

Integrated Multi-period Production Scheduling and Cell Formation for Virtual Cellular Manufacturing Systems

K.L. Mak, J. Ma and L.X. Cui

Abstract—Virtual cellular manufacturing has attracted a lot of attention in recent years because traditional cellular manufacturing is inadequate under a highly dynamic manufacturing environment. In this paper, a new mathematical model is established for generating optimal production schedules for virtual cellular manufacturing systems operating under a multi-period manufacturing scenario. The objective is to minimize the total manufacturing cost over the entire planning horizon. A hybrid algorithm, based on the techniques of discrete particle swarm optimization and constraint programming is proposed to solve the complex production scheduling problem. Although particle swarm optimization performs competitively with other meta-heuristics for most optimization problems, the evolution process may be stagnated as time goes on if the swarm is going to be in equilibrium, especially for problems with hard constraints. Constraint programming, on the other hand, is an effective technique for solving problems with hard constraints. However, the technique may be inefficient if the feasible search space is very large. Therefore, the aim of the proposed hybrid algorithm is to combine the complementary advantages of particle swarm optimization and constraint programming to improve its search performance. The effectiveness of the proposed methodology is illustrated by solving a set of randomly generated test problems.

Index Terms—Backtracking, Constraint programming, Discrete particle swarm optimization, Virtual cellular manufacturing systems

I. INTRODUCTION

As global market becomes more and more competitive, manufacturing industries venture into a new phase, where both opportunities and challenges abound. Nowadays, manufacturing industries face relentless pressure manifested with a growing tendency of greater varieties of products with shorter manufacturing cycles and a highly dynamic manufacturing environment. Manufacturers thus should constantly adopt efficient manufacturing systems to respond to dynamic changes in customers' demand in order to keep their market share. Group technology (GT), a strategy proposed for this reason, is a manufacturing philosophy in which the parts having similarities are grouped together to

achieve higher level of integration between the design and manufacturing process of an industry. Cellular manufacturing (CM) and virtual cellular manufacturing (VCM) are two classical manufacturing layouts of this design and have attracted a lot of attention in recent years.

Cellular manufacturing has long been considered efficient in improving the productivity of batch production systems. In cellular manufacturing systems (CMSs), the parts that undergo similar manufacturing operations are grouped together to form a part family, and the workstations that produce those parts are physically grouped together to form a manufacturing cell for manufacturing these parts. Cellular manufacturing has the advantage in managing material flow easily due to the similarity of parts and proximity of the workstations. However, cellular manufacturing also has many drawbacks such as low machine utilization and unbalanced workload [1], because the machines are usually duplicated to restrict the manufacturing of parts in their respective manufacturing cells. The duplication of machines generates excessive production capacity in the manufacturing systems and increases the operational and maintenance costs.

In order to overcome the deficiencies of cellular manufacturing, a new concept called virtual cellular manufacturing was proposed. The main difference between virtual cellular manufacturing and cellular manufacturing is that the workstations in a virtual manufacturing cell are not grouped physically on the production floor. [2] and [3] reported that the virtual manufacturing cell appears as data files in a virtual cell controller. When a job arrives, the controller will take over the control of the relevant workstations to form a virtual manufacturing cell. The controller will also oversee the manufacturing of the job until it is finished. At the same time, the workstations will not be locked up on the formation of a virtual manufacturing cell, but are free to be assigned to other manufacturing cells to produce other jobs as long as there are excess capacities. When the job has been completed, the virtual manufacturing cell terminates and the workstations will be released and become available for other incoming jobs. A virtual cellular manufacturing system contains a number of virtual manufacturing cells which are dynamically created and controlled within it. Due to the sharing of workstations, it is expected that virtual cellular manufacturing has higher efficiency and flexibility than cellular manufacturing.

Although the concept of virtual cellular manufacturing has many advantages in terms of workstation utilization and workload balancing, production scheduling for virtual

K.L. Mak is Professor at the Dept. of Industrial and Manufacturing Systems Engineering (IMSE), The University of Hong Kong, Hong Kong, (e-mail: makkl@hkucc.hku.hk).

J. Ma is a PhD student at the Dept. of IMSE, HKU, Hong Kong, (e-mail: majun_nankai@yahoo.cn).

L.X. Cui is a PhD student at the Dept. of IMSE, HKU, Hong Kong, (e-mail: cui.lixin@hku.hk).

cellular manufacturing systems (VCMSs) has not received a lot of attention from the research community because of the complexity of the problem. Irani *et al.* [4] proposed a method based on graph theory and mathematical programming for forming virtual manufacturing cells. Baykasoglu [5] proposed a simulated annealing algorithm for developing a distributed layout for virtual cellular manufacturing cells. Mak *et al.* [6] developed a genetic methodology to generate effective production schedules for virtual cellular manufacturing systems operating under a single period scenario. In this paper, research will be extended into the multi-period situation.

The remainder of this paper is organized as follows. Section II presents the mathematical model and section III the hybrid algorithm. Section IV analyses the computation results obtained from solving a set of randomly generated problems. Finally, the conclusions are given in section V.

II. MATHEMATICAL MODEL

In this section, a mathematical model is established to describe the characteristics of multi-period VCMSs for the purpose of generating optimal production schedules. In the model, there are W workstations ($w=1,2,...,W$) with certain production capacities. The planning horizon contains P periods ($p=1,2,...,P$), each of which is further divided into a certain number of time slices with the same length ($s=1,2,...,S$). Some jobs are to be produced in each period and the objective is to minimize the total manufacturing cost over the entire planning horizon. To run the virtual cellular manufacturing systems well, three types of information [6] should be provided by the production schedule: (1) the types of workstations and other production resources that should be grouped to form virtual manufacturing cells; (2) the bottleneck workstations in each virtual manufacturing cell and the most appropriate rates to process the assigned jobs; (3) the times to create and terminate the virtual manufacturing cells.

This research is conducted under following assumptions.

1. Each type of job consists of a certain number of operations that must be manufactured according to the production route;
2. All types of operations of the same job must be handled on the same workstation in each period;
3. The processing time of each operation of a job is deterministic and known. Moreover, the processing time of an operation on any workstation that can handle it is the same;
4. The production volume and due date of each job in each period is deterministic and known;
5. The distance between any two workstations and the transportation cost of each job between any two workstations are deterministic and known;
6. Each period within the planning horizon is divided into a number of time slices of equal length. In addition, no work-in-process is allowed. That is, the processing rate of each job must satisfy the condition such that the production output of an operation in a time slice must be equal to that

of its preceding operation in the last time slice, and that of its succeeding operation in the next time slice;

7. Each workstation can handle at most one operation at a time and each operation cannot be interrupted once started in any time slice;
8. Compared with the inventory-holding cost and the manufacturing cost, the subcontracting cost of each job is much higher;
9. The transportation time of material and the machine setup time can be negligible.

The following parameters are used in the development of the mathematical model:

$r_{j,p}$ =the production route of job j in period p

$w(r_{j,p})$ =the workstation w used in production route $r_{j,p}$

$O_{j,i}$ =the operation i of job j

$DD_{j,p}$ =the due date of job j in period p

$V_{j,p}$ =the volume of customer need of job j in period p

K_j =the number of operations of job j

d_{w_1,w_2} =the distance between workstation w_1 and w_2

PL =the length of a time slice

$MC_{w,s,p}$ =the maximum capacity of workstation w in time slice s of period p

$pt_{j,i,w(r_{j,p})}$ =processing time for producing one unit of operation i of job j on workstation $w(r_{j,p})$

$D(r_{j,p})$ =the total distance of production route $r_{j,p}$

In addition, α_j is the cost of moving one unit of job j per unit distance; ψ_w is the operating cost of workstation w per unit time; $\gamma_{j,p}$ is the inventory-holding cost of job j in period p ; $\sigma_{j,p}$ is the subcontracting cost of job j in period p .

Decision variables:

$PR_{j,i,w(r_{j,p}),s,p}$ =processing rate of operation i of job j on $w(r_{j,p})$ in time slice s of period p

$st_{j,i,w(r_{j,p}),s,p}$ =the start time of operation i of job j on $w(r_{j,p})$ in time slice s of period p

$ft_{j,i,w(r_{j,p}),s,p}$ =the finish time of operation i of job j on $w(r_{j,p})$ in time slice s of period p

$X_{j,i,w(r_{j,p}),s,p}$ =equal to 1 if operation i of job j is launched by $w(r_{j,p})$ in time slice s of period p ; otherwise, it is 0.

$Y_{j,i,w(r_{j,p}),s,p}$ =equal to 1 if operation i of job j is processed by $w(r_{j,p})$ in time slice s of period p ; otherwise, it is 0.

The mathematical model thus has the following form [7]:

Minimize

$$\sum_{p=1}^P \sum_{j=1}^N \alpha_j \bar{V}_{j,p} D(r_{j,p}) + \sum_{j=1}^N \sum_{p=1}^P \gamma_{j,p} IV_{j,p} + \sum_{j=1}^N \sum_{p=1}^P \sigma_{j,p} SV_{j,p} + \sum_{w=1}^W \psi_w \sum_{p=1}^P \sum_{s=1}^S \sum_{j=1}^N \sum_{i=1}^{K_j} Y_{j,i,w(r_{j,p}),s,p} PR_{j,i,w(r_{j,p}),s,p} pt_{j,i,w(r_{j,p})}$$

Where

$$SV_{j,p} = \max\{V_{j,p} - \sum_{s=1}^{DD_{j,p}} \sum_{\forall w(r_{j,p})} PR_{j,K_j,w(r_{j,p}),s,p} - IV_{j,p-1}, 0\} \quad \forall j, p \quad (1)$$

$$IV_{j,p} = \max\left\{\sum_{s=1}^{DD_{j,p}} \sum_{\forall w(r_{j,p})} PR_{j,K_j,w(r_{j,p}),s,p} + IV_{j,p-1} - V_{j,p}, 0\right\} + \sum_{s=DD_{j,p}+1}^S \sum_{\forall w(r_{j,p})} PR_{j,K_j,w(r_{j,p}),s,p} \quad \forall j, p \quad (2)$$

$$D(r_{j,p}) = \sum_{\forall (w_1, w_2) \in r_{j,p}} d_{w_1, w_2} \quad \forall j, p \quad (3)$$

Subject to

$$X_{j,i,w(r_{j,p}),s+i-1,p} = X_{j,i+1,w(r_{j,p}),s+i,p} \quad \forall j, i, w(r_{j,p}), s, p \quad (4)$$

$$PR_{j,i,w(r_{j,p}),s+i-1,p} = PR_{j,i+1,w(r_{j,p}),s+i,p} \quad \forall j, i, w(r_{j,p}), s, p \quad (5)$$

$$\sum_{\forall w(r_{j,p})} \sum_{s=1}^{S-(K_j-1)} \prod_{i=1}^{K_j} X_{j,i,w(r_{j,p}),s+i-1,p} = 1 \quad \forall j, p \quad (6)$$

$$Y_{j,i,w(r_{j,p}),s',p} \leq (1 - X_{j,i,w(r_{j,p}),s,p})G \quad \forall j, i, w(r_{j,p}), p, s' < s \quad (7)$$

$$PR_{j,i,w(r_{j,p}),s,p} \begin{cases} > 0 \Leftrightarrow Y_{j,i,w(r_{j,p}),s,p} = 1 \\ < 0 \Leftrightarrow Y_{j,i,w(r_{j,p}),s,p} = 0 \end{cases} \quad \forall O_{j,i,w(r_{j,p}),s,p} \quad (8)$$

$$\sum_{s=1}^S \sum_{\forall w(r_{j,p})} PR_{j,i,w(r_{j,p}),s,p} = \bar{V}_{j,p} \quad \forall O_{j,i,p} \quad (9)$$

$$st_{j,i,w(r_{j,p}),s,p} > (p-1) \times S \times PL + (s-1) \times PL \quad \forall O_{j,i,w(r_{j,p}),s,p} \quad (10)$$

$$ft_{j,i,w(r_{j,p}),s,p} = st_{j,i,w(r_{j,p}),s,p} + PR_{j,i,w(r_{j,p}),s,p} pt_{j,i,w(r_{j,p})} \quad \forall O_{j,i,w(r_{j,p}),s,p} \quad (11)$$

$$\sum_{j=1}^N \sum_{i=1}^{K_j} Y_{j,i,w(r_{j,p}),s,p} PR_{j,i,w(r_{j,p}),s,p} pt_{j,i,w(r_{j,p})} \leq MC_{w,s,p} \quad \forall w(r_{j,p}), s, p \quad (12)$$

$$IV_{j,0} = 0 \quad \forall j \quad (13)$$

$$X_{j,i,w(r_{j,p}),s,p}, Y_{j,i,w(r_{j,p}),s,p} \in \{0,1\} \quad \forall j, i, w, s, p \quad (14)$$

where G is a large number.

The objective of the mathematical model is to minimize the total manufacturing cost over the entire planning horizon, including material transportation cost, inventory-holding cost, subcontracting cost, and machine operating cost. Equations (1) and (2) denote the method of calculating inventory-holding volume and subcontract volume of each job in each period respectively. Equations (3) show the method of calculating the travelling distance of a production route. Constraints (4) ensure that when operation i of job j has been finished, operation $i+1$ of job j must start immediately. Constraints (5) make sure that the processing rate of an operation in a time slice must be equal to that of its preceding operation in the last time slice, and that of its succeeding operation in the next time slice in each period. It shows that work-in-process is not allowed in the manufacturing system. Constraints (6) ensure that the

starting times of all operations must be within the planning horizon and each job can only have a unique production route in each period. Constraints (7) require that in each period no operation can start before the time slice from which production is launched. Constraints (8) restrict the processing rate to be greater than or equal to zero. Constraints (9) denote the relationship between the processing rate and the production volume of each job in each period. Constraints (10) and (11) describe the constraints governing the starting time and the finishing time of each operation. Constraints (12) make sure that all jobs assigned to a machine can be finished in each time slice of each period. Constraints (13) indicate that there is no inventory of any job at the beginning of the planning horizon and constraints (14) indicate that these variables are binary.

TABLE 1 A SAMPLE OF PRODUCTION OUTPUTS OF A JOB

Serial number of operations		1	2	3
Workstation No.		2	7	4
Time slice No.	1	5(5)	3(3)	0
	2	2(2)	5(8)	3(9)
	3	3(4)	2(5)	5(6)
	4	0	3(3)	2(7)
	5	0	0	3(6)

To illustrate constraints (4) and (5) of the mathematical model, Table 1 provides an example of the production outputs of a job in a period. In this table, the value in the bracket is the maximum processing rate of an operation in a time slice. For example, the maximum processing rate of operation 1 in time slice 1 is 5. The maximum processing rate of operation 2 in time slice 2 is 8, and that of operation 3 in time slice 3 is 6. Thus, the feasible processing rate of operation 1 in time slice 1 (operation 2 in time slice 2 and operation 3 in time slice 3) is 5. This ensures that no work-in-process exists between operations of a job. After scheduling this job, workstation 2 still has a certain amount of remaining capacity in time slice 2, which allows it to be assigned to other manufacturing cells to produce other incoming jobs.

III. HYBRID OPTIMIZATION ALGORITHM

In order to find an efficient and effective production schedule, this paper develops a new hybrid algorithm based on the techniques of constraint programming (CP) and discrete particle swarm optimization (DPSO).

A. Constraint programming

Constraint programming [8] is an effective methodology for solving difficult combinatorial problems by representing them as constraint satisfaction problems (CSPs). A constraint satisfaction problem usually consists of a set of variables, a domain for each variable, and a set of constraints restricting the values that the variables can simultaneously take.

There are mainly two important categories of constraints in production scheduling problems: precedence constraints and capacity constraints. A typical example of precedence constraints in the VCMS production scheduling problem is that an operation can start only after its

preceding operation has been finished. Capacity constraints usually refer to workstations and other production resources. An example in the production scheduling problem is that each machine must have enough capacity to produce the jobs assigned on it.

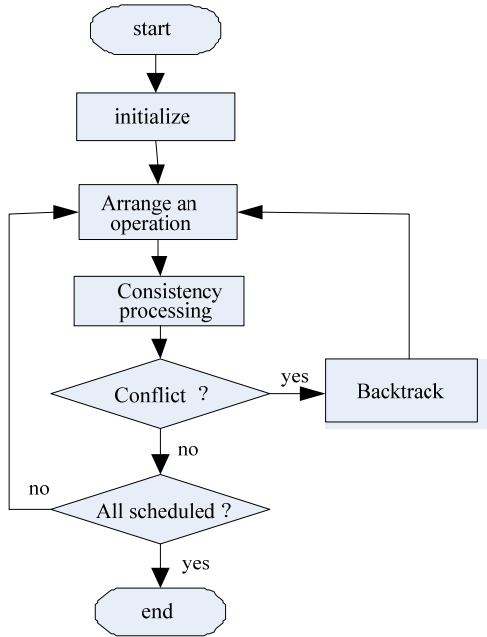


Figure 1 The procedure of constraint programming

Backtracking paradigm is a basic constraint propagation technique used to solve constraint satisfaction problems. The basic operation is to pick one variable at a time, and consider one value in its domain at a time, making sure that the newly picked label is compatible with the instantiated partial solution obtained so far. If the newly picked label violates certain constraints, then an alternative value, if it exists, is picked. If no value can be assigned to a variable without violating any constraint, it will backtrack to the most recently instantiated variable. This process continues until a feasible solution has been found or all possible combinations of labels have been tried and failed [8]. The procedure of constraint programming with backtracking propagation is presented in Figure 1.

B. Discrete particle swarm optimization

Particle swarm optimization (PSO), a population-based optimization approach inspired by the observations of bird flocking and fish schooling, was proposed by Kennedy and Eberhart in 1995 [9]. The basic idea of this approach is to locate the optimal or near optimal solution through cooperation and sharing of information among individuals in the swarm. The swarm is composed of a group of particles in a search space with two important characteristics, namely position and velocity. Each particle represents a potential solution, which flies through the hyperspace and has two essential reasoning capabilities: the memory of its own best position and the knowledge of the global or its neighborhood's best position. Particles within the swarm communicate information with each other and adjust their own position and velocity based on the information. At each step, the velocity of a particle and its new position will be updated respectively according to the following two

equations.

$$V_i = \omega V_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (G - X_i) \quad (15)$$

$$X_i = X_i + V_i \quad (16)$$

where X_i represents the position of particle i , V_i represents the velocity of particle i , P_i is the best previously visited position of particle i , G is the global best position, ω is the inertia weight that controls the impact of previous velocity on its current one, which is usually reduced dynamically to decrease the search area : $\omega_t = (\omega_{\max} - \omega_{\min}) \times \frac{t_{\max} - t}{t_{\max}} + \omega_{\min}$, in

which ω_{\max} and ω_{\min} denote the maximum value and the minimum value of the inertial weight respectively, and t_{\max} is the maximum number of iterations. Generally, the value of the velocity is restricted in the range $[-v_{\max}, v_{\max}]$ to control excessive roaming of particles outside the search space. Usually, v_{\max} is set with value 4.

Figure 2 shows the steps of particle swarm optimization.

```

Initialize parameters
Initialize populations
Evaluate
Do {
    Find the personal best
    Find the global best
    Update velocity
    Update position
    Evaluate
} While(Termination)
    
```

Figure 2 Particle swarm optimization procedure

In practice, many optimization problems such as production scheduling problem are set in discrete search space. To meet this demand, a discrete version of particle swarm optimization (DPSO) was proposed by Kennedy and Eberhart in 1997 [10]. DPSO has two main differences from the original one. First, the particle is composed of binary variables. Second, the velocity must be transformed into the change of probability, which is the chance of the variable taking the value 1. Usually, the transformation is achieved through the following sigmoid function.

$$s(V_i) = \frac{1}{1 + \exp(-V_i)} \quad (17)$$

Where $s(V_i)$ denotes the probability of corresponding bit taking value 1.

In this research, particle k at iteration t can be presented as $X_k^t = (X_k^{t,1}, \dots, X_k^{t,p}, \dots, X_k^{t,P})$, where $X_k^{t,p} = (S_k^{t,p}, M_k^{t,p})$, $S_k^{t,p}$ denotes the job production sequence in period p , and $M_k^{t,p}$ denotes workstation assignment of jobs in period p . The best solution found by particle k until iteration t is denoted as $P_k^t = (P_k^{t,1}, \dots, P_k^{t,P})$ and the best solution found by the swarm until iteration t is denoted as $P_g^t = (P_g^{t,1}, \dots, P_g^{t,P})$. The velocity of particle k at iteration t can be presented as $V_k^t = (V_k^{t,1}, \dots, V_k^{t,p}, \dots, V_k^{t,P})$, where $V_k^{t,p} = (VS_k^{t,p}, VM_k^{t,p})$,

$VS_k^{t,p}$ denotes the velocity of job production sequence in period p , and $VM_k^{t,p}$ denotes the velocity of workstation assignment of jobs in period P .

To facilitate understanding of the proposed methodology, a job production sequence in a period is used as an example to illustrate the construction of a particle.

In $S_k^{t,p}$, $S_k^{t,p} = (S_{k,1}^{t,p}, \dots, S_{k,N}^{t,p})$, and $S_{k,j}^{t,p} = (s_{k,j,1}^{t,p}, s_{k,j,2}^{t,p}, \dots, s_{k,j,N}^{t,p})$. $s_{k,j,d}^{t,p}$ is binary where it is equal to 1 if job j is in the d^{th} position of the production sequence; otherwise, it is equal to 0. For example, suppose the job production sequence in period 1 is (2, 3, 4, 1) in a particle. Then $s_{k,2,1}^{t,1} = s_{k,3,2}^{t,1} = s_{k,4,3}^{t,1} = s_{k,1,4}^{t,1} = 1$ and all other bits are equal to zero.

In $VS_k^{t,p}$, $VS_k^{t,p} = (VS_{k,1}^{t,p}, \dots, VS_{k,N}^{t,p})$ and $VS_{k,j}^{t,p} = (vs_{k,j,1}^{t,p}, vs_{k,j,2}^{t,p}, \dots, vs_{k,j,N}^{t,p})$. High value of $vs_{k,j,d}^{t,p}$ means that job j is more likely to be placed in the d^{th} position in period p , while low value indicates that it is better to place this job in another position. In each iteration, the velocity is updated according to equation (15), and then converted to the change of probability via following sigmoid function.

$$s(vs_{k,j,d}^{t,p}) = \frac{1}{1 + \exp(-vs_{k,j,d}^{t,p})} \quad d = 1, 2, \dots, N_p \quad (18)$$

where N_p is the number of jobs in period p , $s(vs_{k,j,d}^{t,p})$ denotes the probability of placing job j in the d^{th} position of the production sequence in period p .

In the iteration process of discrete particle swarm optimization, each particle should be decoded into a complete production schedule. In the example of job production sequencing, the construction of a job production sequence in period p starts from a null sequence and then places an unscheduled job j in the d^{th} position from $d = 1$ to N_p according to following probability [11]:

$$q_{k,d}^{t,p}(j) = \frac{s(vs_{k,j,d}^{t,p})}{\sum_{j \in U} s(vs_{k,j,d}^{t,p})} \quad (19)$$

where U is the set of of unscheduled jobs in period p , and $q_{k,d}^{t,p}(j)$ is the probability of placing job j in the d^{th} position. A complete job production sequence of a period has been constructed when each of the jobs in this period has been assigned to a position.

C. The proposed hybrid algorithm

Particle swarm optimization is an effective algorithm for solving many types of optimization problems. However, if the swarm is going to be in equilibrium, the evolution process will be stagnated as time goes on [12]. Constraint programming is specialized for solving problems with hard constraints, but may be inefficient when the feasible search space is very large. Hence, a hybrid algorithm (CPSO) which combines their complementary advantages to improve the search process is proposed in this research. The procedure of the proposed hybrid algorithm is summarized briefly in Figure 3.

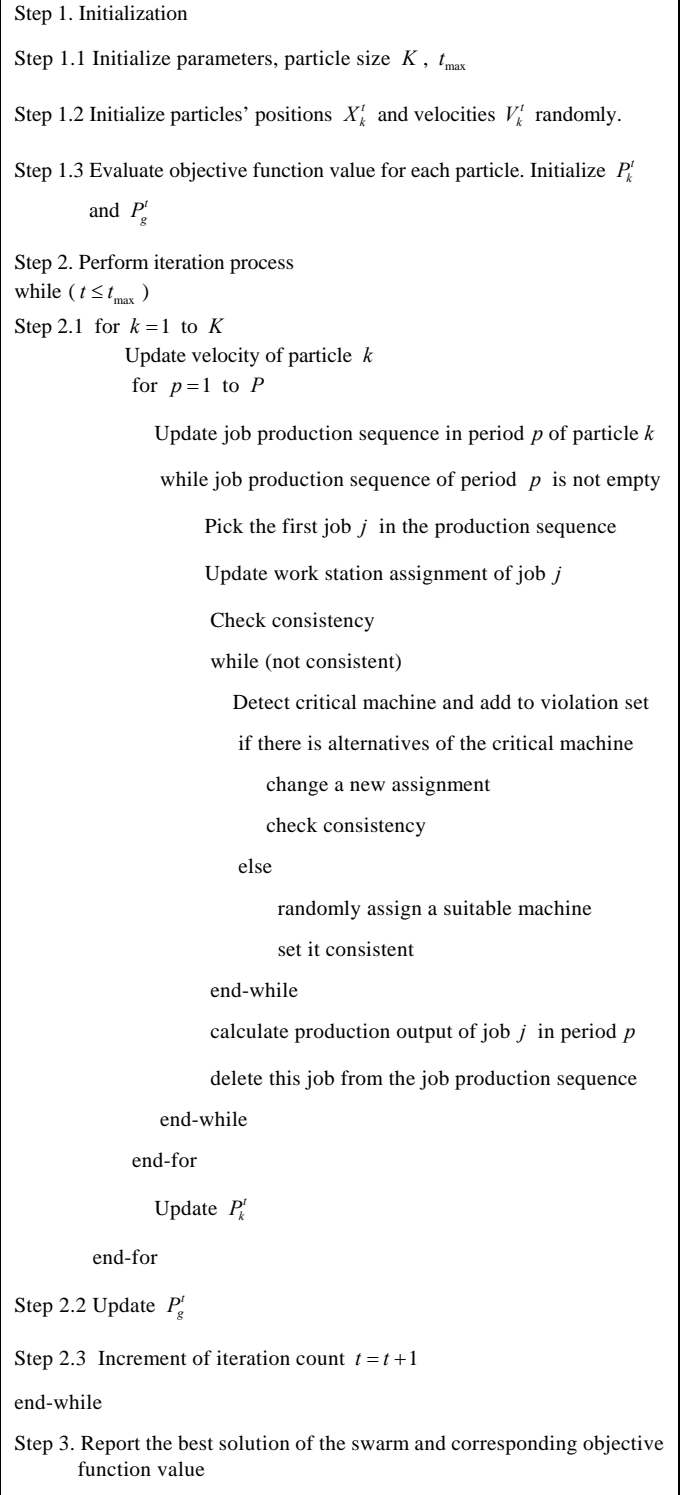


Figure 3 The procedure of the hybrid algorithm

In this research, the planning horizon consists of multi periods. Due to capacity limitations of various workstations, some jobs may not be finished before their due dates in some periods while some workstations may have excess capacities in some periods. Hence, it is necessary to make a trade-off among inventory-holding cost, subcontracting cost, and workstation utilization. A concept called extra capacity [13] is firstly introduced. The extra capacity of a job in a period is the number of units of the job that can be produced in excess in this period without affecting other scheduled jobs. When a job cannot be finished before its due date in a period and has extra capacity in previous periods, the extra capacity will be utilized to produce it and

the inventory is carried to that period. If there is still backlog of this job after utilizing the extra capacity, the amount will be subcontracted in order to meet customers' demand.

In practice, the subcontracting cost of a job is usually much higher than its inventory-holding cost and manufacturing cost. Hence, in the proposed algorithm, if a job cannot be finished before its due date even after utilizing the extra capacity of previous periods, it will be treated as inconsistency and the critical workstation will be detected to try a new assignment. In addition, the procedure in figure 3 indicates that constraint propagation in this research takes the form of single-level backtracking. When inconsistency occurs, the algorithm will detect the critical resource and check whether this critical resource has any alternatives. If the answer is positive, another workstation selected from the alternatives will be assigned to perform the operation; otherwise, the algorithm will not backtrack to the most recently scheduled job and just randomly assign a suitable workstation for it according to DPSO mechanism regardless of consistency, and then continue to schedule the next job until all jobs have been scheduled.

D. Heuristic for production output

However, the proposed hybrid algorithm does not take the production output into consideration explicitly. A heuristic will be adopted to determine the production outputs of each job in each time slice of each period [14]. Figure 4 shows the pseudo-code of the heuristic for determining the production outputs of the jobs in a period.

```

while production sequence of this period is not empty do
    Pick the first job  $j$  in the production sequence
    Set  $s=1, RemQty=V_{j,p}$ 
    while  $RemQty>0$  and  $p \leq DD_{j,p}-K_j+1$  do
         $minPR = RemQty$ 
        for  $i=1$  to  $K_j$ 
            Find maximum  $PR_{j,i,s+i-1}$ 
             $minPR = \min(minPR, PR_{j,i,s+i-1})$ 
        end-for
        if  $minPR>0$  then
             $PR_{j,i,s+i-1} = minPR, \forall i$ 
             $RemQty = RemQty - minPR$ 
            Update capacities of work stations
        end-if
         $s=s+1$ 
    end-while
    Remove job  $j$  from production sequence.
end-while

```

Figure 4 The heuristic for determining production output of jobs

As shown in the example of table 1, the maximum processing rate of an operation in a time slice (shown in brackets in table 1) must satisfy the constraints governing the remaining capacity of the assigned workstation in that time slice. The feasible production output ($minPR$) is the smallest value of the maximum feasible production outputs of all

operations. This ensures that no work-in-process is allowed between operations. If $minPR$ is positive, the remaining capacity of workstations and remaining production quantity ($RemQty$) are updated; otherwise, production of this job can not take place in time slice s of period p . This process repeats until the production outputs of all jobs in this period have been determined.

IV. COMPUTATION RESULTS

In this section, the performance of the proposed hybrid algorithm is analysed by comparing the results obtained from solving a set of randomly generated test problems with that of DPSO.

A. Test Problem Set and Parameters

The values of parameters used in the algorithm are as follows. Particle size is 100, maximum number of iterations is 100, maximum inertial weight is 0.8, minimum inertial weight is 0.2. c_1 and c_2 are both equal to 2, the velocity of particles is restricted in the range $[-4, 4]$. Each period contains 30 time slices, the length of which is 300 seconds. There are 3 or 4 operations required for producing each unit of a job. Each operation has a processing time randomly generated from $[20, 40]$. The number of units that should be produced (a job) in a period is randomly generated from $[50, 80]$. Table 2 shows the scheme used to generate the test problems. In this table, (p, n, d, s, m) is used to denote the parameter combination, where p is the number of periods, n is the number of jobs in each period, d denotes the due date, s denotes job subcontracting cost, and m denotes the number of workstations. For example, scheme 1 means that there are 5 periods in the planning horizon, the number of jobs in a period is randomly generated from $[5, 10]$, the due date of each job is randomly generated from $[\alpha S, \beta S]$, where $\alpha = 0.4, \beta = 0.7$, the subcontracting cost of each job per unit is generated from $[500, 1000]$, the number of workstations is 12.

TABLE 2 PARAMETER SCHEME

No.	Parameter value
1	(5, [5, 10], [0.4, 0.7], [500, 1000], 12)
2	(5, [5, 10], [0.6, 0.9], [500, 1000], 12)
3	(5, [5, 10], [0.4, 0.7], [1000, 2000], 12)
4	(5, [5, 10], [0.6, 0.9], [1000, 2000], 12)
5	(5, [5, 10], [0.4, 0.7], [500, 1000], 20)
6	(5, [5, 10], [0.6, 0.9], [500, 1000], 20)
7	(5, [5, 10], [0.4, 0.7], [1000, 2000], 20)
8	(5, [5, 10], [0.6, 0.9], [1000, 2000], 20)
9	(10, [5, 10], [0.4, 0.7], [500, 1000], 12)
10	(10, [5, 10], [0.6, 0.9], [500, 1000], 12)
11	(10, [5, 10], [0.4, 0.7], [1000, 2000], 12)
12	(10, [5, 10], [0.6, 0.9], [1000, 2000], 12)
13	(10, [10, 15], [0.4, 0.7], [500, 1000], 12)
14	(10, [10, 15], [0.6, 0.9], [500, 1000], 12)
15	(10, [10, 15], [0.4, 0.7], [1000, 2000], 12)
16	(10, [10, 15], [0.6, 0.9], [1000, 2000], 12)

B. Comparison with DPSO

In order to demonstrate the superiority of the proposed

hybrid algorithm over DPSO, five test problems are randomly generated under each scheme. The performances of the hybrid algorithm and DPSO are obtained by averaging the results of running the algorithms five times for each problem. Table 3 shows the performance comparison of the two algorithms after executing both algorithms for 100 iterations. In this table, cost diff and time diff are calculated according to the following equations.

$$\text{cost diff} = \frac{\text{cost of DPSO} - \text{cost of hybrid algorithm}}{\text{cost of DPSO}} \quad (20)$$

$$\text{time diff} = \frac{\text{CPU time of hybrid algorithm} - \text{CPU time of DPSO}}{\text{CPU time of DPSO}} \quad (21)$$

TABLE 3 COMPARISON WITHIN THE SAME ITERATIONS

No.	DPSO		HYBRID		Cost diff	Time diff
	cost	time	cost	time		
1	563218	70	531783	89	5.58%	27.14%
2	529196	87	482444	111	8.83%	27.59%
3	711755	108	578140	143	18.77%	32.41%
4	652706	88	560282	114	14.15%	29.55%
5	372065	79	345057	93	7.26%	17.72%
6	501924	78	486018	88	3.17	12.82%
7	714411	117	614808	140	13.94%	19.66%
8	525589	83	503133	96	4.27%	15.66%
9	1140888	204	1051389	247	7.84%	21.08%
10	1176895	177	1100422	218	6.50%	23.16%
11	1175947	182	1029194	219	12.48%	20.33%
12	1046388	174	927337	206	11.38%	18.39%
13	2724866	568	2460257	793	9.71%	39.61%
14	2711063	510	2431879	709	10.30%	39.02%
15	4435985	620	3646998	867	17.79%	39.84%
16	3784960	629	2960653	890	21.78%	41.49%

From Table 3, it is clear that the proposed hybrid algorithm can obtain much better schedule solutions at the expense of longer computational time. Furthermore, the improvement on solution quality is more obvious when the problem size becomes larger, the due date becomes tighter, or the subcontract cost becomes higher.

Taking scheme 1 for example, figure 5 shows the performances of the two algorithms in locating good solutions within the same number of iterations, and figure 6 compares the computational times of the two algorithms, when the number of iterations is the same.

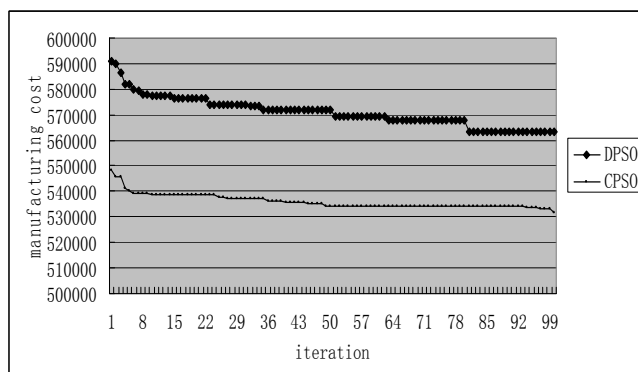


Figure 5 Comparison of cost with the same number of iteration

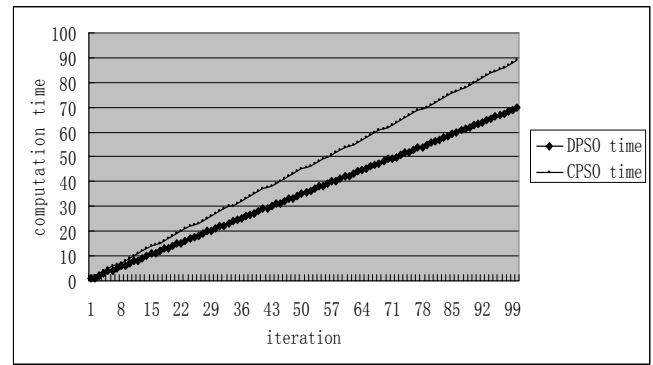


Figure 6 Comparison of computation time with same iterations

Since the hybrid algorithm needs longer computational time for the same number of iterations. It is necessary to compare the performance of these two algorithms with the same computational time. Table 4 lists the comparison results. In this table, T_{ps0} represents the computational time of DPSO running 100 iterations.

TABLE 4 COMPARISON WITHIN THE SAME COMPUTATION TIME

No.	Cost diff			
	$1/3 T_{ps0}$	$1/2 T_{ps0}$	$2/3 T_{ps0}$	T_{ps0}
1	6.29%	6.29%	5.89%	5.58%
2	9.46%	9.23%	9.26%	8.54%
3	20.33%	19.68%	18.06%	18.63%
4	13.65%	13.36%	13.53%	14.08%
5	6.40%	6.26%	6.67%	7.26%
6	3.09%	2.56%	3.10%	2.74%
7	14.16%	14.46%	14.27%	13.94%
8	4.53%	4.44%	3.79%	3.96%
9	7.29%	7.44%	7.82%	7.83%
10	6.17%	6.53%	6.70%	6.46%
11	14.46%	13.56%	13.14%	12.47%
12	12.48%	11.82%	11.95%	11.32%
13	9.63%	9.64%	9.51%	9.70%
14	10.13%	10.10%	10.32%	10.11%
15	17.29%	16.99%	17.40%	17.62%
16	21.88%	22.00%	21.80	21.72%

It is clear that the proposed hybrid algorithm has better performance in locating good schedules within the same computational time. Also, the improvement becomes more obvious when the problem size becomes larger, the due date becomes tighter, or the subcontract cost becomes higher. Hence, the proposed hybrid algorithm is suitable for solving real industrial problems which usually have large problem size.

V. CONCLUSIONS

This paper focuses on solving the integrated production scheduling and cell formation problem for virtual cellular manufacturing systems operating under a multi-period manufacturing scenario. The objective is to minimize the total manufacturing cost within the entire planning horizon, including material transportation cost, machine operating cost, inventory-holding cost, and subcontracting cost. A new mathematical model has been established to describe the characteristics of a virtual cellular manufacturing system and

a hybrid algorithm which combines the advantages of the techniques of constraint programming and discrete particle swarm optimization has been developed to generate effectively the optimal production schedule and cell formation for the manufacturing system. Computational experiments using a set of randomly generated test problems demonstrates that the hybrid algorithm can generate better production schedules and cell formations with the same number of iterations or the same amount of computational times, especially when the size of the problem is large.

ACKNOWLEDGEMENT

The work described in this paper was supported by a grant from the research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKU 715409E).

REFERENCE

- [1] U. Wemmerlov and N. Hyer, "Cellular manufacturing in the US industry: a survey of users," *Int. J. Prod. Res.*, vol. 27, no. 9, pp. 1511-1530, 1989.
- [2] J.R. Drolet, "Scheduling virtual cellular manufacturing systems," PhD Dissertation, Purdue University, West Lafayette, IN, 1989.
- [3] V.R. Kannan and S. Ghosh, "Cellular manufacturing using virtual cells", *Int. J. Op. Prod. Manage.*, vol. 16, no. 4, pp. 99-112, 1996.
- [4] S.A. Irani, T.M. Cavalier, and P.H. Cohen., Virtual manufacturing cells: Exploiting layout design and intercell flows for the machine-sharing problem. *Int. J. Prod. Res.*, vol. 31, no. 4, pp. 791-810, 1993.
- [5] A. Baykasoglu., Capability-based distributed layout approach for virtual manufacturing cells. *Int. J. Prod. Res.*, vol. 41, no. 11, pp. 2597-2618, 2003.
- [6] K.L. Mak, J.S.K. Lau and X.X. Wang, "A genetic scheduling methodology for virtual cellular manufacturing systems: an industrial application," *Int. J. Prod. Res.*, vol. 43, no. 12, pp. 2423-2450, 2005.
- [7] K.L. Mak and J.Ma, "A Novel Hybrid Algorithm for Multi-Period Production Scheduling of Jobs in Virtual Cellular Manufacturing Systems," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2011, WCE 2011, 6-8 July, 2011, London, U.K., pp. 685-690.
- [8] E.P.K. Tsang, Foundation of Constraint Satisfaction. Academic Press, 1993.
- [9] J. Kenndey and R.C. Eberhart, "Particle swarm optimization," *IEEE international conference on neural network*, pp. 1942-1948, 1995.
- [10] J. Kenndey and R.C. Eberhart, "A discrete binary version of the particle swarm optimization," The world multiconference on systems, cybernetics and informatics, pp. 4104-4109, 1997.
- [11] C.J. Liao, C.T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Computers and Operations Research*, vol 34, pp. 3099-3111, 2007.
- [12] P.S. Shelokar, P. Siarry, V.K. Jayaraman, and B.D. Kulkarni., "Particle swarm and ant colony algorithms hybridized for improved continuous optimization", *Applied Mathematics and Computation*, vol. 188, pp. 129-142, 2007.
- [13] K.L. Mak, J. Ma, and W. Su, "Production scheduling for virtual cellular manufacturing systems with workforce constraints using a hybrid algorithm," *Proceeding of International Conference on Natural Computation*, vol. 8, pp. 4445-4449, 2010.
- [14] K.L. Mak, J. Ma, and W. Su, "A constraint programming approach for production scheduling of multi-period virtual cellular manufacturing systems," *Proceeding of International Conference on Natural Computation*, vol. 8, pp. 4440-4444, 2010.