A new single machine scheduling problems with dynamic arrival time and setup consideration in a common due date environment by using the Inver-over CX

S. H. Chen
Department of Information Management, Cheng Shiu University.
No. 840, Chengcing Rd., Niaosong Dist., Kaohsiung City 83347, Taiwan (R.O.C.).
shchen@csu.edu.tw

ABSTRACT

In Rabadi et al. [9], they studied the single machine scheduling problem with setup cost in the common due date environment. Their research assumed the static state when it comes to the problem of single machine scheduling. However, in reality, the dynamic arrival time exists in the scheduling problems. It means unknown numbers of jobs arrive successively. To solve this new scheduling problems on single machine, this research will take Inver-Over [10] as the basis of Inver-over CX. The proposed algorithm combines two-point intersection PMX (Partial Message Crossover). The experimental result shows that the new method, Inver-over CX is better than Inver-over because the proposed algorithm overcomes the slow convergency of inver-over operator.

Keywords

Dynamic arrival time, setup cost, common due date, inver-over operator

1. INTRODUCTION

Among all the scheduling problems, single machine is the fundamental one, which means that the process of all jobs is done by this machine. The order of jobs matters and impacts the overall performance. Most researches use static state to suppose when it comes to the problem of single machine [1, 6, 7, 9]. Take the static arrival time for example, all jobs arrive before the process starts. It thus has nothing to do with the problem of arrival time [5]. However, in reality, the problem of dynamic arrival time exists. That is, unknown numbers of jobs arrive successively when the process starts. When we suppose jobs arrive successively which could make the result much closer to reality. In addition, Rabadi et al. [9] proposed the single machine scheduling problem with setup cost in the common due date environment. To our best knowledge, there is none research studied this schedule problem together with the consideration of dynamic arrival time. As a result, this research might be the first one to study this new problem.

The order of the jobs is crucial to the performance of this new scheduling problem. In the past, Guo & Michalewicz [10] used Inver-over algorithms, which is very effective in solving sequencing problems, such as the Traveling Salesperson Problem (TSP) [11]. This paper uses Inver-over as the basis. However, its drawback is that it becomes slower when it solves the larger size problems. That is why this research will use a genetic operator, Partial Message Crossover (PMX), to reduce its drawback of low sequence perturbation. This research named this proposed algorithm be Inver-over CX and used to solve the new scheduling problem. It is the other contribution of this paper.

The rest of the paper is organized as follows. Section 2 describes the formulations of the new scheduling problems and the related works of the inver-over operator. Later on, Section 3 discussed the new scheduling problems as well as the proposed algorithm. In Section 4, we tested the proposed algorithm against the Inver-over operator. We draw the conclusions in Section .

2. PROBLEM DEFINITION AND RELATED WORKS

2.1 Formulations

In solving the single machine scheduling problem of dynamic arrival time, job sequence would be considered in the research. Therefore, the research is based on Inver-over algorithm with pretty excellent sequence disturbance and route plan in combination with PMX to improve Inver-over algorithm, in which there is a defect that local optimal solution is got into large problem, as a new method named Inver-over CX. The problem to be solved in the research is solved with such new method. Moreover, as target solution is generated from workpiece sequence, the first 10 groups with better tardiness time are inherited to the next generation, while the remained 90 groups are compared in pair to obtain arrival time r_i based on the elite policy of genetic algorithm.

The range value of random number, R_{max}, is calculated as shown in formula (1), where P is processing time, S is the setup time for every workpiece, and N is the number of workpieces.

$$\mathrm{Rmax} = (\mathrm{P} + \mathrm{S}) \times (\mathrm{N}-1) \quad (1)$$

As the common due date K is calculated in consideration of dynamic time, the middle point b in job sequence in

BICT 2014, December 01-03, Boston, United States

DOI 10.4108/icst.bict.2014.257888

consideration of dynamic arrival time has to be resulted in advance, while the formula for calculation of middle point b would be divided according to odd and even numbers of the number of workpieces. Therefore, the formula for calculating middle point b is as formula (2), where b is middle point, N is the jobs.

$$\mathrm{b} = \begin{cases} \mathrm{N} \div 2 & \text{(if n is even)} \\ (\mathrm{N}+1) \div 2 & \text{(if n is odd)} \end{cases} \tag{2}$$

As the arrival time, $\mathrm{r_i}$ of each job is reached, the completion time, C, of each can be calculated using the processing time, P, and arrival time, $\mathrm{r_i}$. The purpose of this study is to consider dynamic arrival time. Therefore, the following equation determines first whether the arrival time $\mathrm{r_i}$is greater than previous completion time, C, before adding the processing time, P, to obtain the completion time, C. Eq. (3) below calculates the completion time, C, where $\mathrm{r_i}$ is the arrival time, P is the processing time, $\mathrm{r_i}$ is the number of jobs and j is the job.

$$\mathrm{C_j} = \{\max(\mathrm{r_j}\ ,\ \mathrm{C_{j-1}}) + \mathrm{P_j}\} \tag{3}$$

When the middle point, b, considering the dynamic arrival time and the completion time, C, considering the dynamic arrival time are calculated, C and b can be used to identify the magnitude of problem and consider the common delivery deadline, K, of the dynamic arrival time. This common delivery deadline, K, is generated by considering early and late arrivals, as shown in Eq. (4).

$$\mathrm{K} = \mathrm{C_b} \tag{4}$$

As the completion time and common delivery deadline, K, are calculated, the deadline can serve as a base point. To go forward is to subtract the completion time, C, from the common delivery deadline, K, and to go backward is to subtract the common delivery deadline, K, from the completion time, C. This will generate the total early arrival time and total late arrival time for each job. The following Eq. (5) and (6) are to calculate the total early arrival time, $\mathrm{E_{total}}$, and total late arrival time, $\mathrm{T_{total}}$, respectively, where j is the job.

$$\mathrm{E_{total}} = \sum_{j=1}^{b} \mathrm{E_j} = \sum_{j=1}^{b} \{\max(\mathrm{K}\ ,\ \mathrm{C_j}) - \mathrm{C_j}\} \tag{5}$$

$$\mathrm{T_{total}} = \sum_{j=b+1}^{n} \mathrm{T_j} = \sum_{j=b+1}^{n} \{\mathrm{C_j} - \max(\mathrm{K}\ ,\ \mathrm{C_j})\} \tag{6}$$

2.2 Review of the Inver-Over Operator

The inver-over operator is proposed by Tao and Michalewicz [10]. The characteristics of the inver-over operator are the original solution competes only with its offspring only, only one operator is used in an evolutionary algorithm, and the number of times the operator is utilized to an individual during a generation is variable. Inver-over operator is based on simple inversion; however, knowledge taken from other individuals in the population influences its action [15]. That is, one is a state, and another is a state set [14]. As a result, to decide an appropriate mating chromosome is also important.

The characteristic of the inver-over operator has been studied by some researchers. It could be classified as an edge-based operator which preserve edges in selected parents and add new edges heuristically [12]. Previous research found that edge-preserving crossover operators are computationally expensive, while inversion-based algorithms (i.e., 2-Opt moves) difficultly escape local optima. A good tradeoff between the two approaches is the inver-over operator that tries to combine the computational efficiency of inversion with the power of crossover [2]. This concludes the advantage of using this operator and it is the state-of-art operator could be used in an evolutionary algorithms (EAs).

The inver-over operator can be found in other EAs. For example, it has been applied with a a memetic ACO-based (M-ACO) algorithm in [8] and S-ACO algorithm for the stationary TSP [3], where inver-over operator is performed among the best and second best ants. Nevertheless, it cannot still compete with domain-specific approaches on large TSPs [13]. Consequently, when this research employs the inver-over operator, it should improve the solution quality of eSSGA. In addition, we may enhance the ability to solve large-size problems.

3. RESEARCH METHOD

A new method, referred to as Inver-over CX, would be proposed based on Inver-over algorithm with PMX and genetic algorithm added to solve the single machine scheduling problem of dynamic arrival time and sequence setup cost in the common due date environment. In the beginning, Section 3.1 depicts the process of Inver-over Algorithm. Section 3.2 shows the Inver-over CX Algorithm of the research.

3.1 Inver-over operator

Guo & Michalewicz [10] mentioned the steps and process of the Inver-over operator. We suppose P1 and P2 are chromosomes. P1' is generated with the Inver-over algorithm. Algorithm 3.1 shows the procedures of Inver-over algorithm which is the combination of Inver-Over and Swap. Swap is two-point mating, i.e. swapping the jobs on the two points. Initially, a limited range of random numbers are generated as the parent generation based on the magnitude of single-machine scheduling issue. The child generation that follows disturbs the sequence of chromosome using the Inver-over algorithm. It disturbs by determining whether the random number, R, is smaller than the threshold, P. Swap is used if yes, and Inver-over if no. Then, the calculation of target functions, such as common delivery deadline, preparation time and completion time, follows. Finally, the target of study is the smallest late arrive time. Therefore, when the calculation of target functions is completed, the solution for chromosome sequence after disturbance (i.e. the sequence solution of jobs) is given to the elite policy for screening and selection. Once the selection is made, it is time to identify whether the solution is achieved. If not, the process goes back before the disturbance sequence.

The following is the detail steps of the Inver-over algorithm, where R is a random number and R is set as $U(0,1)$. P is the threshold value of the experiment. The number of generations i is 1000 for the small size problems and 2000 is for the large size problems.

3.2 Proposed Algorithm: Inver-over CX

Algorithm 1 Procedures of Inver-Over algorithm

1: A starting point c1 is generated randomly on P1;
2: Set the job moving to the right from C1 generated on P1 as the starting point of the Inver-over, cs;
3: Use c1 on P1 to locate c1 on P2;
4: Set the job moving to the right from c1 on P2 as the starting point of the Inver-over, ce;
5: Use ce on P2 to locate ce on P1;
6: Perform Inver-over from cs to ce on P1 and generate P1'.

Algorithm 2 Logics of Inver-Over algorithm

```
for i=1 to 1000 do
  for x=1 to 100 do
    if R < P then
      Swap()
    else
      Inver-over()
    end if
  end for
end for
```

Algorithm 3.2 shows the process of Inver-over CX operator proposed in this study. The Inver-Over operator is combined with PMX as the main axis and the characteristics of genetic algorithm are incorporated. Chromosome is selected as the analog for its job sequence. The process starts by generating random numbers of certain range as the parent generation based on the magnitude of single machine scheduling problems. The Inver-over CX algorithm is used to disturb the child generations that follow. The disturbing process identifies whether R is smaller than P. If yes, PMX is used and Inver-over is used if not. The next is to calculate the common delivery deadline, preparation time, completion time and other target functions. Finally, the target of study is the smallest late arrive time. Therefore, when the calculation of target functions is completed, the solution for chromosome sequence after disturbance (i.e. the sequence solution of jobs) is given to the elite policy for screening and selection. Once the selection is made, it is time to identify whether the solution is achieved. If not, the process goes back before the disturbance sequence.

Algorithm 3 Main procedures of Inver-Over CX

1: Randomly generate the starting point, A, and end point, B, on P1;
2: Set the starting and end points generated on P1 on P2;
3: Duplicate the value between the starting point and the end point of P1 onto the New P2;
4: Duplicate the value between the starting point and the end point of P2 onto the New P1;
5: Duplicate the corresponding P1 and P2 to the remaining blanks in New P1 and New P2 in sequence without repeating;

The following pesudo code is the detail procedures of Inver-over CX, where R is random number and set at $U(0,1)$ and P is the threshold value of experiment.

The following is the PMX logics, where P1 and P2 are two job sequences, New P1 and New P2 are job sequences generated from PMX after two-point mating, random numbers R and R1 are set at U(0,1), A is the starting point and B is the end point. The following is the PMX logics.

Algorithm 4 Inver-Over CX logics

```
for i = 1 to 1000 do
  for x = 1to100 do
    if R<P then
      PMX()
    else
      Inver-over ()
    end if
  end for
end for
```

Algorithm 5 PMX logics

```
R ← U(0,1) and R1 ← U(0,1)
A ← R and B ← R1
NEW P2[A] to NEW P2[B] = P1[A] to P1[B]
NEW P1[A] to NEW P1[B] = P2[A] to P2[B]
for i = 0 to P1 do
  for a =0 to P1 do
    if P1[i] != NEW P1[a] then
      NEW P1[a] = P1[i]
    end if
    if P2[i] != NEW P2[a] then
      NEW P2[a] = P2[i]
    end if
  end for
end for
```

4. EXPERIMENTAL RESULTS

In experiment design and analysis, the research will process the testing based on the issue scales, 10, 15, 20, 25, described by Rabadi et al.[9]. In accordance with the problem types of the adjusted processing time matrix (AP), there are the low, med, and high described in the past literatures. The first group of adjusted processing with average value (10, 60) is low, the medium group with average value (10, 110) is med, and the last group with average value (10, 160) is high. However, the prior research of Rabadi et al.[9] was not to resolve dynamic arrival time issue, and the instance scales which are raised by Rabadi et al. [9] are small. Therefore, the study adopts issue scales, 50, 100, 150, 200, raised by Chen et al.[4] for testing. Since there are 8 instance sizes, 3 issue types (low, med, and high) of each instance scale, and 15 repeated testing cases, the research adopts total 360 (8*3*15) testing cases, and 30 test runs will be processed for each testing case.

4.1 Parameter Settings

The experiment conducts extensive experiment by using design-of-experiments (DOE) to set the optimal threshold values. Then the optimal threshold value will be found based on the experiment results. Due to the limitation of the paper length, this paper applied the paramater configuration directly. The screening threshold value of new method Inver-over CX is 0.5 and for the method to be compared the screening threshold value for the Inver-over operator is 0.1. We compare the screening threshold values of these two methods in this section, comparison table of method threshold values, details are shown in Table 4.1.

Table 1: Method comparison table

Method	Threshold value P
Inver-over CX	0.5
Inver-over	0.1

Hence, the proposed algorithm, Inver-over CX, will be compared to the past one, Inver-over operator. The calculation of objective function can be referred to formula (1) ˜ formula (6). The data analysis and approaches comparing is presented in the next section.

4.2 Comparison and analysis of method data

Through Minitab statistical software, it can be seen from 5 which is the ANOVA analysis table generated by GLM. ANOVA table of 5 compares two screening threshold values of two comparing methods according to previous section. By using ANOVA, it can be learned that the P value of Method in Table is smaller than 0.05, this also represents there are significant factors for the comparison of these two methods, the source Size of 5 is problem size, such as 10, 15, 20 , 25, 50, 100, 150, and 200. Type is problem category, such as low, med, nd high. Value is threshold value, such as 0.1, 0.5, and 0.9. Method is comparison method Inver-over CX and Inver-over algorithm, Degree of freedom is numbers of sample minus 1, Seq SS is sum of squares, Adj SS is sum of squares after the adjustment of error values, Adj MS is sum of squares divide by sum of average squares of degree of freedom, F is the critical value obtained from mean square after adjustment divide by error value, details are shown in 5.

From 5, it shows that the method will have the significant factors; therefore, this study will adopt Bonferroni rating to group and analyze two methods in separate for parts after rating; Table 5 presents the Bonferroni rating of two methods, it can be learned that 95% degree of confidence conducts group to method, group B is all significant in average value and control, that is the new method of this study through results of Bonferroni rating shows the new method of this study Inver-over CX is superior to the past method of Inver-over algorithm.

Table 5 and Table 5 are execution time CPU Time ANOVA tables for the method of this study and comparison method. In Table 5 and Table 5, the P values of Method are all smaller than 0.05, it can be learned that at problem size of 10 to 25 and 50 to 200, comparison of execution time CPU Time will have significant factors. It is the same for the Type, Size, and the interactions between or among these factors.

From Table 5 and Table 5, it can be learned that the comparison of CPU Time has significant factors, therefore, this study will adopt Bonferroni rating to group and analyze two methods in separate for parts after rating, and then to obtain the most significant execution time CPU Time is the most significant method. Table 5 and Table 5 are generated through rating Bonferroni of two methods. In Table 5 and Table 5, it can be learned that conducting group to method and set 95% of degree of confidence when problem size is at 10 to 25 and 50 to 200, group B is all significant in average value and control, that is the new method of this study through results of Bonferroni rating shows the new method of this study Inver-over CX is superior to the past method of Inver-over operator.

Through the parameters screening in section 1 and algorithm comparison in section 2, Table 22 is the method comparison table which integrates section 1 and 2. In the table, Size is the problem size, Type is the problem category, Min is the average minimum value of objective function, Avg is the average value of objective function, Max is the average maximum value of objective function, Best Found is the superiority value that this experiment changes the small problem generation number of problem size to 100000 and large problem is changed to generation number 200000, details are shown in 5.

5. CONCLUSIONS

In this paper, we illustrate a new scheduling problem which considered the dynamic arrival time for the single machine scheduling problem with the setup cost and the common due date. The corresponding instances are generated and we also proposed a new evolutionary algorithm, Inver-over CX, is proposed to solve large-size problems. According to the experimental results, the new algorithm is better than the origin method, Inver-over operator, in terms of the solution quality and the computational time. As a result, both the new scheduling problems and a better evolutionary algorithm are the major contributions in this research. The future work could focus on the local search operators which could further improve the solution quality.

Table 2: Method comparison ANOVA table

Source	Degree of freedom	Seq SS	Adj SS	Adj MS	F	P
Method	1	34339123668	34339123668	34339123668	1667.15	0.000
Type	2	3.58960E+13	3.58960E+13	1.79480E+13	871368.01	0.000
Size	7	8.06616E+14	8.06616E+14	1.15231E+14	5594395.68	0.000
Method*Type	2	782837207	782837207	391418603	19.00	0.000
Method*Size	7	30503496211	30503496211	4357642616	211.56	0.000
Type*Size	14	7.76869E+13	7.76869E+13	5.54907E+12	269404.76	0.000
Method*Type*Size	14	4240620629	4240620629	302901473	14.71	0.000
error	21552	4.43918E+11	4.43918E+11	20597511		
Total	21599	9.20711E+14				

Table 3: Rating table of method comparison

Bonferroni Rating			
Method	N	Average value	Group
Inver-over	10800	133004.7	A
Inver-over CX(Control)	10800	130449.2	B

Table 4: Method comparison small problem CPU Time ANOVA table

Source	Degree of freedom	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.0187151	0.0187151	0.0187151	725.31	0.000
Size	3	0.007691	0.007691	0.002564	9.94	0.000
Type	2	0.00124	0.00124	0.00062	193.48	0.000
Method* Size	3	0.0000256	0.0000256	0.0000085	0.33	0.803
Method* Type	2	0.0001510	0.0001510	0.0000755	2.93	0.054
Type*Size	6	0.0075113	0.0075113	0.0012519	48.52	0.000
Method*Type*Size	6	0.0000512	0.0000512	0.0000085	0.33	0.921
error	10776	0.2780522	0.2780522	0.0000258		
total	10799	0.0000512				

Table 5: ANOVA table of method comparison large problem CPU Time

source	Degree of freedom	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.775954	0.775954	0.775954	15303.62	0.000
Size	3	0.790836	0.790836	0.263612	5199.04	0.000
Type	2	0.039859	0.039859	0.019930	393.06	0.000
Method* Size	3	0.000014	0.000014	0.000005	0.09	0.0965
Method* Type	2	0.000140	0.000140	0.000070	1.38	0.0251
Type*Size	6	0.028690	0.028690	0.004782	94.31	0.000
Method*Type*Size	6	0.000028	0.000028	0.000005	0.09	0.997
Error	10776	0.546386	0.546386	0.000051		
Total	10799	2.181907				

Table 6: CPU Time small problem Bonferroni Rating table

Bonferroni Rating			
Method	N	Average value	Group
Inver-over	5400	0.016	A
Inver-over CX(Control)	5400	0.012	B

Table 7: CPU Time large problem Bonferroni Rating table

Bonferroni Rating			
method	N	Average value	group
Inver-over	5400	0.048	A
Inver-over CX(Control)	5400	0.031	B

6. REFERENCES

[1] V. A. Armentano and R. Mazzini, "A genetic algorithm for scheduling on a single machine with set-up times and due dates," *Production Planning & Control*, vol. 11, no. 7, pp. 713–720, 2000.

[2] R. Baraglia, J. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 6, pp. 613–622, 2001.

[3] X. Bi and G. Luo, "The improvement of ant colony algorithm based on the inver-over operator," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*. IEEE, 2007, pp. 2383–2387.

[4] P. C. Chang, T. Lie, J. Y. Liu, and S. Chen, "A genetic algorithm enhanced by dominance properties for single machine scheduling problems with setup cost," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 5, pp. 1–21, 2011.

[5] S.-H. Chen, M.-C. Chen, and Y.-C. Liou, "Artificial chromosomes with genetic algorithm 2 (acga2) for single machine scheduling problems with sequence-dependent setup times," *Applied Soft Computing*, vol. 17, pp. 167–175, 2014.

[6] H. Emmons, "Scheduling to a common due date on parallel uniform processors," *Naval Research Logistics (NRL)*, vol. 34, no. 6, pp. 803–810, 1987.

[7] X. Luo and F. Chu, "A branch and bound algorithm of the single machine schedule with sequence dependent setup times for minimizing total tardiness," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 575–588, 2006.

[8] M. Mavrovouniotis and S. Yang, "A memetic ant colony optimization algorithm for the dynamic travelling salesman problem," *Soft Computing*, vol. 15, no. 7, pp. 1405–1425, 2011.

[9] G. Rabadi, M. Mollaghasemi, and G. Anagnostopoulos, "A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time," *Computers & Operations Research*, vol. 31, no. 10, pp. 1727–1751, 2004.

[10] G. Tao and Z. Michalewicz, "Inver-over operator for the tsp," in *Parallel Problem Solving from Nature—PPSN V*. Springer, 1998, pp. 803–812.

[11] M. F. Tasgetiren, O. Buyukdagli, D. Kızılay, and K. Karabulut, "A populated iterated greedy algorithm with inver-over operator for traveling salesman problem," in *Swarm, Evolutionary, and Memetic Computing*. Springer, 2013, pp. 1–12.

[12] H. Tsai, J. Yang, Y. Tsai, and C. Kao, "An evolutionary algorithm for large traveling salesman problems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 4, pp. 1718–1729, 2004.

[13] Y. Wang, J. Sun, J. Li, and K. Gao, "A modified inver-over operator for the traveling salesman problem," in *Proceedings of the 7th international conference on Advanced Intelligent Computing Theories and Applications: with aspects of artificial intelligence*. Springer-Verlag, 2011, pp. 17–23.

[14] X. Xie and J. Liu, "Multiagent optimization system for solving the traveling salesman problem (tsp)," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 489–502, 2009.

[15] X. Yan, H. Liu, J. Yan, and Q. Wu, "A fast evolutionary algorithm for traveling salesman problem," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 4. IEEE, 2007, pp. 85–90.

Table 8: Method comparison table

		Inver-over CX				Inver-over		
Size	Type	Best Found	Min	Avg	Max	Min	Avg	Max
10	low	452	592	649	708	785	852	912
	med	608	628	801	888	896	1000	1132
	high	885	924	1075	1270	1116	1259	1464
15	low	1173	1258	1345	1512	1403	1517	1685
	med	1742	1792	1874	2139	1807	2126	2363
	high	2342	2442	2517	2898	2481	2501	3062
20	low	2237	2359	2476	2535	2451	2605	2811
	med	3537	3668	3882	3948	3687	3949	4304
	high	4896	4930	5025	5384	5022	5250	6062
25	low	3617	3816	3953	4149	3917	4110	4418
	med	5710	5977	6071	6362	6041	6200	6725
	high	8101	8234	8521	8923	8624	8825	9331
50	low	13344	13756	15224	16157	19478	20306	21171
	med	22567	24064	25939	27368	29758	31462	33000
	high	29931	32908	37267	39893	38770	42172	45023
100	low	54720	76097	79656	82502	80668	84829	88213
	med	96291	122982	129132	133504	124279	135249	140038
	high	137739	175521	180916	187285	168484	186211	194466
150	low	137719	186418	192654	199777	193030	196825	200844
	med	248994	300750	310933	320150	293114	313756	329006
	high	357821	419430	428912	439554	415209	431204	44502
200	low	238804	334742	350366	358778	343789	353416	363418
	med	502544	524042	561107	583614	552953	570057	587465
	high	605732	748853	781387	795890	759919	785526	799641