International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:2, No:3, 2008

Solving Machine Loading Problem in Flexible Manufacturing Systems Using Particle Swarm Optimization

S. G. Ponnambalam, and Low Seng Kiat

***Abstract*—**In this paper, a particle swarm optimization (PSO) algorithm is proposed to solve machine loading problem in flexible manufacturing system (FMS), with bicriterion objectives of minimizing system unbalance and maximizing system throughput in the occurrence of technological constraints such as available machining time and tool slots. A mathematical model is used to select machines, assign operations and the required tools. The performance of the PSO is tested by using 10 sample dataset and the results are compared with the heuristics reported in the literature. The results support that the proposed PSO is comparable with the algorithms reported in the literature.

***Keywords*—**Machine loading problem, FMS, Particle Swarm Optimization.

I. Introduction

FMS operational decisions consist of pre-release and post-release decisions. FMS planning problems also known as pre-release decisions take into account the pre-arrangement of parts and tools before the process of FMS begins. FMS scheduling problems, which come under the category of post-release decisions, deal with the sequencing and routing of the parts when the system is in operation [1]. The machine loading problem in a FMS is specified as to assign the machine, operations of selected jobs, and the tools necessary to perform these operations by satisfying the technological constraints (available machine time and tool slots constraint) in order to ensure the minimum system unbalance and maximum throughput, when the system is in operation [2]. An attempt has been made to solve the objective function simultaneously to bring the outcomes in close proximity to the real assumption of the FMS environment.

II. Literature Review

Since the objective of this paper is to propose an efficient evolutionary search heuristic to solve problem pertaining to job selection and machine loading in random FMS to optimize the system imbalance and throughput simultaneously, only the relevant literature are reviewed in this section.

Tiwari and Vidyarthi [1] proposed a Genetic Algorithm (GA) based heuristic to solve the machine loading problem of a random type FMS. The proposed GA-based heuristic determines the part type sequence and the operation- machine allocation that guarantee the optimal solution to the problem, rather than using fixed predetermined part sequencing rules.

Swarnkar and Timari [2] proposed a generic 0-1 integer programming formulation and a hybrid algorithm based on tabu-search and simulated annealing (SA) is employed to solve the problem.

Prakash et al. [3] proposed a special Immune Algorithm (IA) named "Modified immune algorithm (MIA)". This method is capable of learning and memory acquisition. This method improves some issues inherent in existing IAs and proposes a more effective immune algorithm with reduced memory requirements and reduced computational complexity.

Chan et al. [4] proposed a fuzzy goal programming approach to model the machine tool selection and operation allocation problem of flexible manufacturing systems. The model is optimized using an approach based on artificial immune systems and the results of the computational experiments are reported.

Tripathi et al. [5] proposed a multi-agent-based approach for solving the part allocation problems in flexible manufacturing systems (FMS) that can easily cope with the dynamic environment.

Akhilesh Kumar et al. [6] extended the simple genetic algorithm and proposed a new methodology, constraint based genetic algorithm (CBGA) to handle a complex variety o f variables and constraints in a typical FMS-loading problem.

Yogeswaran et al. [7] proposed a hybrid algorithm using genetic algorithm and simulated annealing {GASA) algorithm for his problem. They also proposed efficient machine loading heuristics.

The machine loading problem of a flexible manufacturing system is well known for its complexity. This problem encompasses various types of flexibility aspects pertaining to part selection and operations assignments along with constraints ranging from simple algebraic to potentially very complex conditional constraints. In this paper, a particle swarm optimization algorithm is proposed to solve this

S. G. Ponnambalam is with the Monash University, Sunway Campus, 46150 Bandar Sunway, Malaysia (phone: +60-3-55146203; fax: +60-3-55146207; e-mail: sgponnambalam@eng.monash.edu.my).

Low SengKiat was with Monash University, Sunway Campus, 46150 Bandar Sunway, Malaysia. He is now working as Instrumentation and Control Engineer at Kencana Petroleum, Malaysia.

problem. The literature survey clearly supports the proposal of an efficient heuristic to this problem. Besides that, the justification to adopt particle swarm optimization is mainly due to its performance in solving scheduling problems. The advantages of PSO are that PSO is easy to implement and there are only few parameters to adjust. So far, PSO has not been tried to solve the machine loading problem. Problem Environment

The FMS under consideration in this paper consists of a number of multifunctional CNC machines, tools with the potential to execute several operations. The jobs are available in batches and arrive in random sequences with different requirements for processing. The batch size, number of operations, processing time and number of tool slots needed for each job is known initially. There are-two types of operations accessible for a job namely:

Essential operation – job can be only performed in a particular machine

Optional operation – job can be performed in a number of machines available

Optional operation gives the flexibility in routing of the jobs. The FMS considered has four multifunctional machines with each having 480mins of available processing time (8hrs = 1 shift) and 5 tool slots. The notations used in this paper are given in Table I.

III. Objective Function and Constraints

The objective functions and the constraints are discussed in this section. These are adapted from reference [2].

Minimize system unbalance: equals to the sum of the idle time remaining on the machines after allocation of all feasible jobs. The value of system unbalance must be either 0 (100% utilization of the system) or a positive value.

$$Maximize\_F1 = \frac{M \times H - \sum_{j=1}^{M} \left(UT_j - OT_j \right)}{M \times H} \quad (1)$$

Maximize throughput: is equals to the sum of batch size for all the selected jobs during the planning horizon.

$$Maximize\_F2 = \frac{\sum_{i=1}^{N} B_i \times x_i}{\sum_{i=1}^{N} B_i} \quad (2)$$

Thus the overall objective function is:

$$Maximize\_COF = \frac{M \times H - \sum_{j=1}^{M} \left(UT_j - OT_j \right)}{M \times H} + \frac{\sum_{i=1}^{N} B_i \times x_i}{\sum_{i=1}^{N} B_i} \quad (3)$$

TABLE I
Notations

symbol	Meaning
i	index of job; $1 \le i \le N$
j	index of machines; $1 \le j \le M$
N	number of jobs, swarm size
M	number of machines
Bi	batch size of job i
Xi	= 1 if job i is selected or = 0 otherwise
H	length of scheduling period
UTj	underutilized time in machine j
OTj	overutilized time in machine j
SU	system unbalance
TH	Throughput
RTMj	remaining time on machine j
RTSj	remaining tool slot on machine j
S	randomly generated job sequence
AS	assigned jobs
UAS	unassigned jobs
RTMj	Remaining time on machine j
RTSj	Remaining number of tool slots in machine j
t	Iteration number
P_k^t	Position of k^{th} particle at time step t, $(k = 1,N)$
${}^{e}P_k^t$	Position of the best previous position of the k^{th} particle at time step t.
$Z\left({}^{e}P_k^t\right)$	Objective function value of the sequence represented by the position of the particle ${}^{e}P_k^t$.
G^t	The global best particle position at time step t having the minimum objective function value, i.e. $min\ \left\{Z\left({}^{e}P_k^t\right)\right\} k = 1,N\}$.
v_x^t	Velocity of the particle x at time step t.
$\|\| v_x^t \|\|$	Length of the list of transpositions of the particle x at time step t.
C_1, C_2, C_3	Learning coefficients

The objective function prescribed in equation (3) is subjected to the constraints detailed below.

- System unbalance: equals to the sum of the idle time remaining on the machines after allocation of all feasible jobs. System unbalance should be bigger than or equals to 0 (100% of utilization of the system).
- Tool slots: number of slots needed for the operation of the jobs to be performed on a machine must be always be less than or equals to the tool slots available in that machine.
- Unique job routing: despite the flexibility existing in the selection of the machine for optimal operations, once a machine is selected, the operation has to be completed on the same machine.
- Non-splitting of job: once a job is considered for processing, all the operations are to be completed before undertaking a new job.
- Sharing of the tool slots is not considered.

- Number of pallets and fixtures used in the system are sufficient and readily available.
- Parts are readily available on machines so material handling time is negligible.

IV. Proposed Algorithm

PSO is a population-based, bio-inspired optimization method. It was originally inspired in the way crowds of individuals move towards predefined objectives, but it is better viewed using a social metaphor. Individuals in the population try to move towards the fittest position known to them and to their informants, that is, the set of individuals that are their social circle [8]. The objective is to maximize a fitness function. The structure of the proposed PSO algorithm is presented in Fig. 1.

$t \rightarrow 0;$
$for\ (k = 1,N)$
    Generate P_k^t;
    Evaluate $Z(P_k^t)$;
    ${}^eP_k^t \rightarrow P_k^t$
$G^t \rightarrow P_k^t\ having\ \max\ \{Z({}^eP_k^t), k = 1,N\};$
$for\ (k = 1,N)$
    Initialise v_k^t;

//iterative improvement process
do {
    $for\ (k = 1,N)$
        update Position P_k^{t+1};
        update velocity v_k^{t+1};
    Apply local search on all particle positions;
    Evaluate all particles;
    update ${}^eP_k^{t+1}$ and G^{t+1}, $(k = 1,N)$;
    $t \rightarrow t+1$;
} $(while\ t < t_{max})$
$Output\ G^t$

Fig. 1 The proposed PSO algorithm

V. Operations Performed to Update Position and Velocity

Various operations performed for computing particle velocity and updating particle positions are explained below:

Subtraction (position - position) operator: Let x_1 and x_2 be two positions representing two different sequences. The difference $x_2 - x_1$ is a velocity v. For example, subtracting two positions in the velocity update equation results in a velocity which is a set of transpositions.

Addition (position + velocity) operator: Let x be the position and v be the velocity. New position xi is found by applying the first transposition of v to p, i.e., $x_i = x + v$ then the second one to the result etc.

Addition (velocity + velocity) operator: Let v_1 and v_2 be two velocities. In order to compute $v_1 + v_2$, we consider the list of transpositions which contains first the 'ones' of v_1, followed by the 'ones' of v_2.

Multiplication (Coefficient X velocity) operator: Let c be the learning coefficient and v be the velocity, c x v results in a new velocity.

VI. Numerical Illustration

A numerical illustration of the proposed PSO algorithm is presented in this section. The readers are advised to refer [2] for the test problems used to evaluate the proposed algorithm. A sample test problem is shown in Table II.

TABLE II
Sample Problem

Job	Batch Size	Number of Operations	Operation	Machine	Unit Processing Time	Tool Slots	Total Processing Time
1	15	1	1	4	10	2	150
				2	12	2	180
2	10	2	1	1	20	1	200
			2	3	35	2	350
3	12	1	1	1	22	3	264
4	9	1	1	3,2	25	1	225
5	16	2	1	4	30	2	480
				2	25	1	400
				3	27	2	432
			2	1,4	16	1	256
6	11	1	1	2	21	3	231

A. Initialization

Parameters such as swarm size, number of generations (t), constants Cl, C2 and C3 must be initialized. The job/machine data (test problem) will be inputted in this phase. The swarm size used is equivalent to the number of jobs. The termination condition and the Cl, C2, C3 are identified after conducting sensitivity analysis.

B. Particle Generation

The seed sequence is created using the widely used SPT rule. Subsequently the swarm is generated from circular perturbation method [7]. The searching phase is iterated to the predetermined number of iterations. Lastly, the final optimum result will be obtained via the global best variable.

C. Velocity Initialization

After the swarm is initialized, each potential solution is assigned a velocity randomly. Length of velocity of each

particle $||v||$ is generated randomly between 0 and n. And the corresponding lists of transpositions $(i_q, j_q); q = 1, ||v_k||$ are generated randomly for each particle. The above formulation permits exchange of jobs $(i_1, j_1), (i_2, j_2)$ . . $(i_{||v||}, j_{||v||})$ in the given order. Each particle keeps track of the its improvement and the best objective function value achieved by the individual particles so far is stored as local best solution $\left({}^{e}P_k\right)$ and the overall best objective function achieved by all the particles together so far is stored as the global best solution (G_k). The iterative improvement process, as explained in Fig. 1 is continued afterwards.

D. Velocity and Position Update

Particles velocity is continuously updated using equation 4 and the particles position is continuously updated using equation 5.

$$v_k^{t+1} = c_1 U_1 v_k^t + c_2 U_2 \left({}^{e}P^t{}_k - P_k^t\right) + c_3 U_3 \left(G - P_k^t\right) \quad (4)$$

$$P_k^{t+1} = P_k^t + v_k^{t+1}. \quad (5)$$

Where, c_1, c_2, c_3 are integers and U_1 ,U_2 *and* U_3 are random numbers between 0 and 1.

For example, let P_k^t represents a sequence {2,3,4,1} with C1=1, C2=2, C3=2, $U_1 = 0.2$, $U_2 = 0.4$, $U_2 = 0.3$, $||v|| = 2$ and v = ((1, 4),(2, 3)) and ${}^{e}P_k^t$ and G^t be (1,4,3,2) and (3,1,4,2) respectively. Velocity of the particle k at time step $t+1$ namely v_k^{t+1} is obtained using equation 4 as follows.

v_k^{t+1} =1x 0.2 [(1,4),(2,3)] ⊕ 2 x 0.4 [(1,4,3,2) - (2,3,4,1)] ⊕ 2 x 0.3 [(3,1,4,2) –(2,3,4,1)]

Where [(1,4,3,2) – (2,3,4,1)] represents a velocity such that applying the resulting velocity to the current particle (2,3,4,1) yields a position (1,4,3,2).

Thus, v_k^{t+1} = 0.2 [(1,4), (2,3)] ⊕ 0.8 [(2,3), (1,4)] ⊕ 0.6 [(1,2), (1, 4)]
= ((1, 4),(2, 3),(1, 2))

Similarly, position of the particle k at time step $t+1$ namely P_k^{t+1} is obtained using equation 5 by applying v_k^{t+1} over P_k^t as follows.

P_k^{t+1} = (2,3,4,1) + ((1,4), (2,3),(1,2))
= (1,3,4,2) + ((2,3),(1,2))
= (1,4,3,2) + ((1,2))
= (1,4,2,3).

VII. Local Search Mechanism

The concept of the PSO algorithm is very simple and straight forward. It uses equations 4 and 5 to recursively update position of the particles till the termination criterion is met. In the present work, termination criterion is taken as 100 iterations of the PSO algorithm.

In some cases, it is found that PSO get stuck in local optima at an early stage and resulted in non-improvement during remaining iterations. In order to overcome the above problem and to improve its performance, two local search methods were adopted in the proposed PSO to improve the quality of the solution. They are the Job index Based Insertion Scheme (JiBIS) proposed by Varadharajan and Rajendran [9] and the Random Start Adjacent Swapping Scheme (RSASS) proposed by Chandrasekaran et al. [10].

Job Index Based Insertion Scheme

Let us consider a six job problem, whereby the current particle has the sequence, {3,2,5,1,4,6}. Hence, this sequence shall be the first SEED sequence. Element 1 is inserted into all positions within the sequence as the current index number is 1. The fitness value is evaluated for all particles in the population.

Iteration 1

Sequence 1 = {1,3,2,5,4,6}. Fitness (1)= 1.24
Sequence 2 = {3,1,2,5,4,6}. Fitness (2) = 1.56
Sequence 3 = (3,2,1,5,4,6). Fitness (3) = 1.63
Sequence 4 = {3,2,5,1,4,6}. Fitness (4) = 1.18 // Current SEED
Sequence 5 = {3,2,5,4,1,6}. Fitness (5) = 1.79 // SEED for next iteration
Sequence 6 = {3,2,5,4,6,1}. Fitness (6) = 1.50
Then element 2 is inserted into all positions and evaluated.

Iteration 2

Sequence 1 = {2,3,5,4,1,6} Fitness (1) = 1.55
Sequence 2 = {3,2,5,4,1,6} Fitness (2) =1.59 //Current SEED
Sequence 3 = {3,5,2,4,1,6} Fitness (3) = 1.27
Sequence 4 = {3,5,4,2,1,6} Fitness (4) = 1.33
Sequence 5 = {3,5,4,1,2,6} Fitness (5) = 1.88 //SEED for next iteration
Sequence 6 = {3,5,4,1,6,2} Fitness (6) = 1.52

similarly, after going through n = 6 iterations and fitness improvements, the JIBIS local search is intended to improve the fitness of the current particle executed in the PSO algorithm for every PSO iteration.

Position Based Local Search (PBLS)

A position based local search improvement algorithm is applied for the quick search of optimal solution. This local search is applied after finding local best and global best solutions.

1) Select a Particle P;
2) Initialize i =1, j = i+1; Select i
3) Swap i with j to generate a new sequence
4) Evaluate the new sequence;
5) if (new sequence is better)
replace P with the new sequence;
terminate the search process;
else

swap j with i;
j=j+1;
if (j = n)
i = i+1; j = i+1;
go to step 3

VIII. Results and Discussions

The performance of the proposed PSO and the local search heuristics in terms of the objective function value and the CPU times for the ten problems over the best performing heuristic (GASA proposed by Yogeswaran et al. [7]) so far reported in the literature are presented in Table III. The objective function values obtained by the two algorithms are the same. The advantage of the proposed PSO is its shorter computational time. Table III clearly indicates the superior performance of position based local search. The average CPU time of GASA for the ten problems is 4.2842 and for PSO with PBLS is 0.59567. The saving in CPU time form PSO is about 80%, which is very significant. The results clearly indicate the comparable performance of the proposed PSO with the existing better performing heuristic. The performance of the proposed PSO is compared with the heuristics reported in the literature and the results are presented in Table IV. The results show the comparable performance of the proposed PSO over other heuristics.

TABLE III
Performance Comparison between PSO and GASA

Data set	Objective function value		CPU time (s)		
	GASA [7]	PSO+ PBLS	GASA [7]	PSO+PBLS	PSO+JiBIS
1	1.5927	1.5927	7.9091	1.462	2.9301
2	1.8516	1.8516	9.094	0.252	0.252
3	1.9095	1.9095	3.479	0.718	1.6429
4	1.5734	1.5734	0.67	0.209	0.2086
5	1.6651	1.6651	2.152	0.770	0.9921
6	1.8574	1.8574	3.401	0.603	2.26
7	1.6874	1.6874	4.961	0.239	0.2389
8	1.6529	1.6529	8.642	1.157	1.3261
9	1.8391	1.8391	0.873	0.299	0.2989
10	1.7723	1.7723	1.669	0.2496	0.2496

IX. Conclusion

It is clear from this research that the machine loading problem can be solved by using a PSO based heuristic that can tackle the problem in a synergistic way. By combining effective local search and PSO, resource allocation can be done efficiently. This research had also highlighted the efficiency of the local search algorithm in the optimization process. The local search algorithm reduced the time to reach the best fitness value by a considerate amount in some cases. The results presented in Tables III and IV clearly show that the PSO algorithm is comparable to the better performing algorithms reported in the literature and it obtains the best results obtained so far at a faster rate. The results reported in Table III are obtained from PSO with Job Index Based Insertion local search.

The future work is to fine tune the parameters of PSO and proposing efficient machine selection heuristics. The machine selection heuristics play a major role in achieving better results.

References

[1] Tiwari M.K., Vidyarthi N.K., "Solving machine loading problem in flexible manufacturing system using genetic algorithm based heuristic approach", *International Journal of Production Research*, vol. 38, no. 14, pp. 3357-84, 2000.

[1] Swarnkar.R, and Tiwari.M.K, "Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing based heuristic approach", *Robotics and Computer Integrated Manufacturing* , vol. 20, pp. 199-209, 2003.

[2] Prakash, A., Nitesh Khilwani, Tiwari M.K. and Yuval Cohen, "Modified Immune Algorithm for job selection and operation allocation problem in Flexible Manufacturing Systems", *Advances in Engineering Software*, vol. 39, no. 3, pp. 219-232, 2008.

[4] Chan F.T.S., Swamkar R. and Tiwari M.K., 2005, Fuzzy goal-programming model with an artificial immune system (AIS) approach for a machine tool selection and operation allocation problem in a flexible manufacturing system, *International Journal of Production Research*, vol. 43, no. 19, pp. 4147-4163.

[5] Tripathi A.K. and Tiwari M.K., Chan F.T.S., "Multi-agent-based approach to solve part section and task allocation problem in flexible manufacturing systems", *International Journal of Production Research*, vol. 43, no. 7, pp. 1313-1335, 2005.

[6] Akhilesh Kumar, Prakash, M. K. Tiwari, Ravi Shankar, and Alok Baveja, "Solving machine-loading problem of a flexible manufacturing system with constrained-based genetic algorithm", *European Journal of Operational Research*, vol. 175, no. 2, pp. 1043-1069, 2006.

[7] Yogeswaran, M.; Ponnambalam, S.G.; Tiwari, M.K, "An hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS", *International Journal of Production Research*, in-print, 2008.

[8] J. Kennedy, R. Eberhart and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Mateo, CA, USA, 2001.

[9] Varadharajan, T. K., and Rajendran, C, A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs, *European Journal of Operational Research*, vol. 167, no. 3, pp. 772-795, 2005.

[10] Chandrasekaran, S, Ponnambalam, S. G, Suresh, R. K, "Multi-objective particle swarm optimization algorithm for scheduling in flowshops to minimize makespan, total flowtime and completion time variance,", *Proceedings of IEEE International Congress on Evolutionary Computation 2007,* IEEE CEC 2007, Singapore, September 25-28, 2007, pp. 4012-4018.

[11] Shankar,K., and Srinivasulu,A., "Some solution methodologies for loading problems in flexible manufacturing system", *International Journal of Production Research*, vol. 27, no. 6, pp. 1019-1034, 1989.

[12] Mukhopadhyay S. K, Midha S, Murlikrishna, V., "A heuristic procedure for loading problem in flexible manufacturing systems", *International Journal of Production Research*, vol. 30, no. 9, pp. 2213-28, 1992.

[13] Tiwaii,M. K., Hazarika, B., Vidyarthi, N. K., Jaggi, P., and Mukhopadhyay, S. K., "A heuristic solution to machine loading problem of a FMS and its Petri net model", *International Journal of Production Research*, vol. 35, no. 8, pp. 2269-2284, 1997.

[14] Vidyarthi N.K and Tiwari M.K., "Machine loading problem of FMS: a fuzzy-based heuristic approach", *International Journal of Production Research*, vol. 39, no. 5, pp. 953-979, 2001.

[15] Srinivas, Tiwari, M. K. and Allada, V., "Solving the machine loading problem in a flexible manufacturing system using a combinatorial auction-based approach", *International Journal of Production Research*, vol. 42, no. 9,pp. 1879-1893, 2004.

[16] Kumar, R.R., Amarjit Kumar Singh and Tiwari,M.K., "A fuzzy based algorithm to solve the machine-loading problems of a-FMS.and its neuro fuzzy Petri net model", *International Journal of Advanced Manufacturing Technology*, vol. 23, no. (5¬6), pp. 318-341, 2004.

[17] Nagarjuna, N., Mahesh, and Rajagopal,K., "A heuristic based on multi-stage programming approach for machine loading problem in a flexible manufacturing system", *Robotics and Computer Integrated Manufacturing*, vol. 22, no. 4, pp. 342-352, 2006.

TABLE VI
THE PERFORMANCE OF THE PROPOSED PSO OVER THE ALGORITHMS REPORTED IN THE OPEN LITERATURE

Data Set	Reference number of the heuristics collected from the open literature																						Prposed PSO with PBLS	
	[6]		[7]		[8]		[1]		[9]		[10]		[2]		[11]		[12]		[3]		[6]		[7]			
1	1.3557		1.4615		1.4854		1.5927		1.4839		1.5927		1.5927		1.4839		1.5927		1.5927		1.5927		1.5927		1.5927	
	39	253	42	122	42	76	48	14	44	127	48	14	48	14	44	127	48	14	48	14	48	14	48	14	48	14
2	1.4965		1.7578		1.7411		1.6208		1.7984		1.5204		1.7411		1.7984		1.6208		1.7984		1.6208		1.8516		1.8516	
	51	388	63	202	63	234	46	18	63	124	57	500	63	234	63	124	46	18	63	124	46	18	63	22	63	22
3	1.6475		1.8510		1.7943		1.8359		1.8574		1.8359		1.8574		1.8574		1.8245		1.8359		1.8359		1.9095		1.9095	
	63	288	79	286	69	152	69	72	73	128	69	72	73	128	73	128	69	94	69	72	69	72	73	28	73	28
4	1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734		1.5734	
	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819	51	819
5	1.5726		1.8104		1.6651		1.6000		1.6062		1.5726		1.8104		1.6651		1.6062		1.6651		1.6000		1.6651		1.6651	
	62	467	76	364	61	264	53	187	53	175	62	467	76	364	61	264	53	175	61	264	53	187	61	264	61	264
6	1.4132		1.6592		1.7516		1.8210		1.8408		1.5204		1.8408		1.7151		1.8408		1.8210		1.8210		1.8731		1.8731	
	51	548	62	365	63	214	61	28	64	69	57	500	64	69	63	284	64	69	61	28	61	28	64	37	64	37
7	1.5939		1.7696		1.0966		1.6064		1.6064		1.5546		1.6001		1.6001		1.6064		1.6874		1.6064		1.6874		1.6874	
	54	189	66	147	48	996	54	165	54	165	63	486	54	177	54	177	54	165	63	231	54	165	63	231	63	231
8	1.2752		1.2752		1.5320		1.6529		1.6218		1.6482		1.6529		1.6218		1.6218		1.6218		1.6218		1.6529		1.6529	
	36	459	36	459	43	158	48	63	44	13	48	72	48	63	44	13	44	13	44	13	44	13	48	63	48	63
9	1.6571		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391		1.8391	
	79	462	88	309	88	309	88	309	88	309	88	309	88	309	88	309	88	309	88	309	88	309	88	309	88	309
10	1.3869		1.6692		1.7344		1.7723		1.7633		1.7622		1.7723		1.7723		1.7633		1.7723		1.7633		1.7723		1.7723	
	44	518	56	320	55	166	56	122	54	82	54	84	56	122	56	122	54	82	56	122	54	82	56	122	56	122