Parallel Genetic Algorithm for High School Timetabling

Sanjay R. Sutar
Asso. Professor, Dr. B. A. T. University, Lonere,
Research Scholar, SGGSIET,
Nanded, INDIA.

Rajan S. Bichkar
Professor, E&TC and Dean R&D, G. H. Raisoni
College of Engg. & Mgt.,
Pune, INDIA.

ABSTRACT
The high school timetabling problem is a combinatorial optimization problem, proved to be NP-hard. It is a task to assign class–teacher interactions to rooms and timeslots of a weekly schedule. The nature of the problem varies depending on the region and institution. It has several hard and soft constraints. It means finding an assignment, such that no hard constraints are violated and the number of violations of soft constraints is minimized. Large and complex high school timetabling problem taken from real life, often takes long time to do manually. Hence, automated timetabling has attracted researchers since 1980s and many techniques have been proposed to solve it. Genetic Algorithm can be effectively used to solve such difficult problem. We propose the Parallel Genetic Algorithm (PGA) with customized operators and probabilistic repair to solve "hard timetabling" test problems hdtt4, hdtt5 and hdtt6 given by Professor Kate Smith-Miles in OR-Library. The optimal objective function for each of these problems is no clashes and fulfilling teacher's workload on each class in given room. The functions are designed for intelligent operators and repair. The PGA consisting operators augmented with problem specific knowledge and probabilistic repair in crossover converges faster than Simple Genetic Algorithm (SGA) and give solution within ***few*** seconds. The results are compared with the recent work carried out using different methodologies on same data set.

General Terms
Timetable scheduling, Genetic algorithm

Keywords
Simple Genetic Algorithm, Parallel Genetic Algorithm, Hard Timetabling, Repair

1. INTRODUCTION
Timetabling problems are optimization problems, considered to be a subset of scheduling problems. There are various problems in this category such as education timetabling, healthcare scheduling, transportation, sports and entertainment scheduling. The events have to be ordered in time slots while satisfying various constraints. Typically, timetabling problems are classified in three main categories: university, school, and exam timetabling problems.

It is very difficult to solve school timetabling by conventional methods and the computation to find optimal solution increases exponentially with problem size. Hence efficient search methods are used to produce optimal or near optimal timetable satisfying the constraints. Institutes have to perform this task regularly which means a large wastage of time and efforts. Design of techniques for the automated timetables is still of interest. The school timetabling is to assign class, teacher and room tuples to time slots predefined number of times, so as to satisfy the hard constraints, and minimize the violation of soft constraints. The hard constraints include all class-teacher meetings must be included, no class or teacher should appear more than once in a slot, certain classes may need to be split and rearranged. The soft constraints represent expected features of the timetable, however if these constraints are not satisfied it will be still valid. We have to minimize the soft constraint cost and hence to improve the quality of the timetable. Typical soft constraints for the timetabling problem include class or teacher preferences. Research work in school timetabling started in 1960s. It focused on greedy technique and on local search methods such as simulated annealing, tabu search, and genetic algorithms. The metaheuristics techniques seem to be the most efficient and able to generate solutions in reasonable time and they can be adapted to different forms of problems. Hence, metaheuristics are used for large and complex instances. A large number of diverse methods have been proposed for solving timetabling problems, from a number of disciplines like Operations Research, Computational Intelligence, and Artificial Intelligence and development in solving them is a main goal of current research in these areas.

Genetic Algorithm (GA) is a population based algorithm amenable for parallelization. Genetic algorithms are used to solve problems similar to the evolution. Initial population is created with randomly built individuals. Each individual is one of the possible solutions to the problem. The population is then modified with evolutional process similar to nature, i.e. evaluation, selection, crossover and mutation. The algorithm stops if a termination criterion is met. GAs produce timetables of comparative quality to that generated and used by the researchers. The timetable created meets all timetabling requirements without violating the constraints. The main contribution of this paper is a parallel genetic algorithm with customized operators to solve hdtt4, hdtt5, hdtt6 problems within short times which are less than most of the known times so far, reported by various techniques with same experimental set up.

The following section provides previous work employing genetic algorithms to solve the school timetabling problem. Section III describes an overview of the hard timetabling, hdtt. The PGA with customized operators and probabilistic repair in crossover, used to solve hdtt problems is presented in section IV. The performance of the PGA in solving the problems is discussed in section V. A summary and future work is described in section VI.

2. GENETIC ALGORITHMS AND SCHOOL TIMETABLING
D. Abramson and J. Abela [1] showed how the execution time can be reduced by using shared memory multiprocessor while applying GA to the school timetabling problem. Alberto Colorni et al. [2] presented a model, algorithms and programs for the timetable problem, with reference to a real life application. Various versions of simulated annealing and tabu search were compared with a GA based approach. Tabu search gave the best performance and the GA did better than simulated annealing.

Andrea Schaerf [3] proposed an algorithm based on local search techniques. The algorithm used different techniques and different types of moves alternatively. Also the hard constraints were adaptively relaxed. The algorithm worked successfully within some large high schools with different side constraints. Adora E. Calaor et al. [4] presented hybrid technique, parallel genetic algorithm with simulated annealing to solve school timetabling. They explained how to run the algorithms in parallelization on a local network. They compared results on the different parallel models.

Alpay Alkan and Ender Ozcan [5] discussed new operators to be applied in evolutionary algorithms for timetabling, such as exam timetabling. The operators are violation directed mutations, crossovers, and a violation directed hierarchical hill climbing method. Tests gave good results on a small part of a real data.

Leonardo Aparecido Ciscon et al. [6] proposed a memetic algorithm, a hybridization that increased the robustness and the quality of the results, giving more appropriate solutions to real timetabling problem. Two methodologies, simultaneous elimination of open periods and isolated classes were compared. Rushil Raghavjee and Nelishia Pillay [7] proposed genetic algorithms for school timetabling problem. The fitness of the offspring was compared to that of the parent to check the effect of mutation on the quality of the timetable. They found that in the most of cases there was no improvement in the fitness. Hence they tested a "nondestructive" version of the mutation operator. The fitness after each mutation swap compared with the fitness of the parent. If the fitness was better further swaps were not performed and the offspring returned. If there was no improvement then all swaps were performed and the offspring returned. This mutation operator improved the performance of the GA.

Nedim Srndic et al. [8] described a PGA for solving the weekly timetable problem for elementary schools. They proposed methods for chromosome representation and fitness evaluation, and specific recombination and mutation operators. The method used a coarse grained PGA, suitable for execution on a Beowulf cluster. Eugene Ruben Ramirez [9] employed smart operators, Violation-Directed Mutation (VDM), Event-Freeing Mutation (EFM), Stochastic Violation-Directed Mutation (SVDM) and Stochastic Event-Freeing Mutation (SEFM) during the mutation process to solve two high school timetabling problems, with and without fixed master schedule. Violation directed mutation with a one point crossover gave the best result. The second type of problem was solved using GA by increasing population size. Michael Pimmer and Gunther R. Raidl [10] used the international, real-world instances of the benchmarking project for high school timetabling. Timeslot filling heuristic (TFH) was used for creating timetables. Selected timeslots were iteratively filled with sets of events. Nelishia Pillay [11] presented the performance comparison of a selection constructive hyper-heuristic (SCHH), a generation constructive hyper-heuristic (GCHH), a selection perturbative hyper-heuristic (SPHH) and a hybrid hyper-heuristic (GPHH) combination of a generation constructive and a selection perturbative hyper-heuristic, in solving the school timetabling problem. Evolutionary algorithm was used to search the heuristic space. Five problems in the Abramson benchmark set were solved with all hyper-heuristics. SPHH produced the best results for the all the problems. George H. G. Fonseca and Haroldo G. Santos [12] proposed memetic algorithm for the high school timetabling problem. A mixed Simulated Annealing - Iterated Local Search approach (SA-ILS) was applied to all the individuals in the population at iteration. The approach was suitable, especially to small instances of the problem.

Nelishia Pillay [13] provided an overview of the research carried out in school timetabling, details of problem sets and areas for further research. She attempted to provide a standard definition of the problem in terms of hard constraints, soft constraints and problem requirements. She gave an overview of techniques applied to solve the school timetabling problem. R. Raghavjee and N. Pillay [14] applied a genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem. A two-phased approach, with the first phase focusing on hard constraints and the second on soft constraints, was proposed. The genetic algorithm selection perturbative hyper-heuristic (GASPHH) was applied to five different school timetabling problems. The performance was compared to that of other methods, including a GA. GASPHH performed well over all five problems.

3. THE HARD TIMETABLING PROBLEM

We propose GA based solution to the high school timetabling problems hdtt4, hdtt5, and hdtt6, given as "hard timetabling" test data sets in OR-Library. All periods are to be utilized with very less or no options for each allocation. D. Abramson and H. Dang [15], M. Randall, D. Abramson and C. Wild [16], K. A. Smith, D. Abramson and D. Duke [17] originally used these data sets. Table-1 shows the grouping of rows according to venue for hdtt4.

Table-1 hdtt4 requirements matrix

C/T	venue1				venue2				venue3				venue4			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	2	2	1	2	2	5	1	2	2	1	1	2	3	1	2	1
2	1	1	1	2	0	4	3	2	0	0	5	1	1	4	1	4
3	1	1	1	6	1	2	1	0	2	1	4	1	3	3	2	1
4	2	2	3	2	2	2	1	2	6	1	2	1	2	0	1	1

Requirements matrices for hdtt5 and hdtt6 are given in Appendix. It is read as follows:

If there are C classes, T teachers, V venues, and P periods, then the venue1 block of Table-1 indicates the number of each class teacher interactions in venue1 across the P periods. It is a five days week, six periods a day with a total of 30 periods.

All five problems have the following hard constraints:

- A room must be allocated only once to a timeslot.
- A class must be scheduled only once in a period.
- A teacher must be scheduled only once in a period.
- All class-teacher-venue tuples must be scheduled the expected number of times.

4. A PARALLEL GENETIC ALGORITHM WITH CUSTOMIZED OPERATORS FOR HARD TIMETABLING PROBLEM

We have used GAlib, a C++ library of genetic algorithm objects developed by Matthew Wall of Massachusetts Institute of Technology [18]. The library includes tools for using genetic algorithms to do optimization in any C++ program. We applied 'simple' genetic algorithm as described by Goldberg [19] in his book, with varying parameters to solve hard timetabling problems. SGA creates an entirely new population of individuals by selecting from the previous population then mating to produce the new offspring for the new population. The crossover operator defines the procedure for generating a child from two parents. Mutation operator is based on the value of the mutation probability. Our representation is 1D array of integers which contains <Room, Teacher> tuples i.e. weekly timetable. Objective function returns the fitness score of individual chromosome (timetable) based on error value (error), which is sum of number of workload violations (error1) and number of clashes (error2).

It is given by-

Score= [1/ (0.1+error)], hence timetable with zero error value has score 10. Same room or teacher value at the same index in succeeding classes leads to a clash. The values, error1 and error2 are expressed by the following two equations:

$$error1 = \sum |(Wrct - Arct)|$$

$\forall$ 0<=r<rooms, 0<=c<classes, 0<=t<teachers. (1)

W=workload, A=allotment

$$error2 = \sum_{l=0}^{classes-2} \sum_{m=l+1}^{classes-1} \sum_{n=0}^{slots-1} 1 \quad (2)$$

If ((TT [l*slots+n]) == (TT [m*slots+n])), otherwise 0.

Where, TT is individual timetable (chromosome) and

slots= (2*days*hours)

GA operators have been modified which use problem specific knowledge during the evolution process, to speed up the search. Workload requirements are satisfied by customized initializer which initializes the chromosome with room and teacher values. Mutation is probabilistic and ***adaptive***, as the probability is based on number of errors in the individual, more the errors higher is the probability. It checks clash in succeeding classes probabilistically and replaces it with a random number within valid range.

The workload requirements are preserved while doing crossover by not cutting across classes in a chromosome. Our crossover site is determined randomly which lies within each class itself. Two children are generated by copying left part of the respective parent within the respective class to left part of the corresponding class in child. Their right parts are made up of right part of the respective parent. Workload allotment done so far, after copying first part to children is updated in arrays and the difference with actual workload is stored. The entire process is given by Algorithm-1. Probabilistic ***repair*** function resolves overlaps in a child, if any.

(1) Copy <room, teacher> pairs upto 'site' from parent1 and parent2 to corresponding classes of child1 and child2 respectively
(2) Update allotment done so far after copying left parts of parents
(3) Calculate difference between actual requirement and partial allotment
(4) If (difference > 0) then copy corresponding <room, teacher> pairs to right parts of children else fill the positions with invalid value -1
(5) Check new difference and insert <room, teacher> pairs after 'site' as per actual requirement by replacing -1s

Algorithm-1 Intelligent Crossover

The PGA has multiple, independent populations. The populations are created by cloning the chromosome or population that we pass while creating it. A steady-state genetic algorithm is used for each population evolution. The steady-state GA uses overlapping populations with a specified amount of overlap. An algorithm creates a temporary population of individuals per generation, adds them to the previous population, and then the worst individuals are removed to bring the population to its original size. The amount of overlap between generations is specified as the percentage of the population to be replaced each generation. New offspring are added to the population and then the worst individuals are removed.

Some individuals migrate from one population to another in each generation of PGA. A specified number of best individuals of each population migrate to its neighbor. The main population is updated each generation with best individual from each population. The initializer, modified crossover, adaptive mutation, and repair are also used in PGA.

5. RESULTS AND DISCUSSION

Our implementation is in C++ using Dev C++ compiler, version 4.9.9.2 on Intel(R) Core™ 2 Duo CPU 2.4 GHz with 2 GB of memory and Windows Vista. SGA could give solution to hard timetabling, hdtt4 within a few minutes in 80000 generations (Figure-1). Population size was fixed to 100 after performing trial runs.

The Figure-2 shows performance graph of SGA on hdtt4 with modified operators. An algorithm gave solution in 400 generations and within ***few*** seconds compared to that of SGA (80000 and around 9 minutes) [20]. The graphs in Figure-3 show errors vs. number of generations of customized PGA applied to solve hdtt4, hdtt5 and hdtt6 respectively. We obtained the results with 110,250 and 1200 generations, within ***three, fifteen and one hundred twenty*** seconds respectively which is worth noting in hard timetabling. A significant improvement in execution times was observed. Execution times are less than the most of known times so far, out of different methodologies. The performance for problems was tested against the comparison given by Nelishia Pillay [11]. The comparison was carried out on machine ***equivalent*** to our configuration. The following methodologies were applied to the Abramson benchmark set [17].

GA with SPHH- Selection perturbative hyper-heuristic, NN-TT2 and NN-TT3- Neural network approaches, SA1 and SA2- Simulated annealing, TS- Tabu search and GS- Greedy search. Our customized GA performed comparable to SPHH, at par with GS and SA2 while better than NN-TT2, NN-TT3 and TS on hdtt4. The customized PGA ***outperformed*** all (on hdtt4 and hdtt5). It performed ***very well*** except SA1 and GS (on hdtt6). Table-2 lists the performance comparison with other methods. We observed that customized operators and parallel variant helped in giving faster solutions for the GA based timetabling.

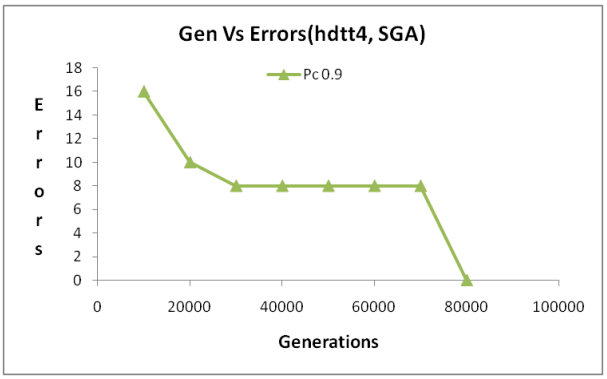

Figure-1 Performance of SGA on hdtt4 (Linear Scaling, Roulette Wheel Selection, Pm=0.001, Ps=100, Errors=216 for 0 generation)

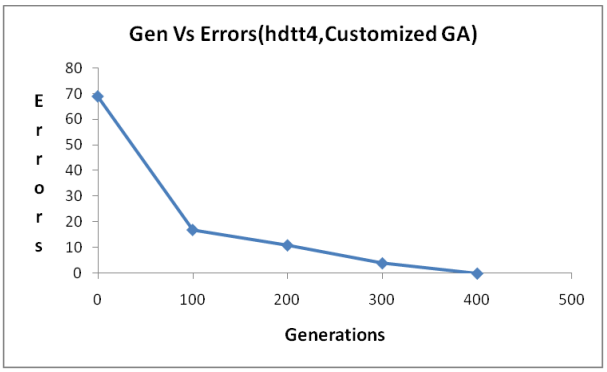

Figure-2 Performance of SGA with customized operators on hdtt4 (Linear Scaling, Roulette Wheel Selection, Ps=100, Pc=1.0)

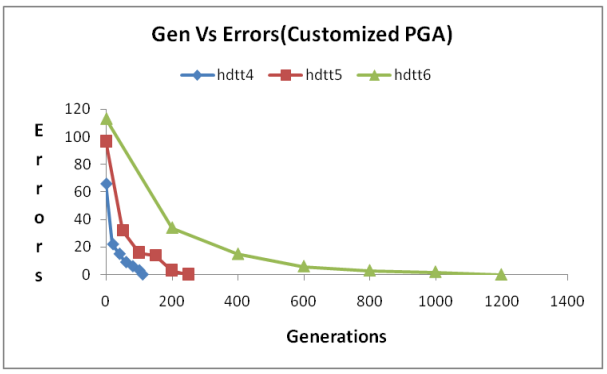

Figure-3 Performance of customized PGA on hdtt4, hdtt5, hdtt6

Table-2 Performance comparison (Execution time in Seconds)

Method	Dataset		
	hdtt4	hdtt5	hdtt6
SPHH	12	120	300
NN-TT2	29	116	291
NN-TT3	109	146	227
SA1	-	72	80
SA2	15	41	123
TS	630	87	1144
GS	16	39	78
Proposed PGA	**3**	**15**	**120**

6. CONCLUSION AND FUTURE WORK

We presented a customized PGA to solve high school timetabling problems hdtt, "hard timetabling", specifically hdtt4, hdtt5 and hdtt6 given as test data sets in OR-Library. The SGA produced a solution that met all the constraints for hdtt4 problem, but was unable to solve higher problems even after longer runs. Use of domain knowledge during the initialization, crossover, and mutation helped to speed up the search. The probabilistic repair was effective in boosting the fitness of the population. An algorithm gave results in a few seconds i.e. in much less time compared to SGA. Customized PGA performed ***better*** than known times given in comparative studies on similar configuration (except SA1 and GS for hdtt6). We will apply the PGA to additional hdtt and school timetabling problems in future.

7. REFERENCES

[1] D. Abramson and J. Abela, "A Parallel Genetic Algorithm for Solving the School Timetabling Problem," Appeared in 15th Australian Computer Science Conference, pp. 1-11, Hobart, February, 1992

[2] Alberto Colorni, Marco Dorigo and Vittorio Maniezzo, "Metaheuristics for Highschool Timetabling," Computational Optimization and Applications- 9, pp. 275–298, 1998

[3] Andrea Schaerf, "Local Search Techniques for Large High School Timetabling Problems," IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 29, No. 4, July, 1999

[4] Adora E. Calaor, Augusto Y. Hermosilla and Bobby O. Corpus Jr., "Parallel Hybrid Adventures with Simulated Annealing and Genetic Algorithms," Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), IEEE, 2002

[5] Alpay Alkan and Ender Ozcan, "Memetic Algorithms for Timetabling," Congress on Evolutionary Computation (CEC), IEEE, 2003

[6] Leonardo Aparecido Ciscon, Humberto Cesar Brandao de Oliveira, Michelle Cristina Alves Andrade, Guilherme Bastos Alvarenga and Ahmed Ali Abdalla Esmin, "The School Timetabling Problem: a focus on elimination of open periods and isolated classes," Proceedings of the 6th International Conference on Hybrid Intelligent Systems (HIS), IEEE, 2006

[7] Rushil Raghavjee and Nelishia Pillay, "Evolving Solutions to the School Timetabling Problem," World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009

[8] Nedim Srndic, Emir Pandzo, Mirza Dervisevic and Samim Konjicija, "The Application of a Parallel Genetic Algorithm to Timetabling of Elementary School Classes-A Coarse Grained Approach," 22nd International Symposium on Information, Communication and Automation Technologies, IEEE, 2009

[9] Eugene Ruben Ramirez, "Using Genetic Algorithms to Solve High School Course Timetabling Problems," A Thesis Presented to the Faculty of San Diego State University in Partial Fulfillment of the Requirements for the Degree, Master of Science in Computer Science, 2010

[10] Michael Pimmer and Gunther R. Raidl, "A Timeslot-Filling Heuristic Approach to Construct High-School Timetables," The 9th Metaheuristics International Conference, MIC, 2011

[11] Nelisha Pillay, "A Comparative Study of Hyper-Heuristics for Solving the School Timetabling Problem," Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, (SAICSIT'13), pp. 278-285, 2013

[12] George H.G. Fonseca and Haroldo G. Santos, "Memetic Algorithms for the High School Timetabling Problem," IEEE Congress on Evolutionary Computation, México, 2013

[13] Nelishia Pillay, "A survey of school timetabling research," Annals of Operation Research, 218:261–293, Springer Science Business Media, New York,2014

[14] R. Raghavjee and N Pillay, "A genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem," Orion Journal, Vol. 31 (1), pp. 39-60, 2015

[15] D. Abramson and H. Dang, "School timetables: a case study in simulated annealing: sequential and parallel algorithms," Lecture Notes in Economics and Mathematics Systems, V.Vidal(Ed.), Springer-Verlag Berlin, Chapter 5, pp. 103-124, 1993

[16] M. Randall, D. Abramson and C. Wild, "A general meta-heuristic based solver for combinatorial optimization problems," Technical report TR99-01, School of Information Technology, Bond University, Queensland, Australia.

[17] K. A. Smith, D. Abramson and D. Duke, "Hopfield neural networks for timetabling: formulations, methods, and comparative results," Computers and Industrial Engineering, Vol. 44, No. 2, pp. 283-305, 2003

[18] Matthew Wall, "GAlib: A C++ Library of Genetic Algorithm Components," Version 2.4, Documentation Revision B, August, 1996, (http://lancet.mit.edu/ga/), Massachusetts Institute of Technology (MIT) (c) 1995-1996, Matthew Wall (c) 1996-2005

[19] D.E.Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989

[20] Sanjay R. Sutar, Rajan S. Bichkar, "Genetic Algorithms based Timetabling using Knowledge Augmented Operators", International Journal of Computer Science and Information Security, (1947-5500), pp.570-579, Vol.14, No.11, 2016

8. APPENDIX

A-1 hdtt5 requirements matrix

C/T	Venue1					Venue2					Venue3					Venue4					Venue5				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	1	0	0	1	2	4	0	3	1	3	2	2	0	0	2	1	2	0	0	1	1	1	1	1	1
2	2	0	1	3	1	0	1	0	1	3	3	2	1	2	1	0	2	3	1	0	0	3	0	0	0
3	0	0	1	2	1	0	1	2	1	1	1	0	3	1	1	2	0	2	0	4	1	2	3	1	0
4	1	2	3	1	0	1	0	3	0	0	3	0	0	0	3	1	3	0	2	1	1	2	1	1	1
5	0	2	1	4	1	2	1	0	2	0	1	0	0	1	1	1	1	0	2	1	1	3	2	2	1

A-2 hdtt6 requirements matrix

C/T	Venue1						Venue2						Venue3						Venue4						Venue5						Venue6					
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	1	0	1	1	0	3	2	1	1	0	2	0	2	1	1	1	0	0	0	2	0	0	2	1	0	1	0	0	0	3	0	1	1	0	2	0
2	0	2	0	0	0	0	0	0	1	0	2	3	1	0	3	0	1	0	0	1	1	2	0	2	0	1	0	3	1	0	0	3	2	1	0	0
3	0	1	0	1	0	0	0	0	0	0	2	1	0	0	1	1	2	1	0	2	0	0	3	0	3	0	1	4	0	2	0	1	0	2	1	1
4	1	0	2	1	0	1	1	0	0	0	0	0	1	1	1	1	1	2	1	1	2	1	0	0	2	0	1	0	0	1	1	2	2	3	0	0
5	2	1	1	2	2	1	0	2	1	2	2	1	1	1	1	0	0	0	1	0	1	0	1	2	2	1	0	0	0	0	1	0	0	0	1	0
6	0	1	1	1	1	2	3	0	1	1	1	0	1	1	2	0	1	0	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1