International Journal of Computer Applications (0975 – 8887)
Volume 114 – No. 18, March 2015

Genetically Evolved Solution to Timetable Scheduling Problem

Sandesh Timilsina
Sharda University
Department of Computer Science and Engineering

Rohit Negi
Sharda University
Department of Computer Science and Engineering

Yashika Khurana
Sharda University
Department of Computer Science and Engineering

Jyotsna Seth
Sharda University
Department of Computer Science and Engineering

ABSTRACT
The simultaneous advancement in genetic modeling and data computational capabilities has prompted profound interest of scientists across the globe in the field of timetable scheduling. The wider usage of timetable scheduling in complex data manipulation and computation has attracted many researchers to put forward their theory regarding the use of genetic algorithms. The progression on this field has increased the efficiency of the timetable to use the limited resources in the given time to get productive results. This paper describes various genetic algorithmic methods.

Keywords
Genetic Algorithm, Timetable, Crossover, Mutation, Constraints, Fitness

1. INTRODUCTION
Genetic Algorithm is the method based on the natural process of biological evolution that can be used to solve the problems which are difficult to solve with classical methods. Genetic algorithm is non-deterministic and is used to solve mainly NP-hard problem like timetable scheduling problem. There are situations where even after spending many hours trying to find a perfect schedule, the optimal solution is not found. Scheduling timetable being an arduous task for manual computation, the use of automated genetic algorithm can profoundly increase the efficiency of the process.

Even with all these benefits, the perfect solution to automated timetable scheduling is yet to be found as the timetable scheduling problem is complex due to the large amount of data being manipulated and the variation of constraints involved in timetable scheduling.

1.1 Genetic Algorithm (GA)
It is a procedure inspired by biologic evolution (Charles Darwin), programmed in computers and oriented to produce solutions for problems that have been difficult to solve with classical approaches (wgonz 2010). It is a non-deterministic algorithm that starts with several solutions of the given problem which is known as the initial generation and finds the best solution evolved from them using the biological operators like selection, crossover and mutation.

1.2 Chromosome Representation
It is important to note that genetic algorithm treats each solution as a string to speed up the algorithm. In fact, each solution is represented as Chromosome using binary strings of 0's and 1's.

Individual	Generation	Fitness	Mating pool
1	101	5	101
2	001	1	011
3	011	3	111
4	111	7	111

Fig 1: Chromosome Representation

1.3 Initial Population
Creating Initial population is the first step of the GA. Initial population is created by the randomly generated solutions. These solutions are correct solutions but not the best. Then by using the evolution process, the best solution is evaluated from the population. Size of the population is not fixed and completely depends on the need. Smaller population will vanish after some generations while larger population will increase the chances of getting more accurate solution.

1.4 Fitness Function
The initial random population is assessed with the help of fitness function. After evaluation those individuals with more fitness are likely to survive and take part in reproduction. Fitness function is mainly chosen according to the constraints.

1.5 Operators
1.5.1 Selection
When each individual in the population is evaluated, those individuals with higher fitness have more probability to get selected and come in the mating pool to produce new generation. The Individuals with the lowest fitness are likely to get extinct. In the example shown in Fig. 2, the individual having the fitness 7(111) has the highest probability to get selected while the individual with the lowest fitness 1(001) gets extinct.

```
0100101110001010 11 Chromosome
01001011|1000101011 Splited into genes
0100|1011|10001|01011 Splited into data
```

Fig 2: Selection

1.5.2 Crossover

Crossover is a process where two or more individuals are selected to produce new individual(s) by exchanging their gene information. Crossover occurs in favor of the individuals having higher fitness.

001111001010101111000101110101	parent 1
001111010001011001001001011001	partent2
001111001010101001001001011001	child 1
001111001010101001111010001011	child 2
001111010001011111000101110101	child 3

Fig 3: Crossover

Crossover probability tells how often the crossover will take place. If the crossover probability is 0%, it will result to offspring same as parents. 100% crossover probability will produce offspring by crossover.

1.5.3 Mutation

Mutation prevents the occurring of the local optima by allowing us to search other parts of the search space. During mutation, the genes of the selected chromosomes are changed randomly. It is one bit change process.

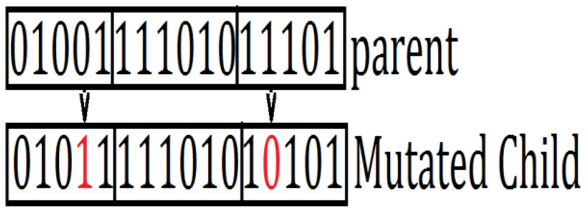

Fig 4: Mutation

Mutation probability tells how often the parts of chromosome will be mutated. If the mutation probability is 0%, the chromosome will not be mutated while in other cases mutation will occur to change the chromosome.

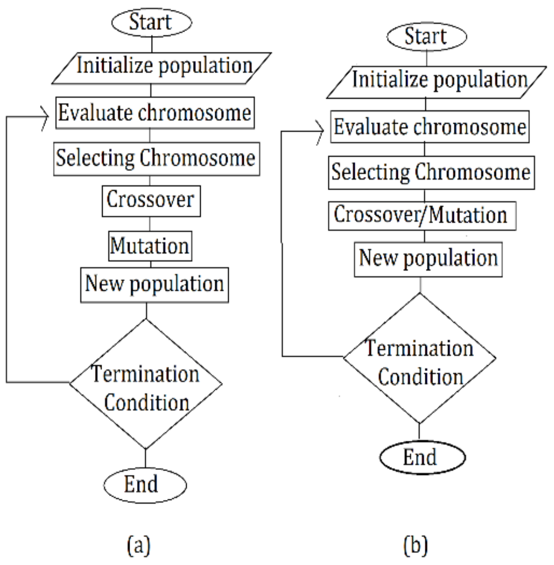

Fig 5: Genetic Algorithm Flowchart

Since crossover and mutation occur according to their respective probability, either one or both will occur after each iteration.

1.5.4 Timetable Scheduling Problem

Timetable scheduling problem can be defined as the distribution of the limited number of resources to different events such that maximum constraints are satisfied. Wren (1996) explained the timetabling as follows:

‘‘Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives.’’

Many researchers around the globe are involved in the timetable scheduling problem, yet the problem is not perfectly solved because constraints vary in each scheduling problem. Using genetic programming, the better solution to various scheduling problems with different constraints can be provided.

Following constraints are to be considered while dealing with the timetable scheduling problem:

a) Hard constraints :

- Same resource (teacher or room) must not be assigned to two events at the same time.
- All the available courses must be scheduled [2].
- Same semester events can be assigned at the same time slot if both the events are of type “lab practice” as each “lab practice” classes for a course scheduled within a week are attended by different student groups. Same semester events of type “theory” or when one event is “theory” and the other event is “lab practice” must not run concurrently.
- Each lecture must be held in one of the rooms that are allocated for the lecture.
- Each room has different availability schedule.
- Each lecture must be assigned to the teacher who belongs to a group of teachers designated for the lecture.
- Specific lectures must be strictly assigned to specific teachers.
- One teacher must be assigned to each class of type “theory” while two teachers must be assigned to each class of type “lab practice” [3].
- Larger room must be assigned to lecture with a large group of students [1].

b) Soft constraints:

- Every teacher has his/her own availability schedule and preferred time periods.
- All teachers have minimum and maximum limit of weekly work-hours.
- If a lecture within a week is divided into more than one noncontiguous lecture, a specific number of days must be left between the lectures.
- The travel time of teachers and students between classrooms can be minimized.

- The time gaps within the schedule of each teacher can be minimized.
- The time gaps within the schedule of each classroom can be minimized [3].

2. GENETIC ALGORITHM IMPLEMENTATION

2.1 Different Approaches

First method to solve the timetabling problem is to consider it as a 3D-cutting problem, where the time table is described as a 3D structure in which days, timeslots and rooms are the three dimensions. The classes are the cubes which should be placed in 3D timetable structure. For the simplicity, the large number of binary variables can be reduced to the acceptable size by eliminating certain dimensions of the problem and taking them as constrains. This makes the individual size of the gene smaller and easier to use genetic algorithm approach. Three auxiliary 3D structures are created to satisfy the constraints. X and Y axis of all three structure represent the day and time. The Z axis varies to denote room, groups or instructors (see fig 6). While checking, if more classes compete for a particular resource, a new constraint has to be generated. Each individual of population (possible solution) represents one time table. Although the approach of genetic algorithm is adopted for solving this problem in which algorithm starts from an infeasible timetable and tries to get the feasible one, but to significantly enhance the performance, modified and better genetic operators should be used[1].

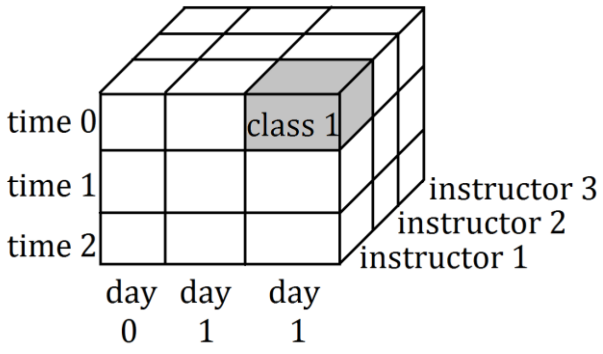

Fig 6: 3D structure of timetable [1]

On the other hand, there can be a simpler and feasible method- Set Evaluation Method- in which an adaptive genetic algorithm model is used for improving effectiveness of automatic arranging examination timetable. The Set-evaluation method explains chromosome as a group of elements (E) and each element comprises of two type of data: course(C), and group (G) i.e. E={C,G}. Set C can be represented by c number of courses $C=\{c_1,c_2,c_3,c_4\ldots\ldots.,c_c\}$ and G can be represented by g number of groups $G=\{g_1,g,g_3,g_4,\ldots..g_g\}$. Chromosomes are evaluated and operated by the genetic operations [2].

As the constraints vary and problem becomes more difficult, the standard and direct approach of Genetic Algorithm does not give the desired optimal automated timetable. Advanced-GA can be used in such cases. It makes the use of an indirect representation of timetable based on the events priorities. In this method, the encoded solution (chromosome) usually represents an ordered list of events, which are placed into the timetable with the help of the timetable builder. The timetable builder uses heuristic and local search to place events into the timetable based on problem constraints. The significance of using a timetable builder is to handle the hard constraints while genetic algorithm is to deal with soft constraints. The number of advanced local search operators including the Micro-GA combinatorial hill-climbing operator can be used in order to avoid local optima in order to fulfill constraints and discover optimal solutions efficiently. In this method the clusters are used to schedule the timetable. Cluster is set of events in which events are defines to satisfy hard constraints. Events are divided into clusters, one cluster for each day. In the cluster events are arranged according to their priorities such that highest priority events always get first position. Events are selected from the clusters priority wise and placed into the timetable [3].

2.2 Evaluation Strategy

The population evaluation is very important step on the genetic algorithm. It enables us to judge each individual (or chromosome or solution) depending on their feasibility. Individuals having higher fitness are likely to survive, rest are likely to get extinct. The choice of the fitness function completely depends on the user requirements. Different condition may require different fitness function.

The fitness function can be defined as:

$F_1(x)$ = (Number of conflicts)*K + Quality

Where,

K is a large constant. The “number of conflicts” represents the number of constraints that have been violated by a particular individual. When an individual does not violate any constraint then it is considered as a feasible solution. The quality of the timetable is determined by earliness of scheduled classes. The genetic algorithm will try to schedule classes as early in the morning as possible indirectly minimizing the number of gaps in the timetable. In this fitness function, the term “quality” is used to represent a particular soft constraint (i.e. earliness of the class) but other soft constraints are not included. And if the “number of conflicts” includes both hard and soft constraints then it will generate individuals violating the hard constraint as a solution [1].

Or, it can also be defined as:

$F_2(x)=\sum^{m}_{i=1} W_i^{hard} \text{ X } COST_i^{\ hard}(x) + \sum^{n}_{i=1} W_i^{\ soft} \text{ X } COST_i^{\ soft}(x)$

Where,

x = Timetable under evaluation.

$COST_i^{\ hard}(x)$ =Measure of violation of the i_{th} hard constraint.

$COST_i^{\ soft}(x)$ =A measure of violation of the i_{th} soft constraint.

$W_i^{\ soft}$ = Weight factor for the i_{th} soft constraint.

W_i^{hard}=Weight factor for the i_{th} hard constraint.

m= Total number of hard constraints under consideration.

n=Total number of soft constraints under consideration [2][3].

The value of both the fitness function described above increases as the number of violated constraints increases. The individual which has lower fitness function value is the better one.

To differentiate between violation of hard constraints and violation of soft constraints and to avoid the violation of constraints:

- The multiplication of large constant K to the “number of conflict” is used in $F_1(x)$[1].

- While, in case of $F_2(x)$ large value of W_i^{hard} can be used[2][3].

2.3 Operation

2.3.1 Selection

The purpose of the selection function is to select the best individuals of the population as it is expected that better parents have better child.

3D-cutting method uses tournament eliminating selection in its improved genetic algorithm, which chooses and eliminates bad individuals from the current population, making room for new children that will be born from the remaining individuals. Probability of elimination will be more for the individual having larger fitness function value as higher the value of the fitness function lesser the fitness of the individual. There also exists a possibility (though very small) to eliminate the best individual from the population, so there must be a protection mechanism to keep the good genetic material safe and this process is known as elitism [1].

While in Set-evaluation method, it used simple selection function which selects the best individuals from the population for the further evolution depending on their fitness value.

Both in 3D-cutting method of improved genetic operators and in Advanced-GA method of advanced genetic algorithm roulette wheel parent selection algorithm is used, in which probability of selection is directly proportional to the fitness value. In this cumulative fitness evaluated as:

$q_k = \sum_{i=1}^{k}$ fitness (individual$_i$) , $D = \max(q_k)$

Where k=1, 2…. POPULATION_SIZE.

The algorithm generates a random number r from the interval (0, D) and selects an individual which satisfies the condition: $\max q_k <= r$ [1],[3].

2.3.2 Crossover

In most of the cases the best choice is to use uniform crossover [1]. It checks all the genes of both the parents. If parents have same gene, this value is written to the child. If values of parents' genes differ, then the algorithm randomly chooses one parent as a dominant one and takes its gene.

```
for each gene in (parent1 , parent2)
{   if(parent1[gene]==parent2[gene])
{     child[gene]=parent1[gene];
 }else{
 child[gene]=random(parent1 ,parent2)[gene];
   }
 }
```

Fig 7: Uniform crossover algorithm [1].

Above algorithm explains that there are no conflicts in case of same value genes but the conflicts may arise in case of different values of parents' gene. Modified crossover operator is used in 3D-cutting method which counts the number of potential conflicts generated by the selection of different genes from both parents' choices. The gene from the parent which causes the fewer conflicts is chosen [1].

A bit different crossover strategy is used in Set-evaluation method. It performs the crossover with respect to both the parents. It first selects the number of positions of the genes of parent 1 and copies the values of these genes to the child 1. After copying, it deletes the genes of parent 2 which have same values as that of selected genes of parent 1 and finally fills the remaining space of child 1 with the residue genes of parent 2. Same principle is adopted by considering genes of parent 2 first. Standard genetic approach was used and result was tested.

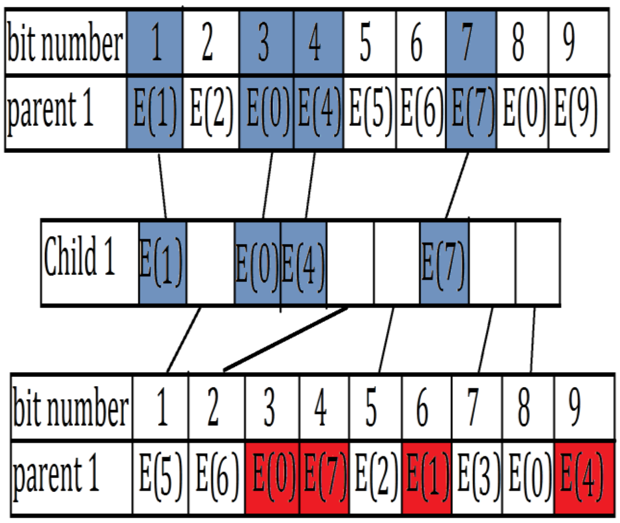

Fig 8: Crossover operation [2].

While selecting the operator, the experimental data can be used. First the standard genetic algorithm is used in which the 5-point crossover with other standard genetic operators such as mutation with probability 0.001 per bit is used and the record of feasibility of result is kept. The operator is selected on the basis of the result. If the result is better than the previous result then the operator is selected. Same technique was used in Advanced-GA method and uniform crossover was selected after conducting simulation experiment as it found to give best result [3].

2.3.3 Mutation

The probability of the mutation is taken as input in first method. For each gene in the chromosome a random value is generated from the interval (0, 1). If the generated random value is found to be smaller than the mutation probability then the gene is changed with a random value.

```
for each gene in individual{   if(p(Random) < $p_m$)
{
gene = get random value from possible values list;
}
```

Fig9: Mutation algorithm [1].

Crossover is modified in 3D-cutting method but such modifications were avoided for mutation. If such modification is applied to mutation operator then this will lead the population to the local optima [1].

Instead of changing the genes to a random value, Set-evaluation method uses the mutation algorithm which selects 2 positions of genes of a chromosome randomly and swaps their values [2].

```
begin
 Define fitness function
 Form prototype of chromosome
 gen ← 0
 Initial chromosome P1(gen), P2(gen)
 while (not terminate condition) do
 begin
   if (random < crossover probability)
   C1(gen) = Crossover P1(gen) ; C2(gen) = Crossover P2(gen )
   else
C1(gen) = Mutate P1(gen) ; C2(gen) = Mutate     P2(gen)
Examine P1(gen), P2(gen), C1(gen), C2(gen)
   gen ← gen + 1
Rank P1(gen), P2(gen), C1(gen), C2(gen)
    P1(gen) =SelectBestRank1; P2(gen) = SelectBestRank2
   End of loop
   end
```

Fig 10: Mutation algorithm [2]

The adaptive genetic algorithm used in Set-evaluation method is explained below. Algorithm performs either crossover or mutation in iteration, not both.

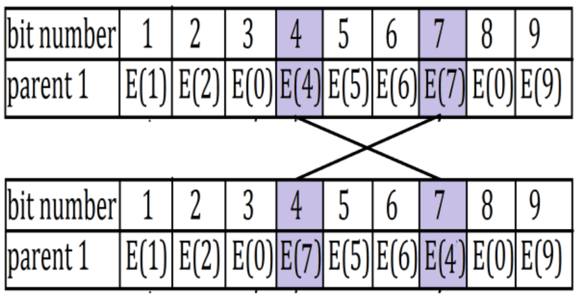

Fig 11: Adaptive genetic algorithm [2]

Bit mutation operator with a probability of 0.001 per bit is used by Advanced-GA method in its advanced genetic algorithm while conducting experiment with standard GA. When the other Genetic operators were added to the standard genetic algorithm then better results were found with Window Mutation operator with a probability of 0.3 and Mutate Chromosome operator with a probability of 0.1. All the operators that enhanced the result are listed below:

a. Uniform Crossover
b. Window Mutation operator with a probability of 0.3
c. Swap Chromosome operator with a probability of 0.1
d. Mutate Chromosome operator with a probability of 0.1
e. Varying fitness Function with square increase.
f. GA population of 200 genotypes, and
g. Micro GA combinatorial hill climbing operator

Apart from all these genetic operator of advanced genetic algorithm the domain specific Hill climbing operators were also adopted for the betterment of the result and to satisfy more constraints. These operators are applied to the best solution of the generation. These operators change the specific field (domain) of the timetable randomly and if the resulting timetable is better than the original timetable then the change is kept otherwise original timetable is restored. These operators are:

a. Change Day Hill Climbing Operator
b. Fix Teacher Hill Climbing Operator
c. Fix Room Hill Climbing Operator
d. Fix Day Hill Climbing Operator.

When the simulation experiments were performed with these domain specific operators they all were found to enhance the result [3].

3. RESULT

The 3D-cutting method was tested on small and large timetable scheduling problems at Faculty of Electrical Engineering and Computing (FER) in Zagreb. The small size problem was obtained from the large size with exclusion of about 70% of classes from the scheduling process. The small problem was solved without conflicts. When solving the large size problem, the basic algorithm stopped at about 95 conflicts. With intelligent operators, algorithm reached to 20 conflicts. Algorithm with the basic operator face problem. After a certain number of conflicts had been reached, very small improvements were achieved through algorithm running time. The algorithm with improved operators shows much better results. Fewer conflicts were achieved in a lesser amount of running time [1].

The Set-evaluation method was tested on final exam scheduling of all courses in 2012 - 2013 curriculum of bachelor degree of school of Engineering of University of the Thai Chamber of Commerce (UTCC) Thailand. The result of fitness function was observed for probabilities of crossover that vary from 0.00, 0.30, 0.40, 0.50, 0.60, 0.65, 0.70, 0.75, and 1.00.Five hundred generations of process was evaluated. Lower fitness value occurred between 0.65-0.75, more specifically, the best result was obtained at 0.75 crossover probability at which no hard constraint is violated [2].

The Advanced-GA method was applied to solve timetable scheduling problem in Technological Educational Institute of Serres, Greece and the result was compared with the man-made solution. The result showed that the man-made solution was able to satisfy all the hard constraints but genetically evolved solution was not able to satisfy all the hard constraints. On the other hand genetically evolved solution managed to satisfy soft constraints better than the man-made solution [3].

4. CONCLUSION

The results show that the basic standard genetic algorithm is unable to satisfy most of real world timetabling problem as the problem gets complicated. Modification to the standard genetic algorithm is needed to achieve better results. The modification can be done in any way that suits the situation. It can be either modification in the genetic operators or it can be the addition of some supporting algorithm or any other modification possible. The modifications or the changes made in genetic algorithm might not be standard because the result may vary as the constraints change. While using the Set-evaluation method best results were found at the 0.75 crossover probability which might not be the same for all scheduling problem. Scheduling problem with the similar constraints may get good result with the crossover probability near 0.75. Algorithms proposed in all three methods achieved much better results, but were unable to achieve the perfect result. These were also unable to check the robustness and efficiency of the methods and to determine the possibility of the method to use as standard method for scheduling. The algorithms have to be tested on many real world timetabling problems with every possible condition.

5. FUTURE WORK

Although the modification of genetic operator is adopted in 3D-cutting method to get the better results, still the proposed modification doesn't ensure 100% satisfaction of constraint. Also the Advanced-GA method which used advanced genetic algorithm with some supportive methods doesn't satisfy all the hard constraints. It clearly makes the need for the further modification in the method of solving the timetabling problem. Modifying the genetic operator is a good idea but even the modified crossover operator in 3D-cutting method doesn't remove the violation of hard constraints. Therefore, enhanced and more sophisticated modification needs to be applied not only to the crossover operator but also to mutation operator.

6. REFERENCES

[1] Branimir Sigl, Marin Golub, Vedran Mornar, "Solving Timetable Scheduling Problem by Using Genetic Algorithms", Information technology interfaces, June 16-19, 2003, pp. 519 – 524.

[2] Supachate Innet ,"A Noval Approach of Genetic Algorithm for Solving Examination Timetabling Problems,"International Symposium on Communications and Information Technologies, 2013.

[3] Spyros Kazarlis, Vassilios Petridis and Pavlina Fragkou, "Solving University Timetabling Problems Using Advanced Genetic Algorithms", 5th International conference on technology and automation, October 15-16, 2005, pp. 131-136.

[4] Henri Larget, "Genetic Algorithms used in Timetable Management", April 2012.

[5] Mitchell Melanie, "An Introduction to Genetic Algorithms," MIT Press, 1998 - 209 halaman.

[6] "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press, 1992 - 819 Seiten.

[7] Victor A. Bardadym, "Computer-Aided School and University Timetabling: The New Wave", first international conference Endiburgh, U.K., August 29-September 1, 1995, pp. 22-45.

[8] Wilhelm Erbern, Jurgen Keepler, "A genetic algorithm solving a weekly course- timetabling problem", first international conference Edinburgh, U.K., August 29-september 1, 1995, pp. 198-211.

[9] Martin Schmidt, "Solving real-life Time-tabling problem", 11th International Symposium, ISMIS'99, Waraw, Poland, june 8-11, 1999, pp 648-656.

[10] Liam T.G. Merlot, Natashia Boland, Barry D. Hughes, Peter J. Stuckey, "A Hybrid Algorithm for the Examination Timetabling Problem", 4th International conference, PATAT 2002, Gent, Belgium, August 21-23, 2002, pp. 207-231.

[11] Philip Kostuch, "The University Course Timetabling Problem with a Three-Phase Approach", 5th International conference, PATAT 2004, Pittsburgh, PA, USA, August 18-20, 2004, pp. 109-125.