Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 398123, 12 pages
http://dx.doi.org/10.1155/2013/398123

Research Article

A Dynamic Intelligent Decision Approach to Dependency Modeling of Project Tasks in Complex Engineering System Optimization

Tinggui Chen[1] and Renbin Xiao[2]

[1] *College of Computer Science & Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China*
[2] *Institute of Systems Engineering, Huazhong University of Science and Technology, Wuhan, Hubei Province 430074, China*

Correspondence should be addressed to Renbin Xiao; rbxiao@hust.edu.cn

Received 2 January 2013; Revised 25 March 2013; Accepted 25 March 2013

Academic Editor: Reza Jazar

Complex engineering system optimization usually involves multiple projects or tasks. On the one hand, dependency modeling among projects or tasks highlights structures in systems and their environments which can help to understand the implications of connectivity on different aspects of system performance and also assist in designing, optimizing, and maintaining complex systems. On the other hand, multiple projects or tasks are either happening at the same time or scheduled into a sequence in order to use common resources. In this paper, we propose a dynamic intelligent decision approach to dependency modeling of project tasks in complex engineering system optimization. The approach takes this decision process as a two-stage decision-making problem. In the first stage, a task clustering approach based on modularization is proposed so as to find out a suitable decomposition scheme for a large-scale project. In the second stage, according to the decomposition result, a discrete artificial bee colony (ABC) algorithm inspired by the intelligent foraging behavior of honeybees is developed for the resource constrained multiproject scheduling problem. Finally, a certain case from an engineering design of a chemical processing system is utilized to help to understand the proposed approach.

1. Introduction

Nowadays, complex engineering projects or design processes with long development times usually involve multiple disciplines and a great deal of effort. In general, the difficulties in design or development do not only simply arise from engineering complexity but also lie in the organizational sophistication necessary to manage this design or development process. Therefore, it is very important to optimize the design or development process. Usually, a complex system includes a large number of tasks or subprojects, and complex dependencies existing among tasks will cause resource competition and coordination. It means that in order to understand the implications of connectivity on different aspects of system performance, it is necessary to model the task dependencies and then sequence all project tasks to reveal the underlying structure of the design or development process.

In most of the literatures, graphs or directed graphs such as flow charts and signal flow diagrams are used to analyze complex processes. For example, digraphs are easy to assimilate until they become quite large and lose their intuitive [1]. Related to the diagraphs, adjacency matrices expedite the decomposition of large systems since computers can easily handle the matrices [2]. In this paper, we focus on two main issues, that is, task clustering as well as its sequencing, where there are more researches concentrating on the first one. For instance, Tseng and Jiao [3] proposed an approach through clustering analysis of a design matrix based on axiomatic design theory and aimed at implementing modular electrical design of electronic products at the system design level. Gershenson et al. [4] discussed the incorporation of modularization into mechanical designs. They also developed the measure of relative modularity and the modular design methodology that encouraged modularity and prevented a

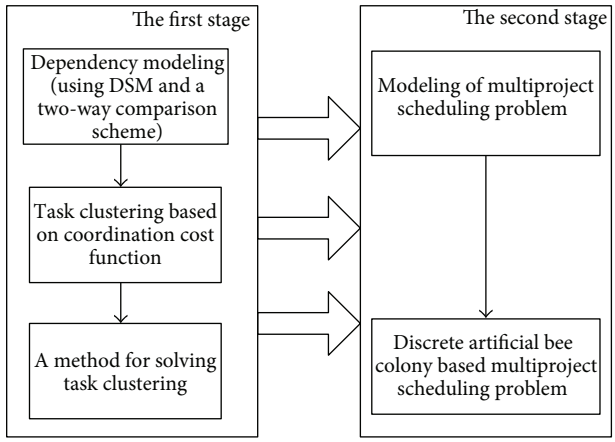


Figure 1: The flow chart of the dynamic intelligent decision approach.

cascade of product design changes due to changes in life-cycle process. Brændeland et al. [5] presented a modular approach to the modeling and analysis of risk scenarios with dependencies such as the electric power supply or telecommunications. In addition, this approach might be used to deduce the risk level of an overall system from previous risk analyses of its constituent systems. Another issue related to the paper is task scheduling, and some representative pieces of research are as follows. Some scholars [6–8] developed a branch and bound approach to solve the task scheduling problem and the differences among them lay in branch schemes as well as elimination rules and other details. Mori and Tseng [9] proposed a genetic algorithm for multimode resource-constrained project scheduling problems (RCPSPs) and compared it with a stochastic scheduling method proposed by Drexl and Gruenewald [10]. Xu et al. [11] illustrated how to combine the idea of rollout with priority rule heuristics and justification for the RCPSP and examined the resulting solution quality and computational cost. They presented empirical evidence that these procedures are competitive with the best solution procedures described in the literature. In addition, Jarboui et al. [12] designed a combinatorial particle swarm optimization (CPSO) algorithm in order to solve RCPSP. The results that have been obtained using a standard set of instances, after extensive experiments, proved to be very competitive in terms of number of problems solved to optimality. Ju and Chen [13] developed design structure matrix (DSM) and an improved artificial immune network algorithm to solve a multi-mode resource-constrained multiproject scheduling problem (MRCMPSP). Moreover, this approach was also tested on a set of random cases generated from ProGen, and the results validated the effectiveness of the proposed algorithm comparing with other famous metaheuristic ones.

However, these researches mentioned earlier only considered one aspect of complex engineering system optimization and neglected connections between task dependencies and task sequencing. Due to these reasons, in this work, we propose a dynamic intelligent decision approach to dependency modeling of project tasks in complex engineering system optimization. It takes this decision process as a two-stage decision-making problem, where in the first stage, large interdependent tasks are decomposed into smaller and manageable task groups by transforming the binary form of task relationships into the quantifiable numerical one. In this stage, three steps are needed. Firstly, DSM is adopted to model dependencies between tasks. And then, a two-way comparison scheme is used to transform the binary DSM into numerical one. Secondly, task clustering based on indices including bid value as well as coordination cost function is realized so as to discompose the large-scale project into some subprojects. Finally, a method for solving task clustering is also developed. In the second one, according to the result of task clustering developed in the first stage, the resource constrained multiproject scheduling model is built firstly, subsequently, a discrete artificial bee colony (ABC) algorithm inspired by the intelligent foraging behaving of honeybee is designed for solving this problem. The whole flow chart of this dynamic intelligent decision approach is shown in Figure 1.

This paper is organized as follows. Firstly, the approach to project task clustering in complex engineering system is proposed in Section 2. Secondly, the mathematic model of multi-project scheduling problem based on the result of task clustering analysis is developed, and a discrete artificial bee colony based multi-project scheduling problem is also proposed in Section 3. Thirdly, an illustrative case from an engineering design of a chemical processing system is given in Section 4. Finally, conclusions and possible future research extensions comprise Section 5.

2. An Approach to Task Clustering in Complex Engineering System

Optimization process in a complex system faces the difficulties not only from technology complexity but also from time pressure. Generally, to decompose large interdependent task groups into small ones by modularization is a very efficient approach to realize engineering optimization. This process usually includes three steps, that is, task dependency modeling, task clustering, and problem-solving strategies.

2.1. Task Dependency Modeling. In this paper, the task-based modeling method is adopted. Here, the design structure matrix representation of the design process is chosen for three reasons. Firstly, it overcomes the size and visual complexity of many graph-based techniques such as PERT and CPM. Secondly, matrices are easy to manipulate and store in computer. Finally, the DSM modeling has been proven by a number of researchers as a useful tool in task scheduling and management [14]. It has been widely applied to design and development [15], such as design optimization [16], tasks sequencing [17], control and monitoring of design tasks [18], and risk analysis and evolution [19].

In general, a simple DSM displays the relationships between components of a system in a compact, visual, and analytical advantageous format. Specifically, in DSM, each row and its corresponding column are identified with the

	A	B	C	D	E
Element A	A		x		x
Element B	x	B	x	x	
Element C			C	x	
Element D	x	x		D	
Element E	x			x	E

Figure 2: Sample of the DSM.

identical labels. Along each row, the marks indicate what other elements the element in that row depends on. A certain column indicates what other elements the element in that row provide to. Diagonal elements do not convey any meaning at this point. Thus, in Figure 2, element A provides something to element B, D, and E, and it also depends on something from element C and E.

In the engineering system optimization process, the typical clustering process is comprised of two sections: one is to define modules it includes, and the other is to define the relationships among different modules which will realize the whole function of a system together. Consider the implications of system performance as well as the characteristics of the DSM, there are two steps needed to model task dependencies: (1) transforming the binary format of task relationships into the quantifiable numerical ones so as to represent dependency strength among project tasks from viewpoints of structure, function, shape, and so on; (2) building up multidimension DSM model and then normalize these multi-dimension data.

2.1.1. Quantitative Approach to Dependencies among Tasks. Dependencies among tasks are determined by their function, structure, shape, and so on. However, the original DSM is populated with "ones" and "zeros" or "X" marks and empty cells. This single attribution was used to convey relationships between different elements namely, the "existence" attributes which signifies the existences or absences of a dependency between the different elements. Compared to binary DSM, numerical DSM could contain a multitude of attributes that provide more detailed information on the relationships between the different system elements. An improved description of these relationships provides a better understanding of the system and allows for the development of more complex engineering system. In this work, we introduce a two-way comparison scheme developed by Su et al. [20] to realize normalization of the binary DSM. The main criteria of this approach are to perform pairwise comparisons in one way for tasks in rows and in another way for tasks in columns to measure the dependency between different tasks. In the row-wise perspective, each task in rows will serve as a criterion to evaluate the relative connection measures for the nonzero elements in that row. It means that for each pair of tasks compared in rows, which one provides more information input than another. Similarly, in the column-wise perspective, each task in columns will serve as a criterion to evaluate the relative connection measures in that column.It also means that for every pair of tasks compared in columns, which one receives more information output than another. Moreover, in order to obtain comparison scale, we also use a single level of analytic hierarchy process (AHP). For example, if ranking of information input for tasks in row T_i is the criterion for comparison matrix, the comparison scale 3 for T_j comparing to T_k denotes that task T_j provides somewhat information input to task T_i than task T_k provides. Conversely, the comparison scale 1/9 for T_j comparing to T_k denotes that task T_j provides much less information input to task T_i than task T_k provides. This two-way comparison scheme contains four phases described as follows.

(1) Select a criterion for every pair of tasks compared and compare which one provides more information input to downstream tasks or which one receives more information output from upstream tasks.

(2) Construct a pairwise comparison matrix as follows:

$$\begin{array}{c|ccc} T_i & T_j & \cdots & T_k \\ \hline T_j & 1 & \cdots & x_{jk} \\ \vdots & \vdots & \ddots & \vdots \\ T_k & x_{kj} & \cdots & 1 \end{array} \qquad \begin{array}{c|ccc} T_l & T_j & \cdots & T_k \\ \hline T_j & 1 & \cdots & y_{jk} \\ \vdots & \vdots & \ddots & \vdots \\ T_k & y_{kj} & \cdots & 1 \end{array}, \tag{1}$$

where T_i or T_l is a criterion for the comparison matrix selected from rows or columns; $\{T_j, \ldots, T_k\} \in \{$nonzero elements of tasks in row i or column $i\}$; $x_{jk}(y_{jk})$ is the comparison scale for T_j compared with T_k when selecting T_i or T_l as a criterion, and $x_{jk}(y_{jk}) \in \{1, 3, 5, 7, 9\}$. In addition, $x_{jk}(y_{jk}) = 1/x_{kj}(1/y_{kj})$.

(3) In order to obtain the relative connection measures between the related tasks in rows and in columns, an eigen-vector is calculated for each pairwise comparison matrix. This eigen-vector denotes the ranking for each comparing task within the comparison matrix.

(4) There are totally n eigen-vectors for n rows and n eigen-vectors for n columns. Combining n rows of eigen-vectors, an $n \times n$ matrix X is formed in which each element in the vector is placed back in its location in the original binary DSM matrix. In addition, another $n \times n$ matrix Y is formed in the same way. The element in matrix X and the one in matrix Y should be combined together to measure all relationships. This may be realized by multiplying the corresponding element of X and Y matrices and taking their geometrical average. In doing so, a numerical DSM matrix which describes quantitative relationships among tasks will be obtained.

2.1.2. Normalized Treatment of Multidimension DSM Model. Numerical DSM model obtained from Section 2.1.1 is a multi-dimensional matrix. Each element in it contains multiple components which defines information relationships among tasks from different viewpoints. Generally, it is difficult to further deal with the multi-dimensional matrix. In order to solve

this problem, a dimensionality reduction method for the multidimensional matrix is proposed, and the corresponding expression is as follows:

$$A_{ij} = \sum_{k}^{N} \left(w_k \times \left(a_{ij}\big|_{\text{View}_k} \right) \right), \tag{2}$$

where $w_1 + w_2 + \cdots + w_N = 1$. A_{ij} is a new element value in DSM after dimensionality reduction. View_k ($k = 1, 2, \ldots, N$) represents analysis viewpoints such as space structure, function, shape, and physical interface. a_{ij} is a dependency strength between task i and j from viewpoint View_k.

2.2. Task Clustering Modeling. The main purpose of task clustering is to make tasks with close connection in the same block so as to form an independent project blocks, while the ones with loose connection will be in different blocks. In doing so, it is easy to realize optimization in engineering system.

At present, there are many methods used to realize task clustering such as similar coefficient method, ranking method, and path search method. However, it is not satisfactory when all these methods are applied to clustering of DSM, especially that group size is unknown in advance. Due to this reason, other researchers developed more efficient methods, where one of the typical ones is suggested by Thebeau [21]. He combined bid and evaluation to realize task clustering. Bid value measures the dependency degree between the element and the group which is proportional to dependent density and is used to determine which group the selected element belongs to. It can be defined as follows

$$B_i = \frac{\left(\sum I_i\right)^{\lambda_d}}{\left(S_i\right)^{\lambda_b}}, \tag{3}$$

where i is the number of groups. B_i is the bid value of group i for the selected element. $\sum I_i$ is the dependency gross in group i. S_i is the number that group i contains. λ_d is a weight exponent. λ_b is a bid exponent of groups.

Another index presented by Thebeau [21] is coordination cost which is the objective function of the bid and evaluation method. It comprehensively describes dependencies among tasks as well as the size of the corresponding group. It is formulated as

$$\begin{aligned} C_I &= \left[\text{DSM}\left(j,k\right) + \text{DSM}\left(k,j\right)\right] S_y^{\lambda_c}, \\ C_E &= \left[\text{DSM}\left(j,k\right) + \text{DSM}\left(k,j\right)\right] S_D^{\lambda_c}, \\ C_T &= \sum C_I + \sum C_E, \end{aligned} \tag{4}$$

where C_I is the coordination cost inside groups (for instance, tasks j and k belong to the same group). C_E is the coordination cost outside groups (for instance, tasks j and k belong to different groups, resp.). $\text{DSM}(j,k)$ and $\text{DSM}(k,j)$ denote relationships between task j and k. S_y is the number of tasks that group y contains. S_D is the total number of tasks that the whole DSM matrix contains. λ_c is a weight exponent of bid groups.

2.3. A Method for Solving Task Clustering. In general, there exist M^N possible combination schemes for the DSM clustering containing M tasks and N groups. According to the characteristics of the problem, we adopt a heuristic approach in this research, and the concrete steps are as follows.

(1) Initialize the problem, where every task is taken as an independent group.

(2) Randomly select one of tasks i and calculate each of the other groups' bid values for it. Subsequently, distribute it to the group which has the highest bid value after repeated calculation.

(3) Delete empty groups, subgroups, and same ones.

(4) Choose a new task to repeat the process mentioned above until all tasks has been traversed and the system reaches a stable state.

3. Multiproject Tasks Scheduling

In the above sections, tasks that have more information relationships will be converged to the same group. Nevertheless, how to arrange these tasks is another important problem to realize engineering system optimization. As a result, a multiproject tasks scheduling problem is proposed in this section. Generally, scheduling process involves allocation of the given resource to projects to determine the start and completion time of the detailed tasks. The allocation of scarce resources then becomes a major objective of the problem, and several compromises have to be made to solve the problem to the desired level of near-optimality [22]. Usually, this process includes two steps: first, build up the model of the multiproject scheduling problem; second, solve the problem using intelligent algorithms.

3.1. The Model of Multiproject Scheduling Problem. The problem consists of the number of projects I, and the following assumptions are taken in to consideration.

(1) Task i cannot start unless all of its predecessors have been completed.

(2) There are only renewable resources, and nonrenewable ones are not considered.

(3) Task preemption is not allowable.

(4) In a multiproject environment, the delay of any project will lead to iterations and alterations of related follow-up work, so we can assume that the objective is to minimize the completion time of all projects but not a certain project.

Based on these assumptions, the problem and the conceptual model will be described as follows. The considered problem consists of M parallel projects, each project $i = 1, \ldots, M$, being composed of J_i tasks ij, $j = 1, \ldots, J_i$. The tasks are interrelated by two kinds of constrains. One is precedence constraints, the other is resource constraints. While being processed, task ij in project i requires q_{ijr} units of renewable resource type $r = 1, \ldots, R$ during each period of its nonpreemptive duration d_{ij}. Each resource type r has a fix and

limit available amount Q_r. In addition, the optimal objective of the problem is to make the makespan of all projects shorter through finding out feasible starting time of tasks and allocation of resources. Therefore, the model of the problem can be described as follows:

$$\min \quad \left\{\sum_{i=1}^{I} \alpha_i \left(F_i - CP_i\right)\right\}, \tag{5}$$

$$\text{subject to}: \quad F_{il} \le F_{ih} - d_{ihm}, \quad \left(i = 1, 2, \ldots I, h = 1, 2, \ldots, J_i\right), \ \left(l \in \wp_h\right), \tag{6}$$

$$\sum_{ij \in BJ(t)} q_{ijr} \le Q_r, \tag{7}$$

where α_i denotes the weight of the ith project and I represents the number of projects. F_i is the completion time of project i. CP_i is the resource unconstrained critical path length of project i. $BJ(t)$ is a task set being executed at time t. $\wp_h$ is the precedence-task set of task h. The objective function (5) seeks to minimize the performance measure. Obviously, minimizing this criterion is equivalent to minimizing the mean resource-constrained completion time of the projects. A constraint (6) imposes the precedence relations between tasks, and a constraint (7) limits the resource demand imposed by the tasks being processed time t to the available capacity. It means the number of available resources will change according to the completion and starting time of tasks.

However, during the scheduling of multiprojects, some tasks will not be performed concurrently due to resource constraints and precedence ones which will also increase the difficulty to solve this problem. In this circumstance, precedence constraints among tasks should be satisfied firstly so as to determine an eligible task set. And then, resource conflicts possibly occurred in this eligible set should be identified in order to decide the task priority values, issued from the select priority rule. Therefore, combined DSM, the following will give a simplifying approach of precedence constraints and the task priority values, respectively: (1) set up DSM clustering model of multiproject based on Section 2; (2) construct an eligible task set, EJ, a task set being executed, BJ, and a task set completed, FJ, where EJ can be generated through DSM. That is to say, if the task satisfies (DSM(ij, :) = K) $\in FJ$ from a row, we can obtain $ij \in EJ$ or $ij \notin EJ$. Similarly, we can determine whether task jk belongs to EJ. In doing so, EJ can be determined; (3) identify resource conflicts between task ij and jk. If exists, perform tasks according to priority value; if not, perform them concurrently and add those tasks which are to be scheduled to BJ; (4) if tasks ij, jk have been fulfilled, update EJ, BJ, and FJ. Determine the next task set which will possibly cause resource conflicts and repeat the process till all of the tasks have been fulfilled.

3.2. Artificial Bee Colony Based Multiproject Scheduling Problem. In this section, the basic artificial bee colony algorithm based on the foraging behavior of honeybees is introduced firstly. Subsequently, the discrete artificial bee colony algorithm used for solving the multiproject scheduling problem is proposed.

3.2.1. Honeybee Modeling. In the basic ABC algorithm, the colony of artificial bees contains three groups of bees: employed bees, onlookers, and scouts. Employed bees determine a food source within the neighborhood of the food source in their memory and share their information with onlookers within the hive, while onlookers select one of the food sources according to this information. In addition, a bee carrying out random search is called a scout. In ABC algorithm, first half of the colony consists of the employed bees and the rest half includes the onlookers. There is only one employed bee corresponding to one food source. That is to say, the number of employed bees is equal to the number of food sources around the hive. The position of a food source denotes a possible solution of the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution.

The initial population of solutions is filled with SN number of randomly generated D-dimensional real-valued vectors (i.e., food sources). Each food source is generated as follows:

$$x_i^j = x_{\min}^j + \text{rand}\,(0,1)\left(x_{\max}^j - x_{\min}^j\right), \tag{8}$$

where $i = 1, 2, \ldots, \text{SN}$, $j = 1, 2, \ldots, D$. $x_{\min}^j$, and $x_{\max}^j$ are the lower and upper bounds for the dimension j, respectively. These food sources are assigned randomly to SN number of employed bees and their corresponding fitness is evaluated.

In order to produce a candidate food position from the old one, the ABC used the following equation:

$$v_i^j = x_i^j - \varphi_i^j\left(x_i^j - x_k^j\right), \tag{9}$$

where $j \in \{1, 2, \ldots, D\}$ and $k \in \{1, 2, \ldots, \text{SN}\}$ are randomly chosen indexes. Although k is determined randomly, it has to be different from i. φ_{ij} is a random number in the range $[-1, 1]$. Once V_i is obtained, it will be evaluated and compared to X_i. If the fitness of V_i is equal to or better than that of X_i, V_i will replace X_i and become a new member of the population; otherwise, X_i is retained.

After all employed bees complete their searches, onlookers evaluate the nectar information taken from all employed bees and choose one of food source sites with probabilities related to its nectar amount. In basic ABC, roulette wheel selection scheme in which each slice is proportional in size to the fitness value is employed as follows:

$$P_i = \frac{\text{fit}\left(x_i\right)}{\sum_{m=1}^{\text{SN}} \text{fit}\left(x_m\right)}, \tag{10}$$

where $\text{fit}(x_i)$ is the fitness value of solution i. Obviously, the higher the $\text{fit}(x_i)$ is, the more probability that the ith food source is selected.

If a position cannot be improved further through a predetermined number of cycles, then that food source is

assumed to be abandoned. The scouts can accidentally discover rich, entirely unknown food sources according to (8). The value of predetermined number of cycles is called "*limit*" for abandoning a food source which is an important control parameter of ABC algorithm.

There are three control parameters used in the basic ABC: the number of the food sources which is equal to the number of employed bees (SN), the value of *limit,* and the maximum cycle number (MEN).

3.2.2. Discrete Artificial Bee Colony for Solving Multiproject Scheduling Problem. The multiproject scheduling problem is a typical NP-hard problem and traditional exact algorithms may cause large computation time and is very difficult to find out the optimal solution. With the last decades, various kinds of optimization algorithms based on swarm intelligence have been designed and applied to function-optimization, task-allocation, and other problems [23]. Some of the popular approaches are ant colony optimization (ACO) [24, 25], particle swarm optimization (PSO) [26, 27], artificial immune systems (AISs) [28], and so on, where Karaboga [29] has described a bee swarm algorithm called artificial bee colony (ABC) algorithm based on the foraging behavior of honeybees. They have compared the performance of the ABC algorithm with those of other well-known modern heuristic algorithms for unconstrained optimization problems, and the results have shown that the ABC algorithm is superior to other ones. However, the basic ABC algorithm was designed to solve continuous optimization problems. In order to make it applicable for solving scheduling problem, a discrete ABC algorithm is proposed in this section, and the concrete steps are as follows.

(1) *Solution Representation*. According to the characteristics of the problem, a direct problem representation is used. Complete information of a schedule for the problem consists of a task priority rule, tasks and their corresponding modes.

(2) *Population Initialization.* To guarantee an initial population with certain quality and diversity, a portion of food sources are generated by using some priority rules while the others are produced randomly. For the project scheduling problem, the smallest slack time (SST), the earliest due date (EDD), and the smallest execution time (SET) rules are commonly adopted to yield the initial schedule. Therefore, this work applies these rules to produce three different solutions. For example, the EDD rule sorts the activities according to their ascending due dates such that $d_{x_{(j)}} \leq d_{x_{(j+1)}}$ and the same to SST and SET.

(3) *Employed Bee Phase.* The employed bees generate food sources in the neighborhood of their position in the ABC algorithm. In this work, three operations including SWAP, INSERT, and INVERSE are used to produce neighboring solutions, where the SWAP operator is defined by interchanging two activities in different positions, while the INSERT one is defined by removing an activity from its original position and inserts it into a new position, and the last one, INVERSE, generates a neighbor by inversing the sequence between two activities in different positions. Note that if the neighboring solutions do not satisfy preference constraints, the old one should be retained. Furthermore, in order to enrich searching region and diversify the population, five related approaches based on SWAP, INSERT, or INVERSE operators are adopted to produce neighboring solutions which are shown as follows:

(1) performing one SWAP operator to a sequence;

(2) performing two SWAP operators to a sequence;

(3) performing one INSERT operator to a sequence;

(4) performing two INSERT operators to a sequence;

(5) performing the INVERSE operator to a sequence.

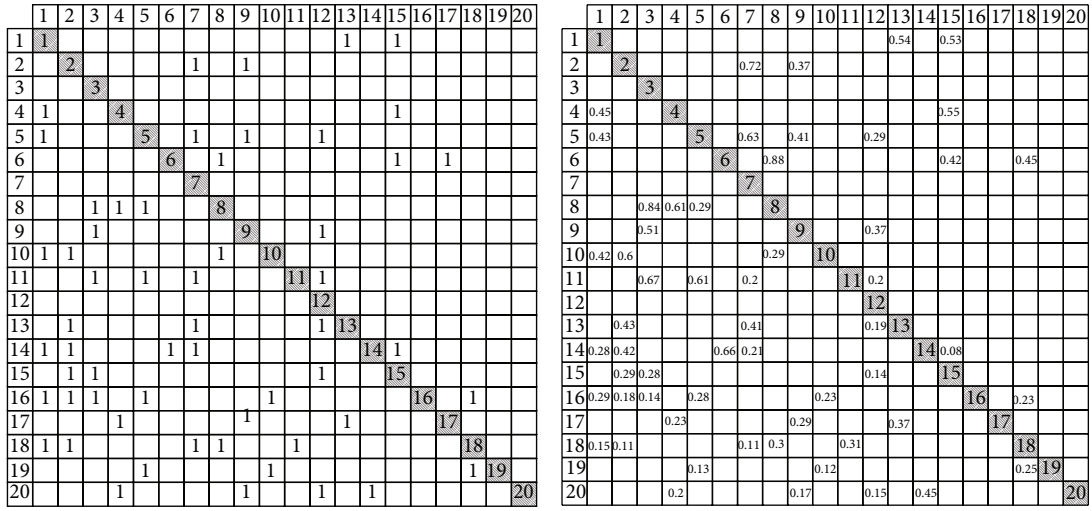
(4) *Onlooker Bee Phase.* In the basic ABC algorithm, an onlooker bee chooses a food source depending on the probability value associated with that food source. In other words, the onlooker bee chooses one of the food sources after making a comparison among the food sources around the current position which is similar to "roulette wheel selection" in genetic algorithm (GA). In this work, we also retain this approach to make the algorithm converge fast.

(5) *Scout Bee Phase.* In the basic ABC algorithm, a scout produces a food source randomly. This will decrease the search efficacy, since the best food source in the population often carried better information than others. As a result, in this paper, the scout produces a food source using several SWAP, INSERT, and INVERSE operators to the best food source in the population. In addition, to avoid the algorithm be trapped into a local optimum, this process should be repeated several times.

4. Simulation Experiments

In this section, a numerical example derived from an engineering design of a chemical processing system is utilized so as to help to understand the proposed dynamic intelligent decision approach firstly. After that, further analysis and discussions about the effect of task clustering analysis on scheduling schemes as well as the performance of ABC algorithm for solving multiproject scheduling problem are also given.

4.1. Numerical Experiment. To demonstrate the effectiveness of our proposed approach, we use 20-task binary DSM matrix derived from Su et al. [20]. In this example, an engineering design of a chemical processing system has 20 tasks, and detailed task information is listed in Table 1. The whole design process usually involves several disciplines. We aggregated these disciplines into four types: systems engineers (r_1), software engineers (r_2), hardware engineers (r_3), and supporting engineers (r_4). Hardware engineers include chemical engineers as well as electrical engineers and supporting engineers consist of the disciplines that act as a supporting role during the design process; for example, manufacturing, logistics, tests, maintainability, reliability, and so forth. Assume that the ready-time is 0. The resource availability $R(r_1, r_2, r_3, r_4)$ is set to (7, 6, 7, 5).



(a) 0-1DSM (b) Numerical DSM

Figure 3: DSM model of design process.

Table 1: Task information for an engineering design of a chemical processing system.

Number	Description of tasks	Resource				Duration (day)	Predecessor
		r_1	r_2	r_3	r_4		
1	Operating structure design	3	0	0	0	6	13, 15
2	Vessel design	5	3	3	2	12	7, 9
3	Plant layout/general arrangement	4	2	4	2	5	
4	Shipping design	4	2	2	2	8	1, 15
5	Structure lifting design	5	0	0	0	4	1, 7, 9, 12
6	Pressure drop analysis	4	2	2	2	4	8, 15, 17
7	Process engineering	4	3	3	2	3	
8	Structural documentation	0	4	2	0	3	3, 4, 5
9	Size valves	2	1	5	3	4	3, 12
10	Wind load design	4	0	0	0	5	1, 2, 8
11	Seismic design	5	0	0	0	9	3, 5, 7, 12
12	Piping design	3	0	0	1	4	
13	Process and instrumentation diagram	3	2	0	1	2	2, 7, 12
14	Equipment support	3	0	3	2	2	1, 2, 6, 7, 15
15	Pipe flexibility analysis	4	2	2	2	2	2, 3, 12
16	Design documentation	5	2	2	0	4	1, 2, 3, 5, 10, 18
17	Foundation load design	5	2	4	2	5	4, 9, 13
18	Insulation structural design	2	2	3	3	5	1, 2, 7, 8, 11
19	Structural bill of material (BOM)	3	2	3	4	6	5, 10, 18
20	Assembly design	4	0	0	0	7	4, 9, 12, 14

In the first stage, according to dependency modelling technology mentioned in Section 2, the DSM model is set up, shown in Figure 3(a), where the empty elements represent no relationships between two tasks and number "1" represents input or output information among tasks. For example, task 1 requires information from tasks 13 and 15 when it executes. Additionally, task 1 must provide information to tasks 4, 5, 10, 14, 16, and 18, otherwise they can not be start. Nevertheless, Figure 3(a) only denotes the 'existence' attributes of a dependency between the different tasks. In order to further reveal their matrix structure, it is necessary to quantify dependencies among tasks.

Subsequently, using a two-way comparison scheme, we can transform the binary DSM into the numerical one. Here, two criterions named task evolution (represented by EC) and task sensitivity (represented by SC) are adopted to perform pairwise comparisons, where the former means the information transfer rate to the element j from i, and the latter means the effect degree of information change to the element i from element j. For example, if the criterion for the

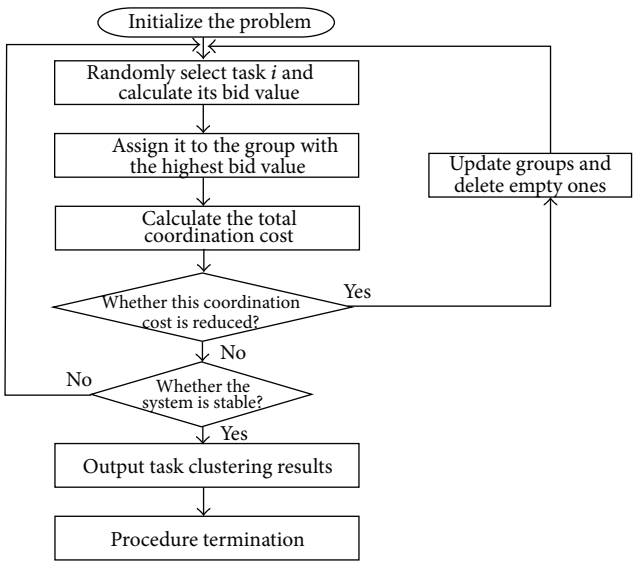


Figure 4: A flow chart of task clustering algorithm.

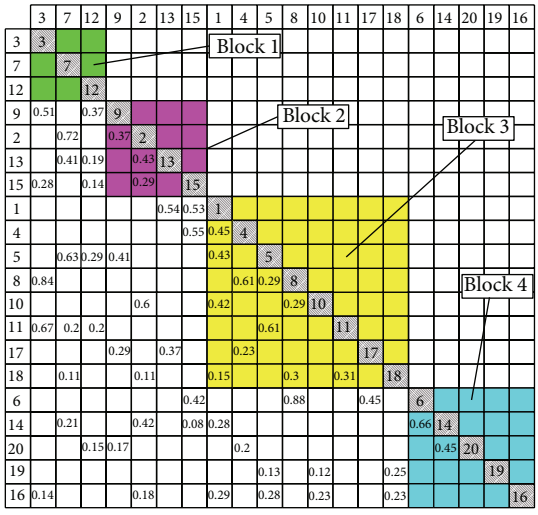


Figure 5: Task clustering result of design process.

comparison matrix is task evolution, the comparison scale 9 for task 4 compared to task 18 indicates task 4 is evolving much faster than task 18 when they receive information from task 1. In addition, the comparison scale 1/3 for task 5 comparing to task 4 represents the evolution degree of task 5 is somewhat slower than that of task 4 when they receive information from task 1. In doing so, we can obtain the comparison matrix of evolution shown as follows:

$$\begin{array}{c|cc} \mathrm{EC}(T_1) & T_{13} & T_{15} \\ \hline T_{13} & 1 & 3 \\ T_{15} & \frac{1}{3} & 1 \end{array} \qquad \begin{bmatrix} T_{13} \\ T_{15} \end{bmatrix} = \begin{bmatrix} 0.949 \\ 0.316 \end{bmatrix} \qquad \lambda_{\max} = 2. \tag{11a}$$

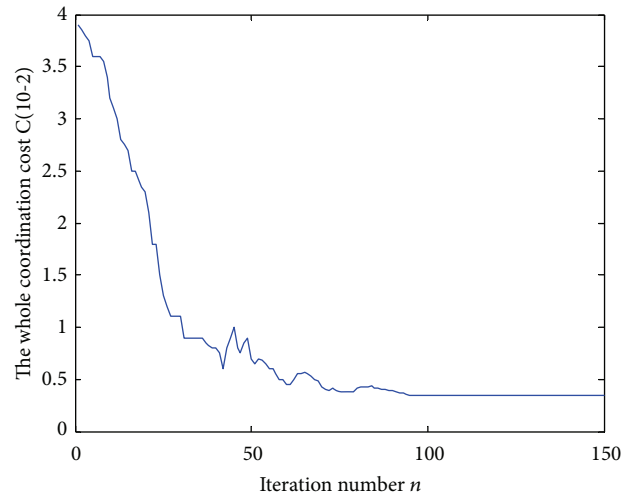


Figure 6: The change curve of coordination cost function.

In the same way, if the criterion for the comparison matrix is task sensitivity, we can get the comparison matrix of sensitivity shown as follows:

$$\begin{array}{c|cccccc} \mathrm{SC}(T_1) & T_4 & T_5 & T_{10} & T_{14} & T_{16} & T_{18} \\ \hline T_4 & 1 & 3 & 5 & 5 & 7 & 9 \\ T_5 & \frac{1}{3} & 1 & 3 & 3 & 5 & 7 \\ T_{10} & \frac{1}{5} & \frac{1}{3} & 1 & 1 & 3 & 5 \\ T_{14} & \frac{1}{5} & \frac{1}{3} & 1 & 1 & 3 & 3 \\ T_{16} & \frac{1}{7} & \frac{1}{5} & \frac{1}{3} & \frac{1}{3} & 1 & 3 \\ T_{18} & \frac{1}{9} & \frac{1}{7} & \frac{1}{5} & \frac{1}{3} & \frac{1}{3} & 1 \end{array} \qquad \begin{bmatrix} T_4 \\ T_5 \\ T_{10} \\ T_{14} \\ T_{16} \\ T_{18} \end{bmatrix} = \begin{bmatrix} 0.847 \\ 0.442 \\ 0.204 \\ 0.186 \\ 0.097 \\ 0.056 \end{bmatrix} \qquad \lambda_{\max} = 6.25. \tag{11b}$$

The detailed computation process including every pair of tasks compared in DSM are not given due to the length limitation of this paper, and the final computation result is shown in Figure 3(b) after normalized treatment mentioned in Section 2.1.2.

After dependency modeling based on DSM, a clustering algorithm based on coordination cost mentioned in Section 2.2 is used. Firstly, initialize the problem, where every task is taken as an independent group. Then, select task i randomly and calculate each group's bid value for it, respectively, and assign it to the group with the highest bid value through loop computation until the total coordination cost is no longer reduced. Finally, update groups and delete empty ones. When all the tasks have been traversed and the system arrives to a stable state, this procedure is over. The detailed task clustering algorithm procedure is shown in Figure 4.

The final task clustering result is displayed in Figure 5, and the corresponding changing curve of the coordination cost with iteration number is shown in Figure 6. We can see

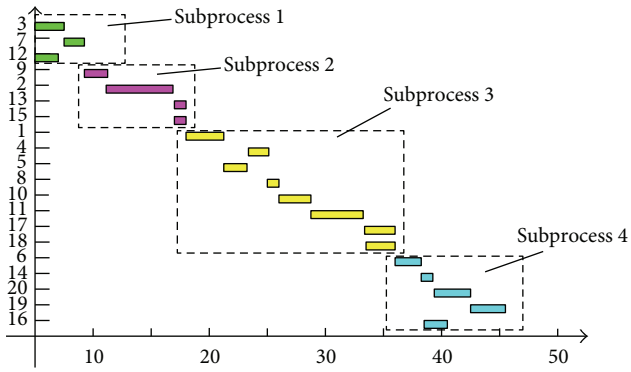


Figure 7: Gantt chart of task planning.

Table 2: Problem instances generated by ProGen for the testing problem subset.

Problem subset	NOI	J_i	R
MP30	20	30	4
MP60	20	60	4
MP90	20	90	4
MP120	20	120	4
MP150	20	150	4
MP180	20	180	4
MP210	20	210	4

NOI: no. of instances; J_i: no. of tasks of a project; R: resources each task can use.

from Figure 5, that the whole design process can be divided into four blocks, where block 1 contains 3 tasks such as 3, 7, and 12, and all of them can be executed without input information from others; block 2 consists of tasks 2, 9, 13, and 15, and they must receive information from block 1; block 3 includes task 1, 4, 5, 8, 10, 11, 17, and 18, and all the tasks must depend on information from blocks 1 and 2; block 4 is comprised of tasks 6, 14, 16, 19, and 20, where tasks 16, 19, and 20 only receive input information from other tasks but provide nothing to others. Furthermore, the cost curve in Figure 6 reveals that the coordination cost converges to a minimum which indicates the whole system has the minimum complexity. Moreover, according to the clustering analysis, each block shown in Figure 5 can be defined as an independent subproject. It means that the original large-scale project is decomposed into four simple, manageable, and small subprojects. The reason that the subprojects are defined is because tasks belonging to the same block have higher correlation degrees, but ones belonging to different blocks have lower correlation degrees. In doing so, a suitable decomposition scheme for a large-scale project is obtained so as to reduce the difficulty of task planning problem using the ABC algorithm below in the second stage.

In the second stage, according to the clustering result, the task planning problem can be transformed into multiproject scheduling one. Considering the objective function of the problem is to minimize the delay time of all projects, we must define the shortest makespans of all projects in advance, and critical path method (CPM) is used to obtain their values of the shortest makespans (i.e., 5, 18, 28, and 13). Moreover, so as to simplify the mathematic model, we set all projects that have the same weight coefficients. That is to say $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$. Subsequently, a discrete ABC algorithm is used to solve this problem mentioned in Section 3.2, and the related parameters are set as follows: SN = 10, limit = 20, MEN = 500. The planning result is shown in Figure 7, and the delay time of the whole process is 41 (obtained from 3 + 8 + 18 + 12).

We can see from Figure 7, tasks in the same project will be executed more tightly and hardly be affected by other tasks from different projects. This is because after clustering operation, tasks with tight dependencies belong to the same project. In addition, the task clustering analysis will help to reduce the search space and further improve the efficiency of solution in the second stage.

4.2. Analysis and Discussions. In this section, the effect of task clustering analysis on scheduling schemes as well as the performance of ABC algorithm for solving multiproject scheduling problem is discussed, and extensive experiments to analyze theses have been illustrated. The project test problems are generated by the project generator ProGen developed by Kolisch et al. [30] and the number of tasks in a project is 30, 60, 90, 120, 150, 180, and 210, respectively. For each problem type, we generated 20 instances. Each task can use up to four resources. To generate the test instances the single project problems were randomly selected network complexity and resource factor, where network complexity means that the network depends on interdependent relations among different tasks. Table 2 shows the related information about projects used for the problem.

4.2.1. The Effect of Task Clustering Analysis on Scheduling Schemes. In order to analyze the effect of task clustering analysis on scheduling schemes, two indexes are introduced: one is the average deviation of project delay time, and the other is the computation time, where the former is used to measure robustness of search algorithms and the latter to compare the complexity of computation. Table 3 shows results obtained on the problem subsets. We can see from it that after task clustering operation, the computation complexity of scheduling problems reduces obviously ($T_c \ll T_s$). The reason is that a large-scale project is decomposed into several simple, manageable, and small sub-projects using a task clustering approach proposed in this work. In addition, after task clustering operation, average deviation of project delay time is also reduced when the project scale is larger. It means that the task clustering analysis will help to decrease the search space and further improve efficiency of algorithms in the second decision stage.

4.2.2. Performance of ABC Algorithm for Solving Multiproject Scheduling Problem. According to task clustering result obtained in the first decision stage, the ABC algorithm is used to solve multiproject scheduling problem in the second one. In order to get the most out of the ABC algorithm, parameter setting mentioned in Section 3.2 was implemented on a

Table 3: The effect of task clustering analysis on scheduling schemes.

Problem subset	NOI	Average deviation of project delay time		Computation time (s)	
		TD_c	TD_s	T_c	T_s
MP30	20	14.3	14.5	0.10	0.22
MP60	20	21.3	23.4	0.53	0.88
MP90	20	29.7	36.9	2.14	3.55
MP120	20	40.2	53.4	3.21	5.71
MP150	20	58.5	81.3	4.35	8.77
MP180	20	82.6	114.8	5.94	12.45
MP210	20	108.7	152.6	7.12	19.88

TD_c: average deviation of project delay time through task clustering; TD_s: average deviation of project delay time without task clustering operation; T_c: computation time through task clustering; T_s: computation time without task clustering operation.

Table 4: Comparison of performance obtained by different algorithms.

Problem subset	Schedules	ABC			SA			ACO			AIS		
		APD	LTM	CPU_t	APD	LTM	CPU_t	APD	LTM	CPU_t	APD	LTM	CPU_t
MP30	1000	21.1	98.6	0.09	**20.1**	**92.5**	**0.06**	22.4	93.5	0.07	25.6	101.4	0.07
	5000	19.3	95.3	0.23	19.3	91.3	0.19	**19.2**	**89.2**	**0.18**	24.3	97.5	0.21
	10000	19.1	89.9	0.67	18.6	91.3	**0.55**	**19**	**89**	0.61	22.1	95.1	0.61
MP60	1000	42.1	130.7	0.61	**41.9**	147.4	**0.41**	42.1	130.4	0.98	**41.9**	**129.8**	0.62
	5000	**39.2**	125.1	3.9	41.5	140.3	**1.79**	42.0	**124.7**	3.7	41.9	127.6	3.9
	10000	**34.8**	**122.3**	7.9	41.5	138.6	**4.5**	41.8	124.5	8.8	41.8	127.5	9.5
MP90	1000	68.5	254.7	2.3	87.4	**253.3**	0.9	**67.3**	287.9	1.2	67	296	**0.7**
	5000	**65.0**	**250.9**	9.5	81.3	252.4	**2.76**	65.1	277.6	5.1	66.9	275	4.8
	10000	**62.4**	**237.6**	23.1	75.6	252	**6.1**	64.8	274.1	10.3	66.9	251.4	11.3
MP120	1000	87.1	345.1	3.1	121.6	345	**1.2**	150.4	387.9	1.5	**86.9**	**340.5**	1.34
	5000	**84.4**	**323.1**	11.6	119.3	340.1	**5.3**	144.3	366.4	6.1	86.5	338.1	5.77
	10000	**81.2**	**317.8**	32.2	115.6	335.7	**10.5**	140.2	354.2	12.9	86	335.2	14.8
MP150	1000	116.9	**410.1**	4.3	125.4	419.9	**2.1**	130.7	443.5	**2.1**	**110.4**	415.3	2.8
	5000	110.4	**397.3**	12.9	124.1	415.3	6.9	127.6	430.1	**6.8**	**109.7**	405.5	8.3
	10000	**105.9**	**392.1**	40.5	122.3	413.5	**13.7**	125.5	427.7	14.9	107.4	401.1	19.3
MP180	1000	147.1	**508.9**	5.9	154.7	520.1	3.2	150.9	519.9	**2.9**	**140.1**	519.4	3.6
	5000	**135.4**	**501.4**	15.9	149.1	515.4	**7.8**	144.3	514.3	8.2	138.4	512.2	10.7
	10000	**129.1**	**492.1**	51.1	147.1	512.5	**19.1**	141.8	510.5	21.7	134.9	508.5	29.1
MP210	1000	**180.9**	**619.5**	7.1	191.4	653.1	**4.1**	199.3	647.9	4.7	189.7	625.1	6.1
	5000	**172.5**	**611.2**	19.3	186.6	651.4	11.2	195.1	641.6	**10.7**	186.9	624.3	15.7
	10000	**167.1**	**606.7**	67.8	185.1	650.1	**23.5**	189.8	639.2	24.1	185.1	621.4	58.1

set of produced multiproject problems after task clustering operation. Table 4 shows results obtained by the various algorithms on the problem subsets. The proposed algorithm is compared with other approaches, including simulated annealing (SA), ant colony optimization (ACO), and artificial immune system (AIS), in view of the average project delay (APD), lower total makespan (LTM) of multiproject, and processing time of CPU (CPU_t). In this table, the first column indicates the problem subset. The second column shows the number of schedules which is used as the stopping criterion. The third to the fifth columns represent the averages of APD, LTM, and CPU_t from various algorithms. From Table 4, it is seen that out of 7 subsets, the ABC algorithm is superior to others when the problem scale is larger. The average project delays and lower total makespan obtained by the four approaches for all the problems are also compared which show that our approach is obviously better than AIS and SA for all the problems but a little worse than ACO when the problem scale is small. In addition, when the number of schedules increases, our approach still searches for the optimum schedule scheme but others are hardly influenced by it. This is because our approach introduces operations including scout bee searching process which is helpful to maintain the diversity of individuals. However, our algorithm may spend more computation time. There are two reasons causing this result: one is that five related approaches based on SWAP, INSERT, or INVERSE operators mentioned in Section 3.2.2 may occupy more time in order to find out the

local optimum; the other is that the new individuals generated by scout bees are introduced to replace the worse ones so as to open up a new searching field.

5. Conclusions

In this study, we have presented a two-stage intelligent decision approach to dependency modelling of project tasks in complex engineering system optimization. Both the dependencies among tasks and sequence of project tasks are clearly identified. In the first stage, in order to optimize the complex engineering system, we further investigate task evolution degree and sensitivity degree based on a two-way comparison scheme. In addition, we have also introduced DSM model that systematically quantifies the strength between related tasks and decomposes large interdependent task groups into smaller and manageable sub-projects. Subsequently, according to decomposition results obtained from the first stage, the discrete artificial bee colony (ABC) algorithm is developed for project task planning in the second stage, where DSM has also adopted to simplify the mathematic model of the task planning. In doing so, a better sequence of project tasks is expected because of less communication links and simpler information flows among tasks in different sub-projects. The major contributions of the research are as follows: (a) the integrated model of complex engineering system optimization is developed. This has led to a visible dependency structure and a simpler task sequence through systematic procedures; (b) the task dependencies in the complex system are clearly identified by quantitative measures. It is very important to offer a great research potential in understanding and analyzing the implications of connectivity on different aspects of complex system performance; (c) the decomposition of a large number of task groups and the planning of projects tasks lay a sound foundation for engineering optimization. In addition, each sub-project obtained from decomposition has been limited to a manageable size so that tasks in the same sub-project have more communication links. The results will serve as the task requirements for efficiently scheduling project tasks.

In this work, we focus on dependency modeling of project tasks in complex engineering system optimization. For sound project task management, it is necessary to consider influences from external factors. Due to this reason, our future research extension will focus on analysing the effects from customers' requirements on engineering optimization process. After that, how to apply this two-stage intelligent decision approach to other optimization fields also needs further study.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant nos. 71071141, 71001088, and 71171089), the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant nos. 200804870070 and 20103326120001), and Humanity and Sociology Foundation of Ministry of Education of China (Grant no. 11YJC630019).

References

[1] S. J. Chen and L. Lin, "A project task coordination model for team organization in concurrent engineering," *Concurrent Engineering Research and Applications*, vol. 10, no. 3, pp. 187–202, 2002.

[2] W. Guo, J. Cha, Y. Fang, and T. C. Woo, "Concurrent design: an algebraic perspective," *International Journal of Information Technology*, vol. 1, no. 1, pp. 17–32, 1995.

[3] M. M. Tseng and J. Jiao, "A module identification approach to the electrical design of electronic products by clustering analysis of the design matrix," *Computers and Industrial Engineering*, vol. 33, no. 1-2, pp. 229–233, 1997.

[4] J. K. Gershenson, G. J. Prasad, and S. Allamneni, "Modular product design: a life-cycle view," *Society for Design and Process Science*, vol. 3, no. 4, pp. 13–26, 1999.

[5] G. Brændeland, A. Refsdal, and K. Stølen, "Modular analysis and modelling of risk scenarios with dependencies," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1995–2013, 2010.

[6] J. P. Stinson, E. W. Davis, and B. M. Khumawala, "Multiple Resource-constrained scheduling using branch and bound," *AIIE Trans*, vol. 10, no. 3, pp. 252–259, 1978.

[7] N. Christofides, R. Álvarez-Valdés, and J. M. Tamarit, "Project scheduling with resource constraints: a branch and bound approach," *European Journal of Operational Research*, vol. 29, no. 3, pp. 262–273, 1987.

[8] E. L. Demeulemeester and W. S. Herroelen, "A branch-and-bound procedure for the generalized resource-constrained project scheduling problem," *Management Science*, vol. 38, no. 12, pp. 1803–1818, 1992.

[9] M. Mori and C. C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem," *European Journal of Operational Research*, vol. 100, no. 1, pp. 134–141, 1997.

[10] A. Drexl and J. Gruenewald, "Nonpreemptive multi-mode resource-constrained project scheduling," *IIE Transactions*, vol. 25, no. 5, pp. 74–81, 1993.

[11] N. Xu, S. A. McKee, L. K. Nozick, and R. Ufomata, "Augmenting priority rule heuristics with justification and rollout to solve the resource-constrained project scheduling problem," *Computers and Operations Research*, vol. 35, no. 10, pp. 3284–3297, 2008.

[12] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008.

[13] C. Ju and T. Chen, "Simplifying multiproject scheduling problem based on design structure matrix and its solution by an improved aiNet algorithm," *Discrete Dynamics in Nature and Society*, vol. 2012, Article ID 713740, 22 pages, 2012.

[14] C. H. Chen, S. F. Ling, and W. Chen, "Project scheduling for collaborative product development using DSM," *International Journal of Project Management*, vol. 21, no. 4, pp. 291–299, 2003.

[15] R. Xiao and T. Chen, "Research on design structure matrix and its applications in product development and innovation: an overview," *International Journal of Computer Applications in Technology*, vol. 37, no. 3-4, pp. 218–229, 2010.

[16] H. Q. Wei, "Concurrent design process analysis and optimization for aluminum profile extrusion product development,"

International Journal of Advanced Manufacturing Technology, vol. 33, no. 7-8, pp. 652–661, 2007.

[17] R. Xiao, T. Chen, and W. Chen, "A new approach to solving coupled task sets based on resource balance strategy in product development," *International Journal of Materials and Product Technology*, vol. 39, no. 3-4, pp. 251–270, 2010.

[18] R. Xiao, T. Chen, and C. Ju, "Research on product development iterations based on feedback control theory in a dynamic environment," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 5(B), pp. 2669–2688, 2011.

[19] M. Danilovic and B. Sandkull, "The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations," *International Journal of Project Management*, vol. 23, no. 3, pp. 193–203, 2005.

[20] J. C. Y. Su, S. J. Chen, and L. Lin, "A structured approach to measuring functional dependency and sequencing of coupled tasks in engineering design," *Computers and Industrial Engineering*, vol. 45, no. 1, pp. 195–214, 2003.

[21] R. Thebeau, *Knowledge Management of System Interfaces and Interactions for Product Development Process*, Massachusetts Institute of Technology, Cambridge, Mass, USA, 2001.

[22] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende, "A genetic algorithm for the resource constrained multi-project scheduling problem," *European Journal of Operational Research*, vol. 189, no. 3, pp. 1171–1190, 2008.

[23] R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey," *International Journal of Bio-Inspired Computation*, vol. 3, no. 1, pp. 1–16, 2011.

[24] Z. Zhang and Z. Feng, "Two-stage updating pheromone for invariant ant colony optimization algorithm," *Expert Systems with Applications*, vol. 39, no. 1, pp. 706–712, 2012.

[25] R. B. Xiao and T. G. Chen, "Relations of swarm intelligence and artificial immune system," *International Journal of Bio-Inspired Computation*, vol. 5, no. 1, pp. 35–51, 2013.

[26] R. B. Xiao, W. M. Chen, and T. G. Chen, "Modeling of ant colony's labor division for the multi-project scheduling problem and its solution by PSO," *Journal of Computational and Theoretical Nanoscience*, vol. 9, no. 2, pp. 223–232, 2012.

[27] J. Upendar, C. P. Gupta, and G. K. Singh, "Modified PSO and wavelet transform-based fault classification on transmission systems," *International Journal of Bio-Inspired Computation*, vol. 2, no. 6, pp. 395–403, 2010.

[28] Y. Liu and R. Xiao, "Optimal synthesis of mechanisms for path generation using refined numerical representation based model and AIS based searching method," *Journal of Mechanical Design*, vol. 127, no. 4, pp. 688–691, 2005.

[29] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Computer Engineering Department, Erciyes University, Kayseri, Turkey, 2005.

[30] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource constrained project scheduling problems," *Management Science*, vol. 41, no. 10, pp. 1693–1703, 1995.