A Study on the Effectiveness of Genetic Algorithm and Identifying the Best Parameters Range for Slots Swapping in the Examination Scheduling

* **Siti Khatijah Nor Abdul Rahim, ***Andrzej Bargiela and ***Rong Qu
* Faculty of Computer & Mathematical Sciences, UiTM Perak, Bandar Baru Seri Iskandar, Bota, Perak, Malaysia.
** School of Computer Science, University of Nottingham (Malaysia Campus), Semenyih, Selangor, Malaysia.
*** School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK.
sitikhatijahnor@gmail.com

Abstract— **In this study, in order to examine the effectiveness of Genetic Algorithm (GA) during the optimization stage in the proposed examination scheduling algorithm, different combinations of parameters have been used. We have experimented a few combinations of parameters (i.e. : additional and different number of parents-iterations) during slots swapping on the benchmark datasets, in order to examine whether the GA could escape from local optima with the increased factors, and if it is the case, therefore to determine the best parameters that would produce the best global optimum. As predicted, it is shown that GA managed to achieve its objective but it has been observed that only certain range of the parameters could offer advantage in terms of improving the quality of the examination schedules.**

Keywords: examination scheduling; domain transformation approach, optimization; genetic algorithm, slots swapping; best parameters range, deterministic pattern.

I. INTRODUCTION

Examination timetabling is one type of timetabling that occurs periodically in many academic institutions. In most of the universities in the world, this type of timetabling can be considered a challenging task due to the many-to-many relationships between students and exams data. The complexities and the challenge arise from the fact that a huge variety of constraints, some of which contradict to each other, need to be fulfilled in different institutions [6].

Numerous approaches have been proposed to solve the examination timetabling problems and can be found in the timetabling literature. These include graph colouring heuristics, local search techniques, evolutionary algorithms, hybridisations, constructive approaches and many more. All of the proposed methods will normally produce solutions, which is the schedule or timetable, but sometimes the quality might be poor.

A good quality timetable could be defined as a timetable that satisfies all the hard constraints, and also it satisfies as much as possible the soft constraints available in the problem. These two types of constraints are the constraints mentioned in the timetabling literature [6]. The timetable that satisfies at least the hard constraint(s) is known as a feasible timetable. A brief explanation of the type of constraints is given below:

- Hard constraints cannot be violated under any situations. For instance, conflicting exams which involve the same students cannot be scheduled concurrently
- Soft constraints are not absolutely crucial to be satisfied, but normally the quality will increase by satisfying many of these constraints. The most common example of soft constraints is maximizing and equalizing the gap between conflicting exams for a particular student.

II. OUR PROPOSED APPROACH

A. *The Proposed Examination Scheduling Algorithm*

We have proposed a Domain Transformation Approach to solve the examination scheduling problems [8] which was inspired by the insights of Granular Computing methodology [1], [2], [3], and [5]. The steps involved in our proposed approach are illustrated in Figure 1 [7], [8], and [9].

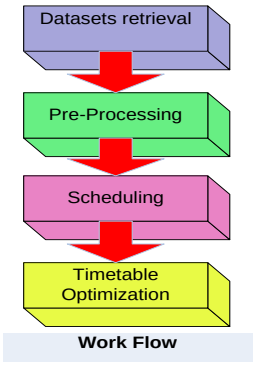

Figure 1: The Steps in Our Proposed Approach

Domain Transformation Approach could be defined as an approach where a problem is transformed into a simple type of problem, normally to multiple or smaller sub problems in order to solve the main problem more effectively. By sub dividing the problems into smaller sub problems, it is easier to tackle the problem phase by phase.

The phases of our proposed examination scheduling method are as follows where the details can be found in [7], [8] and [9].

- Problem domain transformation from student-exam to exam conflict and spread matrix data space
- Generation of a feasible solution via allocation method and backtracking
- Minimization of the overall slots conflicts
- Minimization of the schedule cost by slot swapping
- Minimization of the schedule cost by exam reallocation
- Repetition of the last two steps until no improvement in the schedule cost

III. Optimization in Examination Scheduling

A. Optimization Concept and its Application in Examination Scheduling

In the area of computer science or mathematical programming, optimization can be understood as selecting the best element from some set of available solutions. Therefore, in the examination scheduling context, optimization could be defined as a process to improve the quality of the feasible exams schedule or solution. One can imagine that a feasible timetable can have the ordering of exams that does not satisfy many of the soft constraints. This calls for a separate deployment of optimization to improve the quality of the schedules.

Based on the scheduling steps proposed in the previous section, optimization takes place starting from the third bullet point until the last bullet point. In this particular paper, the focus of our study is the fourth bullet point, i.e.: minimization of the schedule cost by slot swapping. For this step of optimization, in our previous study, the permutations of exams slots are done in order to minimize the cost of the schedule generated by scheduling algorithm by implementing a greedy Hill Climbing (HC) algorithm [8], and [9]. We have been producing good results using this HC, however since HC is a local search procedure; therefore we were motivated to replace this with a global search procedure which we expect would generate better results. As such, we have chosen Genetic Algorithm (GA) to replace the permutations of exams slots in the optimization stage [9].

Even though GA is considered an outdated optimization technique, and recent research have move on to explore meta-heuristics based approaches (eg: harmony search, particle swarm optimization, bee colony optimization, etc) but we have chosen it as an alternative approach to our implementation. This is because, it has been proven a good way of producing high quality examination timetables. In addition, it has been confirmed that the hybridisations of GA with some local search have led to some success in this area [6].

B. Minimization Function

In the examination timetabling literature, the main objective is to minimize the penalty implied by the closeness of consecutive exams taken by each student. This is evaluated by the minimization function proposed by Carter [4] as below:

$$\frac{1}{T}\sum_{i=1}^{N}\sum_{j=i+1}^{N} s_{ij} w_{|pj - pi|} \quad (1)$$

Equation (1) above can be described as N is the number of exams, s_{ij} is the number of students enrolled in both exams i and j, p_j is the time slot where exam j is scheduled, p_i is the time slot where exam i is scheduled and T is the total number of students. Based on this cost function, a student taking two exams that are $| p_j - p_i |$ slots apart, where $| p_j - p_i | = \{1, 2, 3, 4, 5\}$, leads to a cost of 16, 8, 4, 2, and 1, respectively.

C. Minimization of the Schedule Cost by Slot Swapping

We have proposed slots swapping or permutations of exams slots (which we called a Greedy Hill Climbing) in the course of optimization in the examination scheduling process [8] and [9]. This is done by the permutations of rows/columns of the spread matrix of a feasible schedule obtained by the allocation method. The permutations can be considered quite an efficient procedure because the number of available exams slots is most of the time quite small. The effect of doing the slots swapping is new exams ordering are generated with better quality [8] and [9].

Besides the above mentioned procedure, we have also implemented a Genetic Algorithm (GA) optimization to improve the ordering of the exams in the feasible exam schedule generated. GA operators will somehow exchange the exams slots of the parents involved during crossover at certain point, therefore we considered this as a type of slots swapping (but note that it is a different type of swap compared to the above Hill Climbing).

IV. Genetic Algorithm

Genetic Algorithm (GA) is a search heuristic that imitates the process of natural evolution. This heuristic is normally used to generate solutions (which is normally good or useful) to optimization and search problems.

In Genetic Algorithm, a population of candidate solutions will evolve towards better solutions. Each individual in the population has some characteristics (known as chromosomes) which can be mutated and modified.

The evolution process begins from a population of randomly generated individuals and is an iterative process, where by the population in each iteration is called a generation. In each generation, the fitness (quality) of every individual in the population is evaluated based on the objective function selected. Good quality individuals (normally two best individuals) are selected from the current population, and each individual's genome (characteristic) is modified (recombined through cross over and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Normally, the algorithm terminates when either a maximum number of generations has been formed, or a superior solution has been obtained for the population.

The effectiveness of GA is highly dependents on its tweaking parameters. Despite the simplicity of its algorithm, GA needs careful and smart settings of its parameters, for example: method of selecting parents, population size, crossover type and rate, mutation type and rate, number of iterations and etc. A good setting of parameters might cause the algorithm to converge to its best results in an efficient time, meanwhile, on the other hand, wrong configurations of parameters might cause it to take longer time to produce good solutions, and sometimes in some cases it fails to obtain good solutions.

In this study, our objectives are to study the effectiveness of GA and find out the best parameters range (the population size and number of iterations) in the optimization stage in our proposed approach.

A. The Proposed Genetic Algorithm

In our Genetic Algorithm implementation that we proposed [9] we defined the original parent as *P0*, which is a data structure with the initial ordering of slots *(1 .... N)* where *N* is the number of slots. GA creates a new parent by moving position of the rows in blocks from a random index *K* to the end of *P0* to make the first to *K-N* number of rows, to complete the parent then we take the balance (first row to *K*) from *P0* to fill in from *K + 1* to *N* in the new parent. A number of *npar* parents will be created. In general the generation of the new parents is created by shifting the rows which in the end is the new representation of the original parent with a magnitude maximum distance of *npar – 1*. Therefore, if it is just a window shift, there will be identical parents, especially when the number of *N* is smaller than the number of *npar*.

We then produced the new offsprings from these initial parents. The number of offsprings to be produced is equals to the unique permutation of a parent in *P* with other parents in *P*. Each of the parents will be crossed over with all other parents at a certain random point *R*, creating two new offsprings. *Offspring1* will contain the first to *R* slot from *Parent1* and then completed by appending the slot in *Parent2* starting from the first to *N* which is not already in *Offspring1*. The same goes for *Offspring2*. It will contain the first to *R* slot from *Offspring2* and completed by appending the slot in *Parent1*. These two newly created offsprings will be added to *"O"* which is the overall population. We will then eliminate any identical offsprings and replace it with a mutated *P* where we interchange a random slot *t* with a random slot *u*. The best offspring with the lowest cost will be automatically selected to become the next generation parent *P*.

The offsprings generated in the population are the individuals with new orderings of slots due to crossover of the parents involved, which in our study is assumed as a type of slots swapping.

Figure 2 and Figure 3 illustrate the process of slots swapping done using GA.

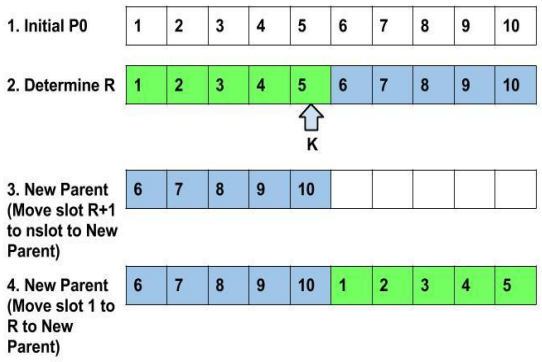

Figure 2: Generation of New Parents in the Proposed GA

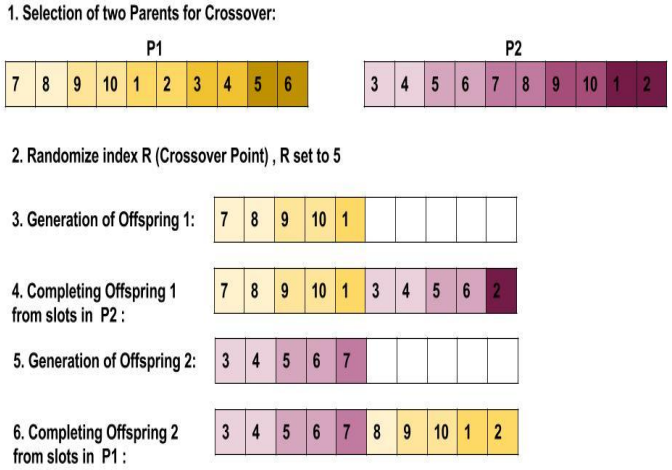

Figure 3: Generation of Offsprings in the Proposed GA

V. COMPUTATIONAL RESULTS AND DISCUSSION

In order to determine the effectiveness of the proposed GA in the optimization step for the examination scheduling, we have conducted experiments on the 12 datasets in the Toronto

benchmark repository. These datasets can be accessed from [ftp://ftp.mie.utoronto.ca/pub/carter/testprob].

Table I listed out the characteristics of each dataset.

TABLE I. THE CHARACTERISTICS OF THE BENCHMARK DATASETS

(a)Name of Dataset; (b) No of Exams; (c) No of Students; (d) No of Enrollments; (e) Required No of Slots; (f) Conflict Density

(a)	(b)	(c)	(d)	(e)	(f)
car-s-91 (I)	682	16925	56877	35	0.13
car-f-92 (I)	543	18419	55522	32	0.14
ear-f-83(I)	190	1125	8109	24	0.27
hec-s-92(I)	81	2823	10632	18	0.42
kfu-s-93	461	5349	25113	20	0.06
lse-f-91	381	2726	10918	18	0.06
rye-f-92	486	11483	45051	23	0.07
sta-f-83(I)	139	611	5751	13	0.14
tre-s-92	261	4360	14901	23	0.18
uta-s-92(I)	622	21266	58979	35	0.13
ute-s-92	184	2749	11793	10	0.08
yor-f-83 (I)	181	941	6034	21	0.29

We have used all 9 combinations of parameters for different number of parents and iterations (i.e.: parents: 10 / 15 / 20 and iterations 10 / 15 / 20) during the experiments.

Recall that as mentioned previously, in our proposed approach [8], the initial schedules were generated using an allocation method before being further optimized. The optimization process was done twice; hence the name First and Second Order Optimization [8] as can be seen in the table.

The results obtained from the experiments were recorded in Table II in the Appendix Section. Note that due to the constraint of page limit in this paper, only a simplified table is presented, where the results of the costs after optimization are rounded to 2 decimal places . Detail results can be found on the author's website.

Based on all the results, the first and foremost point that we would like to state is, it can be seen very clearly that a significant improvement has been achieved for each dataset from the initial cost to the cost produced by GA (from first to last iteration), besides the time taken for the GA optimization is also very efficient with the average of less than 1.5 second CPU time for each dataset. (However note that not any single combination of parameters manage to outperform the results gained by HC in our previous study [8] and [9]). The improvements have been obtained for all combinations of parameters, therefore proving that GA is an effective optimization technique through exams slots swapping. The cost was further reduced by reassigning the exams [8] in the first order optimization (and was further improved during second order optimization for some datasets). This pattern can be observed if one moves from left to right across this table.

The general patterns of the graphs Cost (1) vs. the Total Slot Conflicts for benchmark datasets obtained by our proposed approach have been examined. Again, due to the constraint of page limit, only one graph with the smallest number of exams slots which is ute-s-92 is presented. It has been observed that all the patterns obtained from all datasets are similar [8] and deterministic [9].

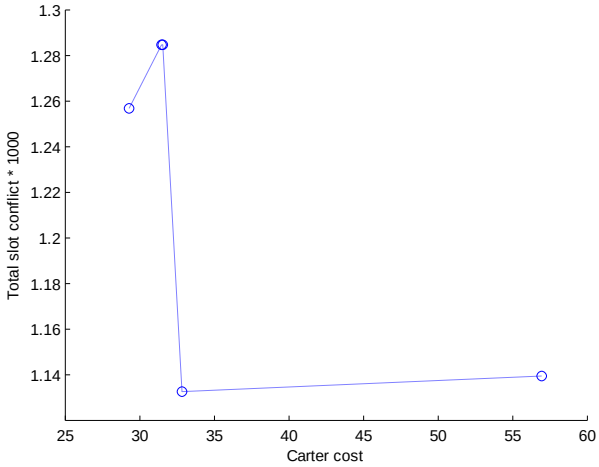

Figure 4: Cost (1) vs. the Total Slot Conflicts for ute-s-92

Based on the graph in Figure 4, we can see that a very significant reduction in terms of the Carter Cost (1) has been achieved by performing GA technique. The initial cost which is 56.9698 has been reduced greatly to 32.8549. The second round GA has further reduced the cost to 31.4724, which later being further optimized by reassignment to produce the cost of 29.3473. It is worth noting here that the pattern of this graph is very deterministic and consistent for all datasets tested during the experimentations. This indirectly proved that GA has been successfully implemented and incorporated in our proposed Domain Transformation Approach framework and it never fails to improve the initial cost of the schedule, hence giving the approach a certainty to produce good results.

In the results presented, if one observes the cost produced by the same number of parents at Iteration 1 (no matter what the value of the maximum iteration), one can see clearly that the costs produced for all cases are the same, for example for car-f-92 (I), the values for the first 3 rows in the table are 7.8894.

Another quick generalization that we can make is that, for the same number of parents, the increased in the number of iterations does not offer advantage in terms of improving the quality of the schedules (or reducing the cost) because for most of the datasets, the costs remain even though the iterations were increased. There are some exceptions for this case however, i.e: for car-f-92 (I), car-s-91 (I) and uta-s-92 (I).

With the increased factor, in terms of the increased in the number of parents, one can also examines that the costs produced are mostly reducing or in other words improving for majority of the datasets, except for car-f-92 (I), ear-f-83 (I), kfu-s-93, lse-f-91, rye-f-92 and uta-s-92 (I) with *npar*=15, 20,

15, 15, 20 and 15 respectively. This can be seen obviously in Figure 5 and Figure 6 which illustrate Carter Cost (1) vs. Number of Parents for datasets sta-f-83 and ute-s-92.

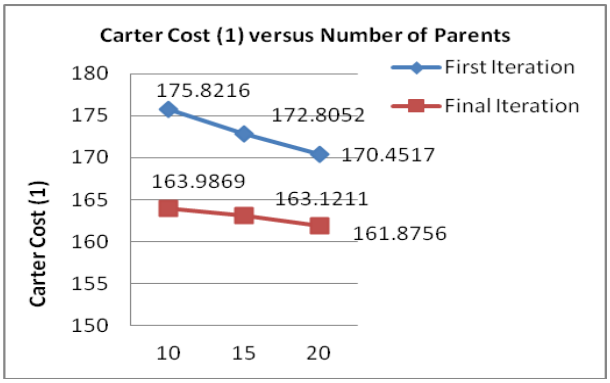

Figure 5: Carter Cost (1) vs. Number of Parents for sta-f-83 dataset

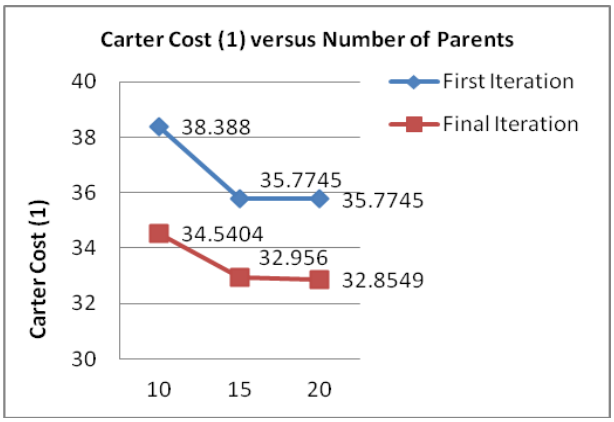

Figure 6: Carter Cost (1) vs. Number of Parents for ute-s-92 dataset

For the second order optimization, the behavior of the results produced by GA is almost the same with the first order optimization, both when moving horizontally (left to right) and vertically (top to bottom) across the table. Most importantly, for all datasets, the costs have reduced significantly before performing GA optimization and from the first iteration to the last iteration.

Based on the results recorded, we can conclude that the values selected for both parameters (i.e.: *number of parents* and *iterations*) in the GA proposed for the slots reordering of the exams schedules are suitable and managed to improve the quality of the schedules. In most cases, the increased in the values of both parameters managed to help the procedure explore the search space efficiently and escape from its local optima.

Having said that, it has been observed that, for *npar* = 10 the cost produced by GA is less encouraging than using *npar* = 15 or 20. Even though GA was run for a few iterations with 10 parents, it seems like the explorations of the search space only managed to find its local optima. Therefore, to avoid the GA from getting stuck in its local optima, according to the overall results, it is suggested to use *npar* = 15 to 20 with the same range of iterations. These suggested values seem to generate better results and have a better chance to arrive at its global minima.

VI. CONCLUSION AND FUTURE WORK

In conclusion, GA has been successfully implemented and incorporated into the proposed Domain Transformation Approach framework to solve the examination scheduling problems. The GA proposed in this study is really an efficient and robust procedure that manages to achieve its objective which is to improve the initial feasible exams schedules (before being optimized). Since it never fails to improve the quality of the initial feasible schedule in very efficient time and producing very deterministic results, we consider the incorporation of GA into our proposed framework is very effective. However, it was examined that this procedure works best at certain range of parameters. For this particular GA for slots swapping, it is suggested to use at least 15 to 20 parents and also 15 to 20 iterations in order to get the best solutions.

For the future research, we are looking forward to analyze and identify the best range for other types of parameters in the GA, besides will try to look at the possibility of incorporating other methods in our existing framework.

REFERENCES

[1] Bargiela A., Pedrycz W., Hirota K., "Granular prototyping in fuzzy clustering," IEEE Transactions on Fuzzy Systems. 2004; 12(5): 697-709. http://dx.doi.org/10.1109/TFUZZ.2004.834808.

[2] Bargiela, A., and Pedrycz, W, "Granular Computing – An Introduction, " Kluwer Academic Publishers. 2002. http://dx.doi.org/10.1007/978-1-4615-1033-8

[3] [Bargiela, A. and Pedrycz, W, "Toward a theory of Granular Computing for human-centred information processing," IEEE Trans. On Fuzzy Systems. 2008; 16(2): 320-330. http://dx.doi.org/10.1109/TFUZZ.2007.905912.

[4] Carter M., Laporte G. and Lee S, "Examination timetabling: algorithmic strategies and applications," Journal of Operations Research Society, 47 373-383, 1996.

[5] Pedrycz W, Smith M.H., Bargiela A, "Granular signature of data," Proc. 19th Int. (IEEE) Conf. NAFIPS'2000, Atlanta. July 2000; 69-73. http://dx.doi.org/10.1109/NAFIPS.2000.877387.

[6] Qu. R, Burke E.K., B. McCollum, L.T.G. Merlot, and S.Y.Lee, "A survey of search methodologies and automated system development for examination timetabling," Journal of Scheduling, 12(1): 55-89, 2009. doi: 10.1007/s10951-008-0077-5.

[7] Rahim, S. K. N. A., Bargiela, A., Qu, R, "Granular modelling of exam to slot allocation," ECMS 2009 Proceedings edited by J. Otamendi, A. Bargiela, J. L. Montes, L. M. Doncel Pedrera (pp. 861-866). European Council for Modeling and Simulation. 2009. doi:10.7148/2009-0861-0866.

[8] Rahim, S. K. N. A., Bargiela, A., Qu, R, "Domain transformation approach to deterministic optimization of examination timetables," Artificial Intelligence Research, 2(1):122-138, January 2013. doi: 10.5430/air.v2n1p122.

[9] Rahim, S. K. N. A., Bargiela, A., Qu, R, "Hill climbing versus genetic algorithm optimization in solving the examination timetabling problem," International Conference on Operations Research and Enterprose Systems, Barcelona, Spain, 2013.

APPENDIX

TABLE II. RESULTS FROM THE EXPERIMENTS CONDUCTED USING OPTIMIZATION PROCEDURES

Cost-1: Cost at First Iteration During First Order Optimization
Cost-2: Cost at Maximum Iteration During First Order Optimization
Cost-3: Cost at First Iteration During Second Order Optimization
Cost-4: Cost at Maximum Iteration During Second Order Optimization
Cost-5: Final Cost after Last Stage of Optimization

Data set/ Initial Cost	npar	Max iter	Cost-1	Cost-2	Cost-3	Cost-4	Cost-5
car-f-92 / 9.43	10	10	7.89	6.76	5.25	5.23	5.12
		15	7.89	6.76	*5.25*	*5.23*	5.12
		20	7.89	6.76	5.25	5.23	5.12
	15	10	8.04	6.78	5.25	5.25	5.25
		15	8.04	6.78	5.25	5.25	5.25
		20	8.04	6.78	5.25	5.25	5.25
	20	10	8.00	6.75	5.25	5.24	5.12
		15	8.00	6.71	5.25	5.24	5.12
		20	8.00	6.71	5.25	5.24	5.12
car-s-91 / 11.77	10	10	9.39	8.02	6.50	6.32	6.31
		15	9.39	7.98	6.50	6.32	6.31
		20	9.39	7.98	6.50	6.32	6.31
	15	10	9.07	7.77	6.50	6.30	6.15
		15	9.07	7.76	6.50	6.30	6.15
		20	9.07	7.76	6.50	6.30	6.15
	20	10	9.06	7.64	6.50	6.29	6.23
		15	9.06	7.64	6.50	6.29	6.23
		20	9.06	7.64	6.50	6.29	6.23
ear-f-83 / 72.69	10	10	56.47	49.19	45.02	44.42	43.75
		15	56.47	49.19	45.02	44.42	43.75
		20	56.47	49.19	45.02	44.42	43.75
	15	10	53.51	48.99	45.02	43.72	43.54
		15	53.51	48.99	45.02	43.72	43.54
		20	53.51	48.99	45.02	43.72	43.54
	20	10	53.71	49.33	45.02	42.78	40.93
		15	53.71	49.33	45.02	42.78	40.93
		20	53.71	49.33	45.02	42.78	40.93
hec-s-92 / 21.88	10	10	19.32	15.19	15.01	13.49	13.38
		15	19.32	15.19	15.01	13.49	13.38
		20	19.32	15.19	15.01	13.49	13.38
	15	10	19.39	14.14	14.64	13.00	12.87
		15	19.39	14.14	14.64	13.00	12.87
		20	19.39	14.14	14.64	13.00	12.87
	20	10	19.00	13.62	14.39	12.91	12.82
		15	19.00	13.62	14.39	12.91	12.82
		20	19.00	13.62	14.39	12.91	12.82
kfu-s-93 / 37.79	10	10	25.88	18.58	17.58	17.58	17.56
		15	25.88	18.58	17.58	17.58	17.56
		20	25.88	18.58	17.58	17.58	17.56
	15	10	25.29	19.03	17.58	17.58	17.56
		15	25.29	19.03	17.58	17.58	17.56
		20	25.29	19.03	17.58	17.58	17.56
	20	10	25.29	18.15	17.58	17.50	17.13
		15	25.29	18.15	17.58	17.36	16.92
		20	25.29	18.15	17.58	17.36	16.92
lse-f-91 / 23.77	10	10	20.04	17.04	16.84	16.05	15.81
		15	20.04	17.04	16.84	16.05	15.81
		20	20.04	17.04	16.84	16.05	15.81
	15	10	19.40	17.20	16.84	15.71	15.67
		15	19.40	17.20	16.84	15.71	15.67
		20	19.40	17.20	16.84	15.71	15.67
	20	10	19.10	16.14	16.84	15.66	15.16
		15	19.10	16.14	16.84	15.66	15.16
		20	19.10	16.14	16.84	15.66	15.16
rye-f-92 / 31.50	10	10	20.49	18.13	11.30	11.10	10.89
		15	20.49	18.13	11.30	11.10	10.89
		20	20.49	18.13	11.30	11.10	10.89
	15	10	19.04	16.46	11.30	10.96	10.93
		15	19.04	16.46	11.30	10.96	10.93
		20	19.04	16.46	11.30	10.96	10.93
	20	10	18.57	17.36	11.30	10.96	10.93
		15	18.57	17.36	11.30	10.96	10.93
		20	18.57	17.36	11.30	10.96	10.93
sta-f-83 / 201.06	10	10	175.82	163.99	169.71	161.70	159.96
		15	175.82	163.99	169.71	161.70	159.96
		20	175.82	163.99	169.71	161.70	159.96
	15	10	172.81	163.12	169.48	161.20	161.13
		15	172.81	163.12	169.48	161.20	161.13
		20	172.81	163.12	169.48	161.20	161.13
	20	10	170.45	161.88	164.65	160.93	160.46
		15	170.45	161.88	164.65	160.93	160.46
		20	170.45	161.88	164.65	160.93	160.46
tre-s-92 / 14.81	10	10	12.63	11.42	10.85	10.63	10.35
		15	12.63	11.42	10.85	10.63	10.35
		20	12.63	11.42	10.85	10.63	10.35
	15	10	12.51	11.01	10.85	10.58	10.50
		15	12.51	11.01	10.85	10.58	10.50
		20	12.51	11.01	10.85	10.58	10.50
	20	10	12.22	10.96	10.85	10.51	10.35
		15	12.22	10.96	10.85	10.51	10.35
		20	12.22	10.96	10.85	10.51	10.35
uta-s-92 / 7.71	10	10	5.92	5.16	4.23	4.22	4.02
		15	5.92	5.13	4.23	4.22	4.02
		20	5.92	5.13	4.23	4.22	4.02
	15	10	5.93	5.01	4.23	4.14	4.07
		15	5.93	5.01	4.23	4.14	3.96
		20	5.93	5.01	4.23	4.14	3.96
	20	10	5.90	5.04	4.23	4.16	4.08
		15	5.90	5.04	4.23	4.16	4.08
		20	5.90	5.04	4.23	4.16	4.08
ute-s-92 / 56.97	10	10	38.39	34.54	31.54	31.54	31.54
		15	38.39	34.54	31.54	31.54	31.54
		20	38.39	34.54	31.54	31.54	31.54
	15	10	35.77	32.96	31.54	31.47	29.35
		15	35.77	32.96	31.54	31.47	29.35
		20	35.77	32.96	31.54	31.47	29.35
	20	10	35.77	32.85	31.54	31.47	29.35
		15	35.77	32.85	31.54	31.47	29.35
		20	35.77	32.85	31.54	31.47	29.35
yor-f-83 / 59.04	10	10	51.67	47.54	44.82	43.01	42.63
		15	51.67	47.54	44.82	43.01	42.63
		20	51.67	47.54	44.82	43.01	42.63
	15	10	50.77	47.50	43.32	42.83	42.54
		15	50.77	47.50	43.32	42.83	42.54
		20	50.77	47.50	43.32	42.83	42.54
	20	10	51.97	46.62	44.35	42.83	42.54
		15	51.97	46.62	44.35	42.83	42.54
		20	51.97	46.62	44.35	42.83	42.54