Disassembly Sequencing Problems with Stochastic Processing Time using Simplified Swarm Optimization

Wei-Chang Yeh, Chen-Min Lin, and Shang-Chia Wei

***Abstract*—With eco-awareness and eco-regulation, the end-of-life (EOL) disassembly sequencing problem (DSP) has become increasingly important in the process of recycling, reclamation, or remanufacturing EOL products. However, most of the studies in the disassembly sequencing plan assume that the processing time is deterministic for each part's disassembling procedure. For a realistic and logical approach to the DSP, we take account of the stochastic concept and redefine each part's disassembling time as an exponentially distributed function. Considering the NP-complete nature of the stochastic problem, this paper presents a revised simplified swarm optimization (SSO) to minimize the total expected disassembling time for the stochastic disassembly sequencing problem (SDSP). The results presented in the study show that the SSO is qualified to effectively schedule the SDSP under limited computation time.**

***Index Terms*—End-of-Life (EOL), exponentially distributed function, simplified swarm optimization (SSO), Precedence Preservative Operator (PPO), stochastic disassembly sequencing problem (SDSP), Group updating mechanism (Group UM).**

I. Introduction

The disassembly scheduling of an end-of-life (EOL) product, first proposed in 1994 [1], is an eco-friendly research topic. The great variability in the lifespan of products makes the retrieval of different kinds of products difficult, automation is hard to achieve, and using manual tools is still the main processing technique in current disassembly environments. Hence, the necessity of this labour intensive approach causes a more significant variance in processing time than more automated industries. Most previous researches, however, assume that processing times are constant and known. In this paper, we assume that the processing time for each part is a random variable which is independent of other variables and follows exponential distribution.

In this paper, we present a heuristic method called PPO-SSO to solve stochastic disassembly sequencing problems (SDSP). In SDSP, the processing time of each component is a random variable which follows certain distribution, and it is assumed to be an independent exponential distribution random variable in this paper. In place of the exhaustive search algorithms which dramatically increase the cost and time with the amount of component growing, PPO-SSO has proved to be a powerful and efficient methodology for dealing with combinatorial problems with precedence relationship constraints. In addition to the proposed PPO-SSO, a revised SSO updating mechanism called Group-UM is also proposed to deal with SDSP.

The paper is organized as follows: A brief literature review is presented in Section 2. In Section 3, we frame the SDSP model. The proposed PPO-SSO and the Group updating mechanism which revises the PPO-SSO are introduced in Section 4. A numerical experiment and its results are presented in Section 5 and the conclusion of this study is given in Section 6.

Manuscript received March 29, 2012; revised April 27, 2012

Authors are with the Department of Industrial Engineering and Engineering Management, National Tsing Hua University (e-mail: yeh@ieee.org).

II. Literature Review

In this section, we discuss the various literatures on disassembly sequencing planning and the methodology we use－PPO-SSO.

A. *Disassembly Sequencing Problems*

For the effective recycling, reclamation, and remanufacturing of EOL products, Gupta and Taleb[1,2] first addressed realistic disassembly sequencing problems (DSP) with well-defined product structures. Subsequently, the DSP has been formulated from many perspectives, such as disassembly precedence and geometric constraints [3, 4], the disassembling process and plan [5-7], and design for disassembly [8,9]. Many kinds of methodology have been proposed to solve the DSP: for example, the Petri net (PN)-based approach [4], Branch and Bound [10], and greedy algorithm [11]. Recently, Soft Computing such as ant colony optimization [12], genetic algorithm [13], and simplified swarm optimization [14] have also been implemented in DSPs.

In these above-mentioned studies, the disassembly processing times of components are assumed to be constant and known. Considering the stochastic control in DSP, Tang et al. [15] built a fuzzy attributed Petri net (FAPN) model which represented these mathematic uncertainties in disassembly and developed an algorithm for the model. Yeh [16] considered the dependence and learning effect of the processing time of each part and executed SSO to optimize the problem. In scheduling problems, many literatures [17-21] have assumed the processing times to be independent exponential distribution random variables, which is more practical in many real-world applications.

B. *Simplified Swarm Optimization*

Simplified swarm optimization (SSO), an emerging swarm intelligence stochastic optimization method, was proposed by Yeh [22] to overcome the drawback of particle swarm optimization (PSO), especially for discrete combinatorial optimization problems. To improve the efficiency and feasibility of SSO in DSP, Yeh [14] modified SSO by combining the feasible solution generator (FSG), self-adaptive parameter control (SPC), and precedence preservative operator (PPO) [23, 24]. The FSG increases the efficiency of initializing the population to satisfy the given precedent rules; the SPC is proposed to dynamically adjust the updating parameters of SSO, which may improve effective and efficient exploration; and the PPO is integrated with SSO to generate a new feasible solution that preserves precedence relationships. Recently, Yeh [16] proposed a learning effect DSP (LDSP) model that considers the learning effect of processing time in DSP, and revised the published SSO by improving the update mechanism and modifying SPC to solve the LDSP.

III. SDSP Formulation

The stochastic disassembly sequencing problem (SDSP) involves the stochastic property of the processing time in the proposed DSP model [7, 9].A classical DSP model has *n* components arranged in a sequence (*X**) to disassemble without overlapping and idle time between them. In SDSP, we regard the basic processing time as the main effect on the SDSP variance so the basic processing time of each component j is assumed to be an independent exponential distribution random variable with rate λ_i (i.e., the population mean of the processing time of part j at time t is $1/\lambda_i$) [14], in which we let t_j^s be the basic processing time in sample s for component j expressed as t_j^s ~ exp (λ_i). Because of the stochastic property, the objective function is performed as an expected value of the total disassembly time which can be derived with Eq. (1).

$$F(X)=E[T(X,B_{1,j})+\sum_{k=2}^{n}T(X,B_{k,j})]+\sum_{k=2}^{n}T(X,D_{k,j})+\sum_{k=2}^{n}T(X,M_{k,j})$$
$$=\frac{\sum_{s=1}^{S}[T_s(X,B_{1,j})+\sum_{k=2}^{n}T_s(X,B_{k,j})]}{S}+\sum_{k=2}^{n}T(X,D_{k,j})+\sum_{k=2}^{n}T(X,M_{k,j}) \quad (1)$$

Where

$$T_s(X,B_{k,j})=\begin{cases}0 & d_{k-1,j}=d_{k,j}=2 \text{ and } m_{k-1,j}=m_{k,j}\\ t_j^s\sim\exp(\lambda_j) & \text{otherwise}\end{cases} \quad (2)$$

$$T(X,D_{k,j})=\begin{cases}0 & d_{k-1,j}=d_{k,j}=2 \text{ and } m_{k-1,j}=m_{k,j} \text{ or no direction change}\\ 1 & 90^\circ \text{ direction change, e.g., } +x \text{ to} +y, -x \text{ to} +z\\ 2 & 180^\circ \text{ direction change, e.g., } +x \text{ to -}x, \text{-}x \text{ to} +x\end{cases} \quad (3)$$

$$T(X,M_{k,j})=\begin{cases}0 & d_{k-1,j}=d_{k,j}=2 \text{ and } m_{k-1,j}=m_{k,j} \text{ or used the same disassembly method}\\ 1 & \text{the disassembly method is changed}\end{cases} \quad (4)$$

$$d_{k,j}=\begin{cases}0 & \text{if component j is not demanded}\\ 1 & \text{if component j is demanded for reuse}\\ 2 & \text{if component j is demanded for recycling}\end{cases} \quad (5)$$

$$m_{k,j}=\begin{cases}A & \text{if component j is made of aluminum}\\ P & \text{if component j is made of plastic}\\ S & \text{if component j is made of steel}\end{cases} \quad (6)$$

In sequence X, we obtain S numbers of $T_s(X,B_{k,j})$ such as Eq. (2), the basic processing time (in seconds) of component *j* in position *k* of sample *s*. $T_s(X,B_{k,j})$ follows an exponential distribution with expected processing time $1/\lambda_i$. In Eq. (3) and (4), the penalty $T(X,D_{k,j})$ and $T(X,M_{k,j})$ we assume to be constant are caused by the neighbours' demand and method relationships based on the condition that the components in positions k-1 and k are not made of the same material (i.e., $m_{k-1,j}\neq m_{k,j}$) or one of them is not for recycling (i.e., $d_{k-1,j}\neq 2 \text{ or } d_{k,j}\neq 2$). Besides, $d_{k,j}$ in Eq. (5) denotes the type of demand of component *j* in position *k* (0: not demanded; 1: demanded for reuse; 2: demanded for recycling) and $m_{k,j}$ in Eq. (6) is the material type of the component *j* in position *k* (*A*: aluminium; *P*: plastic; *S*: steel). In practical, there has an extraordinary condition, in which when components in positions k-1 and k are both demanded for recycling (i.e., $d_{k-1,j}=d_{k,j}=2$) and are made the same material (i.e., $m_{k-1,j}=m_{k,j}$), this two components can be recycled without disassembly so $T_s(X,B_{k,j})$, $T(X,D_{k,j})$ and $T(X,M_{k,j})$ are all equal to zero. To sum up: the objective function of SDSP is constructed in two parts that include the expected processing time and the constant penalty from the change of disassembly directions, the disassembly methods, the demand after disassembling and the material of the component and its neighbours. Notably, there is a priority problem to consider in disassembly sequencing. Fig. 1 gives an EOL product structure as an example, in which the precedence relationships are as follows: component 1 or 2 must be disassembled prior to others; component 7 must be disassembled prior to components 6 and 3 and the components of their lower layers; component 6 must be disassembled prior to components 4 and 5.

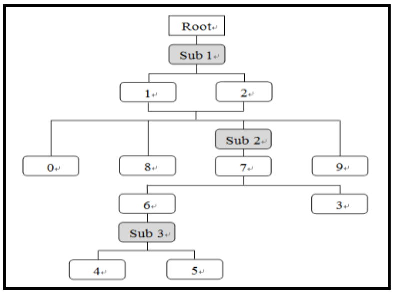

Fig. 1. Example of EOL product structure.

IV. The PPO-SSO with FSG

In this section, the SSO-PPO will be introduced and a revised updating mechanism (UM) of SSO called Group-UM will be proposed. The PPO-SSO preserves the precedence relationships of SDSP which was proved effective in increasing the efficiency of SSO by avoiding the repetition of adjusting infeasible solutions. In addition, the proposed Group-UM increases the global search ability by revising the updating mechanism.

A. PPO-SSO

In SSO, a group of solutions form a population and the amount of solutions is called population size. The selected variable value of each new solution is generated from the current solution, *pBest*, *gBest*, or a random solution. The *pBest* solution is that which has the best function value within its own iterative solutions and the *gBest* solution is that which has the best function value among all solutions. In DSP, the value of x_{id}^t expresses the component number at *d*th position of *i*th solution. To conquer the precedence relationship constraint, Yeh proposed the PPO-SSO [9], which integrates the above-mentioned mechanism with the PPO proposed for the genetic algorithm [7] to preserve the precedence relationship. The revised model is as follow:

$$x_{id}^{t} = \begin{cases} L(X_i^{t-1}) & \text{if } \rho_d \in [0, C_w) \\ L(P_i^{t-1}) & \text{if } \rho_d \in [C_w, C_p) \\ L(G_i) & \text{if } \rho_d \in [C_p, C_g) \\ L(X) & \text{if } \rho_d \in [C_g, 1) \end{cases} \quad (7)$$

Where $C_w = c_w$, $C_p = C_w + c_p$, and $C_g = C_p + c_g = 1 - c_r$ are adjusted dynamically based on the SPC proposed by Yeh [9]; c_w, c_p, c_g, and c_r are the probabilities of the new variable at generation t generated from the current solution, *pBest*, *gBest*, and a random solution in SSO, respectively. X_i^{t-1} is the *i*th solution, P_i^{t-1} is the *pBest* of *i*th solution at generation *t-1*; G_i is the *gBest* before generation *t* and *X* is a random solution. ρ_d is a random number between [0, 1] generated to determine the component at *j*th position. L(•) is the value of the left-most variable in the •solution. See the precedence relationships in Fig.1 as an example, in which it is assumed that the current, *pBest*, *gBest*, and the random solutions at generation *t* are X_i^{t-1} (1708264539), P_i^{t-1} (2819736045), G_i (2793614508) and *X* (1927065843). Each number in parentheses is the component number arranged in the sequence position. The procedure of this example is listed in Table 1. We set the parameters as constant where $C_w = 0.15$, $C_p = 0.4$, and $C_g = 0.75$. In position 1 (*d*=1), since the random number $C_p = 0.4 < \rho_1 = 0.581788 < C_g = 0.75$, let the left-most variable 2 of G_i be the first variable in X_i^t (i.e., $X_i^t = 2$) and delete 2 from all the solutions. In dimension 2 (*d*=2), since the random number $\rho_2 = 0.05323 < C_w = 0.15$, the left-most variable 1of X_i^{t-1} is selected as the second variable and deleted from all the solutions. Proceeding in the same way, X_i^t (2178936405) is generated and must conform to the precedence relationships. The complete precedence is showed in Table I.

TABLE I: EXAMPLE OF THE PPO-SSO

d	1	2	3	4	5	6	7	8	9	10
ρ_d	0.581788	0.05323	0.637805	0.315322	0.868324	0.643801	0.313155	0.418506	0.168064	0.901222
X_i^{t-1}	(1708264539)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)	(~~1708264539~~)
P_i^{t-1}	(2819736045)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)	(~~2819736045~~)
G_i	(2793614508)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)	(~~2793614508~~)
X	(1927065843)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)	(~~1927065843~~)
X_i^t	(2---------)	(21--------)	(217-------)	(2178------)	(21789-----)	(217893----)	(2178936---)	(21789364--)	(217893640-)	(2178936405)

B. Novelty of the Proposed SSO

The new updating mechanism (UM), called Group-UM, is proposed by revising the updating mechanism of the proposed PPO-SSO. In Group-UM, the population is first divided into N_{gro} groups with equal solutions. Within each group, the solution which has the best function value is chosen as the *grBest*. Second, the total generations (*GEN*) are segmented into two parts with given G_{seg} and *GEN*-G_{seg} generations respectively. In the first G_{seg} generations, the role of *gBest* is replaced by *grBest*. The novel model is:

$$x_{id}^{t} = \begin{cases} L(X_i^{t-1}) & \text{if } \rho_d \in [0, C_w) \\ L(P_i^{t-1}) & \text{if } \rho_d \in [C_w, C_p) \\ L(Gr_i) & \text{if } \rho_d \in [C_p, C_g) \text{ and } t \le G_{seg} \\ L(G_i) & \text{if } \rho_d \in [C_p, C_g) \text{ and } t > G_{seg} \\ L(X) & \text{if } \rho_d \in [C_g, 1) \end{cases} \quad (11)$$

where Gr_i is the *grBest* solution of solution *i*. The reason for replacing *gBest* with *grBest* in the first G_{seg} generations is that it can expand the search space to search more possible solutions and increase the diversity of the population to avoid premature convergence. After an extensive search, the *gBest* replaces the role of *grBest* in the UM after the G_{seg} th generation to accelerate the convergence behaviour of the solutions. The implementation steps of PPO-SSO with Group-UM are shown in Fig. 2 and the shaded parts indicate the main changes to the proposed Group-UM.

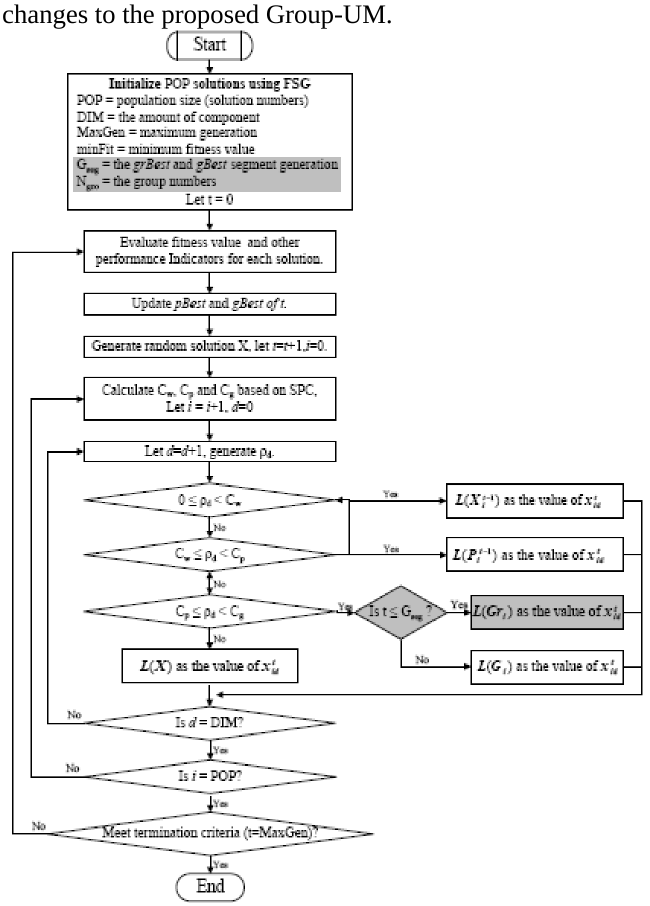

Fig. 2. The flowchart of the PPO-SSO with Group UM.

V. Numerical Results

In this study, the performances of the original UM and the new Group-UM of the proposed PPO-SSO are demonstrated and compared by using ten examples (Ex.1-10) from the literatures [13, 14], in which the data sets and the precedence relationships were presented. There are 10-component set in Ex. 1 [13] and 13-component set in Ex.2-10 [14]. In the SDSP, the original constant $T(B_k)$ in the data sets is the population mean of processing times which are random variables that independently follow exponential distributions.

In the Group-UM, G_{seg} and N_{gro} are two factors, where G_{seg} means the first G_{seg} generations taking the *grBest* over the *gBest* and N_{gro} is the number of groups which are equal to the population size divided by the group size. Given the population size of 40 and 300 generations, the G_{seg} has three levels of 100, 150 and 200 (expressed as *G*1, *G*2, and *G*3 respectively), and the N_{gro} has three levels of 5, 8, and 10 (expressed as *N*1, *N*2, and *N*3 respectively). Hence, there are nine combinations: *G*1*N*1, *G*1*N*2, *G*1*N*3, *G*2*N*1, *G*2*N*2, *G*2*N*3, *G*3*N*1, *G*3*N*2, and *G*3*N*3. The experiment results of the above combinations including the fitness function, standard deviation and computation time, are shown in Tables 2, 3 and 4 respectively. The values in Tables 2-4 are the original UM performance values divided by the corresponding performance values of each group combination. If the value of the combination in these tables is larger than 1, it means the performance of the Group-UM based on the combination setting is better than the performance of original UM, and vice versa. The averages of each G_{seg} and N_{gro} from Tables 2-4 are calculated in Table 5. The statistical analyses of ANOVA results for these performances are shown in Tables 6-8, and Figs. 3-8 illustrate the main effects and interaction plots. According to the results, we observe the following:

The effectiveness of the Group UMs: The fitness functions of the Group combinations are better than the fitness functions of original UM on average in Table 2, so the Group UMs have the ability to improve the effectiveness of SSO in SDSP. More importantly, we can observe from Table 3 that Group UMs are clearly more robust. The Group UMs in Table 4 are less efficient, which is caused by the additional procedure of calculating the *grBest*.

The main effect of G_{seg} and N_{gro} : In Table 5, the *G*3 is more effective and robust than *G*2 and *G*1; there is slight difference between the performances of *N*1, *N*2, and *N*3. From the ANOVA results of the fitness value, N_{gro} has a significant effect on the fitness function in Table 6 and the standard deviation is affected significantly by G_{seg} in Table 7. Fig. 3 illustrates the main effects of G_{seg} and N_{gro} to fitness value and the results are that *T*3 and *N*3 perform better. The main effects G_{seg} and N_{gro} standard deviation are shown in Fig. 4, in which *G*3 and N1 are the most stable.

The interaction of G_{seg} and N_{gro} : From the ANOVA (see Tables 6 and 7), we observe that the interaction of G_{seg} and N_{gro} are both significant, which means that the performance of G_{seg} will be changed in different N_{gro} . Figs. 5 and 6 illustrate the interactions of G_{seg} with N_{gro} to the fitness values and standard deviations. From the effectiveness respect in Fig. 5, *G*1*N*2 is slightly better than all combinations with *G*1, *G*2*N*3 is the best combination with *G*2, *G*3*N*3 is the best combination with *G*3. Fig. 6 illustrates the interaction for SD that *G*1*N*1 is better than all combinations with *G*1, *G*2*N*3 is the best combination with *G*2, *G*3*N*1 is the best combination with *G*3.

TABLE II: The Value of the Average fitness Function Ratio

Ex	original UM	Group-UM								
		T1N1	T1N2	T1N3	T2N1	T2N2	T2N3	T3N1	T3N2	T3N3
1	27.3484	**1.000 065**	0.998 634	0.999 949	**1.001 929**	**1.005 382**	**1.000 482**	**1.001 439**	**1.004 95**	**1.000 308**
2	29.3575 7	0.998 221	0.992 786	0.995 089	0.994 421	0.997 454	0.998 574	0.993 328	0.994 508	0.997 939
3	35.3292 8	0.992 349	0.991 573	**1.005 124**	0.984 805	0.994 664	**1.002 224**	0.990 788	0.994 848	**1.001 588**
4	28.1498	**1.014 26**	**1.018 317**	**1.021 028**	**1.004 158**	**1.017 44**	**1.017 315**	**1.015 571**	**1.015 857**	**1.014 982**
5	29.6715 6	**1.012 141**	**1.025 805**	**1.023 608**	0.982 889	**1.035 145**	**1.035 864**	**1.022 08**	**1.035 432**	**1.013 216**
6	28.9769 1	**1.007 146**	**1.011 665**	0.996 572	0.996 728	**1.004 002**	**1.002 206**	**1.002 248**	**1.007 046**	**1.014 057**
7	39.7922 3	0.997 532	**1.001 494**	0.994 981	0.996 794	**1.002 05**	**1.006 73**	**1.006 609**	**1.005 829**	0.999 356
8	34.8053	**1.013 371**	**1.019 055**	**1.007 742**	0.983 915	**1.014 177**	**1.006 458**	**1.010 66**	**1.011 799**	**1.028 564**
9	41.9658 8	**1.023 185**	**1.012 336**	**1.005 233**	**1.005 788**	**1.013 033**	**1.021 429**	**1.015 373**	**1.002 355**	**1.007 328**
10	39.5569 1	**1.003 214**	0.996 683	0.998 098	0.999 303	0.996 058	**1.001 621**	0.998 899	**1.000 166**	**1.006 061**
Av g.		**1.006 148**	**1.006 835**	**1.004 742**	0.995 073	**1.007 941**	**1.009 29**	**1.005 7**	**1.007 279**	**1.008 34**

TABLE III: The Value of the Standard Deviation Ratio

Ex	original UM	Group-UM								
		T1N1	T1N2	T1N3	T2N1	T2N2	T2N3	T3N1	T3N2	T3N3
1	0.58945 9	0.959 698	**1.039 158**	0.907 177	**1.360 643**	**1.025 225**	**1.407 196**	**1.183 08**	**1.068 955**	**1.513 867**
2	0.45142 1	**1.011 892**	**1.592 387**	**1.504 225**	**1.376 833**	**1.028 345**	**1.056 88**	**1.591 584**	**1.014 473**	**1.190 453**
3	0.96934 8	**1.071 482**	0.887 953	**1.134 655**	0.910 135	**1.081 256**	**1.384 075**	**1.152 36**	**1.005 682**	**1.030 723**
4	0.97694 7	**1.702 719**	**1.904 499**	**1.421 38**	**1.184 381**	**1.905 063**	**1.848 482**	**2.049 181**	**1.858 911**	**1.657 182**
5	1.98854 8	**1.748 07**	**1.697 9**	**1.511 62**	0.936 382	**1.566 44**	**1.353 034**	**2.350 666**	**2.060 268**	**1.713 487**
6	0.95420 5	**1.386 062**	**1.240 298**	**1.016 316**	0.928 695	**1.575 693**	**1.772 61**	**1.324 909**	**1.452 049**	**1.723 907**
7	1.23126 2	**1.931 443**	**1.283 171**	**1.383 097**	**1.297 9**	**1.397 852**	**1.276 362**	**1.497 35**	**1.888 459**	**2.007 065**
8	2.25238 6	**1.414 178**	**1.293 808**	**1.222 81**	**1.382 264**	**1.420 539**	**1.325 604**	**1.642 515**	**1.161 252**	**1.374 273**
9	1.75013 9	**1.268 74**	**1.398 565**	0.946 466	0.922 686	0.848 181	**1.269 759**	**1.753 304**	**1.315 044**	**1.072 722**
10	0.53373 3	0.822 55	0.647 907	0.929 829	**1.023 646**	**1.130 503**	0.937 39	**1.292 667**	**1.037 382**	0.666 028
Av g.		**1.331 68**	**1.298 56**	**1.197 76**	**1.132 36**	**1.297 91**	**1.363 14**	**1.583 76**	**1.386 25**	**1.394 97**

To sum up, the PPO-SSO with Group updating mechanism is conducted to deal with the SDSP and is proved to have excellent efficacy and stability. Moreover, in the experiment we executed, the results show that when a larger segment generation parameter (G_{seg}) is adopted, algorithm stability will be strengthened, and effectiveness will be improved when the group numbers (N_{gro}) setting is increased.

TABLE IV: THE VALUE OF THE COMPUTATION TIME RATIO

Ex	original UM	Group-UM								
		T1N1	T1N2	T1N3	T2N1	T2N2	T2N3	T3N1	T3N2	T3N3
1	3.435633	0.934027	0.989222	0.951708	0.955625	0.976781	0.953883	0.88453	0.9234	0.95413
2	4.446	0.96247	0.992573	0.942542	**1.077122**	0.994075	0.967868	0.956115	0.861483	0.945803
3	4.524533	0.96498	0.992585	0.946364	0.888784	0.996447	0.965955	0.969418	0.906242	0.980503
4	3.925467	0.968382	0.998804	0.968574	0.984624	**1.000663**	0.977425	0.957065	0.916544	0.982423
5	4.1574	0.957265	0.991321	0.956538	**1.131173**	0.988007	0.962034	0.935011	0.930074	0.959208
6	4.031567	0.966347	**1.000257**	0.952526	**1.087399**	0.995121	0.963437	0.942535	0.979487	0.967073
7	4.490733	0.966435	0.989577	0.953818	**1.160966**	0.992764	0.964954	0.963683	0.892252	0.963732
8	4.2713	0.967364	0.99024	0.959117	**1.303444**	0.990362	0.964909	0.938067	0.924851	0.991074
9	4.324833	0.962214	0.98894	0.951552	**1.259954**	0.989876	0.965271	0.828798	0.920145	0.910689
10	4.6223	0.969808	0.99363	0.956213	**1.234797**	0.997647	0.970548	0.883906	0.937586	0.967049
Avg.		0.961929	0.992715	0.953895	**1.108389**	0.992174	0.965628	0.925913	0.919206	0.962169

TABLE V: THE PERFORMANCE RATIO AVERAGES OF G_{seg} AND N_{gro}

Tc	Fit	SD	T	Ng	Fit	SD	T
T1	1.005909	1.276002	0.969513	N1	1.002307	1.349267	0.998744
T2	1.004101	1.264468	1.022064	N2	1.007352	1.327574	0.968032
T3	1.007106	1.454993	0.935763	N3	1.007458	1.318622	0.960564

TABLE VI: ANALYSIS OF VARIANCE FOR FITNESS VALUE

Source of variation	Degree of freedom	Adjusted mean square	F-value	p-value
Ex	9	272.96	5282.87	0
Tc	2	0.07	1.37	0.26
Ng	2	0.24	4.67	**0.012**
Tc*Ng	4	0.22	4.27	**0.004**
Error	72	0.05		
Total	89			
S=0.227306		R-Sq=99.85%,	R-Sq(adj)=99.81%	

TABLE VII: ANALYSIS OF VARIANCE FOR SD

Source of variation	Degree of freedom	Adjusted mean square	F-value	p-value
Ex	9	1.78991	54.29	0
Tc	2	0.16469	5	**0.009**
Ng	2	0.00302	0.09	0.913
Tc*Ng	4	0.10841	3.29	**0.016**
Error	72	0.03297		
Total	89			
S=0.227306		R-Sq=99.85%,	R-Sq(adj)=99.81%	

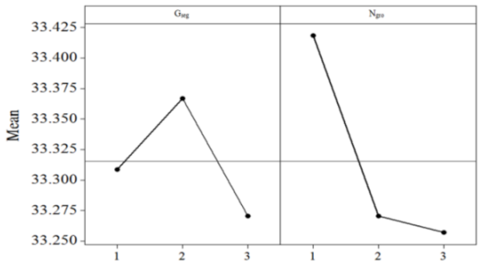

Fig. 3. Main effect in G_{seg} and N_{gro} for fitness value

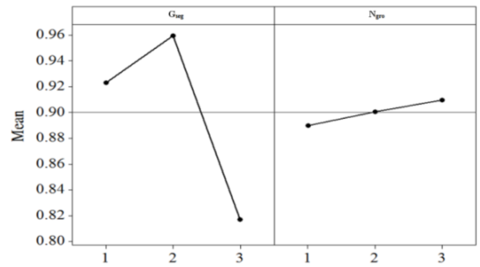

Fig. 4. Main effect in G_{seg} and N_{gro} for SD

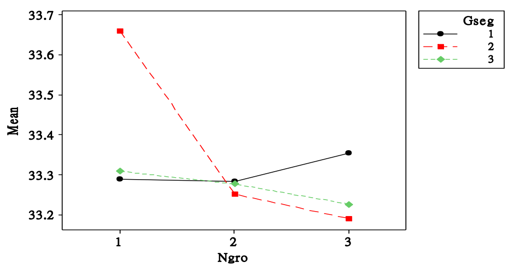

Fig. 5. Interaction for fitness value of G_{seg} and N_{gro}

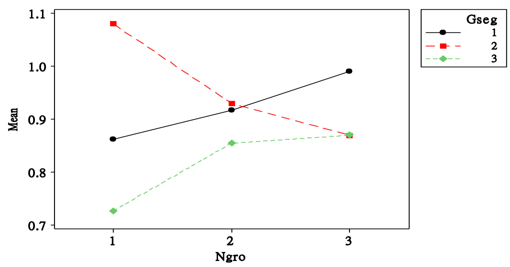

Fig. 6. Interaction for SD of G_{seg} and N_{gro}

VI. CONCLUSION

This paper constructs the SDSP model which considers the stochastic property of the disassembly processing time of the proposed DSP. The model would be close to the real world circumstance of disassembly. The work also proposes a novel SSO by revising the updating mechanism called Group-UM to strengthen the global search ability and enhance the effectiveness of identifying the optimum disassembly sequencing of SDSP. The experiment and statistical results prove the improvement of Group-UM and show the effects of the parameter adjustment for performances such as expected value and standard deviation in SDSP. Future research may study the problem on a larger scale by increasing the numbers of components, or investigating a more general SDSP model such as different distribution or other factors in the model. A different methodology or improvement to SSO could also be proposed to enhance efficiency.

REFERENCES

[1] S. M. Gupta and K. Taleb. Scheduling disassembly. *International Journal of Production* Research.1994, 32 (8): 1857-1866.

[2] K. Taleb and S. M. Gupta. Disassembly of multiple product structures. *Computers and Industrial Engineering*.1997, 32 (4): 949-961.

[3] A. Gungor and S. M. Gupta, Disassembly sequence planning for complete disassembly in product recovery. *Proceedings of the Northeast Decision Sciences Institute Conference, Boston, MA*.1998, 250-252.

[4] K. E. Moore, A. Gungor, and S. M. Gupta, Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships. *European Journal of Operational Research*. 2001, 135(2): 428-449.

[5] L. Brennan, S. M. Gupta, and K. N. Taleb. Operations planning issues in an assembly/disassembly environment. *International Journal of Operations & Production Management*. 1994, 14 (9): 57 – 67.

[6] S. M. Gupta and S. M. McGovern, Multi-objective optimization in disassembly sequencing problems. *Second World Conference on POM and 15th Annual POM Conference, Cancun, Mexico*. 2004, April 30-May 3.

[7] I. M. Langella, Heuristics for demand-driven disassembly planning. *Computers & Operations Research*. 2007, 34 (2): 552-577.

[8] P. Veerakamolmal and S. M., Gupta. Analysis of design efficiency for the disassembly of modular electronic products. *Journal of Electronics Manufacturing (JEM)*. 1999, 9 (1): 79-95.

[9] TC. Kuo, Disassembly sequence and cost analysis for electromechanical products. *Robotics and Computer-Integrated Manufacturing*. 2000, 16(1): 43–54.

[10] A. Gungor and S. M., Gupta. Disassembly sequence plan generation using a branch-and-bound algorithm. *International Journal of Production Research*. 2001, 39(3): 481-509.

[11] S. M. McGovern and S. M.Gupta. Greedy algorithm for disassembly line scheduling. *IEEE International Conference on Systems, Man and Cybernetics*. 2003, 2 (5-8): 1737–1744.

[12] S. M. McGovern and S. M. Gupta. Ant colony optimization for disassembly sequencing with multiple objectives. *International Journal of Advanced Manufacturing Technology*. 2006, 30(5-6):481-96.

[13] E. Kongar and S. M. Gupta. Disassembly sequencing using genetic algorithm. *The International Journal of Advanced Manufacturing Technology*. 2006, 30(5- 6):497-506.

[14] W. C. Yeh. Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization. *IEEE Transactions on Systems, Man and Cybernetics*. 2011, 42 (1): 250-261.

[15] Y. Tang, M. Zhou, and M. Gao. Fuzzy-petri-net-based disassembly planning considering human factors. *EEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*. 2006, 36(4): 718-726.

[16] W.C. Yeh, Simplified swarm optimization in disassembly sequencing problems with learning effects. *Computers &Operations Research*. 2012, 39 (9): 2168-2177.

[17] G. Weiss and P. Michael, Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *Journal of Applied Probability*.1980, 17 (1): 187-202.

[18] J. Bruno, P. Downey, and G. N. Frederickson. Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *Journal of the ACM*. 1981, 28 (1): 100-113.

[19] P. Michael, Stochastic scheduling with release dates and due dates. *Operations Research*. 1983, 31(3):559-572.

[20] A. K. Agrawala, E. G. Coffman, Jr., M. R. Garey, and S. K. Tripathi, A stochastic optimization algorithm minimizing expected flow times on uniforn processors. *IEEE Transactions on Computers*. 1984, **33** (4): 351-356.

[21] S. Goren and I. Sabuncuoglu, Optimization of schedule robustness and stability under random machine breakdowns and processing time variability. *IIE Transactions*. 2009, 42 (3): 203-220.

[22] W. C. Yeh, A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems. *Expert Systems with Applications*, 2009, 36 (5): 9192-9200.

[23] C. Bierwirth, D. C. Mattfeld, and H. Kopfer, On permutation representations for scheduling problems. In: Voigt HM, et al.(eds.). *Parallel problem solving from nature-PPSN IV*. Springer, Berlin Heidelberg New York. 1996, pp. 310-318.

[24] C. Bierwirth and D. C. Mattfeld, Production scheduling and rescheduling with genetic algorithms. *Evolutionary computation*. 1999, 7(1): 1-18.