Stochastic assignment optimization via hybrid genetic algorithm

Huang Ti Dao[a], Cheng Chien Tse[b], Chang Chung Kai[c]

Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Taiwan

[a] henry.2039@gmail.com, [b] pp.pp253@gmail.com, [c] changchungkai0606@gmail.com

Abstract

This project tackles the non-linear and acyclic assignment problems with uncertain processing times. We propose an effective genetic algorithm (GA) hybridized with Hungarian Algorithm, critical path method, Job-order crossover, and so forth, to minimize the total processing time. To build a high-quality and diverse set of initial population of GA, we introduce a heuristic method combined with critical path based Hungarian algorithm. In this hybrid GA, there are two different evolution approaches, breadth search and depth search. The former applies the roulette-wheel copy and roulette-wheel selection integrated with Job-order crossover; the latter infuses critical path-search and Hungarian Algorithm. The breadth search is used to avoid being trapped at a local optimal solution, and the depth search is used to accelerate the local search. This hybrid GA will harness one of the approaches with some specific condition; that is, this algorithm will repetitively switch between these two methods to increase the efficiency of finding the better solution.

Keywords: Hybrid Genetic Algorithm, Stochastic Assignment Optimization, Production, Routing

I. Introduction

In manufacturing industry, people tend to use production routing to specify all the required tasks to create products. ‘Routing’ means an ordered list of tasks. In general cases, production routing can be displayed as a directed graph to demonstrate how a product be manufactured. In this paper, we are interested in worker assignment problem on the directed acyclic production routing, named production routing task assignment problem (PRTAP), and propose a hybrid genetic algorithm (hGA), which combines genetic algorithm (GA) and other methodology or concept such as critical path method, Hungarian Algorithm, Job-order crossover etc., to minimize the makespan of the product. Taking the dynamically changing processing time in real-world manufacturing system into consideration, we introduce the sample path approximation to obtain the optimal worker assignment result in the stochastic condition.

This paper is organized as follows. In Section II, we introduce the mathematical formulation to describe PRTAP. In Section III, we discuss the program implementation of hGA, and hybridize other methodology with GA to enhance the performance of algorithm. In Section IV, we discuss the program implementation of sample path approximation so as to obtain the optimal solution in stochastic condition. Also, we compare the performance in hGA and original GA in Section V. Finally, in Section VI, we summarize the conclusion of our research.

II. Problem Description

Production routing task assignment problem (PRTAP) is an assignment problem for n workers onto an acyclic directed production routing with n tasks. The parameters in PRTAP are described as follows:

1. n, a number to represent the total number of tasks, or total number of workers.
2. The cost table, $C_{ij} \sim Exp(c_{ij})$, is a matrix used to represent the mean value of operation time on jth task by ith worker. C_{ij} are random with exponential distribution with parameter c_{ij}.
3. $G(V,E)$, a production routing graph, is an acyclic directed graph, with task set V as nodes and production dependencies set E as paths. $V\colon\{1,2,\ldots,n\}$ is the set of tasks, where $1,2,\ldots n$ obeying topological ordering of production dependencies, and set E describes the production dependencies of V. For instance, the mathematical expression of the Fig. 1. can be shown as follows:

   $G(V,E)$
   $V\colon\{1, 2, 3, 4, 5\}$
   $E\colon\{(1, 2), (1, 3), (2, 4), (3, 4), (4, 5)\}$

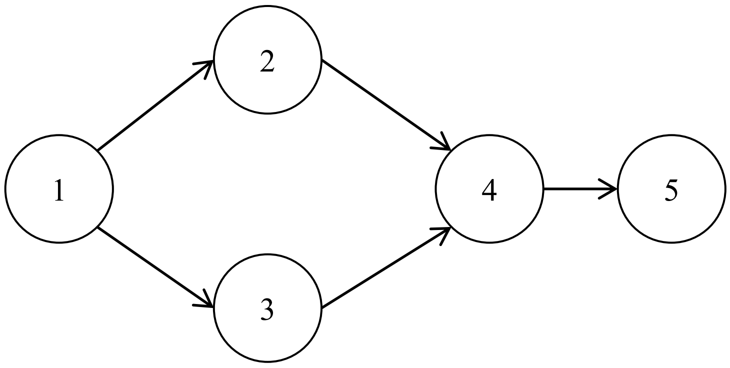


Fig. 1. Sample of production routing graph

4. P_j shows the set of all precedent tasks of the jth task, defined as $P_j = \{v|(v,j) \in E, \forall v \in V\}$. Taking the 1st and 4th tasks in Fig. 1. for example, we can tell that there is no v satisfies the condition in $P_1 = \{v|(v,1) \in E, \forall v \in V\}$, therefore, P_1 is an empty set, which means that there is no precedent tasks before 1st task. For $P_4 = \{v|(v,4) \in E, \forall v \in V\}$, we can realize that $\{(2,4),(3,4)\} \in E$ can fulfill the condition in P_4, which indicates that 2nd and 3rd task are the precedent tasks before 4th task.
5. The ending time of jth task, denoted as ET_j, can be formalized as:

$ET_j = max_{v\in P_j}\{ET_v\} + \sum_{i=1}^{n} C_{ij}x_{ij}, j = 1,2,\dots,n,$ where $max_{v\in P_j}\{ET_v\}$ is used to obtain the last finishing time of all the precedent tasks of jth task, which implies the starting time of jth task. $\sum_{i=1}^{n} C_{ij}x_{ij}$ describes the operation time in jth task. In this paper, we assume that the nth task is the only terminal of the graph. Due to the topological ordering, the makespan of the production routing is ET_n. Taking 4th task in Fig. 1 for instance ($ET_4 = max_{v\in P_4}\{ET_v\} + \sum_{i=1}^{n} C_{i4}x_{i4}$), according to P_4, we already know that 4th task has two precedent tasks, 2nd and 3rd task. In the part of $max_{v\in P_4}\{ET_v\}$, it will be $max(ET_2, ET_3)$. In these two ending time, we pick up the maximum one because 4th task can be started until both 2nd and 3rd task are finished.

For further investigation and research, we introduce a production routing graph as Fig. 2. and cost table C_{ij} in Appendix A, and set up the decision variables and mathematical model as follows:

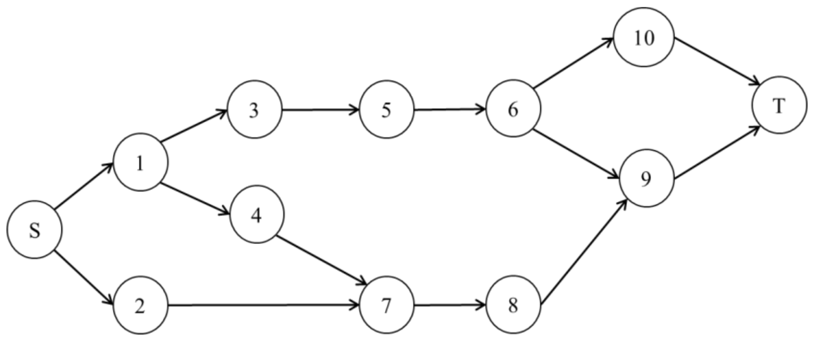


Fig. 2. Targeted production routing graph

The decision variables in PRTAP are represented in equation (1):

$$x_{ij} = \begin{cases} 1, & \text{if worker } i \text{ is assigned to task } j; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The PRTAP model is then written as:

$$\begin{aligned} \min \quad & E(ET_n) = E\left[max_{v\in P_n}\{ET_v\} + \textstyle\sum_{i=1}^{n} C_{in}x_{in}\right] & (2) \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,2,\dots,n & (3) \\ & \textstyle\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,2,\dots,n & (4) \\ & x_{ij} \in \{0,1\}, \forall i,j = 1,2,\dots,n. & (5) \end{aligned}$$

Equation (2) indicates the objective function of PRTAP model, where ET_n are taken expectation to obtain the expected minimum makespan in the stochastic condition. Constraint (3) and (4) make sure that one worker only gets exactly one task in the assignment result.

III. Program Implementation of Hybrid Genetic Algorithm

GA is a multidimensional optimization global search algorithm. In first stage, GA will initialize the population of possible solutions referred to as chromosomes. After applying a series of stochastic genetic operation such as chromosome selection, crossover, mutation, GA will obtain the solution of good quality for the given problem.

To acquire the clear concept of hGA proposed in this paper, the following section will introduce the components of hGA, genetic operators, and other related methodologies.

hGA components:

1. *p_size*: Population size which means the total number of chromosomes in each generation.
2. *max_gen*: Max generation which means the maximum generation that hGA will run through.
3. C_{ij} : Cost table as mentioned in Section II.
4. C_{pool} : It means a set of randomized cost table *C*. In hGA, the algorithm will randomly draw *C* in C_{pool} in advance and execute genetic operation so as to realize the dynamically changing processing time in real-world manufacturing system.
5. *iter_time*: It means total simulation time the algorithm will execute for each chromosome.
6. $chro^k$: The encoded chromosome *k*, for all $k \leq p\_size$, which is a vector. Also, let $chro^k_{i,j}$ denoted as the gene section *i* to *j* of the chromosome *k*.
7. fv_k: The fitness value of chromosome *k*, which represents the quality of chromosome. In this paper, *fv* is exactly the same as the makespan of the product based on chromosome *k*, that is, fv_k is equal to ET_n of chromosome *k*.
8. $f(\cdot)$: Fitness function is used to compute the fv_k.
9. $\alpha_c^{in}, \beta_c^{in}$: α_c^{in} means the proportion will be chosen from previous population in hGA, and the selected proportion of population will be executed according to all genetic operations. As finishing the genetic operation, there are lots of chromosomes that can be chosen to the next population. Therefore, β_c^{in} means the proportion will be selected from those chromosomes in waiting.

Genetic operators

1. Encoded mode

   Converting the solution of PRTAP into the coded chromosome is the key to GA, which strongly affects the efficiency and accuracy of the algorithm. To solve PRTAP, this paper presents a natural number encoding method, which utilizes a one-dimensional decimal array as a chromosome encoding process, we call the expressions based on the relationship of workers and tasks. Each chromosome is composed of several $chro^k_{i,j}$ $\forall i,j=1,\ldots,n$ *and* $i<j$. The example of encoded chromosome is shown in Fig.3:

Task	1	2	3	4	5	6	7	8	9	10
$chro^k$	5	2	1	4	3	9	6	7	10	8

Fig.3. The example of encoded chromosome

In Fig.3, $chro^{k}_{1,3}$ is (5,2,1), which means that the 5th worker is assigned to the 1st task, the 2nd worker is assigned to the 2nd task, and the 1st worker is assigned to the 3rd task

2. Roulette-wheel selection:

The mathematical formulation of Roulette-wheel selection is shown as:

$$prob(chro^{k}) = \frac{fv_k}{\sum_{u=1}^{p\_size} fv_u}, \tag{6}$$

where *prob(chro^k)* indicates the probability of *chro^k* be selected in the population. Roulette-wheel selection is more preferred for maximization problem because the bigger number will have the bigger probability to be chosen. However, PRTAP is a minimization problem. For the sake of the advantage of probability-based selection decision rule, hGA still applies Roulette-wheel selection, which is reformed as equation (7):

$$prob(chro^{k}) = \frac{1/\,fv_k}{\sum_{u=1}^{p\_size} 1/_{fv_u}}, \tag{7}$$

which be suitable for minimization problem.

Procedure 1 ROULETTE-WHEEL-SELECTION

```
Input $population$
Output $chro$
1    $r = \sum_{u=1}^{p\_size} 1/_{fv_u}$
2    randomly select $k$ in $population$ under $\frac{1/fv_k}{r}$ probability
3    return $k$
```

Procedure 2 ROULETTE-WHEEL-SELECT-POPULATION

```
Input $population, p\_size, \alpha$
Output $kid\_pop$
1    $S = \{\}$
2    for $i = 1$ to $p\_size \times \alpha$
3        $k =$ ROULETTE-WHEEL-SELECTION($pop$)
4        $S = S \cup \{k\}$
5    return $S$
```

3. Priority decision rule

In traditional crossover methodology of GA, such as 1-point crossover, k-point crossover, shuffling crossover and so on, it always randomly chooses the gene segments and swaps them. However, this methodology does not have any decision rules supporting it. Therefore, this paper introduces a priority decision rule to determine which task should be assigned first. That is, it states that which kind of gene segments should be kept. Two factors will be considered in priority decision rule:

3.1. Number of path passes through the task in production routing

Path is defined as the route which passes through not only the initial node but also the terminal node in the production routing. For instance, there are two paths in Fig. 1, which are (1,2,4,5) and (1,3,4,5). After listing any possible path, we introduce a function, denoted as $path(j)$, to calculate how many paths pass through jth task. If a node (any possible task) is stopped by many times in different paths, it may imply that the node plays an important role in this production routing. In Fig. 1, we can realize that 1st task, 4th task, 5th task have been passed by twice, and 2nd task, 3rd task have been passed by once. Therefore, $path(1) = path(4) = path(5) = 2$, and $path(2) = path(3) = 1$. In the end, we may claim that 1st task, 4th task, and 5th task are more crucial than others.

3.2. Standard deviation of the operation time on the same task by different workers

Standard deviation of the operation time on the same task by different workers is an important issue to be considered. If a task is quite easy that all workers can finish it in short time and the processing time is stable, it shows that the standard deviation of the operation time on this easy task is quite small. If the standard deviation is small, it implies that no matter which worker is assigned to the task, it will not impact too much on the makespan. In this condition, this task has a low priority to arrange worker. Let *std*(*j*) be the function to compute the variance of operation time on jth task.

With these two factors, the priority decision rule is defined as follows:

$$pri(j) = [path(j)]^2 \times std(j). \tag{8}$$

The higher value returned from $pri(\cdot)$, the higher priority of the task will be. hGA will assign the worker of good skills to those tasks of high priority in advance to make sure those tasks will have the shorter processing time. The following pseudocode illustrates the procedure of priority decision rule:

```
**Procedure 3** PRIORITY-DECISION-RULE
**Input** $C, D$
**Output** $rank$
1    let $rank$ be a new $D.length$ array
```

```
2    for $j = 0$ to $D.length$
3        $rank[j] = path(j)^2 \times std(C_{.j})$
4    return $rank$
```

Procedure 4 PRIORITY-RANK-TO-GENE

```
Input $C, D$
Output $gene$
1    let $rank$ be a new $D.length$ array
2    $rank$ = PRIORITY-DECISION-RULE ($C, D$)
3    $G$ = all job in the top 40% of $rank$
4    sorted $G$ by the $rank$ increasingly
5    return $G$
```

4. Job-order Crossover (JOC):

In PRTAP, the traditional crossover in GA, such as 1-point crossover, k-point crossover, shuffling crossover, and so on, may cause the conflict on constraint (3) and (4). Therefore, this paper considers the concept of JOC described in Bierwirth (1995) and ameliorates the procedure of JOC, which is demonstrated as follows:

Given two chromosomes,

Task	1	2	3	4	5
(parent) *chro*^{a}	**4***	3	**1***	**2***	5
(parent) *chro*^{b}	2	1	4	5	3

and then, it takes 3 steps to finish JOC. First, copy the gene sections with * into *chro*^{c}.

(offspring) *chro*^{c}	**4***		**1***	**2***	

We can realize that 1st, 2nd and 4th workers are already assigned to specific tasks, and 3rd, 5th workers are remaining. Second, check the remaining gene segments in *chro*^{b} if there are any workers can be assigned to the same task in *chro*^{c}. We may notice that 3rd worker in *chro*^{b} can goes to 5th task in *chro*^{c} directly. Then, *chro*^{c} will be:

(offspring) *chro*^{c}	**4***		**1***	**2***	3

Third, assign the rest of workers into *chro*^{c}.

(offspring) *chro*^{c}	**4***	5	**1***	**2***	3

If there is more than one remaining workers needed to be assigned in 3rd step, those workers will be assigned one by one in order.

JOC has a good property to keep gene segments at the same position from both *chro*^{a} and *chro*^{b} without any assignment conflict. Much more detail will be mentioned in the following pseudocode:

Procedure 5 JOB-ORDER-CROSSOVER

```
Input $chro, C, D$
```

```
Output $kid\_chro$
$G$ = PRIORITY-RANK-TO-GENE($C, D$)
let $K$ be a $V.length$ array
# Step 1
for $j$ in $G$
    $W$ = all workers except who has been assigned in $K$
    $C' = \{C_{ij} | i \in W\}$
    $i$ = worker of $\min(C')$
    $K[i] = j$
# Step 2
for $j$ in $V \setminus G$
    $i = chro[j]$
    if $K[i]$ has been assigned
        continue
    $K[i] = j$
# Step 3
$U$ = unassigned worker in $chro$
for $index, j$ in $U$
    $i = index^{th}$ unassigned worker of $K$
    $K[i] = j$
return $K$
```

5. Mutation

   Mutation is a genetic operator used to maintain genetic diversity from one generation to the next. hGA applies 2-point mutation, which will swap any two gene sections randomly, to obtain the global search procedure.

```
Procedure 6 MUTATION
Input $chro$
Output $chro$
randomly select $r$ in $[0, 1]$ under uniform distribution
if $r \leq 0.6$
    randomly select $i$ in $\{1, \dots, chro.length\}$ under uniform distribution
    randomly select $j$ in $\{1, \dots, chro.length\}$ under uniform distribution
    swap $chro_i$ for $chro_j$
return $chro$
```

Related methodologies

hGA utilizes two popular algorithms in scheduling and assignment to enhance the performance. The first one is critical path method (CPM). A critical path is determined by identifying the longest stretch of dependent activities and measuring the time required to complete them from start to finish.

CPM is developed by M. R. Walker and J. E. Kelley Jr. (1950), which assists users in quickly identify its longest series of dependent activities, the longest path of the program. According to CPM, we can realize which task does not have flexible time to spare. Thus, if we improve the processing time of these tasks, the makespan would reduce. The second one is Hungarian Algorithm (HA). HA is a combinatorial optimization algorithm that solves the assignment problem, developed by Harold Kuhn (1955). Based on the concept of these two algorithms, we propose two related methodology as follows:

1. Critical path mutation

   The concept of critical path mutation is quite simple. hGA will find out the critical path under the current assignment result, and apply 2-point mutation to those tasks on the critical path, so called 'critical path mutation'.

```
Procedure 7 CRITICAL-PATH-MUTATION
Input $chromosome, G(V, E)$
Output $kid\_chromosome$
1  get the critical path $A$ of $G$
2  randomly select $s$ in $\{1, \dots, |A|\}$ under uniform distribution
3  $A'$ = randomly select $s$ elements of $A$ under uniform distribution
4  $k' = \{chro_j, \forall j \in A'\}$
5  $k'$ = MUTATION($k'$)
6  $k = k' + \{chro_j, \forall j \in A \backslash A'\}$
7  return $k$
```

2. Hungarian boosting

   Hungarian boosting combines CPM with HA. First, find out the critical path under the current assignment result. Second, reassign the workers, who work on the task passed by the critical path, with HA. This process will optimize the assignment on the critical path so that gradually reduce the makespan.

```
Procedure 8 HUNGARIAN-BOOSTING
Input $chro$, G(E, V)
Output $kid\_chro$
1  let $S$ be an empty set
2  $max_{iter} = 10$
3  $X = chro$
4  for $i = 0$ to $max_{iter}$
5      get the critical path $A$ of $G$ in deterministic simulation
6      if $A \in S$
7          break
8      $S = S \cup \{A\}$
9      $W = \{X_w | w \in A\}$
10     $C' = \{C_{wj} | w \in W, j \in A\}$
```

```
11      $X$ update by HUNGARIAN-ALGORITHM($C'$)
12  return X
```

3. Critical path based Hungarian Algorithm (CPHA)

   CPHA is used to generate the initial population of hGA, and it is quite similar to Hungarian boosting, which will optimize the assignment result on the critical path. However, CPHA will apply HA to different path in production routing over and over again so as to minimize the makespan on the different path. For instance, in Fig. 2, there are 4 possible paths, which are $path_1: (1, 3, 5, 6, 10)$, $path_2: (1, 3, 5, 6, 9)$, $path_3: (1, 4, 7, 8, 9)$, and $path_4: (2, 7, 8, 9)$. Then, CPHA will randomly choose an order of those paths, like $(path_1, path_4. path_3, path_2)$, and apply HA to $path_1, path_4$, and so on. The pseudo-code is shown to demonstrate the detail of CPHA:

```
Procedure 9 CRITICAL-PATH-BASED-HUNGARIAN-ALGORITHM
Input C, G(V, E)
Output kid_population
1   let $K$ be a $V.length$ array
2   $T$ = ALL-PATH($G$)
3   for $M$ in all permutation of $T$
4       let $X$ be a new $V.length$ array
5       $S = V$
6       $W = \{x | x \in X\}$
7       for $A$ in $M$
8           $C' = \{C_{ij} | i \in W, j \in A\}$
9           $O$ = HUNGARIAN-ALGORITHM($C', G$)
10          for $j$ in $A \cap S$
11              $X_j = O_j$
12          $S = S \backslash A$
13          $W = W \backslash \{x | x \in X\}$
14      $K = K \cup \{X\}$
15  return $K$
```

The following section will demonstrate each procedure of hGA:

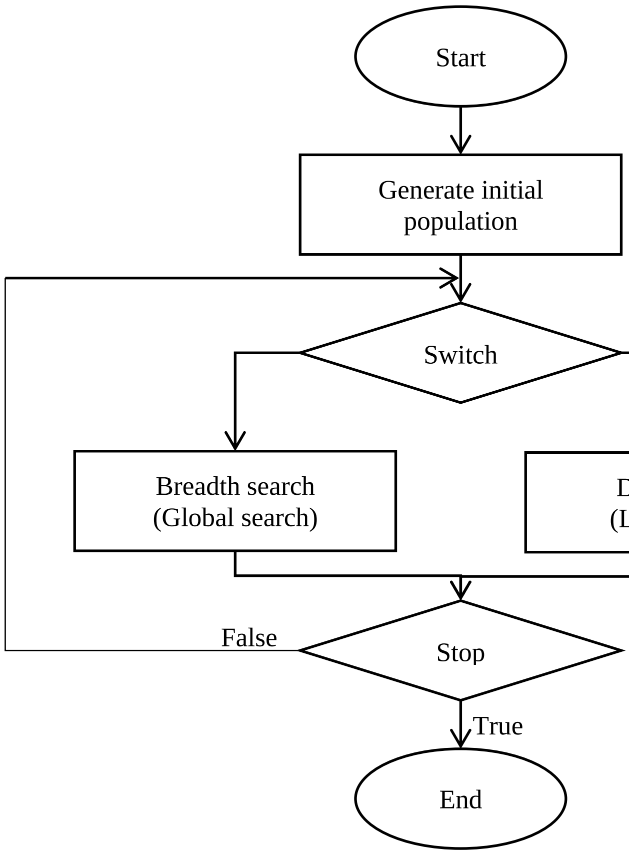

Start
Generate initial population
Switch
Breadth search (Global search)
Depth search (Local search)
False
Stop
True
End

Fig. 4. The flowchart of hGA

Fig. 4 shows that 5 main procedures are involved, which are generating initial population, breadth search, depth search, switch condition, stop condition. For generating initial population, hGA adopts CPHA as mentioned in the section of 'related methodology', which provides adequate population as a good start.

In the stage of breadth search, hGA utilizes the priority decision rule to specify the priority rank of all tasks, and assigns roughly 40 percent of workers into the tasks of high priority in order to gain the incomplete chromosome, so called 'specified chromosome'. For the remaining workers without tasks, hGA takes advantage of roulette-wheel selection to select a chromosome from the previous population, and applies JOC to specified chromosome and the selected chromosome so as to produce a new chromosome. In the end, the new chromosome will go through the mutation process with the mutation rate 0.6. Breadth search will continue until the sufficient chromosomes are generated for the next generation.

Algorithm 1 BREADTH-SEARCH

```
input $pop$
output $kid\_pop$
1    $K = \{\}$
2    # Copy
3    $sorted\_pop = pop$ sorted by fitness value in ascending order
4    for $i = 1$ to $p\_size \times \alpha_{copy}^{out}$
5        $K = K \cup \{sorted\_pop[i]\}$
```

```
6    # JOC
7    S = ROULETTE-WHEEL-SELECT-POPULATION($pop, p\_size, \alpha_{joc}^{in}$)
8    for $i = 1$ to $p\_size \times \alpha_{joc}^{out}$
9        $chro_1$ = ROULETTE-WHEEL-SELECTION($S$)
10       $chro_2$ = ROULETTE-WHEEL-SELECTION($S$)
11       $k_1, k_2$ = JOB-ORDER-CROSSOVER($chro_1, chro_2$)
12       $k_1$ = MUTATION($k_1$)
13       $k_2$ = MUTATION($k_2$)
14       $K = K \cup \{k_1, k_2\}$
15   return $K$
```

The benefit of breadth search is that the population will be much more diverse while using JOC and mutation. It can broaden the searching area in the solution space as a global search procedure.

In the stage of depth search, there are four kinds of cases of reproduce strategy working at the same time, and two methodologies are included, which are mentioned in the section of 'related methodologies'. First is critical path mutation, and second is Hungarian boosting. The first strategy is copying the top 10 percent chromosome from the previous population. About 40 percent of chromosomes will take the second strategy, which relies on Hungarian boosting to optimize the assignment result on the critical path. The third strategy exploits critical path mutation to about 30 percent chromosomes. The rest of the chromosomes, about 20 percent, will go through JOC and mutation, just like the procedure in breadth search.

Algorithm 2 DEPTH-SEARCH

```
input $pop$
output $kid\_pop$
1    $K = \{\}$
2    # Copy
3    $sorted\_pop = pop$ sorted by fitness value in ascending order
4    for $i = 1$ to $p\_size \times \beta_{copy}^{out}$
5        $K = K \cup \{sorted\_pop[i]\}$
6
7    # Job order Crossover
8    S = ROULETTE-WHEEL-SELECT-POPULATION($pop, p\_size, \beta_{joc}^{in}$)
9    for $i = 1$ to $p\_size \times \beta_{joc}^{out}$
10       $chro_1$ = ROULETTE-WHEEL-SELECTION($S$)
11       $chro_2$ = ROULETTE-WHEEL-SELECTION($S$)
12       $k_1, k_2$ = JOB-ORDER-CROSSOVER($chro_1, chro_2$)
13       $k_1$ = MUTATE($k_1$)
14       $k_2$ = MUTATE($k_2$)
15       $K = K \cup \{k_1, k_2\}$
```

```

# **Critical path based Mutate**
S = ROULETTE-WHEEL-SELECT-POPULATION($pop, p\_size, \beta^{in}_{cri-mutate}$)
**for** $i = 1$ **to** $p\_size \times \beta^{out}_{cri-mutate}$
    $k$ = CRITICAL-PATH-MUTATION($S$)
    $K = K \cup \{k\}$

# **Hungarian Boosting**
S = ROULETTE-WHEEL-SELECT-POPULATION($pop, p\_size, \beta^{in}_{hung-boost}$)
**for** $i = 1$ **to** $p\_size \times \beta^{out}_{hung-bosst}$
    $k$ = HUNGARIAN-BOOSTING($S$)
    $K = K \cup \{k\}$
**return** $K$
```

It is obvious that only few chromosomes will experience crossover process in depth search, which is because depth search emphasizes re-optimization of the current chromosome, instead of increasing the diversity between chromosomes. The advantage of depth search is to gradually increase the quality of current solution while making use of Hungarian boosting and critical path mutation.

The switch condition between breadth search and depth search is simple. Either breadth search or depth search cannot lessen the makespan of the production routing within 5 consecutive generations, then hGA will switch to the other searching approach. Last but not least, the stopping condition in hGA follows the max_gen in hGA components. If the number of generation reaches the max_gen, then hGA will stop.

IV. Program Implementation of Sample Average Approximation (SAA)

The principle of Sample Average Approximation (SAA) allows one to tackle such problems with sampling and optimization methods for deterministic problems. In addition, SAA method is easily implementable and can be surprisingly efficient for some classes of stochastic programming problems with convergence properties. To improve the convergence and wisely use the computing capacity, we infuse SAA method into hGA. Fig. 5 indicates the concept of implementation of hGA in SAA format.

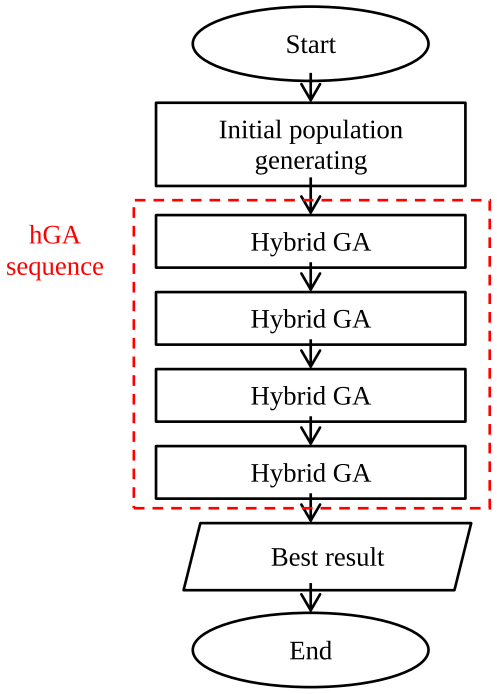


Fig. 5. The flowchart of hGA in SAA format

In Fig. 5, the procedure re-runs hGA for several times, so called 'hGA sequence', and meanwhile, we increase the population size, iteration time, and maximum generation to get close to the optimal solution gradually. In hGA sequence, we do not consume too much computing resource at first. As the procedure has finished more and more layers of hGA, we enlarge the population size, iteration time and so on in order to enhance the accuracy and quality of solutions.

V. Experimental Research and Analysis

Two representative instances are selected for simulation to test the performance of the proposed hGA in this paper. The first one is the 10 tasks production routing showed in Fig. 2, and another one is the 20 tasks production routing shown in Appendix B. For the sake of simulating dynamic processing time in real-world system, we design two kinds of cost table with large and small variance for each instance. The elements in those cost tables are uniformly distributed over the half-open interval shown in Appendix C to E. The results obtained by hGA are compared with those derived from original GA and brute force. All the simulation experiments are performed with AMD Ryzen 1700@3.35GHz, 32GB RAM, Windows 10 1909, Python 3.7.0 64bit. The adopted parameters of hGA in each layer of SAA implementation are listed in TABLE 1:

TABLE 1. Parameters of hGA

Layer / parameter	**1**	**2**	**3**	**4**
p_size	50	100	200	100
iter_time	10	20	400	2000

max_gen	40	20	10	3

The experimental simulations are run for 1000 times. Over 30 percent runs converge to the same chromosome, whose assignment is pretty close to the optimal assignment result. TABLE 2. summarizes the results of experiments on hGA, original GA and brute force:

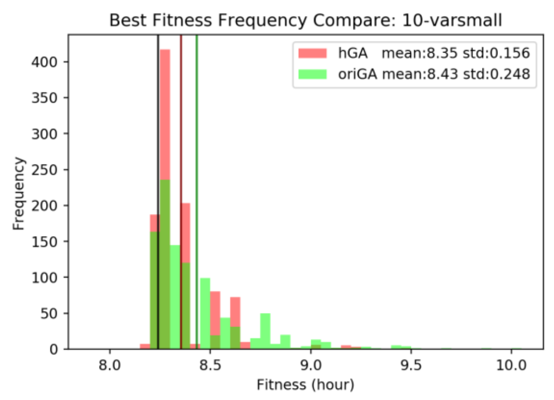
TABLE 2. Performance of hGA and others.

Problem		**10 tasks**		**20 tasks**	
		Small variance	**Large variance**	**Small variance**	**Large variance**
hGA	mean of solution	8.35	349.830	24.15	476.00
	standard deviation of solution	0.156	6.707	0.73	6.562
	optimality gap[a]	97.3%	82.4%	NA[b]	NA[b]
Original GA	mean of solution	8.43	359.672	31.75	522.910
	standard deviation of solution	0.248	8.946	2.251	13.082
	optimality gap[a]	85.7%	32.9%	NA[b]	NA[b]
Brute force	Optimal value	8.24	338.666	NA[b]	NA[b]

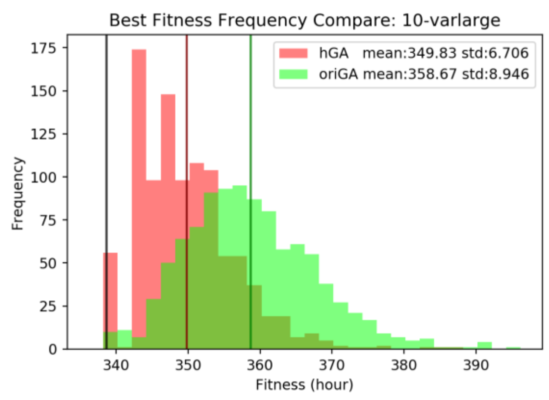
[a] The probability to reach the optimality gap ($^{|opt_{found} - opt_{real}|}/_{opt_{real}}$) which is less than 5%.

[b] It has no chance to get the optimal solution for 20 tasks assignment problem via brute force.

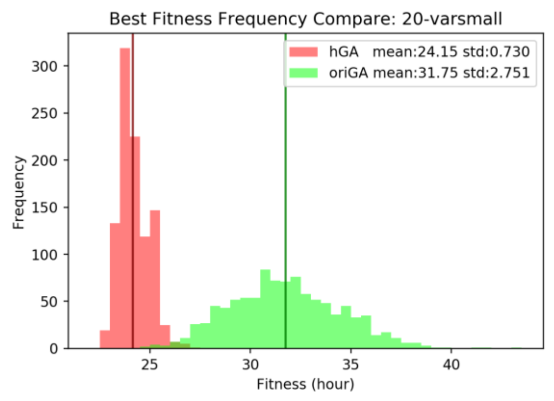
Fig. 6 visualizes the quality of solution on hGA and original GA:



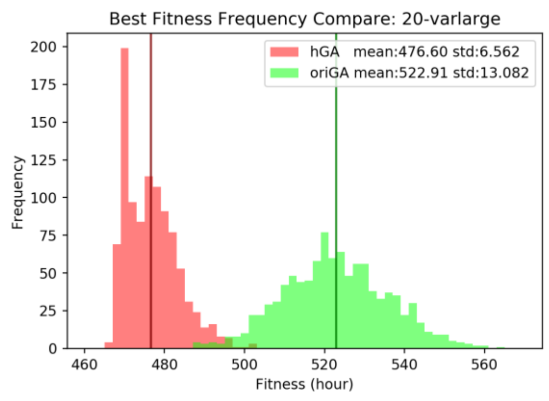
(a) Problem: 10 tasks with small variance



(b) Problem: 10 tasks with large variance



(c) Problem: 20 tasks with small variance



(d) Problem: 20 tasks with large variance

Fig. 6. The histogram of performance of hGA and others. The black line in the 10 tasks situation represents the fitness value of optimal solution discovered by the brute force algorithm.

In Fig. 6, it shows that the distribution of the solution from hGA skews to right and has higher kurtosis. The property of skewing to right implies hGA has the better performance to find the better assignment result to obtain the smaller fitness value, i.e. the makespan of PRTAP. For the property of higher kurtosis, it indicates that the solution derived from hGA is much more stable than others, that is, the variance of fitness value is smaller than that of original GA. Through the graphical result, we may realize that hGA has a better performance than original GA. In addition, we construct hypothesis testing with $\alpha = 0.05$ to determine whether there is enough evidence to conclude that the real mean of fitness value derived from hGA is smaller than that from original GA. In the end, the testing in all instance both has p-value less than 0.05. We may summarize that hGA is better than original GA indeed.

VI. Conclusion

This paper studies the genetic algorithm to solve production routing task assignment problem, and designs a significantly different approach based on Sample Average Approximation to find a better solution under restricted computational resources. The hybrid genetic algorithm in this paper has simple encoding method, and improves genetic crossover to avoid illegal solution. Also, due to Sample Average Approximation, this paper introduces a sequence of genetic algorithm to benefit the searching efficiency. Furthermore, the Critical path based Hungarian Algorithm provides good initial population in reasonable computing time. In addition, switching two modes, breadth search and depth search, in hybrid genetic algorithm among all generation improves the computation efficiency to search for the better solution. To sum up, the algorithm could be applied to solve production routing task assignment problem, which could be extended to the large campaign, such as dispatching human resource in Olympics, and so forth.

Acknowledgement

We would like to acknowledge the role of Dr. Chang, Department of Industrial Engineering and Engineering Management, National Tsing Hua University for efforts towards result validation, cross verifications, and guidance.

REFERENCES

[1] Zhang, H., & Gen, M. (2005). Multistage-based genetic algorithm for flexible job-shop scheduling problem. Journal of Complexity International, 11(2), 223-232.

[2] Zhang, Y., Ogura, H., Ma, X., Kuroiwa, J., & Odaka, T. (2014). A Genetic Algorithm Using Infeasible Solutions for Constrained Optimization Problems. Open Cybernetics & Systemics Journal, 8, 904-912.

[3] Kim, S., Pasupathy, R., & Henderson, S. G. (2015). A guide to sample average approximation. In Handbook of simulation optimization (pp. 207-243). Springer, New York, NY.

[4] González Fernández, M. Á., Rodríguez Vela, M. D. C., Sierra Sánchez, M. R., & Varela Arias, J. R. (2010). Tabu search and genetic algorithm for scheduling with total flow time minimization. In Proceedings of the workshop on constraint satisfaction techniques for planning and scheduling problems (COPLAS). Association for the Advancement of Artificial Intelligence (AAAI).

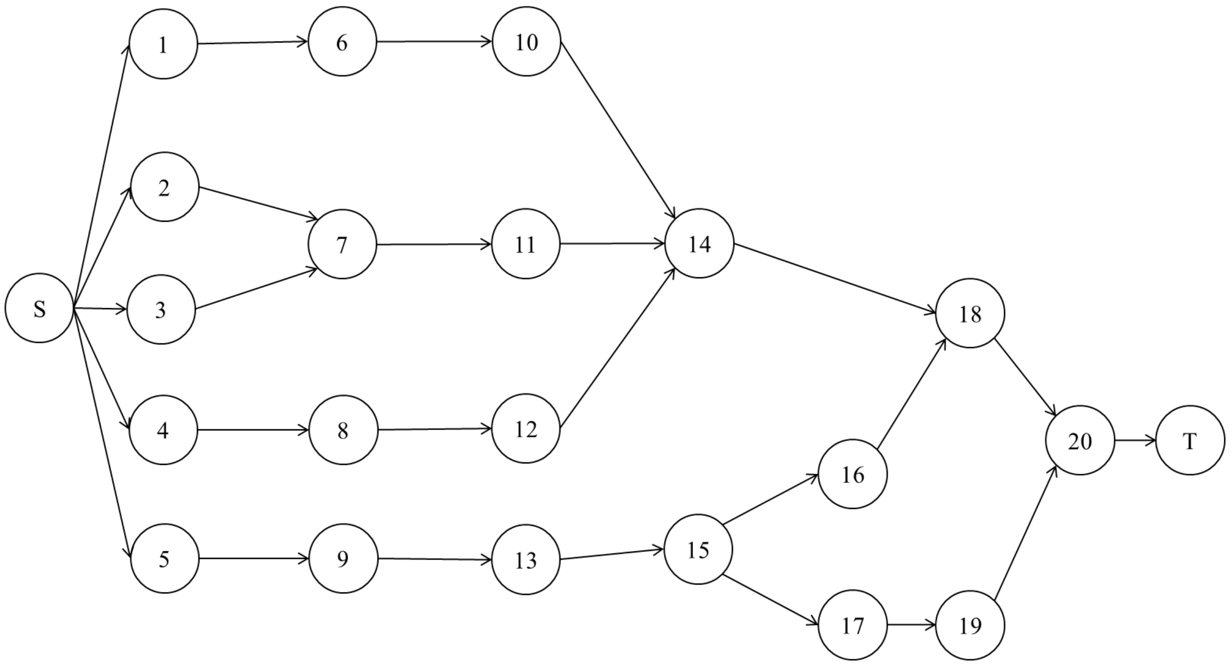
Appendix A

Cost table for 10 tasks production routing. The elements are uniformly distributed over the half-open interval [0, 10).

Worker/Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	7.3	2.7	1.5	0.1	1.3	5.6	5.4	0.7	8.7	4.5
2	4.0	1.6	9.5	5.6	0.7	1.0	7.0	3.2	9.2	3.9
3	3.6	0.2	3.8	2.3	9.2	7.3	0.3	6.9	3.6	8.0
4	9.1	5.1	8.8	6.6	3.3	3.3	9.7	6.8	2.5	6.1
5	7.3	2.4	8.2	1.8	2.0	0.7	4.5	2.0	4.4	3.4
6	6.0	8.1	7.4	0.6	0.6	0.3	6.3	3.9	7.2	3.5
7	9.7	1.5	3.3	3.7	7.4	5.5	1.1	8.3	7.1	0.9
8	7.8	0.1	3.5	6.1	6.9	7.3	1.1	7.9	0.7	6.9
9	0.4	9.7	4.2	3.9	8.6	5.2	4.0	9.9	2.7	2.0
10	5.9	4.5	7.7	2.6	7.1	6.2	3.4	7.2	3.0	1.5

Appendix B

Twenty tasks production routing for the performance validation of different algorithms.



Appendix C

Cost table for 10 tasks production routing with large variance. The elements are uniformly distributed over the half-open interval [50, 80).

Worker/Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	61.7	50.0	52.6	55.9	79.9	58.1	57.0	51.6	59.0	61.3
2	55.9	63.0	79.5	76.1	78.3	54.5	77.9	62.0	52.2	77.4
3	61.0	65.2	51.7	74.0	78.9	59.6	64.8	60.3	51.6	61.9
4	71.2	77.5	58.1	58.3	56.4	74.1	52.3	73.0	59.6	78.2

5	57.9	70.3	55.9	76.0	67.5	65.4	56.0	78.6	50.9	77.2
6	63.2	57.9	52.7	63.5	52.9	52.0	74.2	68.2	67.0	63.2
7	55.6	50.3	67.8	57.1	77.9	69.9	50.7	50.8	51.8	65.4
8	76.4	63.8	67.1	68.6	77.5	64.0	75.9	67.2	58.7	62.8
9	65.1	70.0	54.0	73.9	67.3	50.4	77.5	57.0	76.1	69.3
10	51.8	65.3	55.3	75.7	62.3	73.8	69.6	66.7	76.0	54.1

Appendix D

Cost table for 20 tasks production routing with small variance. The elements are uniformly distributed over the half-open interval [0, 10).

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	6.86	6.16	7.90	4.41	2.98	3.13	8.99	6.95	2.16	8.73	8.49	5.12	6.87	5.42	4.38	7.36	8.22	4.18	4.70	7.82
2	5.60	8.58	4.23	2.95	4.99	2.65	6.07	3.94	3.57	4.80	6.30	8.04	7.07	3.09	7.80	8.66	4.11	7.80	7.75	9.77
3	6.50	6.87	7.86	2.03	4.37	6.83	7.35	5.37	8.01	2.58	3.49	8.40	6.32	4.10	7.09	7.65	7.50	7.77	2.60	5.12
4	6.56	6.21	6.43	4.27	5.23	9.10	6.30	8.52	9.16	6.44	5.44	4.47	4.26	6.13	2.34	8.55	8.69	8.02	3.22	7.72
5	2.94	8.88	5.17	8.24	4.67	3.38	7.39	2.82	8.46	5.74	9.69	9.34	7.23	9.89	9.94	3.41	7.88	3.19	7.85	9.63
6	6.41	8.10	2.66	4.44	4.11	9.49	3.76	5.12	9.99	4.54	8.02	7.66	5.76	4.31	8.83	3.69	7.79	2.30	9.37	2.73
7	5.71	2.67	9.86	8.20	6.01	8.93	4.30	7.93	2.47	5.16	9.81	9.65	3.14	9.15	7.15	8.36	5.60	3.98	2.79	3.94
8	3.81	9.55	7.01	8.68	3.61	7.14	8.55	4.03	5.66	9.42	9.62	8.40	8.85	7.57	3.27	2.09	6.19	6.08	2.36	2.60
9	7.40	7.62	8.34	2.66	9.72	6.91	7.24	9.41	5.48	2.66	3.03	5.56	8.83	2.08	5.61	3.61	9.70	6.32	3.37	2.31
10	2.18	4.18	4.06	9.50	2.16	4.76	7.96	5.11	8.52	5.63	5.86	5.99	6.98	7.98	9.49	2.86	4.25	8.13	2.43	4.55
11	7.18	7.14	5.98	3.88	6.05	6.91	5.98	9.32	9.50	5.76	5.14	6.56	7.74	5.77	2.56	7.91	9.36	5.32	7.51	5.94
12	4.82	2.88	8.74	3.23	9.23	6.16	7.74	7.45	2.90	5.58	5.87	7.48	2.12	7.85	3.86	2.72	8.60	5.83	9.22	9.68
13	9.90	3.61	8.05	7.72	6.73	9.24	7.48	7.90	9.58	8.75	7.88	4.20	8.69	2.15	7.21	4.93	9.36	6.54	9.69	7.55
14	5.31	9.44	6.94	7.39	7.65	7.45	2.74	8.15	3.36	7.40	7.94	8.08	3.78	4.20	5.80	4.42	5.40	2.37	8.70	7.74
15	6.45	3.62	5.29	4.39	4.32	9.24	8.66	9.38	5.29	2.13	9.48	8.74	8.02	9.40	8.52	7.35	5.39	2.97	8.18	8.53
16	7.73	2.32	6.68	8.61	6.52	5.60	3.74	2.29	6.62	7.36	9.05	6.17	8.82	6.96	3.14	5.91	2.39	8.17	9.56	6.30
17	3.30	4.08	9.45	6.91	6.68	3.91	7.91	9.44	3.58	9.22	6.76	2.73	5.38	6.30	6.18	3.92	3.28	2.01	5.34	8.99
18	2.74	3.81	8.46	7.93	8.59	3.97	4.31	9.83	5.95	3.21	7.16	9.85	2.15	3.54	5.44	8.20	5.81	6.22	8.91	9.32
19	9.10	3.17	8.30	3.22	9.02	5.41	3.32	5.16	7.98	9.60	6.96	7.51	5.32	5.28	5.09	2.86	9.36	4.48	2.62	3.12
20	6.73	7.63	8.91	7.70	4.42	6.63	3.74	2.50	3.51	8.37	5.65	8.00	8.30	7.84	4.27	6.71	3.09	2.35	6.48	5.03

Appendix E

Cost table for 20 tasks production routing with large variance. The elements are uniformly distributed over the half-open interval [0, 10).

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	74.3	73.3	55.7	79.9	59.8	59.9	74.2	75.1	63.3	50.8	62.4	77.6	64.0	66.5	69.2	59.2	65.3	66.1	65.9	53.5
2	50.5	79.2	55.4	78.6	67.0	66.4	71.2	78.6	74.0	60.1	73.3	68.2	61.8	70.2	55.0	78.2	52.9	67.0	60.7	51.9
3	59.3	78.0	56.9	51.9	70.2	72.4	59.7	68.6	50.1	73.6	58.0	65.0	73.3	54.0	53.4	79.1	50.0	76.6	73.3	68.9
4	60.3	70.6	75.5	71.3	64.5	58.9	75.1	71.8	58.0	76.5	50.2	77.3	53.4	56.2	58.6	77.8	54.4	65.1	71.0	69.7
5	66.0	74.9	76.3	65.4	61.8	79.3	68.2	70.9	67.8	51.9	73.2	52.3	61.3	77.0	58.7	51.6	63.5	73.8	71.8	53.1
6	58.2	79.3	73.1	66.5	50.3	71.0	60.2	69.8	73.4	69.6	60.7	60.8	62.3	78.0	57.5	73.7	61.8	54.7	55.1	67.1
7	72.7	75.9	52.6	73.4	68.2	68.8	62.0	63.0	77.0	72.9	50.9	51.0	63.1	62.3	66.3	51.3	79.6	75.2	72.3	57.5
8	65.1	66.4	59.1	51.9	68.3	75.0	78.6	73.9	51.8	73.0	58.5	73.1	78.2	68.1	50.2	63.7	67.0	69.4	52.0	61.0
9	61.5	65.8	78.0	65.5	72.9	60.3	69.6	53.9	59.6	54.1	69.7	63.5	65.4	72.0	53.3	79.4	66.0	71.3	65.9	62.6
10	68.8	53.6	53.0	73.8	74.1	65.5	50.3	60.0	60.0	72.4	53.5	67.7	58.8	51.1	60.8	55.1	77.1	64.7	61.2	77.8
11	65.7	63.8	56.8	63.2	74.5	74.0	78.1	66.2	56.7	57.8	77.9	51.7	71.6	50.6	53.9	64.4	59.8	71.1	67.5	78.0
12	56.8	55.5	54.7	57.4	50.5	73.2	57.7	78.4	71.9	54.5	61.1	55.2	72.0	71.8	64.6	72.9	72.4	65.9	76.4	74.2
13	64.2	67.1	76.2	76.2	71.8	55.4	52.6	78.0	76.6	53.8	67.7	52.4	50.9	72.9	62.1	79.9	68.2	64.2	59.7	57.5
14	65.2	58.2	70.5	51.4	79.4	68.0	57.1	79.9	63.5	61.6	75.3	62.0	76.2	66.8	76.4	50.6	67.8	56.4	57.9	50.0
15	59.2	79.4	77.3	67.3	56.2	51.6	69.0	77.6	68.3	63.7	79.5	52.6	64.6	71.7	74.1	50.7	58.3	78.2	64.3	62.7
16	53.2	69.2	72.7	73.4	76.8	53.8	67.8	78.7	62.3	76.3	57.2	71.7	79.7	78.9	66.2	78.2	70.6	76.0	58.2	61.4
17	70.7	62.5	77.4	61.5	64.1	76.3	52.3	76.3	54.8	63.3	51.7	61.0	64.8	54.9	77.3	56.0	77.9	59.4	62.4	65.1

18	79.5	57.0	71.3	73.4	74.0	74.9	60.4	58.7	67.9	77.1	66.6	55.8	67.6	75.1	61.4	68.0	61.6	61.7	72.4	56.0
19	77.1	74.2	51.6	56.8	50.5	66.7	60.3	57.8	73.9	72.3	60.1	53.2	73.8	75.9	51.8	74.0	54.8	52.0	58.7	78.5
20	54.2	69.8	57.8	77.1	69.0	78.3	66.7	77.1	69.1	78.0	73.8	72.6	65.2	64.5	64.5	53.6	72.8	78.0	67.5	66.0