# A reacting approach for design task allocating under resource unavailability

**Meng Wei[1], Yu Yang[1] and Qiucheng Li[2]**

[1]Chongqing University, 174 Shazheng st, Shapingba District, Chongqing P.R. China, 400044

[2]University of Sheffield, Velocity tower APT 146, St Marvs Gate Sheffield, S1 4LS

E-mail: [a]weimeng_cqu@163.com, [b]yuyang@cqu.edu.cn, [c]leeqiucheng@me.com

**Abstract.** Resource unavailability (RU) is inevitable in complex product design (CPD) process, therefore, it is quite difficult to achieve an optimal solution. Based on the controllable characteristic of execution time, a multi-objective model with compressing execution time strategy (CETS) is considered in CPD. There are two stages in this model, in the first stage, we obtain an original allocation with the goal of minimizing completion time. In the second stage, CETS is utilized in response to RU considering the trade-off between compensatory cost and stability. An adaptive multi-objective hybrid genetic algorithm and tabu search (AMOGATS) is developed to solve the first mathematical step, CETS is performed by a basic genetic algorithm (GA). The computational results verify the superiority of hybrid genetic algorithm and proposed strategy.

## 1. Introduction

The guarantee that all the design phrases and sub-processes are under controlled is a key sticking point of efficiency study in CPD process. There are two directions on task allocation, one is allocation optimization model, the other one is algorithm. Jimenez M I et al. [1] present a scheduling model of radar design tasks to achieve both simple design and good performance. CPD process is in a highly uncertain environment due to resource unavailability (RU). RU can deteriorate the system stability and efficiency. Castro, P. M et al. 2] have researched on multi-objective task scheduling of large-complex equipment design, and a continuous time representation method based on uniform time grids is proposed. Ouelhadj D et al. [3] compare the technology of completely reactive scheduling, predictive-reactive scheduling, and robust pro-active scheduling. Regarding RU, there are two main reallocating strategies : allocation repair (AR), and complete reallocating (CR) [4]. Utility and stability measures are used to assess various performance of AR and CR strategies.

Nevertheless, Most of the literature are carried out under the assumption that execution time is fixed. In fact, task execution time is variable and flexible in actual CPD process. When uncertainty emerges, we can compress the time that is allocated to a specific task in order to catch up the original allocation at one point. Sinan Gurel [5] have made a successful anticipative job sequence decisions based on the flexibility of jobs. It is very helpful to reduce rescheduling costs.

Therefore, in this paper, we propose a new model to respond to RU. There are two stages in this model, in the first stage, we obtain the original allocation the goal of minimizing completion time, AMOGATS is used to solve this problem. In the second stage, we propose the compressing execution time strategy (CETS) with variable execution time to respond to RU by considering stability and compensatory cost, a basic genetic algorithm is developed in the second stage.

## 2. Model construction

### *2.1 The first stage: task allocating*

Suppose that design process is decomposed into $n$ tasks $\{T_1, T_2, ..., T_n\}$ and contain $m$ VDUs $\{U_1, U_2, ..., U_m\}$ (Virtual Design Unit, combination of design resource monomer [6]). Task allocation is to assign tasks to VDUs optimally.

**Definition 1.** Design tasks can be decomposed into a series of determined design sequences $\{T_{ij}, j=1, 2, ..., N_i\}$, $N_i$ is the total number of sequences in each design task. $T_{ij}U$ is the set of VDUs which can execute the sequence $T_{ij}$.

**Definition 2.**

$$\begin{cases} X_{ijk} = 1, T_{ij} \text{ is executed by VDU } k \\ X_{ijk} = 0, \text{ others} \end{cases}$$

$$\begin{cases} Y_{ijpqk} = 1, T_{ij} \text{ and } T_{pq} \text{ is executed by VDU } k\text{, and } T_{ij} \text{ has the priority} \\ Y_{ijpqk} = 0, \text{ others} \end{cases}$$

**Definition 3.**

To simplify the study, we make some hypothesizes. Formula (1) guarantees that VDU $k$ can execute the follow-up sequence only after complete the front task. Constraint (2) denotes the propriety of sequence in one task, $T_{ij}$ starts after consummation of $T_{i(j-1)}$. Formula (3) is execution time constraint, (4) denotes $T_{ij}$ can only choose the execution unit from $T_{ij}U$. The completion time constraint is described in the formulation (5) and (6). $S_{ijk}$, $E_{ijk}$, $C_{ijk}$ presents the starting time, execution time, and completion time respectively.

$$S_{pqk} - S_{ijk} - E_{ijk} \geq 0, Y_{ijpqk} = 1, X_{ijk} = 1, X_{pqk} = 1 \quad (1)$$

$$S_{ijh} - S_{i(j-1)k} - E_{i(j-1)k} \geq 0 \quad (2)$$

$$C_{ijk} \geq E_{ijk} \quad (3)$$

$$\sum_k X_{ijk} = 1, k \in T_{ij}U \quad (4)$$

$$C_{ijk} = \max\left\{C_{i(j-1)k}, S_{ijk}\right\} + E_{ijk} \quad (5)$$

$$C_{i1k} = S_{i1k} + E_{ijk} \quad (6)$$

The task allocation of CPD is to assign the tasks to VDUs with certain orders based on time optimization. The original objective function can be described as

$$f_1 = \min\left\{\max\left(C_{ijk} \middle| i = 1, 2, ..., n\right)\right\} \quad (7)$$

### *2.2 The second stage: task reallocating*

When resources are temporarily unavailable, the original scheme is interfered, it seeks response quickly and effectively to ensure system stability with lowest cost. CETS is developed to reduce the influence under the premise of original objective. In CETS, we refer to the Affected Operations Rescheduling (AOR) in job shop scheduling [7]. The reallocated tasks await need to include original allocated tasks and affected tasks directly and indirectly based on the principle of feasibility and optimality. $E_{ijk}$ can be compressed with non-linear growth of compensatory cost $C^a_{ijk}$. $C^a_{ijk}$ is composited by design cost $C_d$ and compression cost $C_c$. $C_c$ is decided by compression amount $y_{ijk}$. $y_{ijk}$ contains optimal compressibility $y^*_{ijk}$ and secondary compressibility $y^2_{ijk}$. $C^a_{ijk}$ can be expressed as a function of $y \geq 0$ as

$$C^a_{ijk} = C_d + C_c \quad (8)$$

$$C_d = X_{ijk} \times R_{ik} \times \left(C_{ik} + C_{ik\to(i+1)k}\right) \quad (9)$$

$$C_c = f(y) = hy_{ijk}^{(a/b)} \quad (10)$$

$$\text{s.t.}\, C_{ik} = \frac{\sum_{z=1}^{z}(n_{ez}\times c_{ez}) + \sum_{s=1}^{s}(n_{ms}\times c_{ms}) + \sum_{t=1}^{t}(n_{tt}\times c_{tt}) + c_{aux}}{L_{ik}}$$

$$\sum_{k}^{m}\left(E_{ijk} - y_{ijk}\right) = D \le \sum_{k}^{m} E_{ijk}$$

$$0 \le y_{ijk} \ge u_{ijk} \le E_{ijk}$$

Where $a \ge b > 0$, $h > 0$, $D$ is the available time of resource. $u_{ijk}$ is the upper of compression amount. Details about $C_d$ refer to [8].

We consider the trade-off between stability (STB) and compensatory cost (CC) in reallocating, the objective function can be described as

$$f_2 = STB = \sum_{i=1}^{n}\sum_{j=1}^{N_i}\left|C_{ijk} - C_{ijk}^{r}\right| \tag{11}$$

$$f_3 = CC = \sum_{i}\sum_{j}\sum_{k} C_{ijk}^{a} \tag{12}$$

Where $C_{ijk}$, $C_{ijk}^{r}$ presents the completion time of original allocation and reallocation respectively. Therefore the objective function of reallocating scheme can be formulated as

$$\min f = \alpha * f_1 + \beta * f_2 + \gamma * f_3 \tag{13}$$

Where $\alpha$, $\beta$, $\gamma$ represents the weight of these objectives.

## 3. Algorithm

### *3.1 Algorithm 1: AMOGATS*

We propose AMOGATS to solve the multi-objective optimization problem. We introduce the unique memory function of TS into the evolutionary search process of GA to construct a new crossover operator TSR. To improve the climbing ability of GA, we treat TS as GA mutation operator TSM, meanwhile introduces the self-adaptive idea to shorten the process of population evolution.

Step 0: Parameter setting. The maximum iteration $N_{gen}$, population size $N_{pop}$, etc.

Step 1: Initialization. Let the number of allocations $n = 0$, $t_{s,n} = 0$ (the $n$th starting time), $W$ is the maximum number of tasks, initialize task states $S_1$, $S_2$, $S_3$ and $S_4$.

Step 2: Perform the following operations in task window.

Step 2.1: Let evolution generation $t = 0$, $S_{ijk} = t_{s,n}$.

Step 2.2: The execution time of the first design sequence $T_{i1}$ of each task can be calculated according to formula (6), if not the $T_{i1}$, make the start time of $T_{i(j+1)}$ equal to the completing time of the $T_{ij}$, then calculate the execution time of $j+1$th design sequence. After obtain the execution time of all the design sequence, we can acquire the fitness value of the design task according to formula (13).

Step 3: Crossover.

Step 3.1: The restructuring randomly generates the number $r_i$ ([0, 1], $i = 1,2,...,N_{pop}$). If $n < p_c$, the $i$th chromosome in the mating pool serves as a crossed parent, produce $N_{pop}$ parent chromosome, $p_c^*$ as the mean value.

Step 3.2: Cross each pair of parents to generate two offspring.

Step 3.3: Adopt the TSR to restructure the offspring obtained through crossing.

The self-adaptive function of crossover probability $p_c$ is defined as $P_c = \begin{cases} P_{c1} - \dfrac{(P_{c1} - P_{c2})(f - f_{avg})}{(f_{max} - f_{avg})}, f \ge f_{avg} \\ P_{c1}, f < f_{avg} \end{cases}$.

Step 4: Mutation. The restructuring randomly generates the number $r_i$ ([0, 1], $i = 1,2,...,N_{pop}$). If $n < p_m$, the $i$th chromosome in the mating pool serves as a crossed parent, produce $N_{pop}$ parent chromosome, $p_m^*$ as the mean value.

The self-adaptive function of mutation probability $P_m$ is defined as

$$P_m = \begin{cases} P_{m1} - \dfrac{(P_{m1} - P_{m2})(f_{max} - f^*)}{(f_{max} - f_{avg})}, f^* \geq f_{avg} \\ P_{m1}, f^* < f_{avg} \end{cases}.$$

Where $f_{max}$ denotes the maximum fitness value, $f_{avg}$ denotes the average, $f$ is the larger fitness of the two individuals, $f^*$ is the fitness value of who is to be mutated.

Step 5: Stopping rule. If the maximal generation is reached, stop and output $F_i$ and allocating scheme. Otherwise, perform the next iteration.

### *3.2 Algorithm 2: CETS*

The function $f(y)$ is increasing and convex. By solving the convex programming function, we can get the optimal compression $y^*_{ijk}$. $f''(y_{ijk}^*)$ is the second derivative of compensatory cost function in the optimal compression $y^*_{ijk}$, and the average slope of compensatory cost function $\Delta$ reflects the rate of cost change when design sequence absorb disturbance. The secondary compressibility $y^2_{ijk}$ reflects the ability of secondary absorption. If $y^2_{ijk} = 0$, it denotes that the design sequence $T_{ij}$ cannot continue to absorb the impact of RU. If $y^*_{ijk} < u_{ijk}$, when $u_{ijk} = y^*_{ijk}$, the first derivative of compensatory cost is equal for the different value of $k$. The greater value of $O_{ijk}$, the stronger ability to absorb disturbance, so we compress the design sequence preferentially. The ability to absorb the disturbance of RU is decided by $E_{ijk}$, $y^2_{ijk}$, $f''(y_{ijk}^*)$, and $\Delta$, which provides the information on the behavior of compensatory cost function. $\Delta$ can be measured as

$$\Delta = \frac{f(u_{ijk}) - f(y^*_{ijk})}{u_{ijk} - y^*_{ijk}} \tag{14}$$

If the optimal compression amount of the interference interval has been overdrawn and still can not match the original scheme, we need to adopt the second compression to further compress for some compressible tasks. It is necessary to determine the order of compression based on the relationship between the secondary compression amount and increasing cost. We use the compound sequencing rules to determine the order of compression, which can be formulated as

$$O_{ijk} = (E_{ijk})^{\alpha_1} * (y^2_{ijk})^{\alpha_2} * (f''(y_{ijk}^*))^{\alpha_3} * (D)^{\alpha_4} \tag{15}$$

$$f''(y_{ijk}^*) = \partial^2 f(y^*_{ijk}) / \partial y^2_{ijk} \tag{16}$$

$$y^2_{ijk} = u_{ijk} - y^*_{ijk} \tag{17}$$

Due to the parameter value $\{\alpha_i\} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ has a greater impact on the reallocation, we use the basic genetic algorithm to achieve the coding, individual evaluation and other genetic manipulation, and then output the optimal parameter value $\{\alpha_i^*\} = (\alpha_1^*, \alpha_2^*, \alpha_3^*, \alpha_4^*)$ in the evolutionary process. The compression sequence of the design sequence can be determined with the estimation time and the duration of RU. The duration is composed of the sum of approximate duration and the time had been performed with RU disturbance. Where we use 4 dimensional nonnegative real number vector $\{\alpha_i\}$ as the individual population, $d_{PIS}/d_{NIS}$ is the fitness value of the giving parameter $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, $d_{PIS}$ and $d_{NIS}$ denote the Euclidean Distance of the two-dimensional vector $(L_{min}, C^a_{ijk})$ to the positive ideal point and negative ideal point respectively in the two-dimensional space composed of matching time and compression cost. $(L_{min}, C^a_{ijk})$ can be computed by the formula (20).

$$\min C = \sum C_c = \sum h y_{ijk}^{(a/b)} \tag{18}$$

$$\text{s.t.} \sum_{T_{ij} \in AT} (E_{ijk} - y_{ijk}) = L_{min} - W_1 - W_2$$

$$0 \leq y_{ijk} \leq u_{ijk} \leq E_{ijk}, \ T_{ij} \in AT$$

Where $AT$ is the set of affected tasks, $L_{min}$ is in the compression the task set, the completion time of the last task before RU is expressed as $W_1$, $W_2$ is the affected time of RU disturbance, which is equal to the sum of approximate duration and the time of the design task which has been performed with RU disturbance.

## 4. Computational study

### *4.1 Experimental design*

A case study is conducted to evaluate the performance of the proposed method and algorithm. We use MATLAB7.0 to write simulation and test program according to the contract terms and actual progress. An aircraft as a complex product consists 4 VDUs, 12 design tasks, and each task has the corresponding design sequence, the competent ability of VDU is different for design task. The essential information of the execution time, competent execution sequence, delivery period, activity type are shown in appendix. Without loss of generality, we let 30 be the number of chromosome population, 100 be the evolutionary generation, and crossover ratio parameters $P_{c1}=0.9$, $P_{c2}=0.5$, mutation ratio parameters $P_{m1}=0.15$, $P_{m2}=0.02$. After emerging the static initial design scheme, we consider that the $U_1$ is unavailable at the time 100, recovers at the time 130. We use the target weight value $\alpha:\beta:\gamma=5:1:1$ [9]. In the formula of compensatory cost, coefficient is randomly emerged from [1.4, 3.0], $a/b$ is randomly defined as [1, 2.7], $u_{ijk}$ is from $E_{ijk} \times$ Uniform [0.4. 1.1].

### *4.2 Computational results*

The Gantt charts of allocating/reallocating are shown in Fig. 1, (a) is the reallocation considering RS, (b) is CR and (d) is CETS. when RU occurs.

**Table 1. Performance of strategy**

| | RS | CR | CETS |
|---|---|---|---|
| Cmax | 266 | 250 | 238 |
| STB | 349 | 310 | 74 |
| CC | 462 | 771 | 434 |
| f | 2141 | 2331 | 1936 |

From Table 1, the $C_{max}$ in static scheme is 238, after responding to RU, $C_{max}$ in RS and CR changes to 266, 250, but it keeps 238 in CETS. In term of stability, we have used RS which only allocates the affected operations and preserve the stability of the resource allocation. In CETS, design sequences are prioritized as $T_{33}, T_{14}, T_{12}, T_{34}, T_{11,3}, T_{12,4}, T_{84}, T_{93}$, the compression time is 9, 7, 6, 6, 5, 5, 5, 4, 4 respectively. In CC, RS is better than the CR which is in keeping with reality, because there is no any change in matching situation in RS, but a lot in CR. However, CC in CETS is much lower because the increasing compressing cost has been counteracted by the declining of design cost. The overall performance shows CETS is significantly predominant than CR and RS theoretically. In addition, coordinated cost of CETS is much lower than them, even though we did not take it into account in this study.

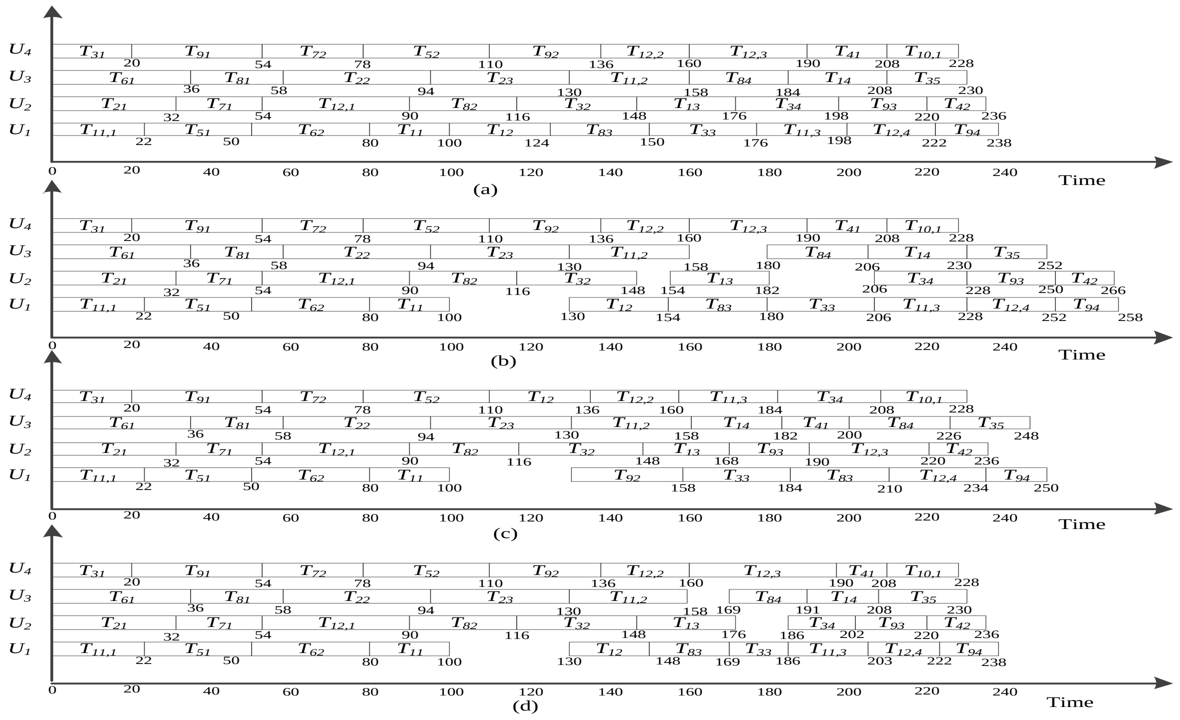

**Fig. 1** Gantt chart of allocating/reallocating

To verify the superiority of the algorithm, we compared AMOGATS with the adaptive multi-objective flexible dynamic scheduling algorithm (AMOFDSA), and the hybrid tabu search genetic algorithm (GATS). After 20 times independently running, the curve of the function value are shown in Fig. 2, the algorithm performance are shown in Table 2. The population size was 100, the maximum evolutionary number was 100, the crossover rate was 0.95, and the mutation rate was 0.15.

**Table 2. Performance comparison of algorithm**

| algorithm | $\overline{f}$ | $\overline{f}_1$ | $\overline{f}_2$ | $\overline{f}_3$ | $\bar{t}$ |
|---|---|---|---|---|---|
| AMOGATS | 3040 | 240 | 240 | 1600 | 16.552 |
| AMOFDSA | 3550 | 250 | 260 | 1790 | 21.536 |
| GATS | 3960 | 260 | 270 | 2130 | 27.658 |

From the result, GATS converges to it after 40 generations. AMOFDSA is in the steady search process between 7 to 17, and get the optimal solution in 22 generation. AMOGATS is in the steady search process between 7 to 10, and the optimal solution emerged in 14 generation. Also in the same operating environment, the running time by GATS is 27.658, AMOFDSA is 21.536, and it's 16.552 by AMOGATS. Thus, the AMOGATS has a higher performance and shorter running time than others.

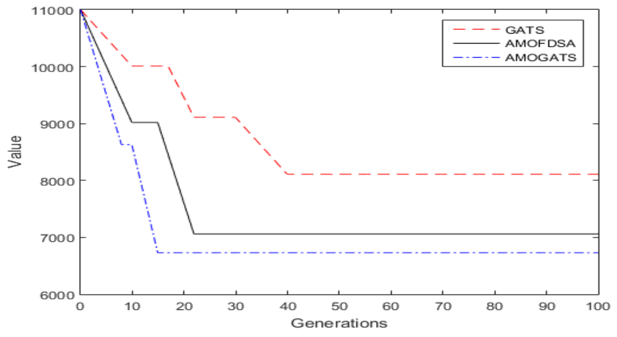

**Fig. 2.** Results of AMOGATS, AMOFDSA, GATS

### *4.3 Further discuss*

In order to choose the strategy reasonably, we try to analyze the applicable condition of CETS and CR. We consider the change of performance with different uncertain factors, such as arrival time, recovery time, and interval of RU. In this study, we assume that RU arrivals at three intervals [80-100], [40-60] [10-20] randomly, the recovery time is in the interval of [10-20], and the interval is [10-20]. We use 1, 2 and 3 to denote intervals [80-100], [40-60] [10-20] respectively, three levels of recovery time (10, 15, 20), and three intervals (10, 15, 20), A and B represent CETS and CR. Therefore, total of 54 experiments are conducted.

**Table 3** Impact of change factor on reallocation performance

| AT | RT | Int. | Stra. | $\bar{f}$ | $\bar{f}_1$ | $\bar{f}_2$ | $\bar{f}_3$ | Dev. | $\bar{t}$ |
|---|---|---|---|---|---|---|---|---|---|
| | | | A | 4025 | 493 | 843 | 224 | 2.20% | 6.51 |
| | | 10 | B | 4718 | 523 | 1265 | 315 | 2.35% | 6.59 |
| | | | A | 4339 | 499 | 1065 | 280 | 3.16% | 6.67 |
| 1 | 10 | 15 | B | 5202 | 547 | 1562 | 358 | 3.21% | 6.91 |
| | | | A | 4677 | 502 | 1359 | 306 | 3.51% | 6.97 |
| | | 20 | B | 6487 | 563 | 2415 | 694 | 4.33% | 7.13 |
| | … | … | … | … | … | … | … | … | … |
| | | 10 | A | 6853 | 603 | 2512 | 723 | 5.91% | 8.73 |
| | | | B | 8366 | 622 | 3898 | 736 | 7.05% | 8.82 |
| | … | 15 | A | 7133 | 609 | 2678 | 801 | 4.35% | 8.96 |
| 2 | 15 | | B | 8683 | 643 | 4012 | 813 | 4.76% | 9.23 |
| | … | 20 | A | 9039 | 910 | 2783 | 796 | 3.79% | 9.77 |
| | | | B | 9065 | 655 | 3823 | 1312 | 4.05% | 9.86 |
| | | | … | … | … | … | … | … | … |
| | | … | … | … | … | … | … | …. | … |
| | | 10 | A | 9327 | 756 | 3896 | 895 | 3.78% | 11.69 |
| | … | | B | 9608 | 772 | 4056 | 920 | 4.83% | 11.78 |
| 3 | | 15 | A | 9822 | 734 | 4155 | 1263 | 6.35% | 11.89 |
| | 20 | | B | 9948 | 759 | 4180 | 1214 | 6.56% | 11.97 |
| | | 20 | A | 10535 | 746 | 4356 | 1703 | 7.26% | 12.05 |
| | | | B | 10589 | 763 | 4425 | 1586 | 7.52% | 12.18 |

Notes: AT: The arrival time; RT, The recovery time; Int.: The interval of RU; Stra.: The CR and CETS strategies; Dev.: The deviation; t: The running time of algorithm.

When RU scale is consistent, the influence on allocation is positively associate with recovery time and arrival interval. Similarly, when recovery time is set, the effect is positively associate with arrival interval as well. RU occurs more forward, the greater the impact. General trend is consistent in other condition. But there is a special point where arrival time is in 3, recovery time is 20, interval is 20, CR is better than CETS in overall performance, but $C_{max}$ is still lower than CETS. When complexity is increasing, the capability of CETS to respond to RU is decreasing. On the contrary, CR keeps a growth trend. Because of the ambiguity in the description of coordination cost, if practitioners only concentrate the time, they can choose the CETS. Otherwise, the balance between compensatory cost, stability and tardiness penalty should be considered according to the actual situation.

## 5. Conclusion and future work

In this paper, CETS is developed for task reallocation in CPD and compared with the performance of CR and RS. The results obtained from an extensive experiment show that CETS outperforms CR and RS in completion time, compensatory cost, as well as stability. The applicable conditions and related influencing factors of CETS are discussed. When the problem is more complex, the performance is no longer in a regular pattern, CR shows the superiority gradually. Furthermore, AMOGATS algorithm is used to solve this problem, computational examples show a high efficiency of the algorithm. In this work, RU is responded based on the existing initial scheme. However, the probability allocations of RU, recovery time, and arrival of new tasks have been investigated in some latest studies. Thus, the priority of compression can combine probability distributions to formulate an anticipative allocation, which can absorb the impact of RU automatically. In this way, loads of manual input and adjustment can be eliminated from process of CPD, which should be completed more precisely in future study.

## References

[1] Jimenez, M. I., Val, L. D., Villacorta, J. J., & Izquierdo, A. (2012). Design of task scheduling process for a multifunction radar. *Iet Radar Sonar Navigation, 6*(5), 341-347.

[2] Pedro M. Castro, Ana P. Barbosa-Póvoa, & Augusto Q. Novais. (2005). Simultaneous design and scheduling of multipurpose plants using resource task network based continuous-time formulations. *Industrial & Engineering Chemistry Research, 44*(2), págs. 343-357.

[3] Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling, 12*(4), 417-431.

[4] Sabuncuoglu, I., & Bayız, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research, 126*(3), 567-586.

[5] Gürel, S., Körpeoğlu, E., & Aktürk, M. S. (2010). An anticipative scheduling approach with controllable processing times. *Computers & Operations Research, 37*(6), 1002-1013.

[6] Cao X B, Xu C D, Hu C S. (2015) Design resource agglomeration methods based on design ability. *Computer Intergated Manufacturing Systems*, *21(9)*:2296-2311.

[7] G. Q. Huang, J. S. K. Lau, K. L. Mak, & L. Liang. (2006). Distributed supply-chain project rescheduling: part ii—distributed affected operations rescheduling algorithm. *International Journal of Production Research, 44*(1), 1-25.

[8] Chandra Mohan, B., & Baskaran, R. (2012). Review: a survey: ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications, 39*(4), 4618-4627.

[9] Rangsaritratsamee, R., Ferrell, W. G., & Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering, 46*(1), 1-15.