value for the GEI (*maxnoimp*) and (ii) a limit on the number of search iterations for each number of cells (*maxit*). The selection of the *maxnoimp* parameter is trivial, with the default value being *maxnoimp* = 1. We used *maxnoimp* = 2 to provide a little extra precaution; however it is unlikely that larger values would be necessary. The computational requirement of the algorithm is primarily contingent on the *maxit* parameter. Fortunately, using a limit as small as *maxit* = 10,000, which is very fast, yielded exceptional results. The *maxit* = 10,000 setting matched the *maxit* = 100,000 setting for all 37 problems when residual cells were permitted, and 36 of the 37 problems when residual cells were forbidden.

### 7.2. Extensions

We focused on the cell formation problem of maximizing the GEI because it is arguably the most common objective function in the recent literature and, therefore, there were plenty of reported results to use as a basis for comparison to our proposed method. Moreover, the maximization of the GEI fits comfortably within the ILS framework. The PERTURBATION phase produces partitions that are not locally optimal; however, they are feasible solutions. The RELOCATION phase refines the partition to local optimality. It is important to recognize, however, that many practical cell formation problems incorporate other objectives and constraints pertaining to various demand and production factors (Pandian & Mahapatra, 2009; Schaller, 2005, 2007). These additional considerations may include intra-cell transfer costs, operation times and processing sequences for parts, and production costs and capacities for machines. The inclusion of many of these considerations would result in a situation whereby a perturbation process might produce infeasibility (e.g., the violation of processing sequence constraints or production capacity limits), not just suboptimality. Accordingly, a greedy construction heuristic can be used to return the solution to feasibility. Under these circumstances, a possible alternative to ILS could be based on the closely related iterated greedy search (IGS) paradigm, which was formalized by Ruiz and Stützle (2007, pp. 2035–2036). Unlike ILS, which iterates over standard local search operations (e.g., transfers, insertions, or exchanges), IGS involves iteration over a construction heuristic. More specifically, when using IGS, solutions are iteratively deconstructed (and rendered infeasible) and re-established using a greedy construction heuristic. Like ILS, IGS has proved effective for a variety of discrete optimization problems (Brusco & Jacobs, 1993; Jacobs & Brusco, 1995; Pan, Tasgetiren, & Liang, 2008; Quevedo-Orozco & Rios-Mercado, 2015; Rodriguez, Lozano, Blum, & Garcia-Martinez, 2013; Ruiz & Stützle, 2007, 2008; Zhu, Zheng, Zhang, & Cai, 2013).

Finally, it is important to recognize that ILS and IGS can be integrated easily with other metaheuristic procedures. For example, Brusco and Jacobs (1993) effectively embedded IGS within a simulated annealing framework, and later used the same approach in an application related to the scheduling of airport ground personnel (Brusco et al., 1995). More recently, Quevedo-Orozco and Rios-Mercado (2015) integrated ILS with variable neighborhood search in the development of a heuristic for a capacitated $p$-center location problem. Accordingly, one promising extension of this paper would be the integration of ILS with metaheuristics in an effort to tackle large part-machine clustering matrices.

## References


Askin, R. G., & Subramanian, S. P. (1987). A cost-based heuristic for group technology configuration. *International Journal of Production Research, 25*(1), 101–113.

Banfield, C. F., & Bassill, L. C. (1977). Algorithm AS 113: A transfer algorithm for non-hierarchical classification. *Applied Statistics, 26*(2), 206–210.

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

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

Brusco, M. J. (2015). An exact algorithm for maximizing grouping efficacy in part-machine clustering. *IIE Transactions, 47*(6), 653–671.

Brusco, M. J., & Jacobs, L. W. (1993). A simulated annealing approach to the cyclic staff scheduling problem. *Naval Research Logistics, 40*(1), 69–84.

Brusco, M. J., Jacobs, L. W., Bongiorno, R. J., Lyons, D., & Tang, B. (1995). Improving personnel scheduling at airline stations. *Operations Research, 43*(5), 741–751.

Brusco, M. J., & Steinley, D. (2007). Exact and approximate algorithms for part-machine clustering based on a relationship between interval graphs and Robinson matrices. *IIE Transactions, 39*(10), 925–935.

Bychkov, I., Batsyn, M., & Pardalos, P. M. (2014). Exact model for the cell formation problem. *Optimization Letters, 8*(8), 2203–2210.

Carrie, A. S. (1973). Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research, 11*(4), 399–416.

Chan, H. M., & Milner, D. A. (1982). Direct clustering algorithm for group formation in cellular manufacturing. *Journal of Manufacturing Systems, 1*(1), 65–74.

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

Chandrasekharan, 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*(3), 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.

Clapham, C. (1996). *The concise Oxford dictionary of mathematics* (2nd ed.). Oxford, England: Oxford University Press.

Crama, Y., & Oosten, M. (1996). Models for machine-part grouping in cellular manufacturing. *International Journal of Production Research, 34*(6), 1693–1713.

Dinkelbach, W. (1967). On non-linear fractional programming. *Management Science, 13*(7), 492–498.

Dong, X., Nowak, M., Chen, P., & Lin, Y. (2015). Self-adaptive perturbation and multi-neighborhood search for iterated local search on the permutation flow shop problem. *Computers and Industrial Engineering, 87*(1), 176–185.

Elbenani, B., & Ferland, J. A. (2012). Cell formation problem solved exactly with the Dinkelbach algorithm. CIRRELT-2012-07 (pp. 1–14), Montreal, Quebec.

Ghosh, T., Sengupta, S., Chattopadhyay, M., & Dan, P. D. (2011). Meta-heuristics in cellular manufacturing: A state-of-the-art review. *International Journal of Industrial Engineering Computations, 2*(1), 87–122.

Gonçalves, J. F., & Resende, M. G. C. (2004). An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering, 47*(2–3), 247–273.

Hartigan, J. A. (1975). *Clustering algorithms*. New York: Wiley.

Jacobs, L. W., & Brusco, M. J. (1995). Note: A local-search heuristic for large set-covering problems. *Naval Research Logistics, 42*(7), 1129–1140.

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*(7), 2059–2079.

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

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

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*(2), 233–243.

Kumar, K. R., Kusiak, A., & Vannelli, A. (1986). Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research, 24*(3), 387–397.

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

Kusiak, A., & Cho, M. (1992). Similarity coefficient algorithm for solving the group technology problem. *International Journal of Production Research, 30*(11), 2633–2646.

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

Li, X., Baki, M. F., & Aneja, Y. P. (2010). An ant colony optimization metaheuristic for machine-part cell formation problems. *Computers and Operations Research, 37*(12), 2071–2081.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In F. Glover & G. A. Kochenberger (Eds.). *Handbook of metaheuristics, international series in operations research & management science* (Vol. 57, pp. 320–352). New York: Springer.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated local search: Framework and applications. In M. Gendreau & J.-Y. Potvin (Eds.). *Handbook of metaheuristics, International series in operations research & management science* (Vol. 146, 2nd ed., pp. 363–397). New York: Springer.

McCormick, W. T., Schweitzer, P. J., & White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research, 20*(5), 993–1009.

Mosier, C. T., & Taube, L. (1985a). The facets of group technology and their impact on implementation. *Omega, 13*(5), 381–391.

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

Noktehdan, A., Karimi, B., & Kashan, A. H. (2010). A differential evolution algorithm for the manufacturing cell formation problem using group based operators. *Expert Systems with Applications, 37*(7), 4822–4829.

Pan, Q.-K., Tasgetiren, M. F., & Liang, Y.-C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers and Industrial Engineering, 55*(4), 795–816.

Pandian, R. S., & Mahapatra, S. S. (2009). Manufacturing cell formation with production data using neural networks. *Computers and Industrial Engineering, 56*(4), 1340–1347.

Papaioannou, G., & Wilson, J. M. (2010). The evolution of cell formation problem methodologies based on recent studies (1997–2008): Review and directions for future research. *European Journal of Operational Research, 206*(3), 509–521.

Paydar, M. M., & Saidi-Mehrabad, M. (2013). A hybrid-genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. *Computers and Operations Research, 40*(4), 980–990.

Quevedo-Orozco, D. R., & Rios-Mercado, R. Z. (2015). Improving the quality of heuristic solutions for the capacitated vertex p-center problem through iterated greedy local search with variable neighborhood descent. *Computers and Operations Research, 62*(October), 133–144.

Rodriguez, F. J., Lozano, M., Blum, C., & Garcia-Martinez, C. (2013). An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. *Computers and Operations Research, 40*(7), 1829–1841.

Ruiz, R., & Stützle, T. (2007). A simple and effective greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*(3), 2033–2049.

Ruiz, R., & Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research, 187*(3), 1143–1159.

Schaller, J. (2005). Tabu search procedures for the cell formation problem with intra-cell transfer costs as a function of cell size. *Computers and Industrial Engineering, 49*(3), 449–462.

Schaller, J. (2007). Designing and redesigning cellular manufacturing systems to handle demand changes. *Computers and Industrial Engineering, 53*(3), 478–490.

Seifoddini, H. (1989). A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. *International Journal of Production Research, 27*(7), 1161–1165.

Seifoddini, H., & Wolfe, P. M. (1986). Application of the similarity coefficient method in group technology. *IIE Transactions, 18*(3), 271–277.

Silva, M. M., Subramanian, A., & Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers and Operations Research, 53*(1), 234–249.

Srinivasan, G., Narendran, T. T., & Mahadevan, B. (1990). An assignment model for the part-families problem in group technology. *International Journal of Production Research, 28*(1), 145–152.

Stanfel, L. E. (1985). Machine clustering for economic production. *Engineering Costs and Production Economics, 9*(1–3), 73–81.

Tariq, A., Hussain, I., & Ghafoor, A. (2009). A hybrid genetic algorithm for machine part grouping. *Computers and Industrial Engineering, 56*(1), 347–356.

Tunnukij, T., & Hicks, C. (2009). An enhanced grouping genetic algorithm for solving the cell formation problem. *International Journal of Production Research, 47*(7), 1989–2007.

Waghodekar, P. H., & Sahu, S. (1984). Machine-component cell formation in group technology MACE. *International Journal of Production Research, 22*(6), 937–948.

Wu, T. H., Chang, C. C., & Chung, S. H. (2008). A simulated annealing algorithm for manufacturing cell formation problems. *Expert Systems with Applications, 34*(3), 1609–1617.

Wu, T. H., Chung, S. H., & Chang, C. C. (2010). A water flow-like algorithm for manufacturing cell formation problems. *European Journal of Operational Research, 205*(2), 346–360.

Zhu, Y., Zheng, Z., Zhang, X., & Cai, K. (2013). The *r*-interdiction median problem with probabilistic protection and its solution algorithm. *Computers and Operations Research, 40*(1), 451–462.