ASHESI

ASHESI UNIVERSITY COLLEGE

AUTOMATED COURSE SCHEDULER FOR ASHESI UNIVERSITY COLLEGE

JENNIFER FENTENG VALENTINA AGBENOWOSI

APRIL 2014

APPLIED PROJECT

ASHESI UNIVERSITY COLLEGE

AUTOMATED COURSE SCHEDULER FOR ASHESI UNIVERSITY COLLEGE

By

JENNIFER FENTENG VALENTINA AGBENOWOSI

Dissertation submitted to the Department of Computer Science,

Ashesi University College

In partial fulfillment of Science degree in Management Information Systems

April 2014

Declaration

I hereby declare that this dissertation is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: …………………………………………………………………………..

Candidate's Name : …………………………………………………………………………..

Date : …………………………………………………………………………..

I hereby declare that the preparation and presentation of the dissertation were supervised in accordance with the guidelines on supervision of dissertation laid down by Ashesi University College.

Supervisor's Signature:...……………………………………………………………………..

Supervisor's Name : ...………………………………………………………………………..

Date : ...………………………………………………………………………..

Acknowledgement

This dissertation would not have been possible without the guidance and help of certain individuals who in one way or the other contributed and extended their valuable assistance in the preparation and completion of this project.

My utmost gratitude goes to my supervisor, Dr. Nathan Amanquah, whose insightful direction and assistance made this project a rewarding journey. I thank him for making me believe I can do it no matter how challenging the project got along the way.

I am grateful to Dr. Ayorkor Korsah and Mr. Aelaf Dafla, lecturers at Ashesi University College Computer Science Department, who provided directions on my choice of a project which led me to this very project.Many thanks also to Mr. Kwadwo Gyamfi Osafo-Marfo whose insightful knowledge into scheduling systems and databases made me understand the various options I needed to consider in dealing with this project.

Also to Mrs. Araba Mbirba Amuasi Hammond, I am very grateful. I could not have gotten this far without your support.

I am most especially thankful to God who gave me the grace and strength and assured me of better outcomes when I encountered challenges. To my entire family whom I cannot do without, and friends, I am thankful to you all for the prayers and support and for always reminding me that you are there for me, no matter what. I love you all.

Abstract

Ashesi University College is faced with the challenge of effectively scheduling courses at the beginning of the semester so that there are no class clashes for both lecturers and students. In an attempt to solve the Course Timetabling Problem at Ashesi University College, five algorithms: Genetic Algorithm, Constraint Programing, Particle Swarm Optimization, Simulated Annealing and Tabu Search algorithm, which are known for their use in solving University Course Timetabling problems have been studied and based on their ease of implementation, their robustness in arriving at feasible solutions, their computational speed and whether an optimal solution is always guaranteed, Particle Swarm Optimization algorithm is chosen to implement a solution to the Ashesi University Course Timetabling problem.

This project is focused on eliminating course conflicts and creating an optimal table based on teachers' preferences for certain timeslots to teach during the week. The paper outlines the assumptions and steps including explanations on Particle Swarm Optimization used in constructing the timetable base on teachers' preferences. Test conducted on the project proved that the use of Particle Swarm Optimization to solve the Ashesi Course Timetabling problems is in the right direction.Finally, the paper proposes a focus on other areas of the course timetabling problem at Ashesi University College, using the same Particle swarm optimization procedures described in the paper to help provide a complete solution to the timetabling problem of the school.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACTP	Ashesi Course Timetabling Problem
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
CP	Constraint Programming
SA	Simulated Annealing
TB	Tabu Search
LS	Linear Search

CHAPTER 1: INTRODUTION

1.1 BACKGROUND INFORMATION

Course scheduling is one of the most important aspects of a university’s administration process yet it is not an easy thing to do. Various universities are faced with the challenge of starting smoothly at the beginning of every semester because there always seem to be conflicts in classroom assignment to various courses. There is also the issue of some students being scheduled to take two or more courses at a particular time, which is impossible to do and is termed as class conflicts.For most universities, course schedules which are prepared manually are ineffective and inaccurate since it is not able to effectively schedule courses. These manual schedules oftenresult in clashes that occur with some courses taking place either in the same classrooms or at the same times with the same students registered for these different courses.In these days of advanced use of computers and other forms of technology, it has become increasingly important for institutions to make use technologies in solving course scheduling problems which they always face in their school’s administration. In view of this, there has been the need for the use of automated scheduling systems to help academicregistrars of universities to produce conflict-free academic timetables for their institutionsfor smooth administration of a school’s semester.

1.2 PROJECT OVERVIEW

Ashesi University College is a Liberal Artco-education in Ghana, situated precisely at Berekuso in the Eastern Region. It has a total enrollment of about five-hundred and seventy students majoring in Bsc. Business Administration, Bsc. Management Information Systems and Bsc. Computer Science.With a population of about six hundred students and growing, it has not been easy on the part of registrars to create course timetables that are devoid of conflicts or clashes for both lecturers and students.Currently, with the increase in student population, Ashesi University College is faced with the challenge of starting smoothly at the beginning of every semester due to conflicts in class and room schedules for students especially. This is envisaged to worsen as the school expanses in terms of the level of enrollment and increase in the number of departments in the school. Right now the course schedules are prepared manually, and it sometimes results inclashes which are mostly as a result of the change in class sizes every year and the number of students retaking courses, who are mostly not accounted for. Another reason can be attributed to the lack of faculty availability within a particular semester. These available times are not consistent throughout the academic year hence new timetables based on lecturers' new available times have to be generated and this results in conflicts easily as manual timetabling can be difficult to deal with. As the number of students, faculty and courses increases the manual method will no longer be effective since there will be so much to manage. Hence, registrars of the school are seeking other means of making the timetabling system betterfor the entire student body and for smooth administration of the academic semester.

This project considers possible ways of dealing with this problem through the study of five algorithms that can be used to solve the timetabling problem at Ashesi University College. The project then focuses on the use of the best algorithm selected based on its characteristics as compared to other algorithms and target solving one aspect of the course scheduling problem, lecturer’s preferences, for Ashesi University College. The project is developed in java (Netbeans) and augmented with the use of text files and MySql database to read in data.

1.3 OBJECTIVE

It is difficult to find an effective generic solution to automatic timetabling problems due to the diversity of course scheduling problems and the variance of constraints and particular requirements for various institutions [1]. For this reason, it has become necessary for an institution to create its own automated scheduling system that suites its needs. It is for this reason that the author attempts to develop an automated scheduling system that will help the registrars of Ashesi University College to come up with a conflict-free course timetable for the school each semester.It would be desirable to build an automatic scheduler to allocate lecture time slots to courses,taking into consideration courses available, lecturers' available time (preferences), lecture times and course sections while avoiding conflict; just to mention a few of the constraints necessary forgenerating an academicschedule.

1.3 IMPORTANCE OF THE SYSTEM AND MOTIVATION

Amidst the many things we can manually manage, there still remain certain processes which can be effectively managed through automation. Course timetabling is one of those processes that need automation in order to be accurately produced when there are many constraints to deal with. Creating an automated academic scheduler for Ashesi is generally very important for the smooth administration. It will help to lessen the frustrations students, faculty, staff and administrators of academic institutions face with timetable clashes by generating a conflict-free timetables. Another important reason for an automated timetabling system for Ashesi is to allow and enable students to take the right courses they wish to take, for their majors, in a particular semester. This prevents students from having to drop certain courses they had wished to take, due to class conflicts. Also, in the midst of too many itemsthat may exceed cognitive capacities of humans to manage, especially fordecision-making under stress, it is important that we make use of technology to aid in the management processes.With the registrars of Ashesi University College finding it difficult to create a conflict-free timetable thattakes into consideration students retaking certain courses, an automated scheduler is an option for a solution to the school's course scheduling problem. An automated scheduler will also help to generate a timetable that takes into consideration the year groups of studentstaking different courses in order to create a timetable devoid of conflicts for the school.

This project has been inspired by the author's quest to help solve the challenges she and some of her mates faced in making sure that they register for their desired courses during their four years education at Ashesi University College. Some of her mates have had to drop courses they had wished to take, in order to graduate on time, simply because these courses were scheduled at periods the students had other classes to attend. An automated course scheduler will be able to deal with these challenges.

1.5 CHALLENGES ASSOCIATED WITH THE SYSTEM

The problem with resource-constrained timetabling is that it is challenging and part of the family of NP-hard problems that no best solution to them is known[1][2]Another challenging aspect of it has to do with the size and the complexity of constraints involve in creating solutions that will satisfy the demands of students and instructors [3]. The problem is even made harder by the need to develop a system that is easy for everyone involved in the process to use and understand, and for them to be satisfied with the results. Due to the differences in constraints and requirement necessary for creating a conflict-free timetable in various institutions

[4],course scheduling has become unique to each institution and as result requires the time and knowledge to be able to build one.Another challenge has to do with the non-violation of constraints of faculty, courses, classrooms, timeslots, or students. Research into course scheduling problems [3] , [2] has shown that it is highly unlikely to have an algorithm that guarantees satisfaction to all exposed constraints of various types of problems because it is not easy to express and formulate precisely the requirements and constraints for real-life timetabling tasks.

1.6 DELIVERABLES

With the adoption of an automated scheduling system designed for Ashesi University College, it should be possible for academic registrars of the school to createconflict-free course timetables for both lectures and students. Based on the systems input parameters,courses should be assigned to appropriate into a particular timeslot based on lecturer's available times and the system should be able to manage the number of classes that can take place at a particular timeslot. This is to ensure that the number of classes that are scheduled for a period are not more than the number of rooms available.

1.7 OUTLINE OF REPORT

This chapter gives a general overview of course timetabling problems with specific reference to the case of Ashesi University College, the reasons for an automated academic scheduler for the school, different sources of information that were gathered in order to fully understand the problem and the deliverables expected at the end of the projects' implementation. The next chapter (chapter 2) presents the literature review on five of the most popular algorithms that have been used for course scheduling problems and forms the bases of decision to use a particular algorithm for the implementation of the Ashesi University Course Timetabling problem. In chapter three (3), the design of the automatic scheduler for Ashesi University College is stated and explained based on the algorithm being used and the problem being solved. Chapter four (4) follows with methodology used for the implementation and an analysis of the project's implementation and a test of the re. Finally, the report ends with conclusion and recommendations in Chapter five (5).

CHAPTER 2: LITERATURE ANALYSIS

Various academic institutions such as Arkansas Tech University [5], Ibb University- Yemen[6] and Universitas Pelita Harapan[7]have all tried using various techniques and mechanisms to solve course timetabling problems.Some of these institutions have used Genetic Algorithm[8], Heuristic Search[7], Constraint Programming[9], Tabu search[10], Particle Swarm Optimization[11], graph coloring algorithms[12] and simulated annealing[13]. Others have also used a combination of these methods to solve the problem of course scheduling[6]. Most of these institutions, in attempts to solvecourse timetabling problems through the use of automated scheduling systems,classified the available constraints into soft and hard constraints involved in determining parametersthat are essential to solving the problem. In addition, rules regarding the system's input parameters were also made to ease the mode of implementation of the system and how the system can be effectively used [14].For example, each lecture unit has identical unit (one hour thirty minutes, in the case of Ashesi University College, is assigned to all slots so that each lecturer has the same length of time to teach) and designated times may not be assigned lecturers when they are not available.

In order to find out the best algorithm to use when implementing a solution to Ashesi University College course timetabling issues, there is the need to study and compare the various algorithms that exist and have been used to solve timetabling or related scheduling problems. This will help to highlight the advantages and disadvantages of each algorithm and why one should be selected over the other for the implementation of an automatic course scheduler for Ashesi University College. Below is a review on five of these algorithms that have been mostly used in solving course timetabling problems and what they each actually entail.

2.1 ALGORITHMS

2.1.1 GENETIC ALGORITHM

Studies on the works of Safaai et el [6] has shown that Genetic algorithm, whichwas introduced by John Holland in the seventies [8], has been used by several researchers to solve scheduling problems. Genetic Algorithm is a problem solving strategy based on the Darwinian Evolution theory of survival of the fittest[15][16]. It is a search technique, which is based on the mechanics of natural genetics, where biological processes aresimulated to allow the consecutive generations in a population to adapt to theirenvironment[16]. Hence, starting with a population of randomly created solutions, better ones are more likely to be chosen for recombination into new solutions, i.e. the fitter a solution, the more likely it is to pass on its information to future generations of solutions and this works through the mechanism of selection and reproduction popularly known as crossover and mutation.

Despite the wide use of Genetic Algorithm by many researchers to solve timetabling problems[1] because a population of potential solution (schedules) is always available[15] (solutions being generated at any point in time because of mutation and crossover during the genetic computation), it is faced with some limitations. One of such limitations is the fact that Genetic Algorithm has longer computational time, emanating from its inability to memorize a potential solution.Also, in Genetic Algorithm, the optimal solution is selected from the last generation; this selection approach may be missing the optimal solution of iterative process [17]. This is so because during the crossover and mutation process, solutions that are deemed weak are rejected from taking part in the crossover and mutation process. However, these rejected or weak solutions may tend out to be more fit in the end as compared to an optimal solution derived from the mutation process. Genetic Algorithms also have to deal with limited population sizes and a limited number of generations. This limitation can lead to premature convergence, which means that the algorithm gets stuck at local optima [18][17]. In order to solve the limitations of Genetic Algorithm, most researchers have enhanced the use of Genetic Algorithmsby combining it with other search algorithms such as heuristic search and simulated annealing to solve time tabling problems. This combination has become necessary because several researchers have concluded that conventional Genetic Algorithms do not give good results among a number of approaches developed for the University Course Timetabling Problems [1].

2.1.2 CONSTRAINT PROGRAMMING

Another, algorithm that is mostly studied and used in solving course timetabling problems is Constraint Programming. It is a relatively new technology developed in the computer science and artificial intelligence communities and has found an important role in scheduling, logistics and supply chain management [19]. The idea of constraint programming is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding a solution that satisfies all the constraints [9]. The earliest ideas leading to Constraint Programming may be found in Artificial Intelligence(AI) dating back to the sixties and seventies [9]. Constraint Programming (CP) has attracted high attention among experts from many areas of study because of its potential for solving hard real-life problems [9]. It has had some early successes in solving problems in Circuit design (Siemens), Real-time control and Container port scheduling at Hong Kong and Singapore [19]. It has been used in scheduling applications such as job shop scheduling, assembly line smoothing and balancing, cellular frequency assignment, nurse scheduling, shift planning, maintenance planning, airline crew roster and scheduling and airport gate allocation and stand planning [19]. Constraint Programming has an inner interdisciplinary nature since it combines and exploits ideas from a number of fields including Artificial Intelligence, Combinatorial Algorithms, Computational Logic, Discrete Mathematics, Neural Networks, Operations Research, Programming Languages and Symbolic Computation [9].

There are currently two branches of constraint programming, namely constraint satisfaction and constraint solving [9]. *Constraint satisfaction* deals with problems defined over finite domains while *Constraint solving* involves describing the problem as a set of constraints and solving these constraints [9].

Constraint Programming has the advantage of taking in more constraints as compared to Genetic Algorithm; as this makes the problem easier to deal with[19]. It is a better option when constraints have few variables[19]. However, the problems associated with constraint programming are thatit is well suited for logic processing and constraint based processing but a weak algorithm for continuous variables due to lack of numerical techniques[19].Constraint Programming algorithm may fail when constraints contain many variables that do not allow constraints to propagate well[19], that is, these may constraints do not allow the problem to be reduce and easily solved by identifying the solvable parts. With Constraint programming, there is the problem of choosing the right constraint satisfaction technique for the particular problem [9].For example, a simple search like chronological backtracking may result in a more efficient solution or timetable than a more expensive constraint propagation technique. Also, the efficiency of constraint programs is still unpredictable and the use of intuition is usually the most important part of decision; when and how to use constraints [9]. This is due to the instability of the model as small changes in data can lead to dramatic changes in performance[9]. Also, the process of performance debugging for a stable execution over a variety of input data, is currently not well understood [9]. Another particular problem in many constraint models is the cost optimization. This is because it is sometimes very difficult to improve an initial solution, and a small improvement takes much more time than finding the initial solution. Hence there is a tradeoff between an “anytime” (but not optimal) solution and “best” ‘(optimal) solution [9].

2.1.3 PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a population based intellect algorithm proposed by Kennedy and Eberhart in 1995, motivated by the flocking behavior of birds [20]. In PSO, there are no DNA inspired operators on the swarm like it is in Genetic Algorithm. Instead, each particle is flying over the search space in order to find promising results and adjusts its flying position according to its’ own previous experience and its’ neighbor experience [21]. In PSO, a bird of a flock is represented as a particle, and the swarm is composed of a group of particles. The position of each particle can be regarded as the Candidate Solution to an optimization problem. Every particle is given a Fitness Function designed in correspondence with the corresponding problem. When each particle moves to a new position in the search space, it remembers its personal best (*P*best), which is the best position the particle in the search area so far. In addition to remembering its own information, each particle will also exchange information with the other particles and remember the global best (*G*best), which is the best position that any of the particles in the population has achieved so far. Then, each particle will revise its velocity and direction in accordance with its *P*best and the *G*best to move toward the optimal value and find the optimal solution [22]. Hence, PSO is an evolutionary technique but it differs significantly from genetic algorithms (GAs) [21].

Among the many algorithms developed for solving scheduling problems, Particle Swarm Optimization (PSO) has been proven to be capable of achieving remarkable performance[22]. The advantages of less parameter settings required and fast convergencemake it a popular algorithm applied to a variety of optimization problems[23]. PSO is also a simple and easy algorithm with decent performance through its robustness in controlling parameters and its high computational efficiency [24]. PSO has a flexible and well-balanced method to improve and adjust to the global and local exploration and exploitation abilities within a short computation time. This is done by the particles ability to update its position and velocity based on the global best position as well as the particles own position. The position and velocity updates areillustrated by the PSO algorithm equation, where equation one (1) is velocity update and equation two (2) is the position update function. After an iteration of the swarm, the velocity update function (equation 1) calculates and updates the velocity of each particle. These updated velocities are then use by the position update function (equation 2) to update all particles' solution.

$$v_{id}^{t+1} = w.\ v_{id}^{t} + c_1.r_1.(p_{id}^{t} + x_{id}^{t}) + c_2.r_2.\left(p_{gd}^{t} + x_{id}^{t}\right) \text{ ...... } (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \text{ .... } (2)$$

v_{id}^{t}: Component in dimension d of i^{th} particle velocity in iteration t.

x_{id}^{t}: Component in dimension d of i^{th} particle position in iteration t.

c_1: Constant weight factor for the individual particle

c_2: Constant weight factor for the global particles.

p_{id}^{t}: Best position achieved so long by particle i.

p_{gd}^{t}: Best position found by the neighbors of particle i.

r_1, r_2: Random factors in the [0,1] interval

w : Inertia weight

From the equations, PSO always has memory, and the knowledge of good solutions which is shared by all particles in the swarm at any point in time[20]. However, despite these numerous advantages of the PSO algorithm, it does not always work well and may need tuning of its behavioral parameters so as to perform well on the problem at hand [25]. Also, any small changes to the PSO implementation can cause dramatic changes in the behavioral parameters that cause good optimization performance[25]. Also despite its robustness and global exploration capability, PSO has the tendency of being trapped in local minima, for a basic PSOdepending on the coefficients used and may also exhibit signs of slow convergence [26].

2.1.4 SIMULATED ANNEALING

Simulated Annealing (SA), proposed by Kirkpatrick et al, is a randomized search method for optimization[27]. It tries to improve a solution by walking randomly in the space of possible solutions and gradually adjusting a parameter called "temperature". At high temperature, the random walk is almost unbiased and it converges to essentially the uniform distribution over the whole space of solutions; as the temperature drops, each step of the random walk is more likely to move towards solutions with a better objective value, and the distribution is more and more biased towards the optimal solutions. SA is a heuristic method that has been implemented to obtain good solutions of an objective function defined on a number of discrete optimization problems[28]. This method has proved to be a flexible local search method and can be successfully applied to the majority of real-life problems[28].

The origin of the algorithm is in statistical mechanics that imitates the annealing process used in metallurgic [29]. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution in order to escape from local minimum. Hence, the design of a good annealing algorithm is vital since it generally comprises three components: (1) neighborhood structure, which defines for each solution a set of neighboring solutions (2) cost function, which is controlled by the temperature and (3) cooling schedule [30]. Hence Simulated Annealing algorithm is basically a three steps process: perturb the solution, evaluate the quality of the solution, and accept the solution if it is better than the new one. SA is an iterative method which accepts a new solution if its cost is lower than the cost of the current solution in each iteration. However, if the cost of the new solution is greater, there is a probability of this solution being accepted. With this acceptance criterion, there is the possibility of the algorithm climbing out of local minima[31].

One advantage of using Simulated Annealing is that Simulated Annealing does not require any mathematical model thus; it can be used to solve a problem if the solution to the problem can be designed so that it can be perturbed and evaluated[29]. Another advantage of using Simulated Annealing is that it provides a means to escape local optima by allowing hill-climbing moves[32], which is made possible by the temperature at which the algorithm operates and the cost associated with new solutions.

However, there are some challenges that come with the use of Simulated Annealing for scheduling. One of the potential drawbacks of using simulated annealing for hard optimization problems is that finding a good solution can often take an unacceptably long time[29] [33]. Hence, it is extremely slow and not suitable for complex optimization problems such as scheduling [34]. Also, Simulated Annealing is an inefficient algorithm at low temperatures when it comes to the acceptance of optimal solution since a lot of computation is required to compute the change in cost yet the solution arrived at can be rejected [33]. Another drawback has to do with the possibility of not finding an optimal solution after the process has been completed [29].Finally, the intensive computational requirements and the practical difficulties involved in the proper choice of Simulated Annealing parameters are factors that reduce its potential in many cases [35].

2.1.5TABU SEARCH

Tabu Search (TS) is a heuristic method originally proposed by Fred Glover in 1986 [36], to various combinatorial problems. Tabu Search is an extension of classical Local Search(LS) methods. Basic TS can be seen as simply the combination of LS with short-term memories. Hence, TS pursues LS whenever it encounters a local optimumby allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, that record the recent history of the search[36].This is achieved by making certain actions “taboo”, meaning not allowing the search to return to a recently visited point in the search space or not allowing a recent move to be reversed [37]. This method minimizes the chance of cycling in the same solution, and therefore creates more chances of improvement by moving into un-explored areas of the search space[38].

The first two basic elements of any Tabu Search heuristic are the definition of its search space (the space of all possible solutions that can be considered or visited during the search) and its neighborhood structure (a set of neighboring solutions in the search space that are defined when local transformations are applied to the current solution at each iteration of the tabu search)[37]. An advantage of the use of Tabu Search is that it can be applied to both discrete and continuous solution spaces [39].Also, for larger and more difficult problems such as scheduling, quadratic assignment and vehicle routing, tabu search obtains solutions that rival and often surpass the best solutions previously found by other approaches[39]. Many computational experiments have also shown that Tabu Search has now become an established optimization technique which can compete with almost all known techniques because of its flexibility [40].

Tabu Search, like any other algorithm has its drawbacks as well. One of its drawbacks is seen in the fact that tabus (spaces not to be revisited during the search forbidden moves) are sometimes too powerful and may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the search process[37].Another drawback is that Tabu Search tends to be too "local", i.e. it tends to spend most, if not all, of its time in a restricted portion of the search space[37]. This may lead to failure to explore the most interesting parts of the search space and thus end up with solutions that are still far from the optimal ones.Furthermore, Tabu Search, just like Simulated Annealing and Genetic Algorithm, is problematic when dealing with optimization problems that contain constraints because it initiates search (generally) with a random solution and apply operators that may not be able to guarantee a feasible solution [41]. Tabu search in itself is also a complex technique that may not be properly formulated if not well understood and not given the right parameters[40].

2.1.6 COMPARISON OF THE FIVE ALGORITHMS

The table below (Table 1) summarizes the differences and similarities among the algorithms that have been discussed above based on their ease of implementation, their robustness in arriving at feasible solutions, their computational speed and whether an optimal solution is always guaranteed.

Algorithm	**Ease of implementation and flexibility**	**Computational Speed and Robustness**	**Convergence**	**Optimal Solution**
Genetic Algorithm	Fairly easy and flexible	Slow and frail	Stuck in local optima (premature)	Not guaranteed
Constraint Programming	Not very easy and not flexible.	Moderate and fairly robust	Global optima	No always optimal
Particle Swarm Optimization	Easy to implement and flexible.	Fast and very robust	Global optima but can be trapped in local optima	Mostly guaranteed
Simulated Annealing	Intensive computational requirements and but flexible	Can be very slow and fairly robust	Escape from local optima	Not always guaranteed

Tabu Search	Complex technique but Flexible	Moderate speed and fairly robust	Can be stuck in local optima	Not always optimal

Table 1 A Comparison of the Five Algorithms under study

From the literature review, it is clear that most of the algorithms used in scheduling may generate feasible but not optimal solutions [6]. Most algorithms proposed for deriving solutions to timetabling problems do not adequately provide solutions on their own due to the presence of limitations such as being trapped in local minima, complexity of implementation, and some specific conditions under which some algorithms operate – level of heat for Simulated Annealing. Other research [21] into finding better solutions for solving timetabling problems have proposed the use of hybrid forms of these algorithms so that they can complement each other in obtaining optimalsolutions. An example is using the Particle Swarm Optimization algorithm and Constraint Based Reasoning used by Ibb University (Yemen) to solve their timetabling problem [21].

2.2 RELEVANCE OF THE RELATED WORKAND APPROACH

Most of the courses scheduling problems mentioned in the literature have similar characteristics as the Ashesi course scheduling problem. However, most of the example problems discussed were solved using a combination of algorithms. This proves the fact that all the algorithms propose to solve timetabling problems have do come with their strength and weaknesses.Most of the solutions to solving the course timetabling problems were also institution specific but gave a general idea about course timetabling problems.

Academic course timetabling problems are unique to every institution [8]. Hence, one solution cannot be applied everywhere but the concepts of development can be applied to course timetabling at different institutions. Based on the reviews and discussion on the various algorithms, their ease of implementation, their robustness in arriving at feasible solutions, their computational speed and the optimality of solutions arrived at after implementation, stated in this paper, the use of Particle Swarm Optimization in solving the Course Timetabling problem at Ashesi University College is proposed. Particle Swarm Optimization has been studied to be an effective algorithm that could be useful for the implementation of an automated scheduler for Ashesi University College.

CHAPTER 3: METHODOLOGY

3.1 PSO ALGORITHM FOR ASHESI COURSE TIMETABLING PROBLEM

PSO is known to have the ability to find a near-optimal solution by updating its flying position and velocity vector with a suitable fitness function [42]. In optimization problems an objective function needs to be defined, and the maximum or minimum value of the function or process are sought after in order to justify conclusions on the system. In the case of the Ashesi course timetabling problem, the objective is want to maximize an objective function which depends on the fitness of the swarm based on the sum of teacher, classroom and student preferences for a particular timeslot.

Hence the objective is to maximize a fitness function:

$$f(x) = \sum_{n=i}^{x}(\text{Lecturers}'preferences) + \sum_{n=i}^{x}(\text{Room Preferences})$$

Where: x = the number of iterations

n = the ith particle in the swarm

The PSO algorithm works by simultaneously maintaining several candidate solutions in the search space called particles. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be thought of as a particle "flying" through the fitness landscape finding the maximum of the objective function. Initially, the PSO algorithm chooses candidate solutions randomly within the search space. The PSO algorithm has no knowledge of the underlying objective function, and thus has no way of knowing if any of the candidate solutions are near to or far away from a local or global maximum [42]. Hence, the PSO algorithm simply uses the objective function to evaluate its candidate solutions, and operates upon the resultant fitness values.

Each particle maintains its position, composed of the candidate solution and its evaluated fitness, and its velocity. Additionally, it remembers the best fitness value it has achieved thus far during the operation of the algorithm, referred to as the individual best fitness[42]. Finally, the PSO algorithm maintains the best fitness value achieved among all particles in the swarm, called the global best fitness, and the candidate solution that achieved this fitness, called the global best position or global best candidate solution.

The PSO algorithm consists of just three steps, which are repeated until some stopping condition is met. The steps are;

1. Evaluate the fitness of each particle
2. Update individual and global best fitnesses and positions
3. Update velocity and position of each particle

Fitness evaluation is conducted by supplying the candidate solution to the objective function [42]. Individual and global best fitnesses and positions are updated by comparing the newly evaluated fitnesses against the previous individual and global best fitnesses, and replacing the best fitnesses and positions as necessary. The velocity and position update step is responsible for the optimization ability of the PSO algorithm. The velocity of each particle in the swarm is updated using the following equation:

$$v_{id}^{t+1} = w.\ v_{id}^{t} + c_1.r_1.(p_{id}^{t} + x_{id}^{t}) + c_2.r_2.\left(p_{gd}^{t} + x_{id}^{t}\right) \text{ .... } (1)$$

The index of the particle is represented by *i*. Thus, $v_{id}^{t}$is the velocity of particle *i* at time *t*in the *d* dimension and x_{id}^{t} is the position of particle *i* at time *t in the d* dimension. The parameters *w,* c_1*,* and c_2 (0 ≤ w ≤ 1.2, 0 ≤ c1 ≤ 2, and 0 ≤ c2 ≤ 2) are user-supplied coefficients where c_1 and $c_2$are the acceleration constants while the *w* is the constriction factor or the inertia weight that is used to control the magnitude of the particle's velocity [43] These range of values have been shown by researchers to be of great positive effects on the performance of the PSO algorithm[44] [45] [46] [47]. The values $r_1$and r_2(0 ≤ r1 ≤ 1 and 0 ≤ r2 ≤ 1) are random values regenerated for each velocity update. The value p_{id}^{t} is the individual best candidate solution for particle *i* at time t, and $p_{gd}^{t}$is the swarm's global best candidate solution at time *t.*

- The first term $(w.v_{id}^{t})$ is the *inertia component*, responsible for keeping the particle moving in the same direction it was originally heading [42].
- The second term $c_1.r_1.(p_{id}^{t} + x_{id}^{t})$, called the *cognitive component*, acts as the particle's memory, causing it to tend to return to the regions of the search space in which it has experienced high individual fitness.
- The third term $c_2.r_2.\left(p_{gd}^{t} + x_{id}^{t}\right)$, called the *social component*, causes the particle to move to the best region the swarm has found so far.

Also, in order to keep the particles from moving too far beyond the search space, we use a technique called *velocity clamping* to limit the maximum velocity of each particle.

Once the velocity for each particle is calculated, each particle's position is updated by applying the new velocity to the particle's previous position:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2)}$$

- $v_{id}^{t+1}$is the newly calculated velocity
- $x_{id}^{t}$is the previous position
- $x_{id}^{t+1}$is the newly updated position

Figure 1below shows the entire process of the Particle Swarm Optimization for University Course timetabling

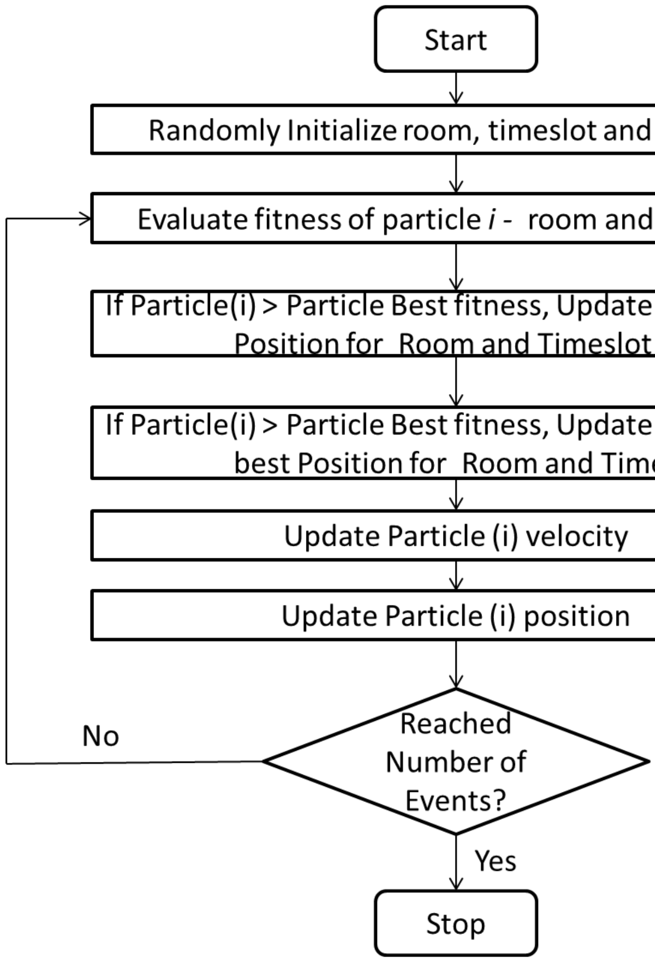
Start
Randomly Initialize room, timeslot and Velocity
Evaluate fitness of particle *i* - room and Timeslot
If Particle(i) > Particle Best fitness, Update Global best Position for Room and Timeslot
If Particle(i) > Particle Best fitness, Update Particle's best Position for Room and Timeslot
Update Particle (i) velocity
Update Particle (i) position
Reached Number of Events?
No
Yes
Stop

Figure 1 A Flow Chart of the PSO Algorithm

3.2 ASHESI UNIVERSITY TIMETABLING PROBLEM

This section describes the timetabling problem at Ashesi University College that Particle Swarm Optimization is used to solve. A semester of the school consists of at most 34 courses, first year to fourth year, at most 35 Lecturers and a total of at most 48 classes. In the Ashesi Course Timetabling Problem, the difference between courses and classes is that a course is the name of the subjects being taken but class refers to a lecturer and the group of students he/she is lecturing. So classes are made up of courses and/or sections of courses. There are a total of 25 timeslots, i.e. 5 periods 5 times a week. The schools' timetabling problem is defined in terms of the problem requirements, hard constraints (constraints that are critical to the systems implementation and will, when not adhered to, lead to solutions rather than the optimal solution) and soft constraints (constraint that can be broken without any significant penalty to the results generated).

The requirements of the problem specify:

- Lecturers' preferences for timeslots that corresponds to their available times and preferred teaching times.
- Room preference for classes due to the number of rooms available
- Different sections of a course as a new class even if the same lecturer teaches those sections

A feasible timetable is one which all events have been assigned a timeslot and a room, so that the following hard constraints are satisfied:

- No lecturer or class clashes. i.e. a lecturer cannot be assigned to more than one class at a time.
- No two classes are scheduled to take place in the same room at the same time.
- Room assigned to course are big enough to contain students registered for the class
- No student is assigned to two or more different classes at the same time.

The soft constraints of the problem are:

- The scheduled time of the class should fall within the preference sets as much as possible.
- The scheduled room of the class should fall within the preference sets as much as possible.
- Lecturer’s preference must be considered before the timetable generation

In building a conflict-free course timetable there is the need to consider teachers’ preference for timeslots, room preference a class and students’ preference as well. These are three important aspect of the course scheduling problem that need not be forgotten. This paper focuses on building a conflict-free timetable based on teachers’ preferences and room allocation penalties using the PSO algorithm described above.Figure 2shows the entire process of using the Particle Swarm Optimization algorithm to solve the timetabling problem at Ashesi University based on teachers' preferences and room allocation.

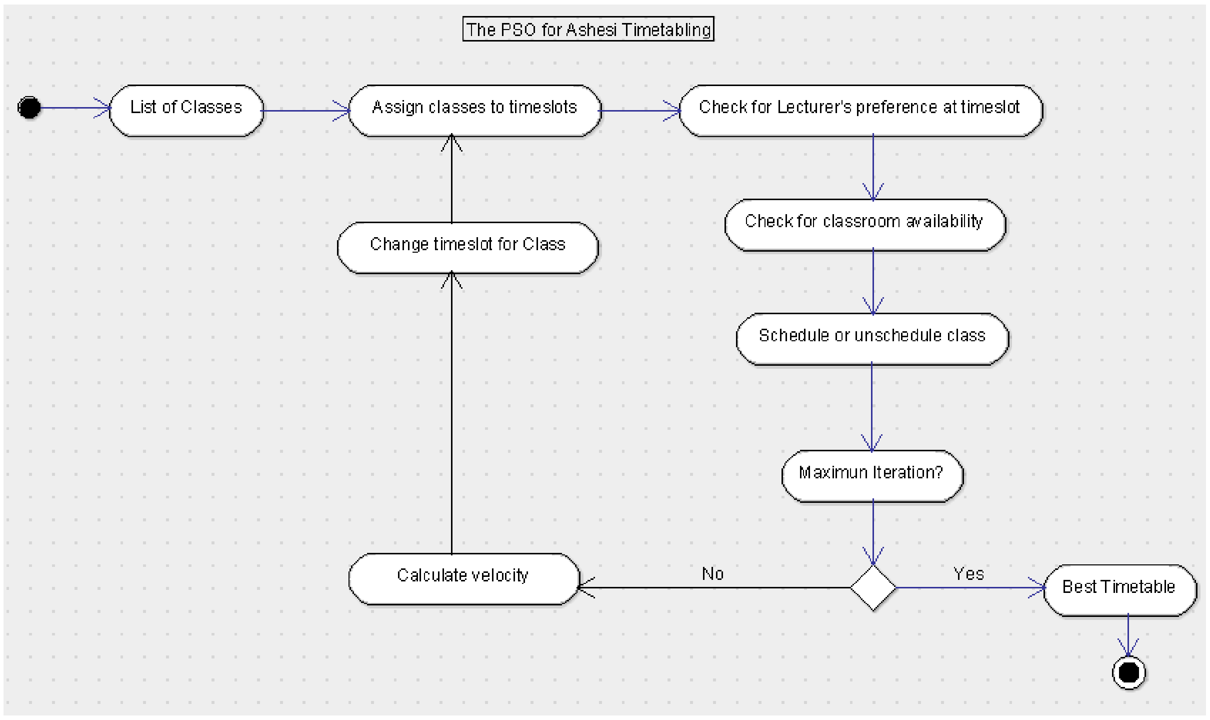
The PSO for Ashesi Timetabling
List of Classes
Assign classes to timeslots
Check for Lecturer's preference at timeslot
Check for classroom availability
Change timeslot for Class
Schedule or unschedule class
Maximun Iteration?
Calculate velocity
No
Yes
Best Timetable

Figure 2An Activity Diagram for the PSO of the ACTP

CHAPTER 4: THE IMPLEMENTATION

This project provides a solution to the Ashesi course timetabling problem based on lecturers preferences for timeslots to which classes are assigned and room allocation, in each iteration of the PSO algorithm. Lecturers' preference is one of the three main components (lecturers' preference, room preference, and student preference) necessary for creating the most optimal solution to the course timetabling problem at Ashesi. However, student preferences as well as classroom preferences for the timetable construction will not be considered in this project. In this project, a simplifying assumption is that all classrooms are of equal capacity and facilities.Hence the main focus of this project is to maximize our objective function which depends on the fitness of the swarm based on the sum of lecturers' preferences for timeslots of the schedule and room allocation penalty.At the end of the optimization process, the highest preference sum among all other preference sums is the optimum solution.

The objective is to maximize a fitness function:

MAX: $f(x) = \sum_{n=i}^{x}(\text{Lecturers}'preferences) + \sum_{n=i}^{x}(\text{Room allocation Penalty})$

Where: x = the number of iterations

n = the ith particle in the swarm

4.1 THE SYSTEM SETUP

In solving the Ashesi timetabling problem using Particle Swarm Optimization, particles were defined to be made up classes, timeslots and lecturers preferences; hence a particle is a timetable that has timeslotsassigned to classes taught by lecturers. In this paper a, as mentioned earlier, a class is made up of courses and the different sections of course. Our main focus will be on class scheduling which is a broader category. This definition was based on the assumption that the same teacher could be teaching two different sections of the same course and each section is given its own code.Hence a class constitutes the course code, course name and the lecturer teaching it and a class can be identified by its code number which is unique for each class. In this project, the class codes were generated based on the count of the classes listed in the system. This coding system was used just for quick references to courses.

For an easier start in implementing the system, 13 lecturers and 20 classeswereused. These figures are representative of the Ashesi System; however actual figures representing the Ashesi Course Timetabling problem were also used for the implementation. For example, class 10 represents Quantitative Methods by Prof. Jackson and class 17 represents Programming by Prof. Jackson. It is typical that a lecturer teaches more than one course.Table 2represents sample class codes that were used for the implementation where T*i* stands for a lecturer out of the 13 lecturers who taught the 20 classes stated below. A simplifying assumption is that no course has two sections; hence all sections of a course are referred to as different classes.

CODE	LECTURER	SUBJECT	CODE	LECTURER	SUBJECT
Class 1	T1	Science	Class 11	T5	Stats
Class 2	T2	Social	Class 12	T9	Finance
Class 3	T3	Math	Class 13	T10	Negotiation
Class 4	T1	English	Class 14	T11	Robotics
Class 5	T2	French	Class 15	T12	Programing
Class 6	T4	Math	Class 16	T9	Trade
Class 7	T5	Script	Class 17	T8	Networks
Class 8	T6	Ecomm	Class 18	T8	Ecomm
Class 9	T7	Mobile	Class 19	T13	Database
Class 10	T8	Quant	Class 20	T5	OpSystems

Table 2 Class code, Lecturers and Courses for the course scheduling

Also, there are 25 timeslots available at Ashesi, that is, 5 periods a day, 5 times in a week. However, in order to simplify the calculations and keeping track of timeslots, an assumption was made that lecturer can teach on every other day. So for a lecturer teaching on Monday, he/she will automatically teach on Wednesday and for a lecturer teaching on Tuesday, he/she will automatically teach on Thursday. This simplifying assumption is in fact the current practice at Ashesi University College. On Fridays, every teacher is scheduled to have a lab section so the schedule for that will be separated from the main schedule. Hence, the number of timeslots used for the Ashesi Course Timetabling problem was reduced to 10 timeslots since the schedule is for two days (Monday and Tuesday) and there are five (5) timeslots in a day. In this case, if a lecturer is scheduled to teach on Monday at 8:30 am, he/she will automatically be assigned to teach on Wednesday at 8:30 am as well.

Timeslots, consisting of one hour thirty minutes (1hr: 30minutes) durations as it is in Ashesi, were used to for the implementation; hence timeslots were labeled from the one to the ten starting from Monday 8:30 am to Tuesday 4:40pm. Table 3shows an example of timeslots that was generated for this problem.

Day\Time	**8:30**	**10:10**	**11:40**	**1:20**	**3:00**
Monday	1	2	3	4	5
Tuesday	6	7	8	9	10

Table 3 Time slots for the ACTP

In order to simplify the algorithm, the timeslots were used as the search space of the particles. An assumption was then made that particles are only allowed to operate in the search space and not outside of it.Hence from a position of 1 to 10, the 20 classes (being used to represent a particle) were distributed across the search space as their initial positions with an availability of 3 classrooms per positions (timeslot). In order to prevent spill overs and allow the particle to stay within the search space, initial positions of one to ten (1 – 10) were assigned to particles. Table 4shows the classes that were used for the implementation and their initial positions.

Code	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Initial Position	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10

Table 4 Initial positions of classes within the search space

With the help of these initial positions, the velocity for each class was calculated based on equation 1.

$$v_{id}^{t+1} = w.\ v_{id}^{t}+c_1.r_1.(p_{id}^{t} + x_{id}^{t})+c_2.r_2.\left(p_{gd}^{t} + x_{id}^{t}\right) \text{ ………………… } (1)$$

The inertia weight, *w*, was set to 0.729 (values could be changed within the range of 0.0 to 1.0).The acceleration constants, *c1* and *c2*, were set to 1.49445 each (the addition of *c1* and *c2* should not exceed 4. and the random values, *r1* and *r2*, were set to random numbers between 0 and 1. For the first velocity update, the initial velocity v_{id}^{t} was assumed to be the same as the initial positions for each of the classes. This assumption was made based on the fact that at the first velocity update, the particles have no velocity but their initial positions. The particles personal best position and the global best position were also set to the initial positions multiplied by 2 and 3 respectively. This assignment was also made based on the assumption that the main focus of the timetable is to get the best schedule that suits lecturers' preferences. Hence, the particles do not have to move towards a particular particle. Each particle should have its own position update based on its updated velocity and then it is scheduled based on the lecturer's preference for it at the position it has assumed. Moving towards other particles in the swarm is not the main focus.

Classes' positions are then updated based on the sum of the velocity calculated and the initial positions as shown by equation 2.

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

However, this resulted in position updates that were outside the range of the search space. In order to maintain all positions within the search space of 1 to 10, a velocity clamping technique was used. Hence for each of the updated positions, if the position specified is greater than 10 (the length of the search space) a mod of 10 was applied to the number and the result is added to one (1) to cater for the exclusion of 0 from the search space and inclusion of 10 in the search space. For example, if a newly updated position resulted in 34 the actual updated position that will be used is 5, that is, (34 mod 10) + 1 = 5.

It is required by the system to state the level of preference that each lecturer attaches to the various timeslots in the system. So for each of the 13 lecturers used for the system implementation,lecturers' preference levels for a position assumed by a class is read in from a text file for that particular lecturer. The preference levels ranged from 1 to 6 all inclusive with the highest preference of 6 and the lowest preference of 1. For example, as in figure 1.7, lecturer 1 has a preference of 5 at timeslot 3. This is interpreted as Lecturer 1 prefers to teach on Monday at 11:50 am as compared to Monday 10:10 am due to the preference levels set to those timeslots.Table 5shows each lecturer and their preference for each timeslot.

Figure 1 1 Lecturers' preferences for timeslots

	PREFERENCES FOR TIMESLOTS									
Lecturer	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
T1	2	1	5	4	2	6	3	1	2	4
T2	4	6	3	1	1	3	2	2	5	5
T3	1	2	5	1	4	4	5	1	3	3
T4	3	1	1	2	1	3	2	3	1	6
T5	2	1	5	2	6	3	1	1	2	2
T6	4	5	3	6	4	5	2	5	4	3
T7	4	2	5	1	1	3	3	1	4	5
T8	6	2	6	4	4	4	6	5	5	5
T9	4	2	1	3	3	6	5	5	5	2
T10	1	3	4	2	2	4	6	2	6	3
T11	4	2	2	2	6	3	1	2	4	2
T12	2	5	6	6	6	5	2	5	5	5
T13	6	2	1	4	6	2	3	3	1	1

Table 5 Lecturers' Preferences for Time slots

4.2.OPERATING IN THE SWARM

Once all elements of a particle have been initialized, a number of iterations are set and a number of particles are created in the swarm. This swarm of particles is used in several iterations specified to determine the best timetable in the swarm,taking into consideration teachers' preferences for classes at specified timeslotsand the availability of classrooms for a class to be scheduled, avoiding conflicts.

Clashes are very critical to the timetable generation. All clashes need to be avoided as much as possible to obtain an optimal timetable. In order to get rid of clashes for this timetable generation, a penalty of negative ten (-10) is allocated to a class as the preference for a teacher if that class happens to assume the same position as other classes being thought by the same lecturer. This penalty of -10 reduces the fitness of that particular particle and does not make it optimal.

Also another issue being addressed in this project is the availability of classrooms for all classes taking place at the same time.Hence for a class to be schedule, it needs to assume a position that will have classroom available for the class. For example in our simplifying example, it is assumed that there arethree classrooms available. Hence for a class to be scheduled into a particular timeslot it needs to be among the first three classrooms to be scheduled into that timeslot. In that case, the class is assigned a positive penalty of five (5). However, if a class happens to be the fourth, fifth and so on, class to be scheduled into a timeslot, it is assigned a negative penalty of six (-6) since there are no classroom available to contain that class. Similarly, when a class is assigned a lecturer’s preference of -10 because it causes a clash, a classroom penalty of -10 is also assigned to the class. At the end, all lecturers’ preferences together with classroom allocation penalties for classes are added to determine the fitness of the particle.

After iteration, each particle in the swarm computes its fitness (the sum of all lecturers’ preferences for timeslots and classroom allocation penalty for classes) and compares it to the fitness of other particles in the swarm. The particle with the highest fitness in the swarm has the global best position in the swarm. When the particle with the best position is found in the swarm, all other particles in the swarm use this global best position together with their current and best positions to calculate their velocity in order to update their positions. This process is repeated several times until the end of the iteration. Each iteration ends by stating the particle with the highest fitness. At the end of the process, the best timetable is derived from the particle with the best fitness throughout the entire swarm.

CHAPTER 5:TESTING,RESULT AND DISCUSSION

5.1 SAMPLE DATA

After setting up the entire system, three different sets of lecturers' preferences were used to test for the validity of the algorithm that has been used for the implementation. 13 lecturers, 20 classes, 3 classrooms, 10 particles and 1000 iterations were used for this implementation. The test resulted in a conflict free timetable for all lecturers regardless of the number of courses they had to teach. Table 6below shows one set of lecturers' preferences and the corresponding best particle or optimal timetable that was obtained using the following PSO parameters: w=0.729, c1=c2=1.49445 and r1 and r2 of random numbers from 0 to 1. The timetable was generated in 5 minutes 23seconds.

PREFERNCES 1	T1	1 5 2 4 1 3 4 3 4 4
	T2	6 5 2 4 3 1 6 1 5 6
	T3	1 6 3 6 4 1 2 6 1 2
	T4	2 6 2 4 1 3 5 4 5 5
	T5	4 2 4 3 5 5 5 6 4 1
	T6	2 6 6 6 3 2 2 5 1 6
	T7	4 1 5 3 3 3 5 1 1 1
	T8	5 1 6 5 3 1 6 2 3 2
	T9	5 4 2 3 6 2 5 4 3 3
	T10	3 2 5 5 1 2 1 5 1 3
	T11	5 3 6 1 2 2 6 1 1 4
	T12	2 6 5 5 1 5 5 5 2 1
	T13	2 3 6 3 3 1 5 6 1 3
RESULT		

```
Teacher 1 has preference of 3 at position 6 with room preference of 5 for class 1
Teacher 2 has preference of 3 at position 5 with room preference of 5 for class 2
Teacher 3 has preference of 6 at position 4 with room preference of 5 for class 3
Teacher 1 has preference of 1 at position 5 with room preference of 5 for class 4
Teacher 2 has preference of 6 at position 10 with room preference of 5 for class 5
Teacher 4 has preference of 5 at position 7 with room preference of 5 for class 6
Teacher 5 has preference of 6 at position 8 with room preference of 5 for class 7
Teacher 6 has preference of 6 at position 3 with room preference of 5 for class 8
Teacher 7 has preference of 1 at position 10 with room preference of 5 for class 9
Teacher 8 has preference of 5 at position 1 with room preference of 5 for class 10
Teacher 5 has preference of 5 at position 7 with room preference of 5 for class 11
Teacher 9 has preference of 5 at position 7 with room preference of 5 for class 12
Teacher 10 has preference of 2 at position 2 with room preference of 5 for class 13
Teacher 11 has preference of 5 at position 1 with room preference of 5 for class 14
Teacher 12 has preference of 1 at position 5 with room preference of 5 for class 15
Teacher 9 has preference of 4 at position 8 with room preference of 5 for class 16
Teacher 8 has preference of 6 at position 3 with room preference of 5 for class 17
Teacher 8 has preference of 5 at position 4 with room preference of 5 for class 18
Teacher 13 has preference of 3 at position 10 with room preference of 5 for class 19
Teacher 5 has preference of 5 at position 6 with room preference of 5 for class 20
--------------------------------------------------------------------------------
Particles fitness is 183
The best Global fitness is 183

BUILD SUCCESSFUL (total time: 5 minutes 26 seconds)
```

Table 6 Timetable Generated from PSO based on Lecturers' preferences and available classrooms

Below in Table 7 is the actual time table representation of the above generated timetable with room allocation.

	8:30-10:00	**10:10-11:40**	**11:50-1:20**	**1:30-3:00**	**3:10-4:40**
Monday	T11 Class 14 Room1	T10 Class 13 Room1	T6 Class 8 Room1	T3 Class 3 Room1	T2 Class 2 Room1
	T8 Class 10 Room2		T8 Class 17 Room2	T8 Class 18 Room2	T1 Class 4 Room2
					T12 Class 15 Room3
Tuesday	T1 Class 1 Room1	T4 Class 6 Room1	T5 Class 7 Room1		T2 Class 5 Room1
	T5 Class 20 Room2	T5 Class 11 Room2	T9 Class 16 Room2		T7 Class 9 Room2
		T9 Class 12 Room3			T13 Class 19 Room3

Table 7 ATimetable with Room Allocations

The use of Particle Swarm Optimization for course scheduling based on teachers' preferences and classroom availability has been demonstrated to be an effective process to generate a course timetable devoid of conflicts. However, as stated earlier in the literature analysis, PSO mostly guarantees an optimal solution but not always. For that reason, the number of iterations and the parameters being used for the timetable generation is very important.

After, running the algorithm with the simplifying example in this project for six times, six optimal solutionswere generated in each run out of the 10 particles and 1000 iterations used for generating a timetable for 13 lecturers, 20 classes and 3 classrooms. These results prove that PSO guarantees an optimal solution

5.2 ACTUAL DATA

The system was also tested using actual values corresponding to the number of classes and lecturers (48 classes and 30 lecturers) that are available in a semester at Ashesi University College. At the end, a timetable that was devoid of class conflicts based on teachers' preference was generated using the following PSO parameters: w=0.729, c1=c2=1.49445 and r1 and r2 of random numbers from 0 to 1.Table 8in appendix shows the lecturers' preference at each of the slots and the timetable that was generated.

5.2.1 A THOUSAND ITERATIONS

With 1000 iterations, 10 particles, 48 classes, 30 lecturers and 7 classrooms, out of six runs conducted, four runs resulted in optimal solutions devoid in any form of conflicts (lecturer and classroom). For each of these runs, an average of 8 minutes was used in generating the timetable.

5.2.2 TWO THOUSAND ITERATIONS

With 2000 iterations, 20 particles, 48 classes, 30 lecturers and 7 classrooms, out of six runs conducted, all six runs resulted in optimal solutions devoid in any form of conflicts (lecturer and classroom) in an average of 11 minutes.

These tests have proven that the PSO algorithm needs to be tuned based on the size of parameters being used forthe system's implementation. Apart from that, it has successfully been used to accomplish the objectives of this project.

5.3CHALLENGES

Many challenges were encountered in this project. Some of these challenges were critical to the systems implementation while others had to do with an understanding of how the algorithms studied in the paper could be applied to timetabling problems.

One of the major challenges theauthor was her ability to interpret the Particle Swarm Optimization algorithm in relation to the timetabling problem she was solving. Though there were lots of literature on how particle swarm optimization had been used to solve course timetabling problems, most of these literature failed to relate the PSO algorithm to the actual implementation of course scheduling. Most of these literatures just restated the PSO algorithm when it came to explaining the implementation process of their solution. This challenge took a lot of this author's time as she tried to interpret,for example, what a "Particle" (as used in the PSO algorithm) stood for when it came to course timetabling problems.

However, despite how challenging this project was, the author is glad she undertook this project which had enlightened her in other fields such as engineering and artificial intelligence.

CHAPTER 6: CONCLUSION AND FUTURE WORK

This paper describes the procedure used in successfully generating a conflict-free timetable for the course timetabling problem at Ashesi University College based on lecturers' preferences through the use of Particle Swarm Optimization algorithm. Careful studies on five algorithms (Genetic Algorithms, Constraint Programming, Particle Swarm Optimization, Simulated Annealing and Tabu Search) presented in this paper served as the bases for which the PSO algorithm was chosen and used in solving the timetabling problem at Ashesi. The goal of optimizing a timetable schedule that is devoid of conflicts was achieved as a conflict-free timetable was generated for the Ashesi Course Timetabling Problem based on lecturers' preferences.

Despite the effort made to create a course timetable devoid of conflict to solve the Ashesi Course Timetabling Problem based on lecturers' preferences, more work needs to be done in creating a schedule that takes into consideration classroom preferences (size of classroom and location eg. Classroom or lab) for courses and students' preferences for timeslots beyond lecturers' preferences that has been dealt with in this paper. The combination of these three solutions to the course timetabling problem will lead to the creation of the most optimum course timetablefor the Ashesi University College course timetabling problem.

This project has been a successful one with a great learning curve for the author. The attempt to help solve the course timetabling problem at Ashesi University College is in the right direction and needs the support of students (by taking up the challenge), lecturers, faculty and academic registrars (by supporting students who wish to continue this project), to help find a solution to the Ashesi course timetabling problem which is yet to become serious as the school's population increases – many students and faculty will have different preferences and not all classroom sizes will be equal.

In conclusion, with conflict-free timetables generated based on lecturers preferences and classroom allocation, PSO has been clearly shown in this project to be capable of addressing the timetabling challenge at Ashesi University College.

REFERENCES

[1] Sadaf Naseem Jat and Shengxiang Yang, "A Guided Search Genetic Algorithm for the University Course Timetabling Problem," in *Multidiscplinary International Conference on Scheduling: Theory and Applications (MISTA)*, Dublin, 2009, pp. 180-191.

[2] Anirudha Nanda, Manisha P Pai, and Abijeet Gole, "An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach," *International Journal of Machine Learning and Computing*, pp. 492-495, 2012.

[3] Keith Murray and Tomas Muller, "Automated System for University Timetabling," West Lafayette, 2006.

[4] Konstantinos E. Parsopoulos and Michael N. Vrahatis, "Particle Swarm Optimization Method for Constrained Optimization Problems," Patras, Greece, 2002.

[5] Sueychyun Fang, "University Course Scheduling System - A UML Application with Database and Visual Programming," *Consortium for Computing Sciences in Colleges: Mid-South Conference*, pp. 160-169, June 2005.

[6] Deris Safaai, Irene Fen Ho Sheau, and Mohd Siti Zaiton Hashim, "Solving University Course Timetable Problem Using Hybrid Particle Swarm Optimization," Johor, Malaysia, 2010.

[7] Samuel Lukas, Arnold Aribowo, and Milyandreana Muchri, "Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study: Universitas Pelita Harapan Timetable," China, 2012.

[8] Rushil Raghavjee and Nelishia Pillay, "An Informed Genetic Algorithm for High School Timetabling Problem," Bela Bela, South Africa, 2010.

[9] Roman Bartak, "Constraint Programming: In Pursuit of the Holy Grail," Czech, 2000.

[10] S. C. Chu and H. L. Fang, "Genetic Algorithms vs. Tabu Search in Timetable Scheduling," *Knowledge-Based Intelligent Information Engineering Systems, Third International Conference*, pp. 492-495,

1999.

[11] Yi-Tin Chen, Shu-Chuan Chu, and Jiun Huen Ho, "Timetable Scheduling Using Particle Swarm Optimization," in *Proceedings of the First International Conference on Innovative Computing, Information and Control*, Shenzhen, 2006, pp. 1-4.

[12] Edmund Burke, Kirk Jackson, Jeff Kingston, and Rupert Weare, "Automated University Timetabling: The State of the Art," Nottingham, UK, 1998.

[13] D Abramson, "Constructing Timetables Using Simulated Annaling: Sequential and Parallel Algorithms," *Management Science*, pp. 98-113, 1991.

[14] Oskar Preinfalk and Helmut A. Mayer, "Automatic Construction of Drama School Timetables Based on Generic Evolutionary Framework for Allocation ans Scheduling Problems," *ACM Symposium on Applied Computing*, pp. 996-1000, 2004.

[15] Vic Ciesielski and Paul Scerri, "Real Time Genetic Schedulingof Aircraft Landing Times," *The IEEE International Conference on Evolutionary Computation*, pp. 360-364, 1998.

[16] A. Tamilarasi and Anantha T. Kumar, "An enhanced genetic algorithm with simulated annealing for job-shop Scheduling," *International Journal of Engineering, Science and Technology*, pp. 144-151, 2010.

[17] Jianguo Jiang, Mingxing Wen, Kaige Ma, Xiuping Long, and Junzheng Li, "Hybrid Genetic Algorithm for Flexible Job-shop Scheduling with Multi-objective," *Journal of Information & Computational Science*, pp. 2197-2205, 2011.

[18] M. J. M. Heijligers, L. J. M. Cluitmans, and J. A. G. Jess, "High-Level Synthesis Scheduling and Allocation using Genetic Algorithms," *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 61-66, 1995.

[19] John Hooker, "Operations Research in Constriant Programming," *Carnegie Mellon University*, pp. 1-329, 2009.

[20] S Sarathambekai and K. Umamaheswari, "Comparison among four Modified DiscreteParticle Swarm Optimization for Task Scheduling in

Heterogeneous Computing Systems," *International Journal of Soft Computing and Engineering*, pp. 2231-2307, 2013.

[21] Irene Fen Ho Sheau, Deris Safaai, and Mohd S. Z. Hashim, "University Course Timetable Planning using Hybrid Particle Swarm Optimization," Malaysia, 2010.

[22] Ruey-Maw Chen and Hsiao-Fang Shih, "Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search," *Algorithms*, pp. 227-244, 2013.

[23] Andeep Kumar, Kawaljeet Singh, and Neeraj Sharma, "AUTOMATED TIMETABLE GENERATOR USING PARTICLE SWARM OPTIMIZATION," *International Journal on Recent and Innovation Trends in Computing and Communication*, pp. 686-692, 2013.

[24] Po-Hung Chen, "Particle Swarm Optimization for Power Dispatch with Pumped Hydro," *InTech*, pp. 132-144, 2007.

[25] Magnus Erik Hvass Pedersen, "Good Parameters for Particle Swarm Optimization," 2010.

[26] Fikret M. Ercan, "Particle Swarm Optimization and Other Metaheuristic Methods in Hybrid Flow Shop Scheduling Problem," *Particle Swarm Optimization*, pp. 155-168, 2009.

[27] Adam Tauman Kalai and Santosh Vempala, "Simulated Annealing for Convex Optimization," *MATHEMATICS OF OPERATIONS RESEARCH*, pp. 1-15, 2006.

[28] Stefka Fidanova, "Simulated Annealing for Grid Scheduling Problem," Sofia, Bulgaria, 2007.

[29] Sergio Ledesma, Gabriel Avina, and Raul Sanchez, "Practical Consideration of Simulated Annealing Implementation," *InTech*, pp. 401-420, 2008.

[30] Tuan-Anh Duong and Kim-Hoa Lam, "Combining Constraint Programming andSimulated Annealing on University Exam Timetabling," Hanoi, Vietnam, 2004.

[31] E. Aycan and T. Ayav, "Solving the Course Scheduling Problem Using

Simulated Annealing," 2009.

[32] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson, "THE THEORY AND PRACTICE OF SIMULATED ANNEALING," *Air Force Office of Scientific Research*, pp. 287-319, 2003.

[33] M.A. Saleh Elmohamed, Geoffrey C. Fox, and Paul and Coddington, "A Comparison of Annealing Techniques for Academic Course Scheduling," *Northeast Parallel Architecture Center*, pp. 1-20, 1997.

[34] Janaki D. Ram, T. H. Sreenivas, and Ganapathy K. Subramaniam, "Parallel Simulated Annealing Algorithms," *JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING*, pp. 207-212, 1996.

[35] Rafael E. Banchs, "Simulated Anealing," 2000.

[36] Michel Gendreau and Jean-Yves Potvin, "Tabu Search," 2010.

[37] Jean-Yves Potvin and Michel Gendreau, "Tabu Search," 2004.

[38] A. R. Mushi, "Tabu Search Heuristic for University Course Timetabling Problem," *African Journal of Science and Technology (AJST)*, pp. 34-40, 2006.

[39] Lei Li, HongRui Liu, and Roberto Lu, "Tabu Search," 2007.

[40] Alain Hertz, Eric Taillard, and Dominique de Werra, "A Tutorial on Tabu Search," 1997.

[41] Sadan Kulturel-Konak, Bryan A. Norman, David W. Coit, and Alice E. Smith, "EXPLOITING TABU SEARCH MEMORY IN CONSTRAINED PROBLEMS," *Journal on Computing*, pp. 1-31, 2001.

[42] James Blondin, "Particle Swarm Optimization: A tutorial," 2009.

[43] Mohd Siti Zaiton Hashim, Deris Safaai, and Irene Fen Ho Sheau, "Incorporating of Constraint-Based Reasoning into Particle Swarm Optimization for University Timetabling Problem," *COMPUTER SCIENCE LETTERS*, pp. 1-21, June 2009.

[44] Qinghai Bai, "Analysis of Particle Swarm Optimization Algorithm," *Computer and Information Science*, vol. III, no. 1, pp. 180-184, 2010.

[45] Kyriakos Kentzoglanakis and Matthew Poole, "Particle Swarm Optimization with an Oscillating Inertia Weight," in *GECCO '09*, United Kingdom, 2009, pp. 1-8.

[46] R. Jayanthi, I. A. Chidambaram, and C. Banusri, "Decentralized controller gain scheduling using PSO for power system restoration assessment in a two-area interconnected power system," *International Journal of Engineering, Science and Technology*, vol. III, no. 4, pp. 14-26, 2011.

[47] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. VI, no. 1, pp. 58-73, 2002.

[48] Goncelo Pereira, "Particle Swarm Optimization," Porto Savlo, Portugal, 2011.

[49] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, 1995, pp. 39-43.

APPENDIX

PREFERENCES	T1	2 3 3 2 2 1 5 6 6 5
	T2	1 6 3 1 5 2 5 3 5 1
	T3	2 5 3 2 1 6 5 5 4 6
	T4	5 1 1 6 2 3 3 1 2 4
	T5	2 4 5 1 4 6 4 2 2 1
	T6	1 6 2 6 4 6 1 4 6 6
	T7	6 6 4 3 6 1 3 6 1 4
	T8	4 1 4 3 5 2 1 2 3 4
	T9	3 3 3 6 2 1 6 3 2 3
	T10	2 5 4 5 4 1 4 2 3 6
	T11	5 3 1 1 5 3 4 5 5 3
	T12	3 4 5 3 5 6 3 5 5 4
	T13	6 6 3 4 5 6 3 2 1 6
	T14	2 6 1 4 4 6 4 5 5 6
	T15	3 1 5 5 1 3 5 5 5 3
	T16	5 4 1 1 2 4 1 6 2 1
	T17	1 3 6 3 6 5 1 1 5 2
	T18	5 1 1 2 1 1 4 4 2 4
	T19	2 5 3 1 3 6 4 5 1 4
	T20	2 5 1 2 6 3 3 2 2 4
	T21	4 3 1 4 3 5 5 3 6 5
	T22	5 3 2 2 4 2 3 5 3 3
	T23	4 3 5 2 6 4 1 2 4 3
	T24	5 6 6 3 6 5 2 3 3 2
	T25	1 6 1 4 6 6 2 5 4 3
	T26	1 2 3 5 1 5 5 4 5 6
	T27	4 3 4 1 4 6 6 4 1 4
	T28	3 4 5 6 4 1 1 6 3 3
	T29	1 4 6 4 2 2 6 3 3 4
	T30	3 2 6 2 6 1 5 3 4 1

RESULT

Teacher 1 has preference of 5 at position 10 with room preference of 5 for class 1
Teacher 2 has preference of 1 at position 10 with room preference of 5 for class 2
Teacher 3 has preference of 3 at position 3 with room preference of 5 for class 3
Teacher 1 has preference of 6 at position 9 with room preference of 5 for class 4
Teacher 2 has preference of 5 at position 9 with room preference of 5 for class 5
Teacher 4 has preference of 2 at position 9 with room preference of 5 for class 6
Teacher 5 has preference of 2 at position 8 with room preference of 5 for class 7

```
Teacher 6 has preference of 2 at position 3 with room preference of 5 for class 8
Teacher 7 has preference of 6 at position 5 with room preference of 5 for class 9
Teacher 8 has preference of 3 at position 4 with room preference of 5 for class 10
Teacher 5 has preference of 5 at position 3 with room preference of 5 for class 11
Teacher 9 has preference of 3 at position 10 with room preference of 5 for class 12
Teacher 10 has preference of 3 at position 9 with room preference of 5 for class 13
Teacher 11 has preference of 5 at position 9 with room preference of 5 for class 14
Teacher 12 has preference of 3 at position 7 with room preference of 5 for class 15
Teacher 9 has preference of 6 at position 7 with room preference of 5 for class 16
Teacher 8 has preference of 3 at position 9 with room preference of 5 for class 17
Teacher 8 has preference of 4 at position 10 with room preference of 5 for class 18
Teacher 13 has preference of 4 at position 4 with room preference of 5 for class 19
Teacher 5 has preference of 4 at position 5 with room preference of 5 for class 20
Teacher 14 has preference of 6 at position 2 with room preference of 5 for class 21
Teacher 15 has preference of 5 at position 8 with room preference of 5 for class 22
Teacher 3 has preference of 6 at position 10 with room preference of 5 for class 23
Teacher 16 has preference of 1 at position 7 with room preference of 5 for class 24
Teacher 17 has preference of 6 at position 5 with room preference of 5 for class 25
Teacher 13 has preference of 2 at position 8 with room preference of 5 for class 26
Teacher 7 has preference of 3 at position 4 with room preference of 5 for class 27
Teacher 18 has preference of 4 at position 8 with room preference of 5 for class 28
Teacher 19 has preference of 3 at position 5 with room preference of 5 for class 29
Teacher 20 has preference of 1 at position 3 with room preference of 5 for class 30
Teacher 21 has preference of 3 at position 5 with room preference of 5 for class 31
Teacher 22 has preference of 4 at position 5 with room preference of 5 for class 32
Teacher 23 has preference of 4 at position 9 with room preference of 5 for class 33
Teacher 24 has preference of 6 at position 2 with room preference of 5 for class 34
Teacher 25 has preference of 5 at position 8 with room preference of 5 for class 35
Teacher 26 has preference of 4 at position 8 with room preference of 5 for class 36
Teacher 27 has preference of 4 at position 3 with room preference of 5 for class 37
Teacher 28 has preference of 6 at position 4 with room preference of 5 for class 38
Teacher 29 has preference of 6 at position 3 with room preference of 5 for class 39
Teacher 30 has preference of 6 at position 3 with room preference of 5 for class 40
Teacher 21 has preference of 4 at position 4 with room preference of 5 for class 41
Teacher 18 has preference of 1 at position 2 with room preference of 5 for class 42
Teacher 22 has preference of 3 at position 2 with room preference of 5 for class 43
Teacher 28 has preference of 3 at position 10 with room preference of 5 for class 44
Teacher 27 has preference of 1 at position 4 with room preference of 5 for class 45
Teacher 19 has preference of 5 at position 2 with room preference of 5 for class 46
Teacher 20 has preference of 3 at position 6 with room preference of 5 for class 47
Teacher 28 has preference of 4 at position 2 with room preference of 5 for class 48
------------------------------------------------------------------------------------
Particles fitness is 424
The best Global fitness is 424
BUILD SUCCESSFUL (total time: 6 minutes 3 seconds)
```

Table 8A simulation of an Ashesi Timetable based on lecturers' preferences.