*International Journal of Production Research*, 2015
http://dx.doi.org/10.1080/00207543.2015.1041575

Taylor & Francis
Taylor & Francis Group

# A hybrid differential evolution and estimation of distribution algorithm based on neighbourhood search for job shop scheduling problems

Fuqing Zhao[a,b]*, Zhongshi Shao[a], Junbiao Wang[b] and Chuck Zhang[c]

[a]*School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou, China;* [b]*Key Laboratory of Contemporary Design & Integrated Manufacturing Technology, Ministry of Education, Northwestern Polytechnical University, Xi'an, China;* [c]*H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA*

(*Received 24 August 2014; accepted 2 April 2015*)

Job shop scheduling problem (JSSP) is a typical NP-hard problem. In order to improve the solving efficiency for JSSP, a hybrid differential evolution and estimation of distribution algorithm based on neighbourhood search is proposed in this paper, which combines the merits of Estimation of distribution algorithm and Differential evolution (DE). Meanwhile, to strengthen the searching ability of the proposed algorithm, a chaotic strategy is introduced to update the parameters of DE. Two mutation operators are adopted. A neighbourhood search (NS) algorithm based on blocks on critical path is used to further improve the solution quality. Finally, the parametric sensitivity of the proposed algorithm has been analysed based on the Taguchi method of design of experiment. The proposed algorithm was tested through a set of typical benchmark problems of JSSP. The results demonstrated the effectiveness of the proposed algorithm for solving JSSP.

**Keywords:** estimation of distribution algorithm; differential evolution algorithm; neighbourhood search; hybrid optimisation; job shop scheduling



## 1. Introduction

The job shop scheduling problem (JSSP) is one of the most complicated combinatorial optimisation problems (Garey, Johnson, and Sethi 1976). In JSSP, there are $n$ ($n \geq 1$) jobs to be processed through $m$ ($m \geq 1$) machines. Each job must pass through each machine only once. Each job could be processed through the machines in a special order, and there are no precedence constraints among different jobs. Each machine can process at most one operation at the same time. There is no set-up or tardiness cost. Each machine has full efficiency. Moreover, the processing time of each operation is fixed and known. In this paper, the minimisation of makespan is selected as the objective, which is to find a schedule to minimise the time required to complete all jobs.

JSSP has been demonstrated to be a strongly NP-hard problem (Garey, Johnson, and Sethi 1976), and it cannot be exactly solved in a reasonable computational time. At first, mathematical methods (Qing-Dao-Er-Ji and Wang 2012) were used to solve JSSP, such as shifting bottleneck procedure, branch and bound and dynamic programming. However, due to the complexity of JSSP, it is unrealistic to solve even a medium-sized problem using reasonable time. Thus, others began to use meta-heuristics to solve JSSP. These methods (Gao et al. 2011) mainly include taboo search, genetic algorithm, simulated annealing, ant colony optimisation, artificial immune system, particle swarm optimisation (PSO) and memetic algorithm, etc. Compared with the mathematical methods, the meta-heuristics make better trade-off between solution quality and computational time, which can be relatively easy to implement and conveniently be adapted to different kinds of scheduling problems. However, the single meta-heuristic neither has an effective local search mechanism for accurately searching near solutions nor global exploration to fully explore the solution space. In order to overcome these drawbacks, estimation of distribution algorithm (EDA) based on global information search and differential evolution (DE) algorithm based on local information search will be incorporated to solve JSSP in this paper.

EDA is a new evolutionary search method. EDA explicitly learns and builds a probabilistic model to capture the parental distribution. Offspring individuals are sampled from the probabilistic model (Hauschild and Pelikan 2011). EDA is good at the automatic discovery and exploitation of problem regularities. A population evolves with intensive communication among all of the promising individuals, readily extracts promising region of search space (Pelikan, Goldberg, and Lobo 2002). Therefore, EDA has strong global exploration, it can be considered as a strongly cooperative

*Corresponding author. Email: fzhao2000@hotmail.com

search method. EDA has been combined with other optimisation algorithms to solve the practical problems (Bai et al. 2012; Ou-Yang and Utamima 2013).

DE algorithm is an optimisation algorithm based on the theory of swarm intelligence (Das and Suganthan 2011). The operations of DE algorithm include mutation, crossover and selection (Das and Suganthan 2011). It solves the optimum problems through competition and cooperation of the population. DE algorithm is easily carried out, has good local search capability, memories the best solution of all individuals and shares the internal information of the population. At present, many improvement measures have been applied to enhance the search ability of DE (Basu 2014; Singh and Srivastava 2014).

In this research, a hybrid differential evolution and estimation of distribution algorithm based on neighbourhood search is proposed (NS-HDE/EDA) to solve JSSP. In NS-HDE/EDA, EDA and DE are selected as two suboptimisation algorithms. Two mutation operators are employed as mutation strategy. The chaotic parameter control (CPC) strategy of reference Wang, Li, and Lai (2009) is employed to enhance the performance of DE. Meanwhile, a neighbourhood search (NS) based on blocks on critical path is proposed to enhance the search ability of the proposed algorithm.

The remainder of the paper is organised as follows. Section 2 reviews recent studies on the JSSP of minimising makespan. Section 3 describes the framework of NS-HDE/EDA. Section 4 discusses the computational study on our algorithm and the parametric sensitivity of our algorithm. Finally, conclusions and future research are drawn in Section 5.



## 2. Literature review

The JSSP is among the hardest combinatorial problems. Not only it is complicated, but it is one of the worst NP-complete class members. Variety of literatures discusses JSSP of minimising makespan with meta-heuristic algorithms which include Tabu search (TS, Ferdinando and Emanuela 1998; Meeran and Morshed 2012), simulated annealing (Elmi, Solimanpur, and Topaloglu 2011), genetic algorithm (Wang 2003), shuffled frog leaping (Cai and Xia 2010), artificial neural networks (Wang and Qi-Di 2007), PSO (Sha and Hsu 2006) and so on.

Recently, many more papers have developed innovative meta-heuristic algorithms for this type of JSSP. For example, Huang and Liao (2008) presented a hybrid ant colony optimisation algorithm with the taboo search algorithm for JSSP, which employs a novel decomposition method inspired by the shifting bottleneck procedure and a mechanism of occasional reoptimisations of partial schedules. Hasan et al. (2009) designed three priority rules, namely partial reordering, gap reduction and restricted swapping, which were used as local search techniques to develop a memetic algorithm. Wong et al. (2010) presented an improved bee colony optimisation algorithm with big valley landscape exploitation (BCBV) as a biologically inspired algorithm to solve the JSSP. The BCBV algorithm simulated the bee foraging behaviour, where information of newly discovered food source is communicated via waggle dances. Pan et al. (2010) proposed a discrete differential evolution (DDE) algorithm with operation-based representation to solve JSSP, which adopts a newly designed mutation and crossover operators. Within this algorithm, an effective local search based on the newly obtained neighbourhoods is presented and imbedded in the DDE algorithm to enhance the local search ability. He et al. (2010) proposed an EDA with probability model based on permutation information of neighbouring operations to solve this classical JSSP. Seo and Kim (2010) proposed an ant colony optimisation algorithm with parameterised search space for job shop problem. The problem is modelled as a disjunctive graph where arcs connect only pairs of operations-related rather than all operations are connected in pairs to mitigate the increase of the spatial complexity. Gao et al. (2011) proposed an efficient MA with a novel local search. Within the local search, a systematic change of the neighbourhood is carried out to avoid trapping into local optimal. And two neighbourhood structures are designed by exchanging and inserting based on the critical path. Wang and Tang (2011) proposed an improved adaptive genetic algorithm for solving the JSSP which was inspired from hormone modulation mechanism, and then the adaptive crossover probability and adaptive mutation probability are designed. Sels, Craeymeersch, and Vanhoucke (2011) presented a genetic algorithm and a scatter search procedure to solve JSSP. The scatter search algorithm splits the population of solutions in a diverse and high-quality set to exchange information between individuals in a controlled way. It has a positive influence on the important balance between intensification and diversification. Qing-Dao-Er-Ji and Wang (2012) proposed a new hybrid genetic algorithm to solve JSSP, who designed some genetic operators and a mixed selection operator based on the fitness value and the concentration value to increase the diversity of the population. Banharnsakun, Sirinaovakul, and Achalakul (2012) proposed an effective scheduling method based on Best-so-far Artificial Bee Colony which biases the solution direction towards the Best-so-far solution rather a neighbouring solution and then use the method to solve JSSP. Li (2012) presented an improve quantum genetic algorithm based on the quantum algorithm theory and quantum chromosome coding knowledge as well as the traditional genetic algorithm for JSSP. Gholami and Sotskov (2013) presented an adaptive algorithm with a learning stage for solving the parallel machines job shop problem. A learning stage tends to produce knowledge about a

benchmark of priority dispatching rules allowing a scheduler to improve the quality of a schedule, which may be useful for a similar scheduling problem. Ponsich and Coello Coello (2013) combined DE with TS to solve the JSSP. A competitive neighbourhood is included within the TS with the aim of determining if DE is able to replace the restart features that constitute the main strengths of i-TSAB. Wang and Duan (2014) proposed a hybrid biogeography-based optimisation (HBBO) algorithm for JSSP. Biogeography-based optimisation (BBO) was a new bio-inspired computation method that was based on the science of biogeography. The BBO algorithm adopted two searching mechanisms including migration and mutation to guarantee the candidates in the search space converge to global optimum solution in faster computational time with stable running process. HBBO algorithm combined the chaos theory and 'searching around the optimum' strategy with the basic BBO to speed up the convergence rate and increase the diversity of candidates in the next running step. Chassaing et al. (2014) addressed JSSPs with an objective of minimising makespan while satisfying a number of hard constraints. An efficient GRASP × ELS approach was introduced for solving this problem. Zhao et al. (2014) proposed a chemotaxis-enhanced bacterial foraging optimisation to solve the JSSP. The new approach was based on a new chemotaxis with the DE operator which aims at solving the tumble failure problem in the tumble step and accelerates the convergence speed of the original algorithm. Murovec (2015) focused on the evaluation of moves for the local search of the job shop problem with the makespan criterion, which introduced an alternative evaluation that relied on a surrogate quantity of the move's potential, which was related to, but not strongly coupled with, the bare criterion. Zhao et al. (2015) employed the shuffled complex evolution (SCE) algorithm to solve the job shop problem. Due to the drawbacks of poor solution and lower convergence rate of the basic SCE algorithm, a new strategy was used to change the individual's evolution in the basic SCE algorithm. The strategy makes the new individual closer to best individual in the current population. Peng, Lü, and Cheng (2015) incorporated a TS procedure into the framework of path relinking (PR) to proposed TS/PR algorithm. TS/PR is able to improve the upper bounds for 49 out of the 205 tested instances, and it solves a challenging instance that has remained unsolved for over 20 years.

Here, we only list a partial of recent literature. It was not a surprise with the new techniques emphasis substantial progress was made in recent period, which shall be called as the boom period for the new algorithms to JSSP, some of the most innovating algorithms were formulated. Although JSSP is an open topic and many researchers have studied it, many of these methods usually trap into local solution and could not get the global optimum. Therefore, the research on the JSSP is still an important issue in the field of production scheduling.



## 3. The proposed algorithm for JSSP

### 3.1 *Notations for the proposed algorithm*

Some symbols used mostly through this paper are summarised as follows:

| | |
|---|---|
| $n$ | number of jobs |
| $m$ | number of machines |
| $o_{ij}$ | the $j$ operation of the job $i$ |
| $J_i$ | the $i$th job |
| $M_i$ | the $i$th machine |
| $t$ | the $t$th generation |
| $\boldsymbol{X}_t$ | the population of $t$th iteration |
| $\boldsymbol{X}'_t(i)$ | the $i$th individual of the population of $t$th iteration |
| $\boldsymbol{x}_i$ | the $i$th individual in population |
| $x_{ij}$ | the $j$th dimension of the $i$th individual |
| $\mu_j$ | the mean of the $j$th dimension in current population |
| $\sigma_j$ | the standard of the $j$th dimension in current population |
| $F_t$ | the value of the scale factor of $t$th iteration |
| $CR_t$ | the value of the crossover rate of $t$th iteration |
| $\gamma_t$ | the value of the differential decisive factor of $t$th iteration |
| $\boldsymbol{v}_i^t$ | the $i$th mutation individual in $t$th generation |
| $\boldsymbol{v}_{ij}^t$ | the $j$th dimension of the $i$th mutation individual in $t$th generation |
| $\boldsymbol{u}_{ij}^t$ | the $j$th dimension of the $i$th crossover individual in $t$th generation |
| $rand$ | a real number between 0 and 1 with uniform distribution |
| $\boldsymbol{\pi}$ | the integer conversion sequence of the individual in population |
| $\pi_d^i$ | the $d$th dimension of the $i$th conversion sequence |
| $\text{Block}_i$ | the $i$th critical sub-block |

### 3.2 *Encoding and decoding*

The operation-based representation is employed to encode a schedule in this paper. This encoding method uses permutation generated by all job's number to encode a schedule. For an $n$-job $m$-machine JSSP problem, its coding sequence is made up of job's number with the length of $n \times m$, the sequence shows the order of jobs be processed on the machine and a job's sequence is represented by its job's number. For an encoding sequence [2 1 1 1 3 2 2 3 3], where 1, 2, 3 represents the job's number of job $J_1$, $J_2$, $J_3$, and the order represents the operation would be processed on the given machine. The encoding sequence is described as [$o_{21}$, $o_{11}$, $o_{12}$, $o_{13}$, $o_{31}$, $o_{22}$, $o_{23}$, $o_{32}$, $o_{33}$], where $o_{ij}$ presents the $j$ operation of the job $i$.

NS-HDE/EDA is a continuous space algorithm, while JSSP's solution space is discrete. Thus, to reasonably map continuous space to discrete space, the sequence mapping is adopted. A detailed three-job three-machine JSSP example is used to illustrate this procedure. The three-job three-machine JSSP includes three jobs ($J_1$, $J_2$, $J_3$) and three machines ($M_1$, $M_2$, $M_3$). Since each job must go through each machine and each machine has three jobs, it has 9! possible ways for this type of JSSP to complete all jobs according to the encoding method. The encoding individual must cover all possible ways. As we must cover 9! possible ways, nine random numbers are generated in interval [0, 1] as an individual. Suppose that a feasible individual in the continuous space is [0.965, 0.517, 0.971, 0.957, 0.485, 0.800, 0.142, 0.422, 0.916], as shown in sequence 1 in the Figure 1. The numbers in sequence 1 are sorted from small to large and their position index will be recorded. This feasible individual can be encoded to an integer series [7, 2, 8, 5, 6, 9, 4, 1, 3] as sequence 2 according to their position index. Then according to their values in the integer series, each number in it will be brought into the equation $\pi_d^i = \lceil \phi_d^i / m \rceil$, $\phi_d^i$ is the number in the sequence 2 and $\lceil \bullet \rceil$ is round up to an integer. So it can be transformed to integer series [3, 1, 3, 2, 2, 3, 2, 1, 1], as shown in sequence 3. The sequence 3 is corresponding to an operation sequence 4.

Through scanning the encoding sequence from left to right, a schedule plan can be obtained by decoding. The operation should be shifted to left as compact as possible to decrease the idle time. Since superfluous idle time can be inserted between operations, an encoding sequence can be decoded into an infinite number of schedules. Therefore, active schedule is adopted in this paper, which has been verified and denoted that it contains optimal schedule in a Venn diagram (Pinedo 2008). Meanwhile, only the active schedule is considered in the decoding approach to reduce the search space.

### 3.3 *Estimation of distribution algorithm*

EDA employs explicitly probability distribution in optimisation. EDA emphasises the global exploration and has fast convergence rate, but its local exploitation ability is relatively limited. Therefore, the global exploration ability of EDA is employed in this paper. In EDA, Gaussian distribution is selected as probability model, and the incremental learning method as its updating way. Gaussian distribution builds a probability model by calculating the mean and standard deviation of the population, which reflects the feature of population. The incremental learning method utilises a part of special individuals to update the mean and standard deviation, which enhances the exploitation ability of EDA so that the population is not quickly trapped into local optima. In our EDA, we choose truncation selection method (Heinz and Dirk 1993) to select special individuals. In the truncation selection method, the candidate solutions are ordered by fitness, and some proportion, $p$, (e.g. $p = 1/2$, $1/3$, etc.), of the fittest individuals are selected and reproduced $1/p$ times, then a large population will produced, whose size is great than the original population. The new population will be selected from the above large population randomly; the individuals in this new population are special individuals. To further explain this procedure in detail, we give an example in Figure 2. The procedure of EDA is illustrated as follows:

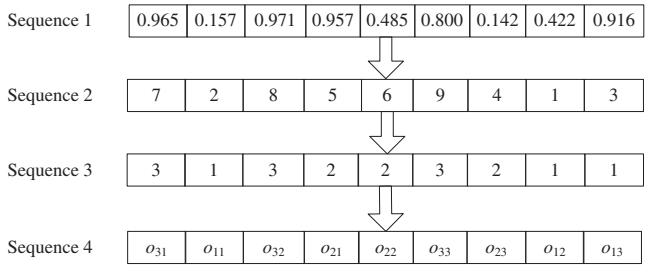


Figure 1. An example of the encoding process.

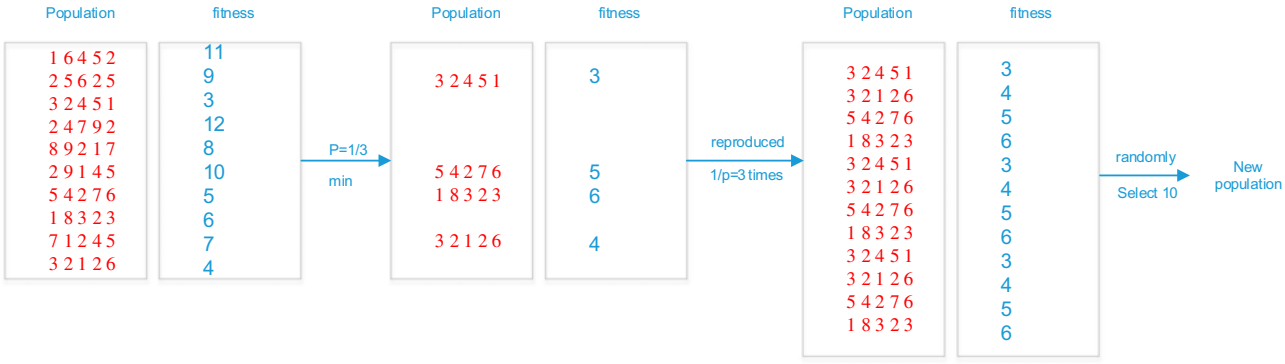


Figure 2. The procedure of truncation selection method.

**Algorithm 1.** (EDA)

**Step 0.** Initialisation.

**Step 0.1.** Randomly initialize the population $\boldsymbol{X}_0$ of size NP.

**Step 0.2.** Set $t = 0$, where $t$ is the current generation.

**Step 1.** Compute the mean $\mu_j$ and standard deviation $\sigma_j$ of the current population $\boldsymbol{X}_t$ with Equation (1).

$$\mu_j = \frac{\sum_{i=1}^{\mathrm{NP}} x_{ij}}{\mathrm{NP}}, \quad \sigma_j = \sqrt{\frac{\sum_{i=1}^{\mathrm{NP}} (x_{ij} - \mu_j)^2}{\mathrm{NP}}} \tag{1}$$

where, $i = 1, 2, \ldots,$ NP, $j = 1, 2, \ldots, D$, $D$ is the number of dimension, NP is the population size and $\boldsymbol{x_i} = (\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}}, \ldots, \boldsymbol{x_{ij}}, \ldots \boldsymbol{x_{iD}})$ is the $i$th individual.

**Step 2.** Use truncation selection method to select a promising population from the current population to compute standard deviation $\sigma_j^k$ with Equation (2).

$$\sigma_j^k = \sqrt{\frac{\sum_{i=1}^{k} (x_{ij} - \mu_j)^2}{\mathrm{NP}}} \tag{2}$$

where $k$ is the number of selection population, $k = \lfloor \eta \times \mathrm{NP} \rfloor$, $\eta$ is the selection probability, $\lfloor \bullet \rfloor$ is rounded down.

**Step 3.** Update $\mu_j$ and $\sigma_j$ with the incremental learning method with Equation (3) and Equation (4), respectively.

$$\mu_j = (1 - \alpha)\mu_j + \alpha(x_{best1,j} + x_{best2,j} - x_{worst,j}) \tag{3}$$

$$\sigma_j = (1 - \alpha)\sigma_j + \alpha\sigma_j^k \tag{4}$$

where $x_{best1}$, $x_{best2}$, $x_{worst}$ is the best, the second best and worst individual in the current population, $\alpha$ is the learning rate.

**Step 4.** Build the probability distribution model $P(\boldsymbol{x})$ with Equation (5).

$$P(\boldsymbol{x}) = \prod_{j=1}^{D} f_N(x_j; \mu_j; \sigma_j) = \prod_{j=1}^{D} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{1}{2}\left(\frac{x_j - \mu_j}{\sigma_j}\right)^2} \tag{5}$$

where $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_j}, \ldots \boldsymbol{x_D})$ is a single individual, $j = 1, 2, \ldots, D$, $D$ is the number of dimension and $N$ is the normal distribution or Gaussian distribution.

**Step 5.** Sample the new population $\boldsymbol{X}_{t+1}$ by the probabilistic model $P(\boldsymbol{x})$.

**Step 6.** Set $t = t + 1$.

**Step 7.** If termination condition is not met, go to Step 1; otherwise end EDA.

In the above EDA, $\mu_j$ and $\sigma_j$ are the parameters of the model. $\mu_j$ restricts centre of the population. $\sigma_j$ controls the diversity of the population and the convergence rate. EDA controls the exploration–exploitation trade-off by the strategies of Equations (3) and (4). The single parent does not jump directly to a desirable location, but rather makes a very small step towards this desirable location. The population will maintain a long-term memory which can provide tiny decision information to the later period of evolution.

### 3.4 *Differential evolution algorithm*

DE algorithm (Das and Suganthan 2011) is an evolutionary algorithm based on vector. DE adopts the similar concept of jump in NS by adding weighted vectors to the target vector to control the evolutionary variation. Therefore, DE has strong exploitation ability and can generate the local information. We select DE as another sub algorithm. In DE algorithm, there are two important parameters that are the scale factor $F$ and the crossover rate $CR$. $F$ decides the scaling of the differential vector produced by differential mutation. It has important influence on the performance of the algorithm, which includes convergence speed and population diversity. $CR$ affects the diversity of population for the next generation. On the one side, if the value of $CR$ becomes larger and larger, the performance of local search ability and the convergence rate tend to become better and better. On the other hand, the smaller value of $CR$ leads to the better performance on maintaining the population diversity and enhancing global search ability. In order to further accelerate the convergence rate and strengthen the search ability of DE, a chaotic strategy named CPC (Wang, Li, and Lai 2009) is introduced. $F$ and $CR$ are updated by Equations (6) and (7), so that $F$ and $CR$ values spread between [0, 1] in the whole region throughout the search process.

At present, there are many versions for DE's mutation operator, which can be expressed by DE/*x*/*y*/*z* (Storn and Price 1997). *x* describes the base vector selection type which includes *rand*, *current*, *best* and so on; *rand* means a randomly chosen population vector; *current* means the base vector comes from the current individual; *best* means the vector of lowest cost from the current population; *y* represents the number of differentials, which is an integer; *z* is the type of crossover operator, which includes *bin* and *exp*; *bin* means a binomial distribution for recombination; *exp* means the exponential distribution for recombination. In our DE algorithm, we selected two classical mutation operators which are shown in Equation (9) DE/*rand*/1 (Das and Suganthan 2011) and Equation (10) DE/*current-to-best*/1 (Bai et al. 2012). DE/*rand*/1 can effectively maintain the diversity of the population. DE/*current-to-best*/1 is used to accelerate the convergence rate. Here, a binomial distribution is selected by crossover operator, so the type of crossover operator is *bin*. The differential decisive factor $\gamma$ is utilised to adjust the proportion of two mutation operators. Such updating way aims at restricting the mutation rate appropriately. The procedure of DE is displayed as follows.

**Algorithm 2.** (DE):
**Step 0.** Initialization.
**Step 0.1.** Randomly initialise the population $\boldsymbol{X}_0$ of size NP.
**Step 0.2.** Set $t = 0$, where $t$ is the current generation.
**Step 0.3.** Initialize $CR_0$ and $\gamma_0$.
**Step 1.** Update parameters.
**Step 1.1.** Update the scale factor $F$.

$$F_{t+1} = 4 \times CR_t \times (1 - CR_t) \tag{6}$$

**Step 1.2.** Update the crossover rate $CR$.

$$CR_{t+1} = 4 \times F_{t+1} \times (1 - F_{t+1}) \tag{7}$$

**Step 1.3.** Update the differential decisive factor $\gamma$.

$$\gamma_{t+1} = 4 \times \gamma_t \times (1 - \gamma_t) \tag{8}$$

for $i$ = 1:NP
**Step 2.** Mutation. If $rand > \frac{\gamma_{t+1}}{2}$,

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + \boldsymbol{F}_{t+1} \times \left(\boldsymbol{x}_{r_2}^t - \boldsymbol{x}_{r_3}^t\right) \tag{9}$$

else

$$\boldsymbol{v}_i^t = (\boldsymbol{F}_{t+1} + \boldsymbol{0.5}) \times \boldsymbol{x}_d^t + (\boldsymbol{F}_{t+1} - \boldsymbol{0.5}) \times \boldsymbol{x}_i^t + \boldsymbol{F}_{t+1} \times (\boldsymbol{x}_b^t - \boldsymbol{x}_c^t) \tag{10}$$

end
where $\boldsymbol{x}_{r_1}^t$, $\boldsymbol{x}_{r_2}^t$, $\boldsymbol{x}_{r_3}^t$ are randomly selected from the current population and $r_1 \neq r_2 \neq r_3 \neq i$. $\boldsymbol{x}_d^t$ is the best solution of the current population, $(\boldsymbol{x}_b^t - \boldsymbol{x}_c^t)$ is a random differential vector with $b \neq c$.
**Step 3.** Crossover.

$$u_{ij}^t = \begin{cases} v_{ij}^t, & \text{if } r \text{ and} < \mathrm{CR}_{t+1} \\ x_{ij}^t, & \text{otherwise} \end{cases} \tag{11}$$

where $\boldsymbol{x}_i^t = \{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{ij}, \ldots, \boldsymbol{x}_{iD}\}$ is the $i$th individual of the current population, $D$ is the number of dimension, $\boldsymbol{u}_i^t = \{\boldsymbol{u}_{i1}^t, \boldsymbol{u}_{i2}^t, \ldots, \boldsymbol{u}_{ij}^t, \ldots, \boldsymbol{u}_{iD}^t\}$ is the crossover individual, $\boldsymbol{v}_i^t = \{\boldsymbol{v}_{i1}^t, \boldsymbol{v}_{i2}^t, \ldots, \boldsymbol{v}_{ij}^t, \ldots, \boldsymbol{v}_{iD}^t\}$ is the $i$th mutation individual.

**Step 4.** Selection.

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t, & \text{if } f(\boldsymbol{u}_i^t) < f(\boldsymbol{x}_i^t) \\ \boldsymbol{x}_i^t, & \text{otherwise} \end{cases} \tag{12}$$

where $f(\bullet)$ is the fitness value.

end for

**Step 5.** If termination condition is not met, let $t = t + 1$, go to Step 1; otherwise end DE.

### 3.5 *The decisive factor*

EDA employs probability distribution to build model, which takes full use of global information to generate offspring and has excellent convergence rate. But poor exploitation ability makes EDA be easily trapped in local optima. DE can use the differential information between vectors (individuals) to jump the local optima and has strong exploitation ability but with low convergence rate. Therefore, EDA and DE are combined by the decisive factor $\zeta$ in this paper. The merits of the above two algorithms are taken for full use. The proposed algorithm balances global exploration and local exploitation.

Through the decisive factor $\zeta$, most of individuals in the population are generated by EDA in the initial evolution. EDA forces the search region to quickly approach a promising area as close as possibly. The minority individuals are generated by DE for maintaining the diversity of the population. These strategies provide the promising area for the later evolution. In the later of evolution, most of individuals in population are generated by DE with the decisive factor $\zeta$ in order to ensure exact search. Nevertheless, as the poor global exploration of DE, EDA generates a little of individuals to maintain the global information and increases their search accuracy. Thus, $\zeta$ is adjusted with Equation (13). When $\zeta$ is larger, EDA plays a leading role. Then $\zeta$ will be gradually lowered in the process of evolution so that DE can occupy dominant position.

$$\zeta_t = e^{-\frac{t}{G_{\max}}} \tag{13}$$

where $t$ is the current iteration, $G_{\max}$ is the maximum value of iteration.

Figure 3 shows that $\zeta$ becomes smaller with increasing evolutionary iteration, and the probability of EDA becomes smaller, while DE becomes larger. Since the population is large, the way of generating individuals by the probability can make the proportion of each algorithm be well in line with statistical regularity. Thus, the proposed algorithm can resist randomness and the search results are more stable.

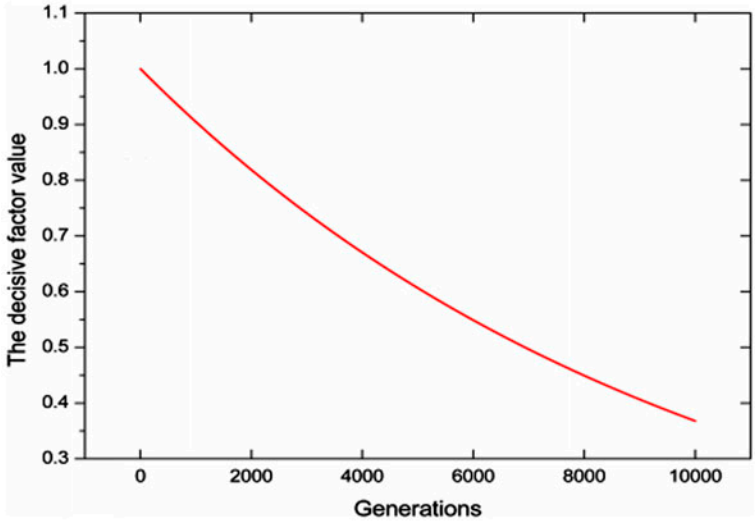


Figure 3. The curve of the decisive factor.

### 3.6 *Local search procedure*

JSSP is a combinatorial optimisation problem and its solution space is limited. But EDA and DE are all continuous algorithms in this paper. According to the encoding scheme, there is a one-to-many relationship between them. So it is insufficient for search ability to simply depend on the hybrid EDA and DE algorithm to solve JSSP. Continuous algorithms do not meticulously search the solution space and one-to-many relationship causes many repeated searching. It consumed a great deal of computational time. Therefore, to reduce the computational time and improve the solution quality, a local search method named a NS based on blocks on critical path is proposed, which is applied to the solutions achieved by the hybrid EDA and DE algorithm. The features of this proposed local search method are discussed in the following subsection.

#### 3.6.1 *Defining neighbourhood*

It is important for the efficiency and the search quality of the algorithm to select the proper neighbourhood structures. Since the neighbourhood based on the block structure of critical path can avoid unnecessary operations and effectively enhance the performance of the algorithms, it is adopted. The disjunctive graph is employed to describe JSSP to illustrate the neighbourhood structures. A brief example of JSSP is introduced, which is shown in Table 1. Figure 4 presents the model of the disjunctive graph of Table 1. The real arcs to precedence relations, the dashed arcs to pairs of operations performed on the same machine.

For a directed disjunctive graph describing feasible schedule, a key component of it is the critical path, which is the longest path from start 0 to end * and its length represents the makespan. The operations on the critical path are called critical operations, which has important influence on the makespan. Once the start of the critical operation is delayed, the makespan inevitably is also delayed. In Figure 5, the length of the critical path is 15 and the critical path is $0 \rightarrow o_{31} \rightarrow o_{21} \rightarrow o_{22} \rightarrow o_{23} \rightarrow o_{13} \rightarrow *$. A maximal sequence of adjacent critical operations that is processed on the same machine is called a critical block. In Figure 4, the critical operations are $\{o_{31}, o_{21}, o_{22}, o_{23}, o_{13}\}$, which include two critical blocks $\{o_{23}, o_{13}\}$,$\{o_{31}, o_{21}\}$.

Since the makespan of a schedule is impossibly less than the length of its critical path, only several critical operations are moved to improve the makespan. The neighbourhood structures based on blocks on critical path are adopted, which includes two neighbourhood structures swap and insert process. As following detailed:

**Neighbourhood 1:** $\boldsymbol{x}$ = swap($\boldsymbol{x}$)

Step 1. Convert the individual $\boldsymbol{x}$ to the sequence $\boldsymbol{\pi}$ according to the encoding scheme.
Step 2. Find the critical path and the critical block **Block** $= \{\text{Block}_1, \text{Block}_2, \ldots, \text{Block}_j\}$ of $\boldsymbol{\pi}$, where $j$ is the number of block, $i = 1$.
Step 3. Swap the two operations $o_n$ and $o_m$ in $\text{Block}_i$ to generate a new sequence $\boldsymbol{\pi}'$.
Step 4. If $f(\boldsymbol{\pi}') < f(\boldsymbol{\pi})$, $\boldsymbol{\pi}_{best} = \boldsymbol{\pi}'$, where $f(\bullet)$ is the function of calculating makespan.
Step 5. Convert $\boldsymbol{\pi}_{best}$ to $\boldsymbol{x}_{best}$. Go back to Step 3, until complete all operation pairs in $\text{Block}_i$, and then go to Step 6.
Step 6. If $i < j$, $i = i + 1$, go back to step 3, otherwise, $\boldsymbol{x} = \boldsymbol{x}_{best}$, return.

**Neighbourhood 2:** $\boldsymbol{x}$ = insert($\boldsymbol{x}$)

Step 1. Convert the individual $\boldsymbol{x}$ to the sequence $\boldsymbol{\pi}$ according to the encoding scheme.
Step 2. Find the critical path and the critical block **Block** $= \{\text{Block}_1, \text{Block}_2, \ldots, \text{Block}_j\}$ of $\boldsymbol{\pi}$, where $j$ is the number of block. $i = 1$.
Step 3. Insert the operation $o_n$ into the front of the operation $o_m$ in $\text{Block}_i$ to generate a new sequence $\boldsymbol{\pi}'$.
Step 4. If $f(\boldsymbol{\pi}') < f(\boldsymbol{\pi})$, $\boldsymbol{\pi}_{best} = \boldsymbol{\pi}'$, where $f(\bullet)$ is the function of makespan.
Step 5. Convert $\boldsymbol{\pi}_{best}$ to $\boldsymbol{x}_{best}$. Go back to Step 3, go back to Step 3, until the all operation in $\text{Block}_i$ are inserted all other possible positions in $\text{Block}_i$, and then go to Step 6.
Step 6. If $i < j$, $i = i + 1$, go back to Step 3. Otherwise, $\boldsymbol{x} = \boldsymbol{x}_{best}$, return.

Table 1. 3 × 3 JSSP problem.

| Job | Machine order | Processing time |
|---|---|---|
| $J_1$ | $M_2$–$M_1$–$M_3$ | 3–4–1 |
| $J_2$ | $M_1$–$M_2$–$M_3$ | 2–3–5 |
| $J_3$ | $M_1$–$M_3$–$M_2$ | 4–2–3 |

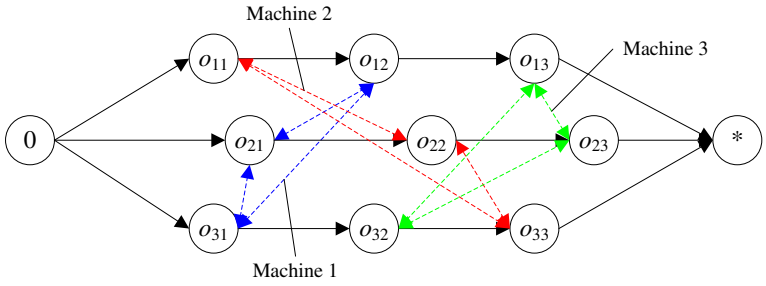


Figure 4. The model of the disjunctive graph in Table 1.

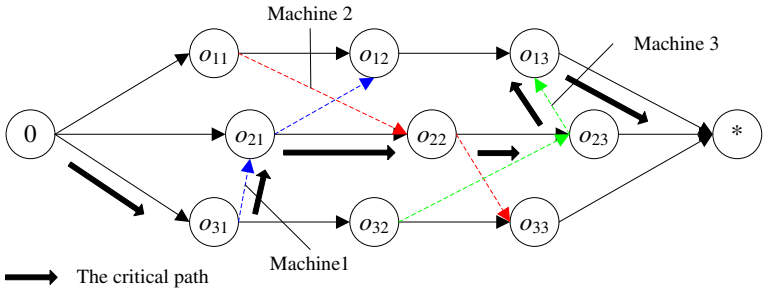


Figure 5. The directed disjunctive graph of the feasible schedule.

Encoding scheme is the mapping from continuous domain to discrete domain, but the above two neighbourhood structures directly handle the operations. Therefore, when the order of the operations is changed, the relevant positions of the individual are also changed to make the individual correspond to the encoding sequence. An example of Table 1 is used to illustrate these processes. The detailed processes are displayed in Figure 6, we suppose the swapped operations are $o_{31}$ and $o_{21}$. In inserting operation, $o_{31}$ is inserted before $o_{21}$.

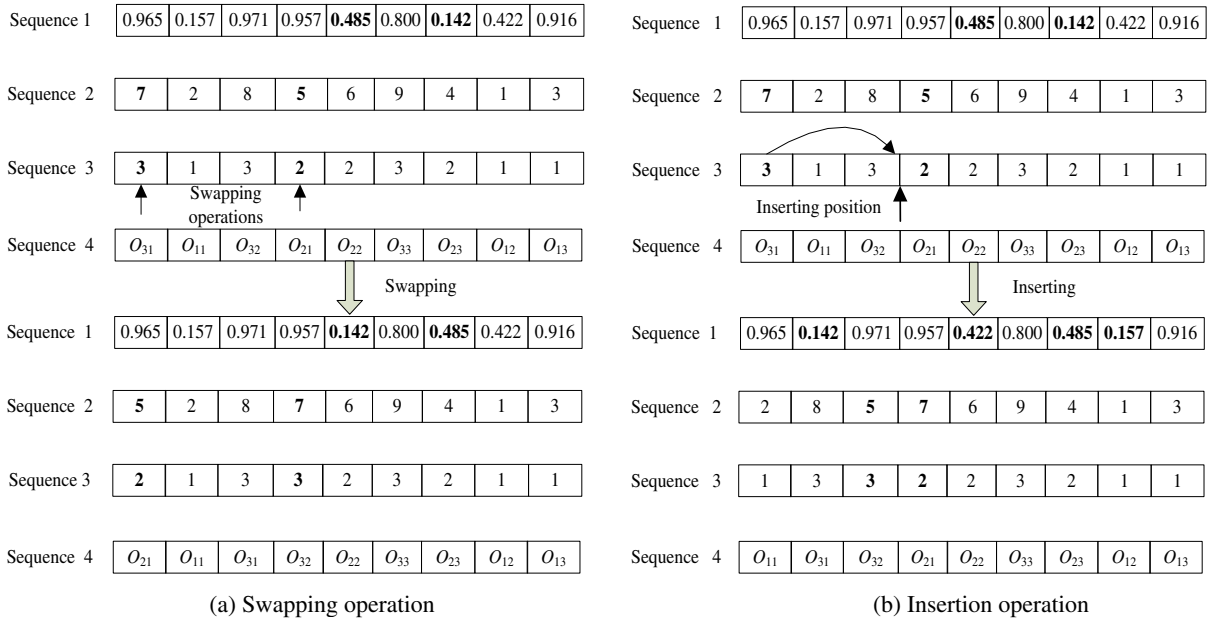


Figure 6. Swapping and insertion operation scheme.

#### 3.6.2 *Local search procedure*

The procedure of the local search algorithm is displayed as follows. To not consume a significant amount of time on local search, the local search algorithm is applied to the best individual of the current population.

**Algorithm 3.** (NS):
**Step 1.** Input the individual $\boldsymbol{x}$.
**Step 2.** Using the Neighbourhood 1 to search, $\boldsymbol{x}' = \text{swap}(\boldsymbol{x})$.
**Step 3.** Using the Neighbourhood 2 to search, $\boldsymbol{x}'' = \text{insert}(\boldsymbol{x}')$.
**Step 4.** $\boldsymbol{x} = \boldsymbol{x}''$, terminate, return $\boldsymbol{x}$.

### 3.7 *The procedure of NS-HDE/EDA*

Based on the above of encoding scheme, EDA, DE and local search strategy, a NS-HDE/EDA is proposed to solve JSSP. The procedure of NS-HDE/EDA is described as follows:
**Algorithm 4.** (NS-HDE/EDA):
**Step 1.** Initialization:
**Step 1.0.** Randomly generate the initial population $\boldsymbol{X}_0$.
**Step 1.1.** Initialise the learning rate $\alpha$, $CR_0 = 0.3$ and $\gamma_0 = 0.2$, then let $t = 0$. **Step 1.3** evaluate the fitness value of each individual in $\boldsymbol{X}_0$.
**Step 2.** Evaluate the fitness value of each individual in $\boldsymbol{X}_0$.
**Step 3.** Conduct the probabilistic mode $P_t(\boldsymbol{x})$.
for $i$ = 1:NP
**Step 4.** Generate the new population $\boldsymbol{X}'_t(\boldsymbol{i})$.
**Step 4.0.** Generate a random number *rand* in interval [0, 1].
**Step 4.1.** if *rand* < $\zeta$,
Generate the new candidate $\mathrm{X}'_t(i)$ by sampling the probabilistic model $P_t$.
else
Update $F_t$, $CR_t$, with $\gamma_t$ Equations (6–8).
Generate the new candidate $\boldsymbol{X}'_t(\boldsymbol{i})$ by DE operators which include mutation and crossover.
end
**Step 5.** Evaluate fitness value: Evaluate the fitness value of the individual $\boldsymbol{X}'_t(\boldsymbol{i})$.
**Step 6.** Selection.

$$\boldsymbol{X}_{t+1}(i) = \begin{cases} \boldsymbol{X}_t(i), & \text{if } f(\boldsymbol{X}_t(i)) < f(\boldsymbol{X}'_t(i)) \\ \boldsymbol{X}'_t(i) & \text{otherwise} \end{cases} \tag{14}$$

where $f(\bullet)$ is the function of calculating makespan.
end
**Step 7.** Local search: use **Algorithm 3** to search the all best individuals in population $X_{t+1}$.
**Step 8.** If the stop criterion is satisfied, then stop. Otherwise, let $t = t + 1$, and turn to Step 3.

In above procedure, stop criterion includes: (1) the number of generations equals the maximum number of generations; (2) the makespan gets the known lower bound.

## 4. Computational results and comparisons

In order to verify the good performance of NS-HDE/EDA algorithm for JSSP, We consider 123 JSSP benchmarks instances from OR-Library (Beasley 1990). These instances include three classes:

(1) FT06, FT10, FT20 ($n \times m = 6 \times 6$, $10 \times 10$, $20 \times 5$) were designed by Fisher and Thompson (1963).
(2) The instance LA01-LA40 ($n \times m = 10 \times 5$, $15 \times 5$, $20 \times 5$, $10 \times 10$, $15 \times 10$, $20 \times 10$, $30 \times 10$, $15 \times 15$) were designed by Lawrence (1984).
(3) Forty instances of eight different sizes ($n \times m = 15 \times 15$, $20 \times 15$, $20 \times 20$, $30 \times 15$, $30 \times 20$, $50 \times 15$, $50 \times 20$, $100 \times 20$) denoted by (TA) due to Taillard (1993).

The experimental environment includes: MATLAB2010b; Windows 7; Intel Core i5-2450M 2.50 GHz (CPU), 2G RAM.

### 4.1 *Parameters analysis*

The parameter settings have significant influence on the performance of the NS-HDE/EDA. NS-HDE/EDA contains some critical parameters: the population size NP, the maximum iteration *Maxgens*, the learning rate $\alpha$ and the selection probability $\eta$. To investigate the influence of these parameters on the performance of the NS-HDE/EDA, we implement the Taguchi (Montgomery 2008) method of design for the experiment to analyse the parametric sensitivity. FT10 is selected as the benchmark test cases. The various values of NP, *Maxgens*, $\alpha$ and $\eta$ are listed in Table 2. Table 3 lists the parameter combinations and the obtained average response variable (ARV) values which is the average makespan value obtained by the NS-HDE/EDA. Table 4 lists each parameter's significance rank according to Table 3. In Figure 7, the trend of each parameter for different value is described.

It is observed from Table 4 that NP is the most significant one among them, which illustrated that NP is crucial to the NS-HDE/EDA. The large population can help the EDA to build accurate model. But a large population size will cause an oversize computational budget. Moreover, $\alpha$ ranks the second, which implies it is also an important factor for NS-HDE/EDA. A small value of $\alpha$ could lead to premature convergence. A large value of $\alpha$ could lead to low convergence rate. From Figure 7, it can be seen that the trend of *Maxgens* and $\eta$ is relatively flat, which expresses their settings slightly impact on the NS-HDE/EDA. According to the above analysis, the parameters in NS-HDE/EDA are set as follows:

- For all FT instances, LA01-LA15, NP = 20 and $\alpha = 0.05$.
- LA16-LA30, NP = 20 and $\alpha = 0.1$.
- LA31-LA35, NP = 10 and $\alpha = 0.05$.
- LA36-LA40, and all TA instances, NP = 15 and $\alpha = 0.05$.
- $\eta = 0.4$, *Maxgens* = 10,000.

For FT and LA instances, we run the NS-HDE/EDA 20 times independently. TA is run 10 times.

### 4.2 *Results and comparison with other algorithms*

Firstly, Relative Error (RE) is defined as follows:

$$\mathrm{RE} = 100 \times (\mathrm{Value} - \mathrm{BKS})/\mathrm{BKS}$$

where BKS is the best-known solution or the best-known upper bound (UB), value is the best solution or the mean solution found by NS-HDE/EDA.

Table 5 gives the computational results obtained by NS-HDE/EDA and the comparison with some algorithms reported in literature (Adams, Balas, and Zawack 1988; Della Croce, Tadei, and Volta 1995; Dorndorf and Pesch 1995; Sabuncuoglu and Bayiz 1999; Coello, Rivera, and Cortés 2003; Ombuki and Ventresca 2004; Sha and Hsu 2006; Hasan et al. 2009; He et al. 2010; Pan et al. 2010). The boldface represents that the better solutions obtain the best known. NS-HDE/EDA can find the best-known solution with 22 instances. For the small instance FT06 and LA01-LA15, almost all algorithms can find the best-known solution. LA31-LA35 can be achieved the best-known solution by NS-HDE/EDA. It also can be seen that the results of NS-HDE/EDA are better than EDA, DDE and $\mathrm{DDE}_1$. It shows that the combination of EDA and DE are effective.

Table 6 shows the number of instances solved (NIS) and the RE of all compared algorithms. Meanwhile, the reduction of RE obtained by NS-HDE/EDA with respect to other algorithms is also presented in the last column. We know that the RE of NS-HDE/EDA is only on average 0.8%. So the proposed algorithm yields a significant improvement in solution quality with respect to all other algorithms.

We also compare the NS-HDE/EDA with DDE, EDA and PSO-priority on the convergence rate. Figure 8 describes the evolutionary curves of the instance FT06, FT10, LA01 and LA20. It can be observed that the convergence rate of

Table 2. The various values of the parameters in NS-HDE/EDA.

| | Values | | | |
|---|---|---|---|---|
| Parameters | 1 | 2 | 3 | 4 |
| NP | 10 | 15 | 20 | 25 |
| Maxgens | 3000 | 5000 | 10,000 | 50,000 |
| $\alpha$ | 0.05 | 0.1 | 0.2 | 0.25 |
| $\eta$ | 0.1 | 0.2 | 0.3 | 0.4 |

Table 3. Parameter combinations and ARV values.

| | Combinations | | | | |
|---|---|---|---|---|---|
| Number | NP | Maxgens | $\alpha$ | $\eta$ | ARV |
| 1 | 1 | 1 | 1 | 1 | 972.7 |
| 2 | 1 | 2 | 2 | 2 | 975.2 |
| 3 | 1 | 3 | 3 | 3 | 984.2 |
| 4 | 1 | 4 | 4 | 4 | 989.0 |
| 5 | 2 | 1 | 2 | 3 | 962.5 |
| 6 | 2 | 2 | 1 | 4 | 987.5 |
| 7 | 2 | 3 | 4 | 1 | 1009.2 |
| 8 | 2 | 4 | 3 | 2 | 971.9 |
| 9 | 3 | 1 | 3 | 4 | 963.4 |
| 10 | 3 | 2 | 4 | 3 | 955.1 |
| 11 | 3 | 3 | 1 | 2 | 955.3 |
| 12 | 3 | 4 | 2 | 1 | 971.6 |
| 13 | 4 | 1 | 4 | 2 | 967.5 |
| 14 | 4 | 2 | 3 | 1 | 950.7 |
| 15 | 4 | 3 | 2 | 4 | 958.1 |
| 16 | 4 | 4 | 1 | 3 | 970.8 |

Table 4. Parameter rank and response values.

| Parameters values numbers | NP | *Maxgens* | $\alpha$ | $\eta$ |
|---|---|---|---|---|
| 1 | 980.3 | 966.5 | 971.6 | 976.1 |
| 2 | 982.8 | 967.1 | 966.9 | 967.5 |
| 3 | 961.3 | 976.8 | 967.5 | 968.2 |
| 4 | 961.8 | 975.8 | 980.2 | 974.5 |
| Response value | 11.594 | 5.506 | 6.133 | 4.360 |
| Rank | 1 | 3 | 2 | 4 |

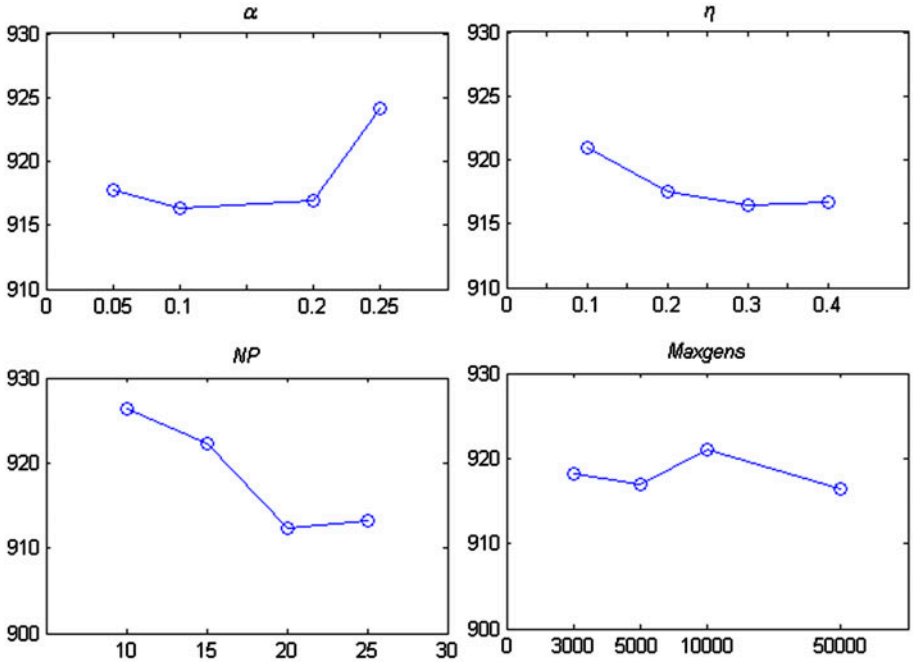


Figure 7. The trend of parameters of the NS-HDE/EDA.

Table 5. Computational results of FT and LA instances.

| Instance | Size | BKS | NS-HDE/EDA | He et al. (2010) | Pan et al. (2010) | Hasan et al. (2009) | Sha and Hsu (2006) | Ombuki and Ventresca (2004) | Coello, Rivera, and Cortés (2003) | Sabuncuoglu and Bayiz (1999) | Della Croce, Tadei, and Volta (1995) | Dorndorf and Pesch (1995) | Adams, Balas, and Zawack (1988) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mean | Best | $T_{av}$(s) | EDA | DDE | $DDE_1$ | MA(PR) | PSO-priority | LSGA | AIS | Beam search | GA | PGA | SBI |
| FT06 | 6,6 | 55 | 55.0 | **55** | 2.31 | 55 | 55 | 55 | – | 55 | – | – | – | – | – | 55 |
| FT10 | 10,10 | 930 | 951.3 | 937 | 328.60 | 937 | 962 | 955 | – | 1007 | – | 941 | 1016 | 946 | 960 | 1015 |
| FT20 | 20,5 | 1165 | 1183.1 | 1178 | 418.03 | 1184 | 1220 | 1199 | – | 1242 | – | – | – | 1178 | 1249 | 1290 |
| LA01 | 10,5 | 666 | 666.0 | **666** | 2.66 | 666 | 666 | 666 | 667 | 681 | – | 666 | 666 | 666 | 666 | 666 |
| LA02 | 10,5 | 655 | 665.2 | **655** | 22.09 | – | – | – | 655 | 694 | – | 655 | 704 | 666 | 681 | 720 |
| LA03 | 10,5 | 597 | 600.5 | **597** | 18.35 | – | – | – | 617 | 633 | – | 597 | 650 | 666 | 620 | 623 |
| LA04 | 10,5 | 590 | 597.4 | **590** | 24.08 | – | – | – | 606 | 611 | – | 590 | 620 | – | 620 | 597 |
| LA05 | 10,5 | 593 | 593.0 | **593** | 0.12 | – | – | – | 593 | 593 | – | 593 | 593 | – | 593 | 593 |
| LA06 | 15,5 | 926 | 926.0 | **926** | 0.39 | 926 | 926 | 926 | 926 | 926 | – | 926 | 926 | 926 | 926 | 926 |
| LA07 | 15,5 | 890 | 890.0 | **890** | 5.08 | – | – | – | 890 | 890 | – | 890 | 890 | – | 890 | 890 |
| LA08 | 15,5 | 863 | 863.0 | **863** | 11.26 | – | – | – | 863 | 863 | – | 863 | 863 | – | 863 | 868 |
| LA09 | 15,5 | 951 | 951.0 | **951** | 4.47 | – | – | – | 951 | 953 | – | 951 | 951 | – | 951 | 951 |
| LA10 | 15,5 | 958 | 958.0 | **958** | 0.15 | – | – | – | 958 | 958 | – | 958 | 958 | – | 958 | 959 |
| LA11 | 20,5 | 1222 | 1222.0 | **1222** | 1.63 | 1222 | 1222 | 1222 | 1222 | 1222 | – | – | 1222 | 1222 | 1222 | 1222 |
| LA12 | 20,5 | 1039 | 1039.0 | **1039** | 2.11 | – | – | – | 1039 | 1039 | – | – | 1039 | – | 1039 | 1039 |
| LA13 | 20,5 | 1150 | 1150.0 | **1150** | 8.66 | – | – | – | 1150 | 1150 | – | – | 1150 | – | 1150 | 1150 |
| LA14 | 20,5 | 1292 | 1292.0 | **1292** | 0.07 | – | – | – | 1292 | 1292 | – | – | 1292 | – | 1292 | 1292 |
| LA15 | 20,5 | 1207 | 1207.0 | **1207** | 13.87 | – | – | – | 1207 | 1232 | – | – | 1207 | – | 1237 | 1207 |
| LA16 | 10,10 | 945 | 977.1 | 956 | 90.87 | 945 | 978 | 964 | 994 | 1006 | 959 | 945 | 988 | 979 | 1008 | 1021 |
| LA17 | 10,10 | 784 | 787.3 | **784** | 81.59 | – | – | – | 785 | 833 | 792 | 785 | 827 | – | 809 | 796 |
| LA18 | 10,10 | 848 | 864.9 | 855 | 102.82 | – | – | – | 861 | 901 | 857 | 848 | 881 | – | 916 | 891 |
| LA19 | 10,10 | 842 | 877.4 | 852 | 103.20 | – | – | – | 896 | 895 | 860 | 848 | 882 | – | 880 | 875 |
| LA20 | 10,10 | 902 | 910.9 | 907 | 92.16 | – | – | – | 967 | 963 | 907 | 907 | 948 | – | 928 | 924 |
| LA21 | 15,10 | 1046 | 1087.8 | 1058 | 504.17 | 1071 | 1126 | 1100 | 1090 | 1201 | 1097 | – | 1154 | 1097 | 1139 | 1172 |
| LA22 | 15,10 | 927 | 967.4 | 952 | 640.03 | – | – | – | 985 | 1046 | 980 | – | 985 | – | 998 | 1040 |
| LA23 | 15,10 | 1032 | 1046.5 | 1038 | 292.30 | – | – | – | 1043 | 1146 | 1032 | – | 1051 | – | 1072 | 1061 |
| LA24 | 15,10 | 935 | 991.2 | 973 | 662.72 | – | – | – | 986 | 1082 | 1001 | – | 992 | – | 1014 | 1000 |
| LA25 | 15,10 | 977 | 1014.2 | 1000 | 701.42 | – | – | – | 1077 | 1107 | 1031 | 1022 | 1073 | – | 1014 | 1048 |
| LA26 | 20,10 | 1218 | 1268.8 | 1229 | 894.96 | 1257 | 1306 | 1262 | 1303 | 1409 | 1295 | – | 1269 | 1231 | 1278 | 1304 |
| LA27 | 20,10 | 1235 | 1327.9 | 1287 | 972.42 | – | – | – | 1328 | 1437 | 1306 | – | 1316 | – | 1378 | 1325 |
| LA28 | 20,10 | 1216 | 1283.5 | 1275 | 975.70 | – | – | – | 1328 | 1434 | 1302 | 1277 | 1373 | – | 1327 | 1256 |
| LA29 | 20,10 | 1152 | 1254.1 | 1220 | 998.91 | – | – | – | 1267 | 1359 | 1280 | 1248 | 1252 | – | 1336 | 1294 |
| LA30 | 20,10 | 1355 | 1396.7 | 1371 | 956.76 | – | – | – | 1363 | 1517 | 1406 | – | 1435 | – | 1411 | 1403 |
| LA31 | 30,10 | 1784 | 1784.0 | **1784** | 296.60 | 1789 | 1801 | 1784 | 1784 | 1886 | 1784 | – | 1784 | 1784 | – | 1784 |
| LA32 | 30,10 | 1850 | 1851.2 | **1850** | 671.69 | – | – | – | 1850 | 2000 | 1850 | – | 1850 | – | – | 1850 |
| LA33 | 30,10 | 1719 | 1719.0 | **1719** | 184.71 | – | – | – | 1719 | 1832 | 1719 | – | 1719 | – | – | 1719 |

(*Continued*)

Table 5. (*Continued*).

| Instance | Size | BKS | NS-HDE/EDA | He et al. (2010) | Pan et al. (2010) | Hasan et al. (2009) | Sha and Hsu (2006) | Ombuki and Ventresca (2004) | Coello, Rivera, and Cortés (2003) | Sabuncuoglu and Bayiz (1999) | Della Croce, Tadei, and Volta (1995) | Dorndorf and Pesch (1995) | Adams, Balas, and Zawack (1988) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mean | Best | $T_{av}$(s) | EDA | DDE | $DDE_1$ | MA(PR) | PSO-priority | LSGA | AIS | Beam search | GA | PGA | SBI |
| LA34 | 30,10 | 1721 | 1738.6 | **1721** | 917.05 | – | – | – | 1740 | 1876 | 1758 | – | 1780 | – | – | 1721 |
| LA35 | 30,10 | 1888 | 1889.2 | **1888** | 604.61 | – | – | – | 1898 | 2027 | 1888 | 1903 | 1888 | – | – | 1888 |
| LA36 | 15,15 | 1268 | 1324.4 | 1315 | 697.07 | 1292 | 1352 | 1334 | 1389 | 1357 | 1357 | 1323 | 1401 | 1305 | 1373 | 1351 |
| LA37 | 15,15 | 1397 | 1487.0 | 1465 | 654.04 | – | – | – | 1544 | 1494 | 1494 | – | 1503 | – | 1498 | 1485 |
| LA38 | 15,15 | 1196 | 1288.6 | 1244 | 628.99 | – | – | – | 1367 | 1338 | 1338 | 1274 | 1297 | – | 1296 | 1280 |
| LA39 | 15,15 | 1233 | 1308.8 | 1291 | 749.59 | – | – | – | 1342 | 1343 | 1343 | 1270 | 1369 | – | 1351 | 1321 |
| LA40 | 15,15 | 1222 | 1290.1 | 1277 | 720.32 | – | – | – | 1357 | 1311 | 1311 | 1258 | 1347 | – | 1321 | 1326 |

Table 6. Average relative deviation to the BKS.

| Algorithms | NIS | RE | | Reduction |
|---|---|---|---|---|
| | | Others | NS-HDE/EDA | |
| EDA (He et al. 2010) | 11 | 0.92 | 0.80 | 0.12 |
| DDE (Pan et al. 2010) | 11 | 3.10 | 0.80 | 2.30 |
| $DDE_1$ (Pan et al. 2010) | 11 | 1.96 | 0.80 | 1.16 |
| MA (PR) (Hasan et al. 2009) | 40 | 3.60 | 1.34 | 2.26 |
| PSO-priority (Sha and Hsu 2006) | 43 | 6.82 | 1.29 | 5.53 |
| LAGA (Ombuki and Ventresca 2004) | 25 | 5.22 | 2.14 | 3.38 |
| AIS (Coello, Rivera, and Cortés 2003) | 24 | 1.51 | 1.44 | 0.07 |
| Beam search (Sabuncuoglu and Bayiz 1999) | 41 | 4.02 | 1.32 | 2.70 |
| GA (Della Croce, Tadei, and Volta 1995) | 12 | 2.38 | 0.73 | 1.65 |
| PGA (Dorndorf and Pesch 1995) | 37 | 4.62 | 1.49 | 2.13 |
| SBI (Adams, Balas, and Zawack 1988) | 43 | 3.89 | 1.29 | 2.60 |

NS-HDE/EDA is much better than other algorithms. The global exploration ability of the EDA in NS-HDE/EDA plays a key role, which effectively helps to improve the convergence rate.

For TA problems, we implemented two classical algorithms which are genetic algorithm (GA) and PSO as the compared objects. We implemented a simple GA for solving JSSP. Each individual is represented by a real chromosome, which is encoded by the operation-based representation method. We use POX crossover and INV mutation as reproduction operators. The population size is set to 30. Then, we implemented a PSO, which is come from reference Lin et al. (2010).

The computational results obtained by NS-HDE/EDA, PSO and GA are summarised in Table 7. The computational results include RE of the best solution (BRE), RE of the average solution (ARE), RE of the worst solution (WRE), the

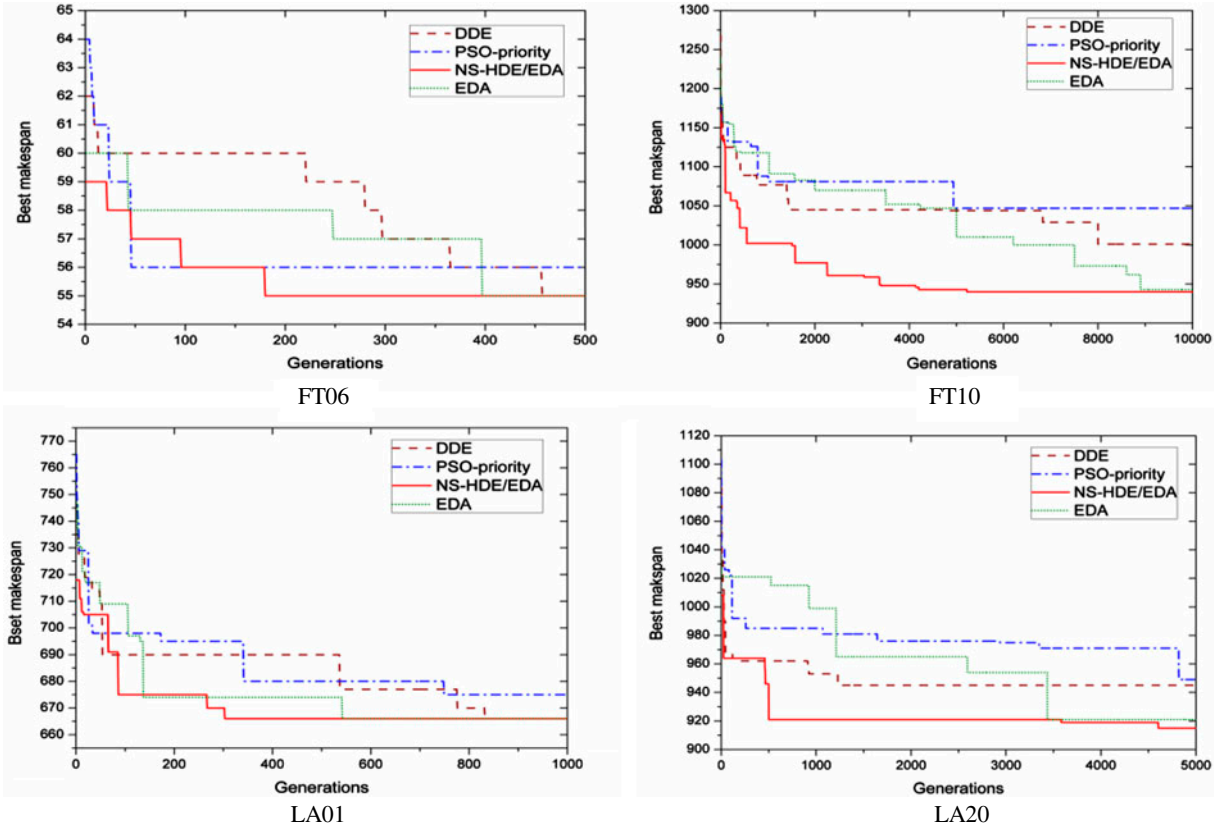


Figure 8. The evolutionary curves for JSSP.

Table 7. Comparison with PSO and GA for TA instances.

| | | NS-HDE/EDA | | | | | PSO | | | | | GA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Size | BRE | ARE | WRE | ASD | $T_{av}$(s) | BRE | ARE | WRE | ASD | $T_{av}$(s) | BRE | ARE | WRE | ASD | $T_{av}$(s) |
| TA01-TA10 | 15,15 | **4.49** | **5.26** | **5.84** | **8.67** | 692.5 | 13.98 | 14.22 | 15.35 | 9.06 | 448.7 | 12.28 | 12.63 | 13.74 | 10.62 | 539.3 |
| TA11-TA20 | 20,15 | **8.76** | **8.96** | **9.41** | **5.54** | 1383.3 | 16.18 | 16.67 | 17.68 | 10.51 | 591.1 | 16.00 | 16.64 | 17.35 | 9.49 | 719.5 |
| TA21-TA30 | 20,20 | **8.25** | **8.52** | **10.57** | **5.52** | 1597.7 | 16.47 | 17.16 | 19.72 | 11.98 | 795.7 | 14.43 | 15.48 | 17.83 | 13.65 | 954.3 |
| TA31-TA40 | 30,15 | **6.53** | **7.58** | **7.91** | **12.89** | 2384.4 | 18.09 | 19.32 | 19.87 | 16.51 | 1182.5 | 15.70 | 17.04 | 17.56 | 17.27 | 1085.7 |
| TA41-TA50 | 30,20 | **7.64** | **8.55** | **8.81** | **12.04** | 3272.7 | 18.37 | 20.19 | 20.69 | 24.15 | 1576.8 | 14.55 | 15.97 | 16.47 | 19.68 | 1666.5 |
| TA51-TA60 | 50,15 | **2.57** | **3.16** | **3.33** | **11.41** | 5899.8 | 16.56 | 18.15 | 18.52 | 29.76 | 1958.0 | 13.60 | 14.59 | 14.91 | 19.32 | 2086.7 |
| TA61-TA70 | 50,20 | **4.36** | **4.95** | **5.19** | **12.18** | 7649.6 | 16.47 | 17.46 | 17.94 | 21.68 | 2658.1 | 14.26 | 15.69 | 16.14 | 28.33 | 2792.6 |
| TA71-TA80 | 100,20 | **3.01** | **3.27** | **3.47** | **13.36** | 11,254.3 | 15.61 | 16.27 | 16.46 | 29.77 | 5224.6 | 14.38 | 14.98 | 15.18 | 22.67 | 5659.3 |
| MRE | | **5.70** | **6.28** | **6.81** | | | 16.46 | 17.43 | 18.28 | | | 14.40 | 15.38 | 16.18 | | |

average standard deviation of the makspan (ASD) and the average computational time ($T_{av}$(s)). The best one for each scale has been marked with boldface. It can be found that, the results obtained by the proposed algorithm are better than the two classical algorithms. The mean ARE value obtained by NS-HDE/EDA is 6.28% which is lower than the previously obtained results of 17.43% from PSO and 15.38% from GA. The graphical representation in Figure 9 shows the BRE values of NS-HDE/EDA, PSO and GA for each TA instance. The BRE values of NS-HDE/EDA are lower than other two algorithms, namely, the best solution of the proposed algorithm is close to the best-known solution which have found at present. Although NS-HDE/EDA does not obtain the best-known solutions at present generations for large problems, the evolutionary trend of the population does not stagnate. That is to say NS-HDE/EDA can further be optimised to obtain the better solutions. The WRE values obtained by NS-HDE/EDA are better than BRE values of PSO and GA. It can be concluded that NS-HDE/EDA is more effective than the compared algorithm. Additionally, the ASD obtained by NS-HDE/EDA is lower than that of obtained with PSO and GA. It demonstrates that the strong robustness of the proposed algorithms. Therefore, NS-HDE/EDA can achieve a good trade-off between global search and local search and is very suitable for JSSP. In particular, the best makespan, the average makespan and the average computational time of each TA instance obtained by NS-HDE/EDA, PSO and GA has been listed in Table A1 of Appendix 1.

By compared with the computational time of the three algorithms in Figure 10, we see that PSO and GA consume comparable amount of time while NS-HDE/EDA requires apparently more time to run. Meanwhile, Table 7 also lists the average CPU time of fixed generations for each scale with three compared algorithms. With increase of problem

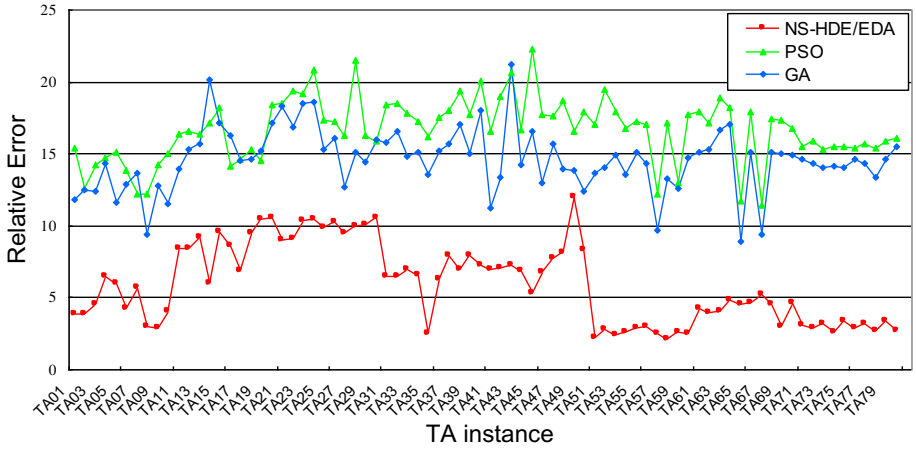


Figure 9. The RE of NS-HDE/EDA, PSO and GA for TA instance.

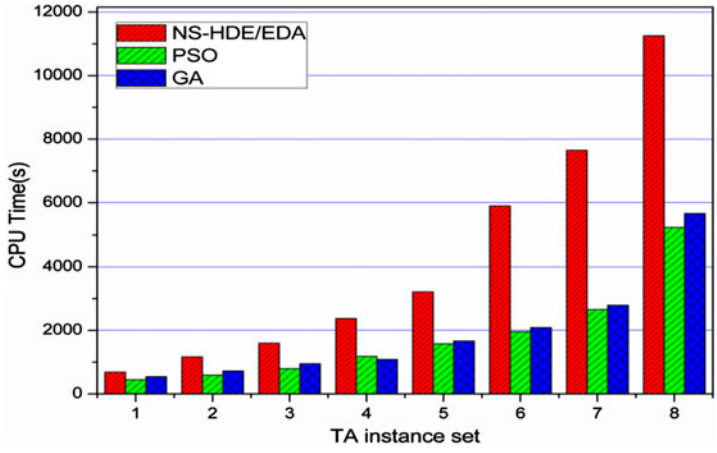


Figure 10. A comparison of computational time for TA instance.

size, NS-HDE/EDA will consume more time. Due to the fact that, in NS-HDE/EDA, there is about 50% computation time spent on local search process. Compared with classical GA and PSO, the NS effectively improves the quality of the solution, but it also takes too much computation time.

## 5. Conclusions

In this paper, we first review recent studies on the JSSP of minimising makespan. Then, a novel hybrid optimisation algorithm, NS-HDE/EDA, was proposed for solving the JSSP with minimisation of makespan as the objective. In NS-HDE/EDA, EDA is integrated with DE by the proposed decisive factor to improve the searching efficiency. Local information and global information are effectively incorporated together. The CPC strategy is introduced to DE to strengthen its search ability. Meanwhile, a local search algorithm, NS, is designed to further search the solutions to improve the quality of the solutions.

The parametric sensitivity of the proposed algorithm was analysed based on the Taguchi method of design of experiment. The proposed algorithm was tested on 123 classical benchmark problems. The computational results and comparisons show that our presented NS-HDE/EDA algorithm is better than the reported approaches in terms of solution quality. However, the performance of the proposed algorithm on the large-scale JSSP was not ideal, especially TA scheduling problems, the obtained results of NS-HDE/EDA algorithm show large gap with the best-known solutions which have found.

Our future researches will include the following topics: First, we will improve the performance of NS-HDE/EDA algorithm and verify its efficiency on large-scale scheduling problem. Second, applying improved NS-HDE/EDA to other kinds of combination optimisation problems may also be a promising direction. Third, due to the consumption of large time in local search, developing novel and effective neighbourhood structures in the local search is meaningful and challenging.



## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work was financially supported by the National Natural Science Foundation of China [grant number 51365030]. It was also supported by the General and Special Program of the Postdoctoral Science Foundation of China [grant number 2012M521802], [grant number 2013T60889]; the Science Foundation for Distinguished Youth Scholars of Lanzhou University of Technology [grant number J201405]; Lanzhou Science Bureau project [grant number 2013-4-64].

## References

Adams, J., E. Balas, and D. Zawack. 1988. "The Shifting Bottleneck Procedure for Job Shop Scheduling." *Management Science* 34 (3): 391–401.

Bai, L., J. Wang, Y. Jiang, and D. Huang. 2012. "Improved Hybrid Differential Evolution–Estimation of Distribution Algorithm with Feasibility Rules for NLP/MINLP Engineering Optimization Problems." *Chinese Journal of Chemical Engineering* 20 (6): 1074–1080.

Banharnsakun, A., B. Sirinaovakul, and T. Achalakul. 2012. "Job Shop Scheduling with the Best-so-far ABC." *Engineering Applications of Artificial Intelligence* 25 (3): 583–593. http://www.sciencedirect.com/science/article/pii/S0952197611001461.

Basu, M. 2014. "Improved Differential Evolution for Short-term Hydrothermal Scheduling." *International Journal of Electrical Power & Energy Systems* 58: 91–100. http://www.sciencedirect.com/science/article/pii/S0142061513005413.

Beasley, J. E. 1990. "OR-library: Distributing Test Problems by Electronic Mail." *Journal of the Operational Research Society* 41 (11): 1069–1072.

Cai, T. X., and L. Xia. 2010. "Optimization of Job Shop Scheduling Based on Shuffled Frog Leaping Algorithm." *Journal of Shenzhen University Science and Engineering* 27 (5): 391–394.

Chassaing, M., J. Fontanel, P. Lacomme, L. Ren, N. Tchernev, and P. Villechenon. 2014. "A GRASP×ELS Approach for the Job-shop with a Web Service Paradigm Packaging." *Expert Systems with Applications* 41 (2): 544–562. http://www.sciencedirect.com/science/article/pii/S0957417413005678.

Coello, C. A. C., D. C. Rivera, and N. C. Cortés. 2003. "Use of an Artificial Immune System for Job Shop Scheduling." In *Proceeding Second International Conference on Artificial Immune Systems, ICARIS-2003*, edited by J. Timmis, P. Bentley, and E. Hart, 1–10. Edimburgh: Springer.

Das, S., and P. N. Suganthan. 2011. "Differential Evolution: A Survey of the State-of-the-art." *IEEE Transactions on Evolutionary Computation* 15 (1): 4–31.

Della Croce, F., R. Tadei, and G. Volta. 1995. "A Genetic Algorithm for the Job Shop Problem." *Computers & Operations Research* 22 (1): 15–24. http://www.sciencedirect.com/science/article/pii/0305054893E0015L.

Dorndorf, U., and E. Pesch. 1995. "Evolution Based Learning in a Job Shop Scheduling Environment." *Computers & Operations Research* 22 (1): 25–40. http://www.sciencedirect.com/science/article/pii/0305054893E0016 M.

Elmi, A., M. Solimanpur, and S. Topaloglu. 2011. "A Simulated Annealing Algorithm for the Job Shop Cell Scheduling Problem with Intercellular Moves and Reentrant Parts." *Computers & Industrial Engineering* 61: 171–178.

Ferdinando, P., and M. Emanuela. 1998. "A Tabu Search Method Guided by Shifting Bottleneck for the Jobshop Scheduling Problem." *European Journal of Operational Research* 120 (2000): 297–310.

Fisher, H., and G. Thompson. 1963. *Probabilistic Learning Combinations of Local Job-shop Scheduling Rules*, 225–251. Englewood Cliffs, NJ: Prentice-Hall.

Gao, L., G. Zhang, L. Zhang, and X. Li. 2011. "An Efficient Memetic Algorithm for Solving the Job Shop Scheduling Problem." *Computers & Industrial Engineering* 60 (4): 699–705. http://www.sciencedirect.com/science/article/pii/S0360835211000143.

Garey, M. R., D. S. Johnson, and R. Sethi. 1976. "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1 (2): 117–129.

Gholami, O., and Y. N. Sotskov. 2013. "Solving Parallel Machines Job-shop Scheduling Problems by an Adaptive Algorithm." *International Journal of Production Research* 52 (13): 3888–3904. Accessed February 7, 2015. http://dx.doi.org/10.1080/00207543.2013.835498

Hasan, S. M. K., R. Sarker, D. Essam, and D. Cornforth. 2009. "Memetic Algorithms for Solving Job-shop Scheduling Problems." *Memetic Computing* 1 (1): 69–83. http://dx.doi.org/10.1007/s12293-008-0004-5.

Hauschild, M., and M. Pelikan. 2011. "An Introduction and Survey of Estimation of Distribution Algorithms." *Swarm and Evolutionary Computation* 1 (3): 111–128.

He, X. J., J. C. Zeng, S. D. Xue, and L. F. Wang. 2010. "An Efficient Estimation of Distribution Algorithm for Job Shop Scheduling Problem." *Swarm, Evolutionary, and Memetic Computing* 6466: 656–663.

Heinz, M., and S.-V. Dirk. 1993. "Predictive Models for the Breeder Genetic Algorithm." *Evolutionary Computation* 1 (1): 25–49.

Huang, K.-L., and C.-J. Liao. 2008. "Ant Colony Optimization Combined with Taboo Search for the Job Shop Scheduling Problem." *Computers & Operations Research* 35 (4): 1030–1046. http://www.sciencedirect.com/science/article/pii/S0305054806001481.

Lawrence, S. 1984. *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*. Technical report. Pittsburgh, PA: Graduate School of Industrial Administration, Carnegie Mellon University.

Li, D.-W., 2012. "To Solve the Job Shop Scheduling Problem with the Improve Quantum Genetic Algorithmed." In *2012 Third Global Congress on Intelligent Systems (GCIS)*, 88–91.

Lin, T.-L., S.-J. Horng, T.-W. Kao, Y.-H. Chen, R.-S. Run, R.-J. Chen, J.-L. Lai, and I. H. Kuo. 2010. "An Efficient Job-shop Scheduling Algorithm Based on Particle Swarm Optimization." *Expert Systems with Applications* 37 (3): 2629–2636. http://www.sciencedirect.com/science/article/pii/S0957417409007696.

Meeran, S., and M. Morshed. 2012. "A Hybrid Genetic Tabu Search Algorithm for Solving Job Shop Scheduling Problems: A Case Study." *Journal of Intelligent Manufacturing* 23: 1063–1078.

Montgomery, D. C., 2008. *Design and Analysis of Experiments*. Arizona, AZ: Wiley.

Murovec, B., 2015. "Job-shop Local-search Move Evaluation without Direct Consideration of the Criterion's Value." *European Journal of Operational Research* 241 (2): 320–329. http://www.sciencedirect.com/science/article/pii/S0377221714007309.

Ombuki, B. M., and M. Ventresca. 2004. "Local Search Genetic Algorithms for the Job Shop Scheduling Problem." *Applied Intelligence* 21 (1): 99–109. http://dx.doi.org/10.1023/B:APIN.0000027769.48098.91.

Ou-Yang, C., and A. Utamima. 2013. "Hybrid Estimation of Distribution Algorithm for Solving Single Row Facility Layout Problem." *Computers & Industrial Engineering* 66 (1): 95–103. http://www.sciencedirect.com/science/article/pii/S0360835213001800.

Pan, Q., L. Wang, L. Gao, and H. Sang. 2010. "Differential Evolution Algorithm Based on Blocks on Critical Path for Job Shop Scheduling Problems." *Journal of Mechanical Engineering* 46 (22): 182–188. http://dx.doi.org/10.3901/JME.2010.22.182.

Pelikan, M., D. E. Goldberg, and F. G. Lobo. 2002. "A Survey of Optimization by Building and Using Probabilistic Models." *Computational Optimization and Applications* 21 (1): 5–20. http://dx.doi.org/10.1023/A:1013500812258.

Peng, B., Z. Lü, and T. C. E. Cheng. 2015. "A Tabu Search/Path Relinking Algorithm to Solve the Job Shop Scheduling Problem." *Computers & Operations Research* 53: 154–164. http://www.sciencedirect.com/science/article/pii/S0305054814002160.

Pinedo, M. L. 2008. *Scheduling Theory, Algorithms, and Systems*. Englewood Cliffs, NJ: Prentice-Hall.

Ponsich, A., and C. A. Coello Coello. 2013. "A Hybrid Differential Evolution – Tabu Search Algorithm for the Solution of Job-shop Scheduling Problems." *Applied Soft Computing* 13 (1): 462–474. http://www.sciencedirect.com/science/article/pii/S1568494612004188.

Qing-Dao-Er-Ji, R., and Y. Wang. 2012. "A New Hybrid Genetic Algorithm for Job Shop Scheduling Problem." *Computers & Operations Research* 39 (10): 2291–2299. http://www.sciencedirect.com/science/article/pii/S0305054811003601.

Sabuncuoglu, I., and M. Bayiz. 1999. "Job Shop Scheduling with Beam Search." *European Journal of Operational Research* 118 (2): 390–412. http://dx.doi.org/10.1016/S0377-2217(98)00319-1.

Sels, V., K. Craeymeersch, and M. Vanhoucke. 2011. "A Hybrid Single and Dual Population Search Procedure for the Job Shop Scheduling Problem." *European Journal of Operational Research* 215 (3): 512–523. http://www.sciencedirect.com/science/article/pii/S0377221711005601.

Seo, M., and D. Kim. 2010. "Ant Colony Optimisation with Parameterised Search Space for the Job Shop Scheduling Problem." *International Journal of Production Research* 48 (4): 1143–1154.

Sha, D. Y., and C.-Y. Hsu. 2006. "A Hybrid Particle Swarm Optimization for Job Shop Scheduling Problem." *Computers and Industrial Engineering* 51 (4): 791–808. http://dx.doi.org/10.1016/j.cie.2006.09.002.

Singh, H., and L. Srivastava. 2014. "Modified Differential Evolution Algorithm for Multi-objective Var Management." *International Journal of Electrical Power & Energy Systems* 55: 731–740. http://www.sciencedirect.com/science/article/pii/S0142061513004328.

Storn, R., and K. Price. 1997. "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *Journal of Global Optimization* 11 (4): 341–359. http://dx.doi.org/10.1023/A%3A1008202821328.

Taillard, E., 1993. "Benchmarks for Basic Scheduling Problems." *European Journal of Operational Research* 64 (2): 278–285. http://dx.doi.org/10.1016/0377-2217(93)90182-M.

Wang, L. 2003. *Shop Scheduling with Genetic Algorithms*. Beijing: Tsinghua University Press.

Wang, X., and H. Duan. 2014. "A Hybrid Biogeography-based Optimization Algorithm for Job Shop Scheduling Problem." *Computers & Industrial Engineering* 73: 96–114. http://www.sciencedirect.com/science/article/pii/S036083521400117X.

Wang, Y., B. Li, and X. Lai. 2009. "Variance Priority Based Cooperative Co-evolution Differential Evolution for Large Scale Global Optimization." *IEEE Congress on Evolutionary Computation, 2009. CEC'09*, 1232–1239. Trondheim: IEEE.

Wang, W. L., and W. Qi-Di. 2007. *Production Scheduling Intelligent Algorithm and Application*. Beijing: Science Press.

Wang, L., and D.-B. Tang. 2011. "An Improved Adaptive Genetic Algorithm Based on Hormone Modulation Mechanism for Job-shop Scheduling Problem." *Expert Systems with Applications* 38 (6): 7243–7250. http://www.sciencedirect.com/science/article/pii/S0957417410013904.

Wong, L.-P., C. Y. Puan, M. Y. H. Low, and Y. W. Wong. 2010. "Bee Colony Optimisation Algorithm with Big Valley Landscape Exploitation for Job Shop Scheduling Problems." *International Journal of Bio-Inspired Computation* 2 (2): 85–99.

Zhao, F., X. Jiang, C. Zhang, and J. Wang. 2014. "A Chemotaxis-enhanced Bacterial Foraging Algorithm and Its Application in Job Shop Scheduling Problem." *International Journal of Computer Integrated Manufacturing*: 1–16.

Zhao, F., J. Zhang, C. Zhang, and J. Wang. 2015. "An Improved Shuffled Complex Evolution Algorithm with Sequence Mapping Mechanism for Job Shop Scheduling Problems." *Expert Systems with Applications* 42 (8): 3953–3966. http://www.sciencedirect.com/science/article/pii/S0957417415000226.

## Appendix 1.

Table A1. Results using NS-HDE/EDA, PSO, GA for TA instance.

| | | | NS-HDE/EDA | | | PSO | | | GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | Size | BKS or (UB) | Best | Mean | $T_{av}$(s) | Best | Mean | $T_{av}$(s) | Best | Mean | $T_{av}$(s) |
| TA01 | 15,15 | 1231 | 1279 | 1283.9 | 689.2 | 1421 | 1423.5 | 435.3 | 1376 | 1379.5 | 552.4 |
| TA02 | 15,15 | 1244 | 1292 | 1301.4 | 702.5 | 1401 | 1403.7 | 481.6 | 1373 | 1384.7 | 532.8 |
| TA03 | 15,15 | 1218 | 1274 | 1286.0 | 695.2 | 1392 | 1396.1 | 493.1 | 1369 | 1381.9 | 567.5 |
| TA04 | 15,15 | 1175 | 1251 | 1255.5 | 720.8 | 1348 | 1352.4 | 491.2 | 1343 | 1346.1 | 521.7 |
| TA05 | 15,15 | 1224 | 1298 | 1302.6 | 687.4 | 1409 | 1411.3 | 432.8 | 1366 | 1368.2 | 521.0 |
| TA06 | 15,15 | 1238 | 1291 | 1297.6 | 710.9 | 1410 | 1410.7 | 428.9 | 1397 | 1403.8 | 551.1 |
| TA07 | 15,15 | 1227 | 1297 | 1299.2 | 665.7 | 1377 | 1378.5 | 446.8 | 1395 | 1400.3 | 543.2 |
| TA08 | 15,15 | 1217 | 1254 | 1258.6 | 682.5 | 1366 | 1372.1 | 436.9 | 1331 | 1336.9 | 526.1 |
| TA09 | 15,15 | 1274 | 1311 | 1328.2 | 698.1 | 1456 | 1457.3 | 427.5 | 1437 | 1443.8 | 539.8 |
| TA10 | 15,15 | 1241 | 1292 | 1321.9 | 672.6 | 1427 | 1430.7 | 412.4 | 1384 | 1395.4 | 537.5 |
| TA11 | 20,15 | (1357) | 1471 | 1484.6 | 1169.7 | 1579 | 1584.2 | 588.1 | 1547 | 1550.1 | 740.7 |
| TA12 | 20,15 | (1367) | 1482 | 1482.0 | 1629.4 | 1594 | 1600.8 | 611.8 | 1576 | 1582.4 | 699.4 |
| TA13 | 20,15 | (1342) | 1466 | 1469.4 | 1193.2 | 1562 | 1570.1 | 578.0 | 1553 | 1564.8 | 695.3 |
| TA14 | 20,15 | 1345 | 1426 | 1427.2 | 1578.0 | 1576 | 1579.3 | 602.3 | 1616 | 1628.7 | 732.1 |
| TA15 | 20,15 | (1339) | 1468 | 1471.6 | 1154.4 | 1583 | 1588.4 | 591.1 | 1569 | 1573.1 | 727.8 |
| TA16 | 20,15 | (1367) | 1485 | 1490.4 | 1610.8 | 1561 | 1569.4 | 587.4 | 1589 | 1590.1 | 692.5 |
| TA17 | 20,15 | 1462 | 1563 | 1563.0 | 1173.8 | 1676 | 1685.8 | 573.2 | 1675 | 1682.0 | 719.3 |
| TA18 | 20,15 | (1396) | 1528 | 1516.7 | 1582.1 | 1610 | 1618.6 | 621.5 | 1600 | 1613.2 | 721.9 |
| TA19 | 20,15 | (1332) | 1471 | 1477.2 | 1141.5 | 1525 | 1532.8 | 572.9 | 1534 | 1551.5 | 737.5 |
| TA20 | 20,15 | (1348) | 1490 | 1493.9 | 1599.6 | 1596 | 1600.4 | 591.4 | 1579 | 1589.3 | 728.7 |
| TA21 | 20,20 | (1642) | 1790 | 1794.3 | 1597.7 | 1946 | 1950.9 | 810.6 | 1943 | 1957.9 | 965.7 |
| TA22 | 20,20 | (1600) | 1745 | 1752.4 | 1629.4 | 1910 | 1922.9 | 821.5 | 1869 | 1880.0 | 955.8 |
| TA23 | 20,20 | (1557) | 1718 | 1720.5 | 1556.8 | 1856 | 1863.2 | 807.5 | 1845 | 1861.2 | 983.5 |
| TA24 | 20,20 | (1644) | 1816 | 1817.6 | 1578.0 | 1987 | 1994.4 | 781.2 | 1950 | 1972.0 | 973.4 |
| TA25 | 20,20 | (1595) | 1752 | 1760.0 | 1621.1 | 1872 | 1885.7 | 793.1 | 1839 | 1856.1 | 934.8 |
| TA26 | 20,20 | (1643) | 1811 | 1813.2 | 1610.8 | 1927 | 1935.8 | 783.4 | 1908 | 1934.9 | 969.0 |
| TA27 | 20,20 | (1680) | 1839 | 1846.4 | 1624.6 | 1953 | 1970.3 | 793.5 | 1893 | 1910.8 | 931.8 |
| TA28 | 20,20 | (1603) | 1763 | 1767.3 | 1582.1 | 1948 | 1958.0 | 791.2 | 1845 | 1863.2 | 938.7 |
| TA29 | 20,20 | (1625) | 1789 | 1792.1 | 1574.9 | 1890 | 1897.3 | 785.8 | 1859 | 1867.4 | 958.2 |
| TA30 | 20,20 | (1585) | 1752 | 1756.7 | 1599.6 | 1837 | 1859.6 | 789.0 | 1839 | 1858.3 | 931.6 |
| TA31 | 30,15 | 1764 | 1879 | 1893.2 | 2379.3 | 2088 | 2110.3 | 1191.2 | 2043 | 2057.8 | 1093.9 |
| TA32 | 30,15 | (1785) | 1901 | 1917.1 | 2389.4 | 2116 | 2141.1 | 1203.5 | 2081 | 2104.1 | 1094.7 |
| TA33 | 30,15 | (1791) | 1916 | 1923.2 | 2405.1 | 2110 | 2117.5 | 1219.3 | 2057 | 2073.4 | 1067.8 |
| TA34 | 30,15 | (1829) | 1949 | 1983.0 | 2364.9 | 2145 | 2173.9 | 1174.5 | 2105 | 2148.9 | 1053.5 |
| TA35 | 30,15 | 2007 | 2057 | 2061.5 | 2390.8 | 2331 | 2341.8 | 1181.6 | 2280 | 2301.1 | 1085.7 |
| TA36 | 30,15 | 1819 | 1934 | 1978.4 | 2355.7 | 2138 | 2171.0 | 1185.9 | 2095 | 2128.5 | 1096.2 |
| TA37 | 30,15 | 1771 | 1911 | 1915.1 | 2376.8 | 2090 | 2103.1 | 1162.1 | 2049 | 2074.5 | 1127.5 |
| TA38 | 30,15 | 1673 | 1789 | 1812.3 | 2428.0 | 1998 | 2031.8 | 1179.9 | 1959 | 1992.3 | 1056.8 |
| TA39 | 30,15 | 1795 | 1937 | 1942.6 | 2398.2 | 2113 | 2124.2 | 1184.8 | 2065 | 2079.0 | 1113.5 |
| TA40 | 30,15 | (1669) | 1790 | 1821.8 | 2355.6 | 2004 | 2036.5 | 1142.2 | 1970 | 1985.4 | 1067.8 |
| TA41 | 30,20 | (2005) | 2145 | 2161.3 | 3176.6 | 2337 | 2348.3 | 1556.8 | 2230 | 2256.7 | 1632.5 |
| TA42 | 30,20 | (1937) | 2074 | 2095.1 | 3297.8 | 2305 | 2378.0 | 1621.7 | 2196 | 2235.2 | 1678.1 |
| TA43 | 30,20 | (1848) | 1982 | 1991.7 | 3256.3 | 2230 | 2253.5 | 1563.4 | 2241 | 2249.0 | 1603.3 |
| TA44 | 20,20 | (1979) | 2115 | 2134.8 | 3324.8 | 2308 | 2375.2 | 1624.5 | 2260 | 2315.7 | 1705.7 |
| TA45 | 30,20 | (2000) | 2107 | 2120.1 | 3309.4 | 2446 | 2458.1 | 1610.5 | 2331 | 2353.1 | 1698.1 |
| TA46 | 20,20 | (2004) | 2139 | 2151.0 | 3402.6 | 2360 | 2412.6 | 1593.1 | 2265 | 2283.1 | 1655.2 |
| TA47 | 30,20 | (1894) | 2040 | 2047.2 | 3180.7 | 2228 | 2239.0 | 1607.8 | 2192 | 2235.2 | 1682.3 |
| TA48 | 30,20 | (1943) | 2101 | 2149.2 | 3314.0 | 2307 | 2375.7 | 1593.4 | 2214 | 2232.0 | 1658.7 |
| TA49 | 30,20 | (1961) | 2196 | 2203.6 | 3141.1 | 2286 | 2298.4 | 1543.5 | 2232 | 2246.5 | 1683.4 |
| TA50 | 30,20 | (1924) | 2085 | 2107.5 | 3324.5 | 2269 | 2291.2 | 1613.8 | 2162 | 2194.3 | 1669.7 |
| TA51 | 50,15 | 2760 | 2821 | 2827.2 | 5921.8 | 3232 | 3241.2 | 1941.8 | 3138 | 3149.2 | 2073.7 |
| TA52 | 50,15 | 2756 | 2834 | 2856.1 | 5931.3 | 3293 | 3329.8 | 2010.1 | 3142 | 3174.2 | 2045.1 |

(*Continued*)

Table A1. (*Continued*).

| Instance | Size | BKS or (UB) | NS-HDE/EDA | | | PSO | | | GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Best | Mean | $T_{av}$(s) | Best | Mean | $T_{av}$(s) | Best | Mean | $T_{av}$(s) |
| TA53 | 50,15 | 2717 | 2784 | 2791.0 | 5843.1 | 3203 | 3216.0 | 2011.4 | 3123 | 3137.4 | 2114.6 |
| TA54 | 50,15 | 2839 | 2912 | 2943.5 | 5927.8 | 3314 | 3395.7 | 1934.5 | 3225 | 3279.0 | 2131.6 |
| TA55 | 50,15 | 2679 | 2757 | 2759.3 | 5911.8 | 3141 | 3153.4 | 1903.5 | 3085 | 3112.1 | 2123.5 |
| TA56 | 50,15 | 2871 | 2956 | 2988.2 | 5936.5 | 3362 | 3447.2 | 1978.8 | 3282 | 3311.8 | 2017.8 |
| TA57 | 50,15 | 2943 | 3018 | 3018.4 | 5925.7 | 3302 | 3322.3 | 1995.6 | 3227 | 3239.2 | 2047.4 |
| TA58 | 50,15 | 2885 | 2946 | 2978.5 | 5910.1 | 3381 | 3473.1 | 1998.5 | 3269 | 3315.6 | 2085.9 |
| TA59 | 50,15 | 2655 | 2724 | 2724.7 | 5895.9 | 3001 | 3015.3 | 1932.6 | 2989 | 3001.4 | 2080.6 |
| TA60 | 50,15 | 2723 | 2792 | 2821.6 | 5891.4 | 3205 | 3284.8 | 2005.4 | 3124 | 3161.4 | 2111.2 |
| TA61 | 50,20 | 2868 | 2989 | 2993.1 | 7629.6 | 3381 | 3397.2 | 2638.1 | 3302 | 3339.1 | 2822.6 |
| TA62 | 50,20 | 2869 | 2982 | 3005.2 | 7695.6 | 3360 | 3391.1 | 2734.5 | 3309 | 3385.2 | 2825.7 |
| TA63 | 50,20 | 2755 | 2868 | 2872.5 | 7627.2 | 3276 | 3295.1 | 2694.4 | 3214 | 3240.8 | 2813.1 |
| TA64 | 50,20 | 2702 | 2833 | 2875.0 | 7673.8 | 3195 | 3214.3 | 2714.8 | 3163 | 3203.5 | 2875.8 |
| TA65 | 50,20 | 2725 | 2849 | 2857.4 | 7702.3 | 3044 | 3067.6 | 2689.1 | 2968 | 2994.5 | 2808.4 |
| TA66 | 50,20 | 2845 | 2976 | 3000.6 | 7723.5 | 3354 | 3389.8 | 2723.3 | 3276 | 3299.0 | 2810.7 |
| TA67 | 50,20 | 2825 | 2973 | 2981.3 | 7655.1 | 3149 | 3175.4 | 2655.1 | 3091 | 3114.0 | 2764.5 |
| TA68 | 50,20 | 2784 | 2910 | 2941.2 | 7712.5 | 3269 | 3302.4 | 2702.3 | 3205 | 3284.1 | 2821.6 |
| TA69 | 50,20 | 3071 | 3162 | 3169.0 | 7634.6 | 3605 | 3632.8 | 2613.2 | 3531 | 3561.2 | 2755.6 |
| TA70 | 50,20 | 2995 | 3134 | 3147.1 | 7624.8 | 3497 | 3546.7 | 2719.7 | 3441 | 3483.3 | 2810.0 |
| TA71 | 100,20 | 5464 | 5632 | 5641.8 | 11,258.5 | 6311 | 6324.8 | 5178.6 | 6264 | 6273.4 | 5623.3 |
| TA72 | 100,20 | 5181 | 5332 | 5347.7 | 11,314.2 | 6002 | 6056.1 | 5278.5 | 5922 | 5945.2 | 5736.3 |
| TA73 | 100,20 | 5568 | 5744 | 5752.2 | 11,283.8 | 6420 | 6442.4 | 5212.4 | 6352 | 6365.7 | 5703.6 |
| TA74 | 100,20 | 5339 | 5478 | 5490.1 | 11,303.1 | 6165 | 6211.2 | 5267.0 | 6094 | 6171.2 | 5782.5 |
| TA75 | 100,20 | 5392 | 5574 | 5581.4 | 11,224.3 | 6227 | 6239.2 | 5187.1 | 6151 | 6167.1 | 5709.8 |
| TA76 | 100,20 | 5342 | 5495 | 5512.6 | 11,321.5 | 6166 | 6205.4 | 5312.2 | 6126 | 6159.0 | 5764.3 |
| TA77 | 100,20 | 5436 | 5609 | 5615.3 | 11,306.4 | 6287 | 6301.0 | 5255.9 | 6213 | 6236.2 | 5634.7 |
| TA78 | 100,20 | 5394 | 5540 | 5578.5 | 11,337.8 | 6226 | 6278.8 | 5229.1 | 6115 | 6184.2 | 5726.5 |
| TA79 | 100,20 | 5358 | 5542 | 5548.3 | 11,213.5 | 6211 | 6233.1 | 5286.6 | 6143 | 6154.2 | 5625.4 |
| TA80 | 100,20 | 5183 | 5326 | 5343.6 | 11,346.8 | 6015 | 6091.5 | 5301.4 | 5988 | 6036.7 | 5723.6 |