A Parallel Evolution based Intelligent Optimization Algorithm to Solve Large-scale Production Planning Problem in Defense Industry

Yu Zhou
College of Information System and Management, National University of Defense Technology, Changsha 410073, P. R. China
Email: zhouyu_gfkd@126.com

Jiang Jiang
College of Information System and Management, National University of Defense Technology, Changsha 410073, P. R. China
Email: jiangjiang_gfkd@126.com

***Abstract*—Since the multistage production planning problem possesses high-dimensional variables and large-scale solution space, it is hard to be solved in an acceptable time. To deal with this challenge, we propose a parallel evolution based intelligent optimization algorithm. The proposed algorithm employs the differential evolution as the algorithm framework to implement the primary mutation and crossover operations, then the entire variables are clustered into several sub-populations according to the problem structure, finally a parallel evolution strategy is proposed to speed up the convergence and progress the search precision of the sub-population. A case of weapons production planning is studied to validate the proposed algorithm. The results show that this algorithm has the fastest convergence and the best global searching capability, compared with classical differential evolution algorithm, genetic algorithm and particle swarm optimization algorithm.**

***Index Terms*—production planning; multistage stochastic programming; differential evolution; parallel evolution; variables clustering**

I. Introduction

The scenario tree based stochastic programing as a kind of mathematical programming technique is more and more popular to model the Large-scale Production Planning Problem in Defense Industry [1-3]. However, the challenge of this application is that the variable dimensions and the solution space exponentially increase with the planning stage and the scenarios. Moreover, they are usually larger than 1000 dimensions and 10^{1000} solutions respectively in the context of weapons production planning. In recent literatures [4-6], the scenario tree based multistage stochastic programming model is solved by some business software such as CPLEX.

However, the weapons production planning (WPP) problem belongs to the nonlinear integer programming with real exponent and it is hard to be converted into the standard form which could be solved by CPLEX. Some vigorous mathematical approaches are also studied to solve the multistage stochastic programming model, but these approaches require the model to satisfy some strict restraints such as convex programming [7], linear programming [8] and/or 0-1 mixed integer programming [9]. Unfortunately, the multistage stochastic programing model in the context weapons production planning does not satisfy these constraints.

In recent years, more and more researchers developed the intelligent optimization algorithms to attack the large-scale production planning problem in many fields. Wu et al. developed a new heuristic search algorithm to solve the dedicated and flexible capacity planning problems [10]. Jiao et al. proposed a generic algorithm (GA) with specific encoding scheme to synchronize product portfolio generation and selection in production planning [11]. Chen et al. introduced a binary particle swarm optimization algorithm (PSO) with dynamic inertia weight and mutation mechanism for the production planning problem in the thin film transistor Array process [12]. Interested readers may refer to [13-16].

However, neither did these studies consider the scenario tree based variables structure in multistage stochastic programming model nor systematically develop an evolutionary algorithm to specifically handle the high-dimensional variables in production planning. At the same time, differential evolution (DE) which is considered as an important branch of the evolutionary algorithms is widely used to solve optimization problems in many fields [17-20]. The simple mutation operation and one-on-one competition strategy of DE reduce the complexity of traditional genetic operations, which brings to a strong global convergence and robustness.

Based on these surveys, we seek to develop an intelligent optimization algorithm to solve the multistage stochastic programming problem in the context of weapons production planning. In order to handle the high-dimensional variables and the large-scale solution space, we propose two novel optimization strategies which are the variable clustering and parallel evolution. To the best of our knowledge, this intelligent optimization algorithm is first proposed here and could be applied to other engineering fields.

II. Multistage Stochastic Programming Model

A. Overview of Weapons Production Planning

Suppose that there are C capability requirements and M weapon categories, in which each capability requirement is supported by more than one weapon category and vice versa. Each weapon category contains a number of weapon models. For each weapon model, the capability values depend on the corresponding production time and production budget. In each weapon category, the weapon models have different capability values when they are assigned with equivalent production time and production budget. To understand the concept of weapons production planning, we need to know the weapons multi-period production process which is introduced in Figure 1.

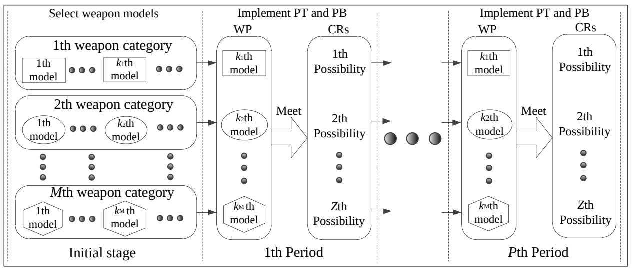


Figure 1. Weapons multi-period production process

In Figure 1, the weapons production process consists of two kinds of variables, one is to select the optimal weapon model from each category which will be implemented at initial stage, and the other one is to optimize the production time (PT) and the production budget (PB) of each selected weapon which will be implemented in multi-period. All of the selected weapons constitute a weapons portfolio (WP). The objective is to minimize the capability gaps between the weapons portfolio and the capability requirements (CRs) over the planning horizon. In this context, both of the capability deficiencies and surpluses between capabilities of the weapons portfolio and the capability requirements are recognized as the capability gaps.

B. Formulations of Capabilities for the Weapons Portfolio

Each capability of the weapons portfolio comes from the corresponding one of each weapon in this portfolio. Thus, we firstly calculate the capability value of each weapon. Given the production time and the production budget in each period, if this weapon does not have the cth capability, $WC_{k_m,p,c}$ = 0; otherwise, $WC_{k_m,p,c}$ is calculated as follows:

$$WC_{k_m,p,c} = \begin{cases} (\sum_{l=1}^{L} a_{k_m,p,c,l} \bullet t_{k_m,p}^{\alpha_{k_m,p,c,l}}) \times (\sum_{h=1}^{H} b_{k_m,p,c,h} \bullet q_{k_m,p}^{\beta_{k_m,p,c,h}}); p == 1 \\ (\sum_{l=1}^{L} a_{k_m,p,c,l} \bullet t'^{\alpha_{k_m,p,c,l}}_{k_m,p}) \times (\sum_{h=1}^{H} b_{k_m,p,c,h} \bullet q'^{\beta_{k_m,p,c,h}}_{k_m,p}); 1 < p < P \end{cases} \tag{1}$$

$$\text{s.t. } t'_{k_m,p} = t_{k_m,p} \times \left(1 + \sum_{p'=1}^{p-1} (t_{k_m,p'} \Big/ (T \times (p - p')))\right); \tag{2}$$

$$q'_{k_m,p} = q_{k_m,p} \times \left(1 + \sum_{p'=1}^{p-1} (q_{k_m,p'} \Big/ (qu_{k_m,p'} \times (p - p')))\right); \tag{3}$$

$$a_{k_m,p,c,l}, b_{k_m,p,c,l}, \alpha_{k_m,p,c,l}, \beta_{k_m,p,c,l} \in \mathbb{R}^+ \tag{4}$$

where Eq. (1) represents the cth capability value of the k_mth weapon in the *pth* period. The L-order (resp. H-order) polynomial represents the relation between the cth capability and the production time (resp. production budget). The production time (resp. budget) of the previous $i-1$ periods add the effective production time (resp. budget) in the ith period according to Eqs. (2, 3).

Since the cth capability requirement is that the quantity $Rq_{p,c}$ is needed at the time $Rt_{p,c}$, we need to further calculate the cth capability value of the weapon at the time $Rt_{p,c}$, it is calculated as follows:

$$WC'_{k_m,p,c} = \begin{cases} WC_{k_m,p,c} \times (t_{k_m,p} / Rt_{p,c}) & t_{k_m,p} < Rt_{p,c} \\ WC_{k_m,p,c} & t_{k_m,p} == Rt_{p,c} \\ WC_{k_m,p,c} \times (Rt_{p,c} / t_{k_m,p}) & t_{k_m,p} > Rt_{p,c} \end{cases} \tag{5}$$

where Eq. (5) represents that $WC'_{k_m,p,c}$ is smaller while the disparity between $t_{k_m,p}$ and $Rt_{p,c}$ is larger. This is because the weapon has not been completely developed when $t_{k_m,p} > Rt_{p,c}$, on the contrary, the weapon lacks enough funding to maintain the achieved capability when $t_{k_m,p} < Rt_{p,c}$.

After acquiring the cth capability value of each weapon model, we can calculate the corresponding capability value of the weapons portfolio through weighted sum, which is calculated as follows:

$$WC_{p,c} = \sum_{m=1}^{M} w_{m,p,c} \times WC'_{k_m,p,c} \tag{6}$$

where $w_{m,p,c} = 0$, if the mth weapon category does not have the cth capability.

C. Formulations of the Uncertain Capability Requirements

In a practical defense engineering environment, the capability requirements are incrementally evolving over the whole planning horizon while they have uncertain values in each period. Considering this property, the uncertain capability requirements are formulated as follows:

$$\{R_{p,z,c} \,|\, R_{p,z,c} = (Rt_{p,z,c}, Rq_{p,z,c})\} \tag{7}$$

$$\text{s.t. } Rt_{p',z,c} \le Rt_{p,z,c},\ Rq_{p',z,c} \le Rq_{p,z,c},\quad p' < p, \tag{8}$$

$$p, p' = 1,2,...,P,\ \ c = 1,2,...,C,\ \ z_p = 1,2,...,Z_p \tag{9}$$

$$Rq_{p,z,c} \in \mathbb{R}^+,\quad Rt_{p,z,c}, P, C, Z_p \in \mathbb{N}^+ \tag{10}$$

where Eq. (7) represents that the cth capability requires the value $Rq_{p,z,c}$ at the time of $Rt_{p,z,c}$ in the *pth* period. $R_{p,z,c}$

has Z_p possibilities in each period. Constraint (8) represents $Rq_{p,z,c}$ and $Rt_{p,z,c}$ increase with the period.

D. Multistage Stochastic Programming Model

The multistage stochastic programming model (MSP) provides different production times and production budgets to hedge against each possible group of capability requirements. Since the variables in *p*th period correlate with all of the ones in the previous (*p*-1)th period, the scenario tree could be used to represent the variables structure corresponding to the possible evolution processes of the capability requirements. The possibilities of the capability requirements equal to the branches of the scenario tree in each period. A scenario tree with 2-2-2 branches is shown in Figure 2.

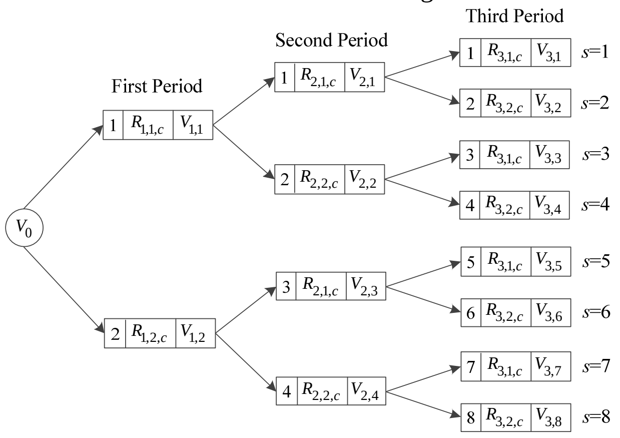


Figure 2. The 3-period scenario tree with 2-2-2 branches

In Figure 2, the path from the root node (circular node) to a leaf node (rectangular node) is called a scenario *s*, i.e. a possible evolution process of the capability requirements over the entire planning horizon. The variable set V_0 in root node corresponds to the selected weapon models to encounter all of the scenarios. Then, the variable set $V_{p,n}$ corresponds to the production time and the production budget in each leaf node.

The objective of weapons production planning is to minimize the weighted sum of the capability gaps between the weapons portfolio and the capability requirements in each leaf node. Thus, the multistage stochastic programming model for weapons production planning is formulated as:

$$\min \sum_{p=1}^{P}\sum_{n_p=1}^{N_p}\pi(n_p)\times\sum_{c=1}^{C}\left|WC_{p,n,c}-Rq_{p,z,c}\right| \tag{11}$$

$$\text{s.t.} \quad z_p=\begin{cases} Z_p; & \bmod(n_p/Z_p)=0 \\ \bmod(n_p/Z_p); & \bmod(n_p/Z_p)\neq 0 \end{cases} \tag{12}$$

$$N_p=\prod_{p'=1}^{p} Z_{p'} \qquad P, N_p, Z_p\in\mathbb{N}^{+} \tag{13}$$

$$t'_{k_m,p,n}=t_{k_m,p,n}\times\left(1+\left(\sum_{p'=1}^{p-1} f_{pre}^{p'}(t_{k_m,p,n})\Big/(T\times(p-p'))\right)\right) \tag{14}$$

$$q'_{k_m,p,n}=q_{k_m,p,n}\times\left(1+\left(\sum_{p'=1}^{p-1} f_{pre}^{p'}(q_{k_m,p,n})\Big/(qu_{k_m,p'}\times(p-p'))\right)\right) \tag{15}$$

$$0<t_{k_m,p,n}\le T; \quad t_{k_m,p,n},\ T\in\mathbb{N}^{+} \tag{16}$$

$$ql_{k_m,p}\le q_{k_m,p,n}\le qu_{k_m,p}; \quad q_{k_m,p,n}, ql_{k_m,p}, qu_{k_m,p}\in\mathbb{N}^{+} \tag{17}$$

$$\sum_{m=1}^{M} q_{k_m,p,n}\le Q_p \tag{18}$$

where $\pi(n_p)$ is considered as the unconditional probability of the n_pth possibility of the capability requirements in each node. Constraints (14, 15) describe the relation between n_p and Z_p. mod(•) represents the remainder operation. $f_{pre}^{p'}(\bullet)$ in constraints (14, 15) locates the variables of the ancestor node in the p'th period. Furthermore, the variables in all leaf nodes (circular nodes) must share the same variables of the weapon models in the root node (rectangular node). Constraint (16) and (17) represent the available range of the production time and the production budget, respectively. Constraint (18) represents the overall available budget for the weapons portfolio in each period.

III. Algorithm Developments

A. Classical DE Algorithm

We first develop the classical differential evolution (CDE) algorithm to solve the addressed multistage stochastic programming model. It is introduced as follows:

Step 1 (Population Initialization). Generate the value of each variable randomly in the available range according to constraints (16, 17). The individual encoding is shown as follows:

$$V_0\ V_{1,1},\ldots,V_{1,N},\ldots,V_{p,n},\ldots,V_{P,N}$$
$$\begin{cases} \overbrace{k_1,\ldots,k_m,\ldots,k_M}^{V_0} \\ \overbrace{t_{k_1,p,n},\ldots,t_{k_m,p,n},\ldots,t_{k_M,p,n}\ q_{k_1,p,n},\ldots q_{k_m,p,n},\ldots,q_{k_M,p,n}}^{V_{p,n}} \end{cases} \tag{19}$$

Step 2 (Evaluation). For each current individual, calculate the objective value by Eq. (11), and then calculate the violation of the overall budget constraint by Eq. (18). Record the current global optimization individual. If there is no individual who satisfies the overall budget constraint, take the individual with minimum violated degree of the budget constraint as the global optimization one.

Step 3 (Mutation). Execute mutation operation which is formulated as

$$U_i=X_{i_1}+F\times(X_{i_2}-X_{i_3}) \tag{20}$$

where X_{i_1}, X_{i_2} and X_{i_2} are the randomly chosen individuals from the current population X. The mutation scaling factor *F* is a real constant number which is often set to 0.5 [22].

Step 4 (Crossover). Select each variable of the current individual X_i with certain probability to replace the corresponding bit of the temporary individual U_i, which is formulated as

$$U_i(j)=\begin{cases} U_i(j), & R_j(0,1)<cr \\ X_i(j), & otherwise \end{cases} \tag{21}$$

where $R_j(0,1)$ is a uniform random number between 0 and 1, and *cr* is the crossover rate which is often set to 0.9 [22].

Step 5 (Handling constraints of variables). Judge each bit in each individual whether or not violate the constraints (16, 17) in the temporary population. The bit which violates the corresponding constraint randomly obtains a number as its value within the available range.

Step 6 (Selection). Evaluate the objective value and the violation of budget constraint for each temporary individual U_i. If the current individual X_i and temporary individual U_i both satisfy the budget constraint, the individual with minimum objective value is selected as new individual. If only one of them violates the budget constraint, the individual without violation of the constraint is selected. If U_i and X_i both violate the constraint, the individual with minimum violated degree is selected. Through above operation, the new population is generated. Then, the global optimization individual is updated.

Step 7 (Loop). Record the current global optimization individual. Repeat steps 3 to 7 until the maximum number of iterations or the desired solution is obtained.

B. *Parallel Control Strategy*

As shown in Figure 2, the individual contains one root variable unit V_0 and fourteen leaf variable units. According to the Figure 2 and the encoding equation (19), if V_0 is fixed, the leaf variable units obviously can be clustered into two completely uncorrelated parts. Moreover, if V_1 and V_2 are also fixed, the else leaf variable unit can be clustered into four completely uncorrelated sub-individuals. Meanwhile, the corresponding objective can also be decomposed two child-objectives and/or four grandchild-objectives, because objective function (11) is the weighted summation of all the node objective functions.

To handle the crucial problem on how to fix V_0, we seek to dynamically determine the V_0 during the iteration process. The idea is that if V_0 of the global best individual keeps invariant within consecutive *I* iterations, each variable has obtain the best value. Thus, the V_0 of the global best individual is taken as the fixed root variable unit of each individual in the clustering process.

The specific evolution procedure based on the variables clustering-1 named as PEDE-1 is introduced as follow:

Step 1. Define the Gv_0 as the root variable unit of the global best individual. Define $cou = 0$ and a constant *I*.

Step 2. Implement iterations of the fundamental evolution process.

Step 3. Judge whether each variable in Gv_0^t equivalent to the corresponding one in Gv_0^{t-1}. If yes, $cou = cou + 1$; else, $cou = 0$.

Step 4. If $cou == I$, implement the variables clustering-1 operation and take the Gv_0^t as the root variable unit of each individual, else, go to Step 1.

Step 5. Implement the mutation and crossover for each part sub-individuals independently.

Step 6. Implement the evaluation operation and calculate the sub-objective value corresponding to each sub-individual.

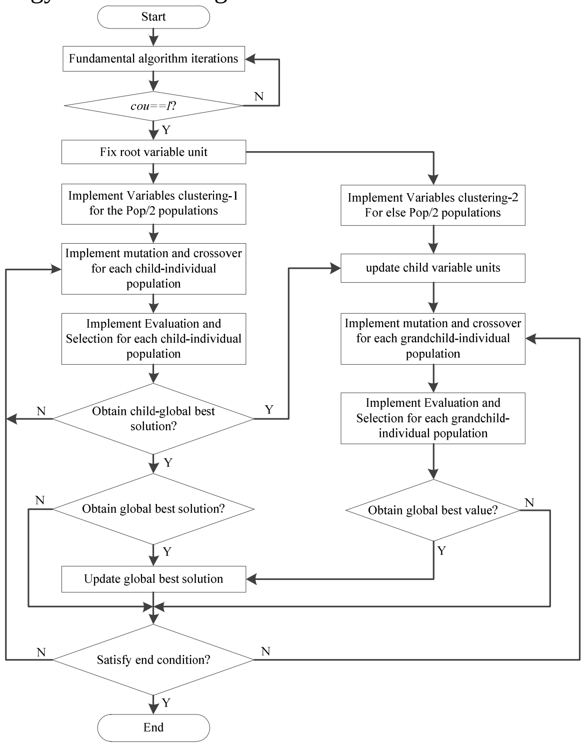
Step 7. Implement selection operation for each sub-individual independently.

Step 8. Aggregate the sub-individuals and sub-objective values into the comprehensive individual and objective value, respectively. Then, update the global best individual.

Step 9. Repeat steps 5 to 7 until the maximum number of iterations or the desired solution is obtained.

C. *Parallel Evolutionary Optimization*

The parallel evolutionary strategy is proposed to improve the performance of the fundamental differential evolution algorithm, which is named as PEDE. The purpose of PEDE is to simultaneously implement the evolution operations on the clustering-1 based populations and clustering-2 populations. This is because, if we only use the variables clustering-1 strategy which is named as PEDE-1, the solution space of the child variable units is still far larger than the one of the ideal scale of the variables. Thus, the child variable units could not converge to the global best values in short iterations. Furthermore, if we only use the variables clustering-2 strategy to implement evolution operation which is named as PEDE-2, the algorithm will be premature and trapped a local optimum. Thus, the parallel evolutionary strategy is shown in Figure 3.


Start
Fundamental algorithm iterations
cou==I?
N
Y
Fix root variable unit
Implement Variables clustering-1 for the Pop/2 populations
Implement Variables clustering-2 For else Pop/2 populations
Implement mutation and crossover for each child-individual population
update child variable units
Implement Evaluation and Selection for each child-individual population
Implement mutation and crossover for each grandchild-individual population
Implement Evaluation and Selection for each grandchild-individual population
Obtain child-global best solution?
N
Y
Obtain global best solution?
Obtain global best value?
Update global best solution
Satisfy end condition?
End
Figure 3: Flowchart of PEDE

As shown in Figure 3, the left-hand represents the optimization of one half populations at the level of child individual corresponding to PEDE-1, and the right-hand represents the optimization of the else half populations at the level of grandchild individuals corresponding to PEDE-2. Meanwhile, if PEDE-1 obtains child global best solution, it will update the child variable unit of the other half populations. This technique holds the optimization on grandchild individual population not to be trapped in local optimum. Therefore, PEDE could update the global best solution based on the variables clustering-1 and clustering-2 simultaneously during the iterations, it make the CDE possess the faster convergent speed and the higher searching precision.

In fact, the variables clustering-1 based sub-population corresponds to the local search in a large scale and the variables clustering-2 based sub-population corresponds to the local search in a small scale. Generally speaking, the algorithm could have a better global searching capability at the earlier iterations, in order to find more global optimization solutions. Then, the algorithm could have a better local searching capability at the later iterations, in order to improve the searching precision and convergence. To meet these demands, we propose an adaptive control strategy for each sub-population scale, which is shown in Figure 4.

```
$j = 1;$
$for\ \ g = 1:G$
  $if\ \ \ g/G > j \times \theta$
    $subpop_1 = round(pop \times (1 - g/G));$
    $subpop_2 = pop - subpop_1;$
    $j = j + 1;$
  $end$
$end$
```

Figure 4: Adaptive control strategy for each sub-population

In Figure 4, $subpop_1$ (resp. $subpop_2$) corresponds to the variables clustering-1 (resp. clustering-2) based sub-population; θ is an judgment coefficient and belongs to (0, 1) according to the empirical settings; *g* is the current iteration times and *G* is the overall iteration times. The most individuals join subpop1 to hold global searching capability at the initial stage. Then, the individuals in the subpop1 migrate to the subpop2 with the increase of the iteration times, in order to strength the searching precision and convergence. Meanwhile, to balance the stability of the evolution on the two sub-populations, migration do not occur in each generation but occur when g/G increases beyond the *j* times of θ.

IV. Computational Results

A. Case Description

Intelligence, surveillance and reconnaissance (ISR) weapons are indispensable components of military operations, they must work together to provide commanders and soldiers with a comprehensive understanding of battlefield situation. Thus, multiple ISR weapons need to be planned to satisfy the capability requirements. There are five weapon categories need to be considered, and the corresponding variables and their ranges are shown in Table 1.

TABLE 1
THE VARIABLES AND THEIR RANGES OF THE WEAPON CATEGORIES

Weapon	Model	Time	Budget1	Budget2	Budget3
RS	[1,5]	(0, 15)	(0, 200)	(0, 300)	(0, 200)
AWCS	[1,4]	(0, 15)	(0, 200)	(0, 200)	(0, 150)
MHAUAS	[1,6]	(0, 15)	(0, 100)	(0, 100)	(0, 100)
BLSDL	[1,3]	(0, 15)	(0, 50)	(0, 100)	(0, 100)
C2S	[1,4]	(0, 15)	(0, 100)	(0, 100)	(0, 120)

All of the notations in Table 1 are listed in the following: (1) RS is the reconnaissance satellite. (2) AWCS is the airborne warning and control system. (3) MHAUAS is the medium to high altitude unmanned aerial system. (4) BLSDL is the beyond line of sight data link. (5) C2S is the command and control system. Each category weapon has three kinds of variables, i.e. model selection variable k_m, production time variable $t_{k_m,p}$ and production budget variable $q_{k_m,p}$. For example, the data on RS in Table 1 represents that it has five optional weapon models. The production time must be planned within a given interval i.e. 0 to 20 in each period and the unit of production time is month. The production budget must be planned within 0 to 300, 0 to 400 and 0 to 300 for each period, respectively, and the unit of production budget is million dollars.

To explore the performance of the proposed algorithm, we construct the scenario trees from 3-3-3 to 5-5-5 branches corresponding to the different scale of the possible evolving capability requirements. The related data are shown as in Table 2.

TABLE 2
PARTIAL DATA ON THE SCALE OF THE SIX INSTANCES

Branches	3-3-3	4-4-4	5-5-5
Children clusters	3	4	5
Grandchildren clusters	9	16	25
Leaf nodes	39	84	155
Variables	395	845	1555
Constraints	434	925	1710
Scale	$>10^{780}$	$>10^{1680}$	$>10^{3100}$

The scenario trees with different branches have the same structure which was shown in Figure 3. Since there are five weapon categories to be planned, each leaf node has corresponding five production time variables and five production budget variables.

B. PEDE Versus GA

We first use PEDE to solve the MSP model. Then, we replace the evolution operations of DE with GA. For the variables take the decimal integer encoding, the heuristic crossover based GA is appropriate to this encoding [23], which is called GA-1. Then, in order to strengthen the stochastic search capability of GA-1, we revise the heuristic crossover of the GA-1 from Eq. (22) to Eq. (23):

$$p_{ch,i} = p_{fa,i} + R(0,1) \times (p_{ma,i} - p_{fa,i}) \qquad (22)$$

$$p_{ch,i} = p_{fa,i} + R_i(0,1) \times (p_{ma,i} - p_{fa,i}) \qquad (23)$$

where $p_{fa,i}$, $p_{ma,i}$ and $p_{ch,i}$ are the *i*th bit of the parent, mother, and child, respectively. The difference between Eq. (22) and Eq. (23) is that $R(0,1)$ (resp. $R_i(0, 1)$) is fixed (resp. variable) in response to each bit of the mother and father. The GA-1 revised by Eq. (24) is called GA-2. Finally, we further amplify the stochastic search capability through the revision of crossover probability *pc* and mutation probability *pm* from Eq. (24) to Eq. (25):

$$pc = 0.9; \qquad pm = 0.05 \tag{24}$$

$$pc = 0.4 + R(0,1)\times 0.5; \qquad pm = R(0,1)\times 0.1 \tag{25}$$

The GA-2 revised by Eq. (25) is called GA-3. The three versions of GA and the PEDE run 30 times, respectively, to solve the addressed case. The best results of the three versions of GA and the worst result of PEDE are shown in Figure 5.

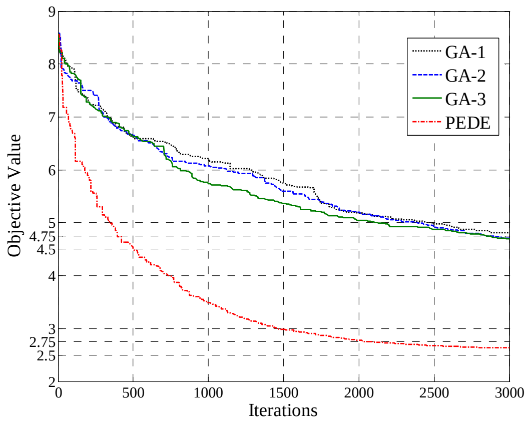


Figure 5: The comparison between PEDE and GA

As shown in Figure 5, the worst result of PEDE is superior to the best results of the three versions of GAs. The final global optimization value of GA-1, GA-2 and GA-3 are comparable with each other, and they are 4.7510, 4.7034 and 4.6965, respectively. It is implied that the boost of the stochastic search is neither improve nor weaken the performance of GA. Furthermore, the inertia evolution operation of DE is superior to the GA in this context.

C. PEDE Versus PSO

Next, we compare PEDE with PSO algorithm. For updating the velocity of a particle in PSO, one kind of updating equation is presented as follows [24]:

$$V_i^d = w\times V_i^d + c_1\times R(0,1)_{1,i}\times(pbest_i^d - X_i^d) + c_2\times R(0,1)_{2,i}\times(gbest^d - X_i^d) \tag{26}$$

Another kind of updating equation is presented as follows [25]:

$$V_i^d = w\times V_i^d + c_1\times R(0,1)_{1,i}^d\times(pbest_i^d - X_i^d) + c_2\times R(0,1)_{2,i}^d\times(gbest^d - X_i^d) \tag{27}$$

The difference between Eq. (28) and Eq. (29) is that the former has the same random number $R(0,1)$ for each dimension of the *i*th particle; the latter has different random number $R(0,1)^d$ for each dimension of the *i*th particle. Hence, we use these two updating equations to explore the performance of the PSO, respectively. The PSO based on (28) is called PSO-1; the PSO based on (29) is called as PSO-2. We further strengthen the randomness of PSO-2 to investigate the improvement of the search space. Thus, we revise *w* and *c* from the empirical setting (28) to the randomness setting (29):

$$w = 0.5; \qquad c = 2 \tag{28}$$

$$w_i = R(0,1)_i; \quad c_{j,i} = 1 + R(0,1)_i\times 2, \quad j = 1,2 \tag{29}$$

The PSO-2 algorithm revised by Eq. (40) is called PSO-3. The three versions of PSO run 30 times, respectively, to solve the addressed case. The best results are shown in Figure 6.

As shown in Figure 6, PSO-1 is of premature convergence due to the same random numbers for all dimensions, which leads to a smaller search space. PSO-2 and PSO-3 show the better convergent efficiency and search space than PSO-1. Moreover, the final global optimization value of PSO-3 achieves 3.494. But the PSO-3 is still inferior to PEDE on global searching capability, even though it has the better convergence speed than PEDE before the previous 415 iterations. Thus, we conclude that PEDE has more advantages to solve the MSP model than the addressed PSO algorithms.

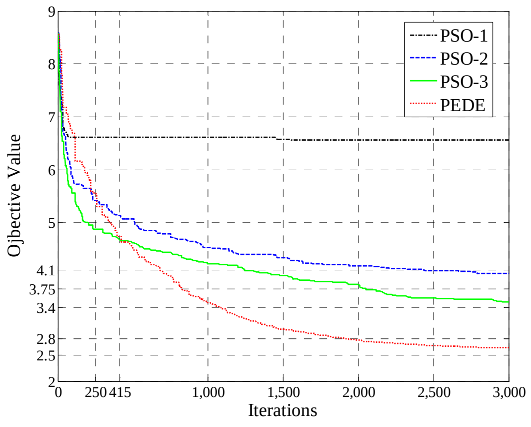


Figure 6. The comparison between PEDE and PSO

D. PEDE Versus CDE

For *F*=0.5 and *cr* =0.9 is the empirical settings of CDE[26], we change CDE algorithm as follow revisions: (1) *F*=0.5, *cr*=0.9, which is named as CDE-1; (2) *cr*=0.9, *F* is replaced with uniform random number between 0 and 1, which is named as CDE-2; (3) *F*=0.9, *cr* is replaced with uniform random number between 0 and 1, which is named as CDE-3. These algorithms run 30s repeatedly, respectively. The best results are shown in Figure 7.

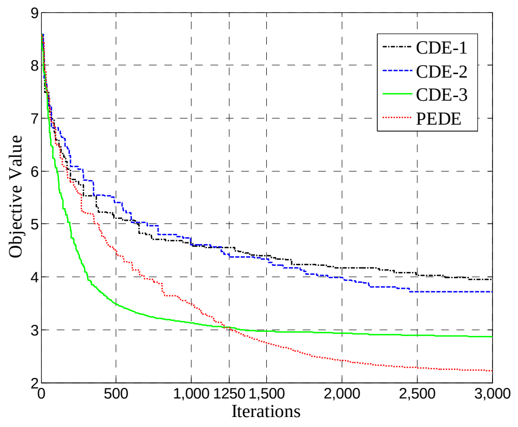


Figure 7. The comparison between PEDE and CDE

As shown in Figure 7, the curves of the three revised algorithms get the best values 3.9442, 3.7793 and 2.8234, respectively. The curve of the PEDE is in response to the best value of 30s runs. PEDE still outperformances than the other revised algorithms. It is referred that the only *F* and *cr* are set as uniform random numbers simultaneously, the classical DE i.e. all fixed one can be improved to the best performance. Furthermore, PEDE obtains an enough large searching space through the entire randomized parameters. The statistical comparisons of the addressed algorithms are shown in Table 3. PEDE has the best stability than the other algorithms according to the standard variation (Std.), and only the stability of classical DE closes to the PEDE.

TABLE 3
STATISTICAL COMPARISON UNDER DIFFERENT SCALES

	Index	3-3-3	4-4-4	5-5-5
GA-3	Best	4.4624	5.3654	7.3900
	Mean	4.6965	5.6456	7.7908
	Worst	4.5319	5.9265	8.1917
	Std.	0.0191	0.1556	0.2222
PSO-3	Best	3.4940	3.6395	3.7989
	Mean	3.5444	3.9198	4.1989
	Worst	3.5790	4.2007	4.5991
	Std.	0.0104	0.0617	0.0119
CDE-3	Best	2.8234	4.0337	4.4972
	Mean	3.2570	4.3072	4.8162
	Worst	3.6991	4.5804	5.1358
	Std.	0.1341	0.1516	0.1768
PEDE	Best	2.2439	3.2645	3.3411
	Mean	2.4876	3.2957	3.3835
	Worst	2.6516	3.3274	3.4252
	Std.	0.0103	0.0172	0.0231

Next, we analyze the performances of PEDE and the other algorithms in each instance which is shown in Table 3.As shown in Table 3, PEDE performs the best, followed by CDE-3 and PSO-3, and the worst is GA-3. According to the measures of Best and Mean, we can conclude that PEDE has the best convergence and searching efficiency than the other three algorithms. Furthermore, the measures of Std and Worst imply that PEDE has the best robustness and stability than the other algorithms.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an intelligent optimization algorithm to solve the multistage stochastic programming model in weapons production planning, in which a population parallel evolution based approach is proposed to handle the high-dimensional variables and the large-scale solution space. That is, the populations are partitioned into several children clusters, and which are implemented the DE evolution operations independently. Meanwhile, the global best optimum individual of grand-child populations is updated in the process of iterations.

The developed algorithm is applied to solve a case of the multistage stochastic programming model in weapons production planning. Six instances with different scale of the variable dimensions and solution space are adopted to test this algorithm. The computational results show that PEDE has the best global searching capability but a slow convergence speed, compared with GA, PSO and DE. Fortunately, the variables clustering approach can speed up the convergence and enlarge the searching space of CDE, while each strategy of this approach is also validated to be effective independently. PEDE performs the best effectiveness between 3-3-3 to 5-5-5 branches, in other words, when the variables scale does not exceed more than approximate 2000 dimensions. It can be found that the standard deviation (Std.) of PEDE is the smallest among these algorithms under each scale of the branches. It is also implied that the parallel evolutionary approach also brings robustness and stability for CDE.

ACKNOWLEDGMENT

This research is supported in part by the National Science Foundation of China under Contract no. 71031007, 71001104 and 71201168. We are grateful for these financial supports. We are also grateful to the anonymous reviewers for their valuable comments and constructive criticism.

REFERENCES

[1] S. Ahmed and N. V. Sahinidis, "An approximation scheme for stochastic integer programs arising in capacity expansion," *Operations Research*, vol. 51, pp. 461-371, 2003.

[2] N. Geng, Z. Jiang, and F. Chen, "Stochastic programming based capacity planning for semiconductor wafer fab with uncertain demand and capacity," *European Journal of Operational Research*, vol. 198, pp. 899-908, 2009.

[3] Y. Feng and S. M. Ryan, "Scenario construction and reduction applied to stochastic power generation expansion planning," *Computers & Operations Research*, vol. 40, pp. 9-23, 2013.

[4] Z.-L. Chen, S. Li, and D. Tirupati, "A scenario-based stochastic programming approach for technology and capacity planning," *Computers & Operations Research*, vol. 29, pp. 781-786, 2002.

[5] S. S. Kara and S. Onut, "A two-stage stochastic and robust programming approach to strategic planning of a reverse supply network: The case of paper recycling," *Expert Systems with Applications*, vol. 37, pp. 6129-6138, 2010.

[6] M. K. Zanjani, M. Nourelfath, and D. Ait-Kadi, "A multi-stage stochastic programming approach for production planning with uncertainty in the quality of raw materials

and demand," *International Journal of Production Research*, vol. 48, pp. 4701-4713, 2010.

[7] S. Vigerske and I. Nowak, "Adaptive discretization of convex multistage stochastic programs," *Mathematical Methods of Operations Research*, vol. 65, pp. 361-383, 2007.

[8] M. S. Casey and S. Sen, "The scenario generation algorithm for multistage stochastic linear programming," *Mathematics of Operations Research*, vol. 30, pp. 615-631, 2005.

[9] L. F. Escudero, M. A. Garín, M. Merino, and G. Pérez, "On BFC-MSMIP strategies for scenario cluster partitioning, and twin node family branching selection and bounding for multistage stochastic mixed integer programming," *Computers & Operations Research,* vol. 37, pp. 738-753, 2010.

[10] C.-H. Wu and Y.-T. Chuang, "An efficient algorithm for stochastic capacity portfolio planning problems," *Journal of Intelligent Manufacturing,* vol. 23, pp. 2161-2170, 2012.

[11] J. R. Jiao, Y. Zhang, and Y. Wang, "A heuristic genetic algorithm for product portfolio planning," *Computers & Operations Research,* vol. 34, pp. 1777-1799, 2007.

[12] Y.-Y. Chen and J. T. Lin, "A modified particle swarm optimization for production planning problems in the TFT Array process," *Expert Systems with Applications,* vol. 36, pp. 12264-12271, 2009.

[13] C. Liu, S. Yang, "A serial insertion schedule generation scheme for resource-constrained project scheduling," *Journal of Computers,* vol. 6, pp. 2365-2375, 2011.

[14] T.-L. Chen and H.-C. Lu, "Stochastic multi-site capacity planning of TFT-LCD manufacturing using expected shadow-price based decomposition," *Applied Mathematical Modelling,* vol. 36, pp. 5901-5919, 2012.

[15] S. Kébé, N. Sbihi, and B. Penz, "A Lagrangean heuristic for a two-echelon storage capacitated lot-sizing problem," *Journal of Intelligent Manufacturing,* vol. 23, pp. 2477-2483, 2012.

[16] P.-F. Tsai, "A label correcting algorithm for partial disassembly sequences in the production planning for end-of-life products," *Mathematical Problems in Engineering,* vol. 2012, p. Article ID 569429, 2012.

[17] D. Shen, Y. Li, "Multimodal optimization using crowding differential evolution with spatially neighbors best search," *Journal of Software,* vol. 8, pp. 932-938, 2013.

[18] W. Liang, L. Zhang, M. Wang, "The Chaos differential evolution optimization algorithm and its application to support vector regression machine," *Journal of Software,* vol. 6, pp. 1297-1304, 2011.

[19] C. Deng, C. Liang, B. Zhao, et al., "Structure-encoding differential evolution for integer programming," *Journal of Software,* vol. 6, pp. 140-147, 2011.

[20] L. Peng, Y. Wang, D. Dai, et al., "A novel differential evolution with uniform design for continuous global optimization," *Journal of Computers,* vol. 7, pp. 3-10, 2012.

[21] S. Martello and P. Toth, *Knapsack problems*. West Sussex, England: John Wiley & Sons Press, 2001.

[22] W. Gong, Z. Cai, "Adaptive parameter selection for strategy adaptation in differential evolution for continuous optimization," *Journal of Computers,* vol. 7, pp. 672-679, 2012.

[23] R. L. Haupt and S. E. Haupt, *Practical genetic algorithms*. New Jersey, USA: John Wiley & Sons Inc., 2004.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceeding of IEEE International Conference on Neural Network*, Perth, Western Australia, 1995.

[25] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," presented at the 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995.

[26] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*. Berlin, Germany: Spring-Verlag Inc., 2005.

Yu Zhou is currently a Ph.D. candidate at College of Information Systems and Management in National University of Defense Technology, China. Her main research interests include intelligent optimization algorithm and combinatorial optimization.

Jiang Jiang is an associate professor at college of information system and management, National University of Defense Technology, China. His research interests span the areas of knowledge-based systems, expert system and its applications.