CHAPTER II
LITERATURE REVIEW

2.1 Introduction

This chapter makes an analysis of the timetable literature broadly organized by algorithmic techniques and discusses the course timetabling problem. Various Major timetable solution generation algorithms have been presented. A detailed examination of the academic literature is provided within the context of these fundamental solution generation algorithms.

2.2 Course Timetabling Problem

The course timetabling problem is a kind of personnel scheduling problem that appears in the education domain. However, arranging objects such as pairing a course and a lecturer in the scheduling process can be viewed as assigning a lecturing job for a lecturer to take care of that course. Moreover, arranging a pair of course and lecturer at certain time slots can also be viewed as allocating or assigning resources (teachers and time slots) for an event.

The course timetabling problem has to consider two types of constraints, namely hard constraints and soft constraints (Burke, Jackson, Kingston and Weare, 1997). Hard constraints are essential and must be rigidly enforced to the schedule, while soft constraints are not essential but desirable (Burke and Petrovic, 2002). Quality of the schedule is measured by an objective function.

Historically the course timetabling problem has been studied extensively for about fifty years as one of very interesting research subject. Among the approaches that have been used to construct scheduling algorithm are sequential approach, cluster approach, constraint based approach, and improvement approach (Burke and Petrovic, 2002; Carter and Laporte, 1997).
The sequential approach selects the events one by one using a certain order, and then places the event into a safe tune slot. In order to prevent a potential conflict, a data structure, like a graph in the graph coloring problem, must be prepared in advance. The vertices of the graph represent the events and the edges connect conflicting events. The cluster approach provides a number of clusters to place a group of conflict events. To construct the schedule, the group is then assigned into a time slot using a certain method.

The constraint based method treats schedule components as variables. It constructs the schedule from an empty set by selecting element by element that comprise of schedule component. The improvement approach selects any arbitrary solution then improves the solution by heuristic methods.

A scheduling activity usually produces a kind of table called a timetable so that some researchers referred the scheduling activity as timetabling activity (Duong and Lam, 2004; Gudes, Kuflik and Meisels, 1990). Relationship between scheduling and timetabling in detail can be found in (Wren, 1995).

Willemen (2002) defines course timetabling as a sub-class of scheduling for which the events take place at educational institutions. More formal and detail definition of the course timetabling problem stated (Tripathy, 1984) as:

"A certain number of meeting which are to be attended by a specific group of students and a teacher or teachers over a definite period of time, requiring certain resources in conformity with the availability of resources and fulfilling certain other requirements".

For a large university that consists of an enormous number of students, lecturers, rooms and courses, a significant effort is required to construct both of the timetables for every academic period. It happens due to the search space of the problem grows as the combinatorial function that has factorial complexity.
In terms of computational process, one essential change such as additional course sections or unavailability of any resources may require some significant additional time to the construction process of the course timetable.

2.3 Solution Generation Algorithms

There are many solution generation techniques that can be used to find a solution for the course timetabling problem such as

1. Integer Programming / Linear Programming
2. Simulated Annealing
3. Tabu Search
4. Genetic and Evolutionary Algorithms
5. Constraint Satisfaction Programming
6. Ant System Algorithm
7. Hybrid Algorithm

2.3.1 Linear Programming/Integer Programming

The Linear and Integer Programming techniques, the first applied to timetabling were developed from the broader area of mathematical programming. Mathematical programming is applicable to the class of problems characterized by a large number of variables that intersect within boundaries imposed by a set of restraining conditions (Thompson, 1967). The word "programming" means planning in this context and is related to the type of application (Feiring, 1986). This scheme of programming was developed during World War II in connection with finding optimal strategies for conducting the war effort and used afterwards in the fields of industry, commerce and government services (Bunday, 1984).

Linear Programming (LP) is the subset of mathematical programming concerned with the efficient allocation of limited resources to known activities with the objective of meeting a desired goal such as maximizing profits or
minimizing costs (Feiring, 1986). Integer Programming (IP) deals with the solution of mathematical programming problems in which some or all of the variables can assume non-negative integer values only. Although LP methods are very valuable in formulating and solving problems related to the efficient use of limited resources they are not restricted to only these problems (Bunday, 1984). Linear programming problems are generally acknowledged to be efficiently solved by just three methods, namely the graphical method, the simplex method, and the transportation method (Makower and Williamson, 1985).

The construction of a linear programming model involves three successive problem-solving steps. The first step identifies the unknown or independent decision variables. Step two requires the identification of the constraints and the formulation of these constraints as linear equations. Finally, in step three, the objective function is identified and written as a linear function of the decision variables.

Schniederjans and Kim (1987) described an application of a 0-1 goal programming approach at the University of Nebraska. The results of this application show the model’s capability to provide an assignment that satisfies departmental course offerings and teaching load objectives by considering personal preferences. However, the need for significant computational power limited the application of this algorithm.

Birbas, Daskalaki and Housos (1997) presented an efficient solution to the timetabling problem for the secondary educational system in Greece. Using an integer programming approach and utilizing commercial tools that were available for a class of scheduling problems, the process of locating the optimal solution was facilitated. Demonstrating the effectiveness of the model, it was illustrated that their method produced optimal results for the specific constraints of the problem.
Drexl and Spreacher (1999) used a branch-and-bound algorithm in relation to semi-active schedules. It was demonstrated that these techniques were effective in providing solutions to these types of problems. Although no conclusion was given it was suggested that the algorithm would generate an optimal solution, which need not be semi-active.

Boronico (2000) presented a model (applied at Monomuth's University School of Business) that was used in scheduling undergraduate courses. The model presented relied on a hierarchical mathematical integer programming model in conjunction with discrete event simulation.

A binary integer programming model (Akif Bakir and Cihan Aksop, 2008) is applied to solve school timetabling problem. The timetable problem is represented as a linear 0,1 integer programming problem (Christelle Guéret, Christian Prins and Marc Sevaux, 2000) and the solution technique based on simplex method is used to obtain the solution.

### 2.3.2 Simulated Annealing Algorithm

This algorithm is inspired from the annealing process of metals or glasses they assume a low energy configuration will be constructed with an appropriate cooling schedule. In finding a solution, this algorithm begins with an arbitrary solution so that is considered as an improvement approach algorithm. The quality of the solution is measured as a fitness function and the next solution is searched from the neighborhood of the previous solution.

The fundamental idea of this meta-heuristic algorithm is to allow, with a certain probability, non improving moves in order to escape from the local solution that happens in the descent method. The probability of moving into non improving solution is kept decreasing during the search process.
Eglese and Rand (1987) incorporated an Annealing algorithm to help conference organizers develop timetables. This was the first time that an algorithm used in the natural sciences had now been used as the fundamental solution generation algorithm to solve the timetable problem. The authors claimed that the developed program permitted the user to update the designed timetable and provided a sample demonstration which illustrated this.

Abramson (1991) applied simulated annealing to construct school timetables. It simulates the tuples as the atoms and the course timetable cost as the system energy. The improvement process is done by swapping elements using two consecutive processes namely remove and insert process. The cost of swapping is measured from the cost of removing the element from one period and the cost of inserting the element in the other period. The temperature is used to the improvement process. This work was intended to construct a timetable for a school where students can be grouped into some disjoint classes. The core process of this work is to place a combination of class, teacher and room which is called an element into a time slot, where the required combination has been prepared previously. The objective Function is to minimize the class cost that is measured by the sum of number of violations appears in each period.

Concerned with the use of Simulated Annealing in the solution of the multi-objective examination timetable problem, (Thompson and Dowsland, 1996) proposed a solution method that optimized groups of objectives in different phases. Depending on the solution quality with respect to earlier phases in the algorithm, some decisions may be altered which do not deteriorate the final result. These limitations may disconnect the solution phase by causing optimal or near optimal solutions to be missed. It was suggested that a Simulated Annealing method can be used to overcome these variants.
A comparison of six different simulated annealing cooling schedules, for solving the school timetabling problem (Abramson, Krishnamoorthy and Dang, 1999) showed that incorporating a way of calculating the temperature that is used as the reheating point produced better results. The basic geometric cooling schedule, a scheme using two cooling rates and four schemes allowing reheating as well as cooling were examined. The third and fourth reheating schedules incorporated computing the temperature used as the reheating point. Extensive testing showed that the fourth reheating schedule produced higher quality and quicker results.

Chainate, Thapatsuwan and Pongcharoen (2008) proposed a repair process during neighborhood search process in the Simulated Annealing Algorithm to the course scheduling problem that consider lecturer preference to certain time slots. The aim of this experiment is to set some Simulated Annealing parameters that have better performance for their problems that include initial temperature, final temperature, neighborhood search operators and cooling schemes. The objective function is to minimize the total violation index which is the sum of weighted penalty of the constraint violation indices.

2.3.3 Tabu Search Algorithm

Tabu Search Algorithm guides the search process by enforcing restrictions. It restricts some alternatives by putting them in a forbidden (Tabu) list in order to prevent cycle as well as to guide the move into the more promising solution.

As a local search algorithm, the Tabu Search Algorithm begins with initial or arbitrary solution. It starts by assuming that all options are acceptable. Subsequently the algorithm will search another solution as best among neighborhood of the current, based on a certain criteria the process will update the Tabu list by adding unacceptable solutions. The history of search process is kept in a flexible memory. Such information will be used to
guide the move from current solution to the next solution in the neighborhood (Colorni et al., 1996; Glover and Laguna, 1997; Hertz, Taillard, and de Werra, 1997; Riaz, Wang and do Li, 2004). The process will continue until the required condition is satisfied.

Alvarez-Valdes, Crespo and Tamarit (2002) used a set of heuristic algorithms in a program for solving course timetabling related problems. A Tabu Search procedure had several strategies developed for it and tested, leading to a potent and quick algorithm which produced satisfactory results.

The work that used the Tabu search in the scheduling problem is done by Santos, Ochi and Souza (2005). They propose a method for class/teacher timetabling problem based on Tabu search algorithm that elaborate the diversification strategy. The objective function of this work is the sum of Cost of more than one teacher teaches the same class in a certain period, the number of times that a class has no activity in a certain period, number of class having the same teacher in a day more than a certain limit and dissatisfaction of personal requests from teachers. Besides using the short term memory with dynamic value of Tabu tenure, the proposed method applies two types of long term Memory for the diversification strategy. The first, Transition-Based Long-Term Memory, keep the number of movement involving teacher i and class j in a two dimensional matrix, while the other, Residence-Based Long-Term Memory, keeps number of iterations involving lesson m, teacher i class j and time slot k, in a sparse four dimensional matrix.

Tabu search algorithm to solve class/teacher or course timetabling problem is presented (Haroldo G. Santos, Luiz S. Ochi and Marcone J.F. Souza, 2005, Cagdas Hakan Aladag and Gulsum Hocaoglu, 2007).
2.3.4 Evolutionary and Genetic Algorithms

Evolutionary Algorithms (EAs) are a class of direct, probabilistic search and optimization algorithms gleaned from the model of organic evolution. A Genetic Algorithm (GA) is a type of EA and is regarded as being the most widely known EA in recent times (Back, 1995). GAs are a class of stochastic search algorithms based on biological evolution whose search strategy mimics natural selection by using an automated version of the “survival of the fittest” analogy (Sanchez, Shibata and Zadeh, 1997). It is a problem solving and optimization method that uses genetics as its model problem and applies the rules of reproduction, general crossover and mutation to pseudo-organizations so those organizations can pass beneficial and survival-enhancing traits to the new (next) generation (Chambers, 1995).

A GA differs from other search techniques in the following ways:

- GAs optimizes the trade-off between exploring new points in the search space and exploiting the information discovered thus far. This was proved using the K-armed bandit (an extension of the one-armed bandit) problem (Buckles and Petry, 1992).
- GAs have the property of implicit parallelism. Implicit parallelism means that the GA’s effect is equivalent to an extensive search of hyper planes of the given space, without directly testing all hyper plane values (Goldberg, 1989). Each schema denotes a hyper plane.
- GAs are randomised algorithms, in that they use operators whose results are governed by probability. The results for such operations are based on the value of a random number (Buckles and Petry, 1992). This means GAs use probabilistic transition rules, not deterministic rules.
- GAs operate on several solutions simultaneously, gathering information from current search points to a direct subsequent search. Their ability to maintain multiple solutions concurrently makes them less susceptible to the convergence problem of local maxima and noise (Goldberg, 1989).
• GAs work with a coding of the parameter set, not the parameters themselves (Goldberg, 1989).
• GAs search from a population of points, not a single point (Goldberg, 1989).
• GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge (Buckles and Petry, 1992).

An abstract view of the problem solution phase using a GA can be illustrated as follows (Buckles and Petry, 1992)

Generate an initial population, \( G \) (at time=0)
Evaluate the fitness function \( f[G(0)] \)
\( t = 0 \)
Repeat
\( t = t + 1 \)
Generate \( G (t) \) using \( G (t-1) \)
Evaluate \( f [G (t)] \)
Until either an acceptable solution is found or population convergence is achieved.

The representation of the timetable problem as a GA is very problem specific. Unlike LP/IP (where an almost “exact” mathematical formulation can be ascribed) the encoding scheme, which represents the template for the solution, must be determined.

A heavily constrained university timetable problem using a Genetic Algorithm based approach for solution generation was described by (Erben and Keppler, 1996). To facilitate this, a problem-specific chromosome representation and knowledge-augmented genetic operators was developed. It was asserted that these operators “intelligently” avoided building illegal timetables. The prototype timetable system that was presented was implemented in C and Prolog, and included an interactive graphical user interface. The test data included in this paper indicated promising results.
It is claimed that GAs have a greater capacity than any other search method in finding the largest number of possible solutions and depict the best possible results (Maxfield, 1997). GAs demonstrate real-world significance when applied to timetabling problems where a complex set of scheduling constraints and a various collection of individuals exist. It is generally argued that a timetabling GA develops the best possible schedule. However, GAs still fall short in the choice of control parameters, the exact roles of crossover and mutation, and the characterization of search landscapes for optimization and convergence characteristics (Srinivas and Patnaik, 1994). Some applications where GAs have been used successfully are analogue circuits, fuzzy logic, and neural networks (Maxfield, 1997).

A multi-objective genetic algorithm was proposed for a class/teacher timetabling problem, incorporating two distinct objectives, the minimization of the violations of both types of constraints, hard and soft, while respecting the two competing aspects - teachers and classes (Carrasco and Pato, 2000). The characteristics of a genetic multi-objective meta-heuristic and a non-dominated sorting genetic algorithm were discussed, using a standard fitness-sharing scheme improved with an elitist secondary population and favourable results were obtained through an application of the algorithm to a real example.

A two-phase genetic algorithm (TGA) was used to examine university timetabling (Ueda, Ouchi, Takahashi, and Miyahara, 2000). Two kinds of populations, class scheduling and room allocation were looked at, developed independently, and a cost value for each person assigned. Instances where the lowest costs occur were selected from each population, and these subjects were combined to calculate the fitness values. To assess its performance, TGA was applied to several problems and outcomes compared with those obtained by the simple GA (SGA). In the case that was used it was found that TGA obtained a better solution than the simple GA when the room utilization ratio was high. Therefore in some instances TGA can find a solution that satisfies all constraints where SGA cannot find a feasible solution.

Lewis Rhydian and Paechter Ben (2004) proposed various Crossover operators including day-based crossover and new repair strategy have been discussed. Based on this various new crossover operators such as selective one-point, selective two-point and selective uniform are proposed in this dissertation. The new repair strategy is also proposed.


2.3.5 Constraint Based Approaches

Constraint satisfaction problems or CSPs are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which are solved by constraint satisfaction methods. CSPs are the subject of intense research in both artificial intelligence and operations research, since the regularity in their formulation provides a common basis to analyze and solve problems of many unrelated families. A solution to a timetabling problem using CSP is described as follows: (Deris et al., 1997).

The problem depicts each course as offering several subjects per semester and each subject having a specified number of lessons per week. Each lesson can thus be defined as the contact hours between lecturers and students at a specific time and place (room). Each lesson lasts for a specific period of time.
Thus the timetabling problem can be defined as an assignment of time $t_i$, $1 < j < m$ and rooms $r_k$, $1 < k < p$ to lessons $s(i)$, $1 < i < n$ taught by a lecturer $L(S(i))$ such that all constraints $C(S(i))$ are satisfied. $L(S(i))$ and $C(S(i))$ are lecturers and constraints of a lesson respectively, while $m$, $p$ and $n$ are infinity variables. Thus the general CSP for a timetabling problem is as follows:

- A finite set of variables $X_1 \ldots X_n$.
- For each variable $X_i$ a set of domains $D_1 \ldots D_n$ containing possible values of $X_i$.
- A finite set of constraints, $C_1 \ldots C_q$ representing relations between variables.

A solution to the CSP involves assigning values from domains of all variables such that all constraints are satisfied.

In terms of the CSP, a timetabling problem can be formulated by representing a timeslot and a room of a lesson as variables of the CSP, available timeslots and rooms as values of the CSP, whereas constraints are the various relationships between lessons. Therefore the CSP model for timetabling problems can be formulated by deciding the variables, values and constraints.

CSP is a decision making tool that satisfies all constraints and its major advantages are as follows:

1. Constraint propagation reduces the search space so it takes less time to search thus minimizing backtracking;
2. Memory requirement is smaller since the search space has been reduced;
3. All available resources are represented in the form of constraints and hence user preferences and requirements can be easily satisfied.

The process of Constraint Satisfaction involves finding values for problem variables subject to constraints on acceptable combinations of values. The satisfaction of all constraints can sometimes lead to conflicts. In these cases
constraints have to be prioritized so that the more important constraints are satisfied first. Furthermore, the satisfaction of some constraints could also give rise to conflicts with the stated objective. Therefore, in some cases it is impossible to solve the problem completely and thus a subset of the problem is solved which is called partial constraint satisfaction (Freuder and Wallace, 1992).

A solution to CSP involves assigning values from domains of all variables such that all constraints are satisfied. Constraint Logic Programming is the integration of two methods: logic programming and constraint solving. Logic programming has the capability of supporting declarative programming where formulation is in terms of true or false. Constraint solving reduces the search space by pruning all impossible values through constraint propagation. Constraint based reasoning is a reasoning process that uses an arc consistency technique to propagate constraints. The arc consistency algorithm is derived from the representation of constraints in a graph that can be associated with a constraint network where nodes correspond to variables in the constraint network and edges link nodes \( i \) and \( j \) if there is a relation \( R_{ij} \) between nodes. During assignment an algorithm (arc consistency algorithm) is used to check the consistency of labels for each couple of nodes that are related by a binary constraint and labels that cannot satisfy constraints are removed (Deris, Omatu, Ohta and Samat, 1997).

Tripathy (1992) noted that a special characteristic of the timetable problem is the highly dense conflict matrix. Consequently it was recommended that computer-aided decision making be used to arrive at a desirable timetable. The related problems were discussed in terms of lack of an objective function and quality. The main conclusion of this paper was that it is desirable to actively involve the decision maker in the timetable generation process.

Cooper and Kingston (1993) described a timetable specification language which allowed the program to handle the many idiosyncratic constraints in a uniform manner. It was claimed that the new algorithms introduced could tackle
the problem more intelligently than any of the traditional search methods. The conclusions drawn indicated that an assignment algorithm which is unaware of a tightly constrained sub-problem lying in its path will fail to solve that sub-problem. On the other hand an assignment algorithm, which is designed to solve that sub-problem, in particular will actually benefit from the tight constraints imposed upon it.

By applying Object Oriented Constraint Logic Programming Deris, Omatu and Ohta (1995) described the solution for the timetable problems of privately owned colleges. The results showed that the highly constrained problems could be solved using minimal computing resources compared to substantial man-months. It was proposed that this method (Object Oriented Constraint Logic Programming (OCLP)) offered many advantages, as the model presented could be applied to similar types of institutions with a minimum number of changes.

Frangouli, Harmandas and Stamatopoulos (1995) used a method incorporating Constraint Logic Programming to construct optimum timetables for the University of Athens Department of Informatics courses. The software platform used for the implementation was an instance of the Constraint Logic Programming class of languages, the ECLIPS system, and proved that the system was an appropriate vehicle for managing the complexity of the timetable problem.

Babes and Quilliot (1995) developed an interactive approach, which was expressed to resolve and formulate logical, resource type and temporal constraints. To achieve the above some parts of a Prolog program were used and a determination of flows, along with the constraint propagation and least engagement techniques.

Gueret, Jussien, Boizumault and Prins (1996) claimed that CSP formalism improved the flexibility of the Constraint Logic Programming (CLP) paradigm. It was claimed that this system provided the ability to rapidly design...
efficient search procedures that provide feasible solution. These implementations were developed using CHIP.

Also using Constraint Logic Programming Boizumault, Delon and Peridy (1996) developed an application where the CLP was used over finite domains. A brief overview of OR approaches for solving examination timetable problems was provided. Also discussed was the examination timetable problem for universities showing how Constraint Logic Programming over finite domains can be used to solve the problem efficiently. It was concluded that the experiences of the authors illustrated the important potentialities of Constraint Logic Programming for the prototyping and implementation of real life applications.

Monfroglio (1988) described an approach in which he reduced the large set of constraints into their component rule base. This rule base was then reordered to minimize backtracking and to afford the maximum amount of the parallelism. This highly computational approach was illustrated by including implementations of it in both Prolog and Parlog systems along with an assessment of its performances. It was claimed that the same system, with little changes, could also solve analogous scheduling, resource allocation and planning problems. Again, for an algorithm that promised so much, it disappeared without trace.

Deris, Omatu and Ohta (2000) viewed the timetabling problem as being combinatorial and dynamic, and thus proposed that any approach used must be efficient, flexible, portable and adaptable. The solution procedure used was based on a constraint based reasoning technique and implemented in an object-oriented approach. It was argued that since the object-oriented approach was implemented the system proposed could be easily modified and adapted to support changes. It was suggested that the most feasible and best timetable for the college was found using the proposed algorithm.
A different approach by Brailsford, Potts and Smith (1999) asserts that many scheduling and timetabling problems can be formulated as Constraint Satisfaction Problems (CSP's). They defined and examined basic techniques for solving CSP's, then compared them with other approaches such as integer programming, branch and bound and simulated annealing. As opposed to more developed OR methods the authors viewed CP as being a relatively new area, however they still see that there are problems for which it is competitive. With further improvements to CP methodology being anticipated, it is considered that awareness of CP as a technique for tackling combinatorial optimization problems is important for researchers in this area.

2.3.6 Ant System

A new approach for solving distributed problem and optimization was proposed by Colomn, Dorigo and Maniezzo (1992) called the Ant System. The system was inspired by the fact that ants as a colony manage to find the shortest path to reach their food. Each time an ant moves one place to another place it leaves some substances that called pheromone on its trail. The more ants passing the path the more pheromone left behind.

If there are some paths between nest and a place of food and there is a colony of ants back and forth from the nest to the place of food, in the beginning the member of ants passing the path will distribute more of less evenly. It can be understood that in a certain period of time ants passing the shortest path will be more frequent in comparison with other paths. It means that the colony will put more pheromone in the shortest path.

Survey by Dorigo, Di Caro and Gambardella (1999) reported that the Ant System Algorithm has been attempted to solve most of the popular problem, such as Traveling Salesman Problem, Quadratic Assignment Problem, job-shop Scheduling, Problem, Vehicle Routing Problem, Sequential Ordering Problem, Graph Coloring Problem, Connection-oriented Network
Routing Problem, and Connection-less Network Routing Problem with very promising results.

2.3.7 Hybrid Algorithm

Difficulties of scheduling or NP problems in general and the availability of some algorithms direct researchers to combine some methods or algorithms as a hybrid algorithm to enhance their algorithm. (Alvarez-Valdes, Martin and Tamarit, 1996) used hybrid algorithm that construct Spanish school timetabling using hybrid of parallel heuristic algorithm and Tabu search.

Burke, Elliman and Weare (1994) discussed an automatic timetable generation method that used applications such as Graph Colouring and Genetic Algorithms. It was claimed that their method was being implemented at some United Kingdom Universities.

Making comparisons between Genetic Algorithms (GA), Simple Heuristics (SH), and Simulated Annealing (SA) on a collection of real timetable problems, Ross and Corne (1995) concluded that SH and SA are generally the best strategy as far as solution quality is concerned. However for a fairly small range of particular problems the GA either equals or betters the performance of SA and SH, only with the added value of a large number of usefully distinct, equally good solutions. This work has value in being one of the very few to perform a direct comparison between different approaches on the same specific problem.

Burke, Newall and Weare (1996) claimed that Hybrid Evolutionary Algorithms yielded better results than Evolutionary Algorithms even though evolutionary techniques with general purpose have optimization capabilities. A Mimetic algorithm was used which incorporated local search methods into the Hybrid approach. It was noted in conclusion that this Hybrid Evolutionary method gave better results.
A dual approach using a combination of Evolutionary Algorithms (EA) and greedy algorithms was proposed by Corne and Ross (1996), in which they retained the speed of the simpler method while adopting the greedy method to bootstrap the process.

Drexl and Salewski (1997) showed that several types of constraints may be modeled using the unifying framework of partially renewable resources. Presented are two-phase parallel greedy randomized and genetic methods. An instance generator for the generation of a representative set of instances was also given. Computational results showed that this method solved the instances investigated close to optimality.

Stallaert (1997) implemented a course timetable system for the Anderson School of Management at UCLA. She divided the overall problem into two sub-problems: 1) to schedule the core courses using an integer-programming algorithm; and 2) to generate a timetable by using a heuristic algorithm to solve a variant of a quadratic assignment problem. She used DSS for generating solutions that she claimed would work faster.

It is suggested that the use of genetic algorithms for solving timetable planning problems is problematic due to the ambiguity in deciding the fitness function (Deris, Omatu, Ohta and Saada, 1999). Their view is that most solutions to constraint satisfaction problems using genetic algorithms are problem-dependent and therefore difficult to apply to real-world problems. (Deris et al., 1999) suggested using a hybrid algorithm consisting of a genetic algorithm and constraint-based reasoning. They tested the proposed algorithm using real data for university timetable planning. Unlike other techniques, this approach did not require modifications to the components of the genetic algorithm procedures and is therefore generic and problem independent. Consequently, the algorithm can be adapted to different kinds of problems with minimum changes and applied to constraint satisfaction problems.

2.4 Conclusion

In this chapter various Solution generation algorithms that have been used for course timetabling problem are described and a detailed examination of the academic literature is provided within the context of this fundamental solution generation algorithms. Based on this study various new crossover operators are proposed in the next chapter. A new repair strategy is also proposed in the next chapter.