Winter 3-2021

PERSONALIZED LEARNING PATH GENERATION BASED ON GENETIC ALGORITHMS

Lumbardh Elshani
University for Business and Technology - UBT

Programi për Shkenca Kompjuterike dhe Inxhinieri

PERSONALIZED LEARNING PATH GENERATION BASED ON GENETIC ALGORITHMS

Shkalla Bachelor

Lumbardh Elshani

Mars / 2021
Prishtinë

Programi për Shkenca Kompjuterike dhe Inxhinieri

Punim Diplome
Viti Akademik 2017 – 2018

Lumbardh Elshani

PERSONALIZED LEARNING PATH GENERATION BASED ON GENETIC ALGORITHMS

Mentori: Krenare Pireva Nuçi, PhD

Mars / 2021

Ky punim është përpiluar dhe dorëzuar në përmbushjen e kërkesave të pjesshme për Shkallën Bachelor

ABSTRACT

A substantial disadvantage of traditional learning is that all students follow the same curriculum sequence, but not all of them have the same background of knowledge, the same preferences, the same learning goals, and the same needs. Traditional teaching resources, such as textbooks, in most cases orient students to follow fixed sequences during the learning process, thus impairing their performance. Curriculum sequencing is an important research issue for learning process because no fixed learning paths will be appropriate for all learners. For this reason, many research papers are focused on the development of mechanisms to offer personalization on learning sequences, considering the learner needs, interests, behaviors, and abilities. In most cases, these researches are totally focused on the student's preferences, ignoring the level of difficulty and the relation degree that exists between various concepts in a course. This work presents a genetic algorithm-based model to offer personalization on learning paths, considering the level of difficulty and relation degree of the constituent concepts of a course. The experimental result shows that the genetic algorithm is suitable to generate optimal learning paths based on learning object difficulty level, duration, rating, and relation degree between each learning object. Furthermore, it indicates that using the proposed genetic-based approach for personalized learning path generation is superior to the traditional curriculum sequencing.

ACKNOWLEDGEMENTS

For the successful completion of this diploma thesis, I would like to express my sincere gratitude and acknowledgements to my research supervisor, Professor Krenare Pireva Nuçi, for providing invaluable guidance and insightful feedback, that brought my work to a higher level. She was the main person who encouraged me in this research, staying close to me throughout the conduct of this work, giving me advice, suggestions, valuable comments and providing relevant theoretical and practical resources for modeling and finalizing this thesis.

Also, I would like to thank the academic and administrative staff of UBT because without their help, I would not have been able to get here. The care and support shown to me during these three years of studies has encouraged me to work harder and overcome various challenges.

For a cherished time spent together and for the source of support when things got discouraging, I would like to thank my friends and colleagues.

Finally, I would like to express my gratitude to my parents and family, who have always stayed close to me even in the most difficult moments, giving me courage and support throughout my studies. I am extremely grateful to them for their love, care and sacrifice for my upbringing and achievement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF CODE SNIPPETS

LIST OF TABLES

LIST OF EQUATIONS

GLOSSARY

GA – Genetic Algorithm

EA – Evolutionary Algorithm

LPG – Learning Path Graph

PSO – Particle Swarm Optimization

SOM – Self Organizing Map

AACS – Attribute-based Ant Colony System

BPT – Bayesian Probability Theory

PLS – Personalized Learning System

CAT – Computer Adaptive Testing

CBR – Case-based Reasoning

MOOC – Massive Open Online Courses

PLP – Personalized Learning Path

PMX – Partially Mapped Crossover

RDM – Relation Degree Matrix

XML – eXtensible Markup Language

SA – Simulated Annealing

HC – Hill Climbing

1. INTRODUCTION

The rapid change of technology and the increase in the internet usage has affected many life processes, and one of them that has been greatly affected is the learning process, as one of the most sensitive processes in the cycle of human activity. Learning is the act of acquiring new knowledge, skills, behaviors, or values, or modifying and reinforcing them, and can involve the synthesis of different types of information. This ability is possessed by humans, animals, and some types of machines. Learning is a continuous process that occurs throughout a person's life, but the path followed by everyone is not suitable for everyone equally. Hence, the need to personalize learning pathways is essential to increase performance and effectiveness of the knowledge gained for a learner. But first, to understand the importance of providing personalization in the learning paths, we must understand what a learning path is? A learning path can be described as a chosen path, taken by a learner through a variety of learning activities, allowing him / her to build knowledge progressively. Studies on pathways help us to explore and explain student behaviors during learning processes, since they have unique knowledge structures based upon their previous experiences and abilities. Learning paths also reveal the learning trails while learners traverse any interactive environment [1]. A learning path allows people to gain knowledge in small chunks, giving students the role of helmsman. It can be customized in the way it creates self-taught and self-paced learning experiences. Now, we need to familiarize ourselves with personalized learning. Personalized learning refers to a wide variety of educational programs, academic strategies, learning experiences, and systematic pedagogical approaches designed to address student's specific learning needs, aspirations, and interests on an individual basis. Whereas the general purpose is to put in the primary consideration the individual learning needs. The United States National Education Technology Plan 2017 defines personalized learning as follows:

"*Personalized learning refers to instruction in which the pace of learning and the instructional approach are optimized for the needs of each learner. Learning objectives, instructional approaches, and instructional content (and its sequencing) may all vary based*

on learner needs. In addition, learning activities are meaningful and relevant to learners, driven by their interests, and often self-initiated".

The purpose of personalized learning is to facilitate academic success for each student by first defining the needs, interests, and aspirations of students, then continuing to provide personalized learning experiences for each of them. The aim of this thesis paper is to provide a genetic-based curriculum sequencing approach to generate personalized learning paths considering course difficulty level and the relation degree that exists between different concepts of that course. Also, the approach presented in this paper tends to use different population initialization methods and different genetic operators, so that we can compare the results in the end. Here, a brief introduction is given to the topic that will cover this thesis, while the rest of this thesis is structured as follows: chapter 2 describes in detail the problem statement on which this thesis is built and defines the objectives to be achieved, then continuing chapter 3, where we review the previous existing literature and describe some existing techniques, which are used to generate personalized learning pathways. Chapter 4 describes the methods and manner of research used to conduct both the research and experiment. The design and implementation description of the proposed approach takes place in chapter 5, while chapter 6 focuses on presenting the results of the experiment and deals with the evaluation of the results obtained using different methods of initialization, selection and recombination. This thesis ends with a conclusion regarding the proposed approach and the work that can be done in the future.

2. PROBLEM DECLARATION AND OBJECTIVES

Given the fact that in each course, a given sequence of topics is followed by a considerable number of students, neglecting their needs, abilities, and goals leads to a situation where not all students can benefit uniformly by following this course. Some of them fit this sequence quite well, thus achieving high results, but for others this is an obstacle, which directly affects the impossibility of achieving the course objectives. Students are individuals, and their ability to learn, knowledge, and performance differs substantially and changes constantly. These individual characteristics pose significant challenges when attending a course. This approach also causes difficulties for curriculum developers and professors, as it is impossible to design a curriculum that suits a huge number of students. One way to overcome this problem is to consider student requirements and then develop a curriculum driven by those requirements. However, such a thing is possible only in cases when the number of students who will attend the course is very small, while in other cases such an approach would be impossible. Having a huge number of students means that we have a huge number of requests from them as well, so the process of compiling a curriculum suitable for all, becomes complex and requires an extremely large amount of time.

Fortunately, with the use of process optimization algorithms, such as genetic algorithms, the process of compiling an individual sequence of topics, driven by the needs and abilities of students can be automated. Such an approach will compile a learning path, based on the individual characteristics of the students. This automation, as the main pillar will have a genetic algorithm, which will use data from individual characteristics of a student and will provide a personalized learning sequence, which is evaluated by an objective function. The objective function will be used by the genetic algorithm to evaluate the learning sequence and the tendency lies in the best possible optimization of this function by the algorithm. In this way the automatic approach will provide a learning sequence for each student, depending on their individual characteristics. The purpose of this thesis is to address the problem of providing personalized learning paths, using genetic algorithms, to develop an automated system, which will be able to provide personalized learning sequences, relying on in the

student's abilities, specifically in the level of difficulty of the topics of a course as well as the relation between them. The objectives of this thesis are as follows:

1. Review different approaches used to deliver personalized learning,
2. Build a genetic algorithm that considers the level of difficulty of course topics, duration or granularity, rating, and relation degree between them,
3. Implement different techniques of population initialization, selection and recombination,
4. Comparison of results from the use of different population initialization, selection and recombination techniques.

3. LITERATURE REVIEW

In this chapter, we will review the various methods used to generate personalized learning paths, as well as review the existing research literature and. In particular, the focus of this chapter will be on examining the use of genetic algorithms to provide personalization in learning paths. So, this chapter is organized into these separate sections:

i. Techniques used to provide personalized learning.
ii. Personalized learning path using Genetic Algorithm approach.

3.1 Techniques Used to Provide Personalized Learning

Lately researches, have experiment with several techniques to generate adaptive learning paths. Such techniques as Learning Path Graph, Ontology, Swarm Intelligence, Neural Networks, Bayesian Networks, Evolutionary Algorithms, to name a few.

Learning path graph [2] is a directed acyclic graph which represents all possible learning paths that matches the learning goal in hand. Student model consists of learner knowledge space, cognitive characteristics, and preferences. Based on the learner's attributes, a personalized learning path is selected from the graph which contains all learning paths. This path is selected using shortest path algorithm from the weighted graph, which is constructed using sustainability function to weight each connection of the LPG.

Ontology technique is capable to provide adaptive e-Learning environments and reusable learning resources. The ontology is formed through a group of abstract topics and semantic connection between them [3]. The topic that constitutes the knowledge of the treated field is collected in the concept class, which contains a data type property to identify a concept and other object properties that allow establishing different relations between domain topics. This way, the learning materials that best fit the learner's individual needs are selected through a comparison between student's learning style and learning style possibilities.

Swarm Intelligence is a type of Artificial Intelligence based on self-organized decentralized systems. The most popular algorithm known in Swarm Intelligence is Particle Swarm

Optimization. PSO is an evolutionary optimization algorithm, which tends to mimic the behavior of social insects like bees. Population particles are randomly initialized and share the information they gather during the flight through solution space. Each particle adjusts its velocity by observing the velocities of the neighboring particles and the currently known best solution, determined by a fitness function and this way each particle cooperates towards finding a solution [4]. As the number of swarms increases, the swarm will gradually move towards a global optimal solution. Student's competency records are created to determine course restrictions, and course metadata records are updated to reflect course learning outcomes and pre-requisites. Once these things are defined and the problem is established, PSO parameters are set to find out the best solution.

In **Neural Networks**, instead of generating individual learning pathways for each student, students are first grouped into different clusters, according to the learning style, and then a learning path that belongs to a particular cluster is generated. In this technique, an intelligent agent called Learning Assistant is used [5]. The role of this agent is two folded:

1. Train the neural network model, by classifying every student into different cluster, based on their preferences, expertise and learning style.
2. Generate an appropriate learning path for each cluster.

To classify similar students into different clusters, SOM (Self Organizing Map) is used. The first step of SOM is training. It is also called vector quantization, which is a very competitive process to build a map using input examples. Then mapping automatically classifies a new input vector (student) to a specific cluster. In the end a learning path is generated for each cluster and so, every student will have its own learning path according to the cluster in which he/she is located.

Ant Colony is another approach to provide personalized learning paths, taking into consideration student's profile. This technique is inspired by the natural phenomenon of ant's lifestyle. Ants construct networks of paths to link available food sources with their nests. This network is constructed in such a way, that the path from the nest to the food source is minimal, thus enabling ants to expend the minimum energy to bring the food into the nest. This

approach can be also used to optimize learning paths for different individuals, based on earlier learning paths followed by other students. As an extension of Ant Colony is Attribute-based Ant Colony System (AACS) [6]. It is a method based on the most frequent learning paths followed by previous students and then tends to find the most suitable learning path for the current student. Such a system updates learning paths from different levels of knowledge and learning styles to build a dynamic and powerful learning path search mechanism. To offer personalization in the learning paths through this approach, three prerequisites must be met:

1. Student's attributes, such as: knowledge level and learning style, must be known,
2. Course attributes, must be annotated from the professor or content provider,
3. Students and course should have a relationship in-between.

Bayesian Network [7, 8, 9] is a directed graph, whose nodes represents the discrete variables of interest and whose edges are the influential links between the nodes. Each variable is associated with a node probability table, which contains conditional probability values. The process of generating a learning path through Bayesian Network consists of two steps. In the first step, a node probability table is created based on BPT. This table indicates the probability of various subsequent nodes, which could be traversed from the actual node. Each node represents a candidate learning path and the probability value is assigned based on student's preferences, expertise and learning style. The second step is used to construct Bayesian Network, which calculates probability value for each unit in the learning path, then from this network the shortest path is selected as an appropriate learning path for a student.

Evolutionary algorithms are a group of algorithms based on the evolution theory of Charles Darwin [10]. The principle followed by these algorithms is known as survival of the fittest, which implies that environmental pressure causes natural selection in a population of individuals. As a class of general, random search heuristics, these can be applied to many different tasks. EAs tend to find optimal solutions within specific constraints by mimicking biological mechanisms, such as mutation, recombination, and natural selection.

Figure 1 describes how all evolutionary algorithms work. First, a real-life problem must be represented in a machine-readable form, then the algorithm starts working with a population whose individuals are randomly generated. From this population, each candidate solution is evaluated and selected according to its fitness and then their recombination is done to form optimal solutions.

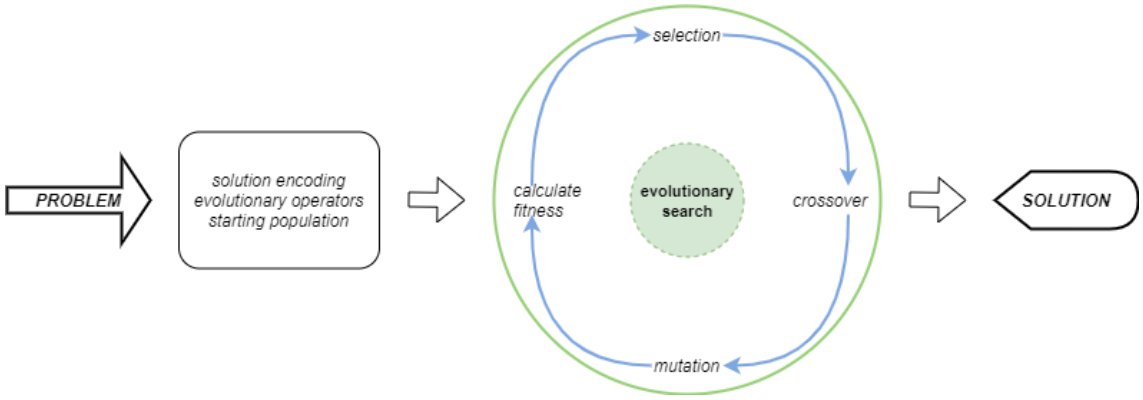

Figure 1 - The basic cycle of Evolutionary Algorithms

Genetic Algorithm as part of evolutionary algorithm is the main approach in our thesis, therefore in the following section will be explained in detail.

3.2 Personalized Learning Path using Genetic Algorithm approach

A number of machine learning approaches are applied for learning purpose, including inductive logic programming, decision tree, deductive reasoning, fuzzy logic, neural networks, and genetic algorithm. Inductive logic programming methods are hard to understand and implement, although they are based on predicate calculus and perform quite well. Decision tree methods apply the information gain theory to determine the attributes for constructing a minimal-attributes decision tree, but they cannot adequately handle exceptional cases. In the other hand, fuzzy logic is difficult to design a reasonable membership function and it is mostly used to clarify ambiguous situations. Learning environment patterns of neural networks are hard to explain because they are like being embedded in a black box and they have been used for a long time in financial problems. On the other side, a genetic algorithm (hereafter: GA) is usually used as an optimization technique to search the global optimum of a function and due to the huge search space, the

optimization problem is often very difficult to solve. GAs have shown a good performance a wide variety of problems. In [4] learning path sequence is modelled as a classical constraint satisfaction problem, and since then, research on usage of GAs to offer personalized learning has emerged. In addition, Jebari *et al.* (2011), in his research paper presented a solution based on GAs to find suitable dynamic personalized learning sequences of exercises, respecting the constraints, and maximizing success of the student [11]. To determine an optimal learning path based on GAs, different approaches have been explored in PLS. As can be observed from the reviewed paper [12], GA is applied to generate personalized learning paths based on CBR, considering curriculum difficulty level and continuity of successive curriculums. Here, authors have used a database to store course information along with the coefficient of difficulty. Then, based on the test results, the appropriate courses are selected, so that they require the lowest level of work coefficient. Finally, a GA generates an optimal individual training program for each student using the data obtained in the selection of courses ranked according to the work coefficient. Moreover in [13], authors applied GAs to propose personalized curriculum sequencing based on pre-test and pre-learning performance. The pre-test is based on incorrect test responses of each individual learner, which is performed at the start of the lesson to collect incorrect course concepts of learners through CAT, then the personalized mechanisms rely on the results of this pre-test. Furthermore, Bhaskar *et al.* (2010), used GAs to evolve learning path generation into a learning scheme generation. This learning scheme accommodates each student's entire context, including their profile, preferences and learning contexts. Depending on the learning goals and intentions of the student, the right content is selected, which could be of different types: presentation, media, content. So, this approach has considered the content of learning. The GA in this approach is used to select the type of content that best suits the needs and goals of a student [14]. In Spain, a team of researchers from the University of Alcala has conducted research on the dynamic selection of learning objects using GA to construct a learning path depending on the learner's competencies as an input and the planned learning outcomes as an output [15]. The problem with the approaches mentioned above, is that they do not generate personalized learning paths adapted to each individual student, but just orient the student to navigate through a range of learning activities [16]. In [17], a group of researchers from an education university in

Taiwan, have worked on constructing a personalized e-Learning system based on case-based reasoning approach and GA. Unlike most other works that consider only the interests, preferences and behaviors of students, the authors in their work [17], have considered the level of difficulty of the learning units and the ability of the student. Thus, their approach is based on the technique of evolution through computerized adaptive testing (CAT), then GA and case-based reasoning (CBR) are used to build an optimal learning path for each student. In another research paper [18], authors aim to construct an optimal individual educational trajectory, based on GA, being as close as possible to the features and real possibilities of each listener of the online course, with the possibility of real-time correction. In addition, they continue to formulate the problem, built into the structure of a massive open online course (MOOC), trying to provide solutions by classifying students into 4 groups, differing in the dominant learning style: visual learners, aural learners, read-write learners and kinesthetic learners and formulating a mathematical model. On this model the generation of individual learning route continues, applying the sequence of genetic algorithm steps to optimize the learning route. At the end, the obtained results are evaluated, and the generated solutions are analyzed against the optimal solutions. Moreover, Bellafkig *et al.* (2010), conceives an adaptive educational system based on the modeling of the description of pedagogical resources. In this paper, this raised problem is considered as a problem of optimization, and to build a sustainable solution the proposed system uses genetic algorithms to seek for an optimal path that starts from the student profile towards pedagogical objectives, passing through intermediate courses. The contribution of this paper is the construction of a system architecture, the adaptation of algorithms in this system and the modeling of a standard content module. The author through this paper targets important parts for the complete construction of an adaptive learning system, starting with the definition of the system architecture, which is generally divided into two parts: the student space where his identifiers fit, and the space of expert seeking the integration of new resources into the system. The work then continues with the adaptation module, which includes GA and adapter functionality, to proceed then with the implementation of the scenario in modeling and adaptation. The general structure of the architecture and adapters is presented with the help of diagrams, while the modeling part of the learning object is realized through the multi-part

standard, to facilitate research, evaluation, reuse, and acquisition, for example by students, instructors, or automated software. To meet the conditions of the presented approach, the author assumes that the student has a precise intellectual baggage, being able to understand a certain course and this set of knowledge is called as pre-requisite concepts. On the other hand, each course is accompanied by a certain set of knowledge to be gained after the course, and this was called as post-acquired concepts [19]. However, from the aforementioned research works, our work differs in the proposed approach of building a hybrid system to generate optimal learning paths considering factors such as: topic difficulty level, duration, rating, and relation degrees of topics, and it aims to contribute on increasing the quality of learning path generated, as well as increasing the performance of the system by developing different techniques for population initialization, selection and recombination, which will be discussed further in the following sections.

4. METHODOLOGY

In this chapter, is described the approach of conducting this research and the processes followed to model the experiment, which is the most important part of this thesis. Before deciding on how to structure and carry out the work, we have reviewed the existing literature in this domain and summarized all the findings in a survey paper, which then served as an important input for the expansion of research and experiment modeling. The survey paper included theoretical aspects and the provision of various ways to provide personalization of learning path (hereafter: PLP), but the main energy was focused on examining GA as a new and very fruitful technique in marking the possibility of use in providing personalization during the learning process. During the review of dozens of papers on the use of GA for PLP generation, a similar flow of research and steps followed by the authors was observed.

Being part of EA, GA before starting the process of searching for optimal solutions from the search space, must have defined two things, which are also the basic requirements of a GA:

1. Solution Encoding - Indicates the representation of a solution within the computation space. This is a process of transforming a solution from the phenotype (representation of actual real-world solution) to genotype space (representation of a solution in computation space that can be easily understood and manipulated using a computing system) [20]. This process is illustrated in Figure 2.

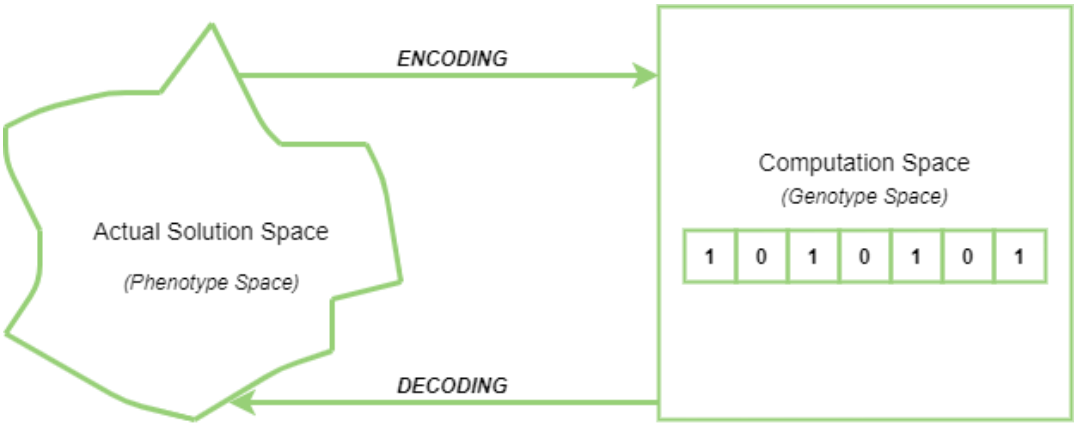

Figure 2 - The process of encoding and decoding a solution

2. Fitness Function - Determines which solutions within a population are better at solving the problem under consideration. This function simply takes a solution as input and indicates how suitable that solution is regarding the studied problem, and it should be highly optimized and must evaluate the quality of a solution in quantitative values.

Immediately after the solution representation and fitness function are defined, GA begins to initialize a population of solutions randomly, or using other combined approaches. Then the population improvement it is done by repeatedly applying recombination and selection operators. Figure 3 describes the process that need to follow in GA.

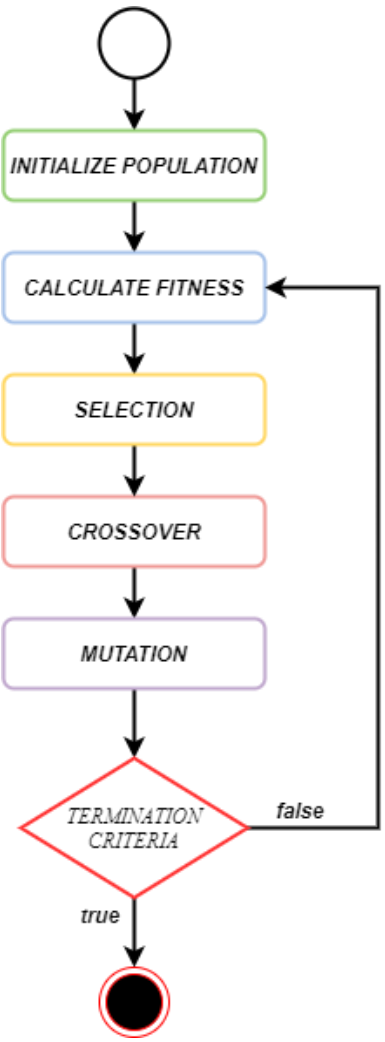

Figure 3 – Process flow of GA

After each iteration, a new generation of population with individuals is formed and in each generation the fitness of each solution is evaluated. The process of creating new generations and evaluating solutions is repeated until a termination condition is met. The general steps to follow when applying GA in optimizing a particular problem are: Population Initialization, Fitness Calculation, Selection, Crossover and Mutation [21]. In the following these steps will be described at an abstract level, while in the next chapter they will be explained at the level of details when describing the implementation structure of the approach offered.

The initial step in the GA process is Population Initialization. Population can be defined in different ways; however, it represents a subset of solutions in a current generation. These constraints should be considered during this step:

- The size of the population should not be too large because it affects the speed of GA, but at the same time it should not be too small as it affects the insufficiency of a good mating pool. So, the size of the population can be decided by trial and error.
- The population must have diversity, to avoid premature convergence.

The main known methods for creating the initial population are:

- Random Initialization - The solutions that make up the initial population are completely random generated solutions.
- Heuristic Initialization - The generation of the initial population is realized with the use of any known heuristic for the problem.

The initialization of the whole population is not suggested to be done totally with the use of any of the heuristics, because this approach can result in the reduction of diversity. Meanwhile, it has been noticed that random solutions are the ones that lead to optimality.

Once the initial population is formed, fitness calculation can be applied to the current population. From this population the solution that fits most is taken as the current optimal solution and placed again in the new generation ensuring the elitism of the algorithm. The definition of the function for calculating the quality of a solution is given in the next chapter.

The next step is the selection of solutions from the population known as parent solutions, in which operators will be applied for their recombination and the creation of new solutions aiming at the evolution of genes. This step is based on Charles Darwin's theory of evolution, which is known as “survival of the fittest” [22]. Different selection policies have been developed over time, but the most popular are:

1. Roulette Wheel Selection – A circular wheel is divided into n pies, assigned to each individual solution, where the portion of the circle is proportional to its fitness value. A fixed point is chosen on the circle circumference and after the wheel is rotated, the region of the circle that comes in front of the fixed point is selected. From this, we can observe that the solution with the highest fitness value, has a higher probability to be selected, due to the size of the portion in the circle.
2. Tournament Selection – From the entire population, we randomly select k individuals and from these solutions, we select the best one to be as a parent solution according to its fitness value.

After selecting the parent solutions, recombination operators are applied to these old solutions, to create new individuals from old ones. These operators are divided into two groups based on their arity, unary and n-ary.

Mutation is considered as a unary variation operator, and it is applied to a single genotype to form a new offspring, with slight modifications from the parent solution. The output from the application of the mutation to a genotype depends on the outcomes of a series of random choices, keeping this operator always stochastic. Meanwhile, crossover is part of binary variation operator. From the name itself, we note that the purpose of this operator is to combine features from two genotypes into one or two new off-springs. Both operators are applied based on a probability, the mutation is usually applied with a lower probability than the crossover, because a high probability of mutation, would cause the genetic algorithm to get stuck in a random search. Mutation contributes to the exploration of the search space and it is the essential operator for GA convergence, while crossover is related to the exploitation of the search space [23].

Once the GA has gone through all these steps, it must be assessed whether the repetition of these steps should continue, or whether GA has reached the moment of stopping this process. To determine this, the GA termination condition is used. The following conditions are usually used to assess whether GA should be stopped [24]:

- An absolute number of generations is reached,
- Fitness of an individual solution has reached a certain pre-defined value,
- The fitness improvement remains under a threshold value for a given period (i.e., number of generations or fitness evaluations).

The workflow explained above is a common pattern observed during the literature review, and as such can be followed to conduct the research and design the experiment in our case, with some necessary minor changes. In the next chapter, we will describe in detail the design and implementation of our approach, utilizing GA in providing personalized learning paths. All the above steps are an integral part of the structure of a GA and as such should be followed, to provide an approach based on these algorithms.

5. DESIGN AND IMPLEMENTATION OF EXPERIMENT

In this chapter, we will present our GA-based solution for providing personalization in learning paths. Furthermore, we will introduce and explain the design of the experiment and the methods used. Initially, we will design a system flowchart on an abstract level, which will then serve as a reference point for the following sections of this chapter and will encapsulate the entire process as well as the steps the system will go through. Then, we will explain in detail each part of the system, starting with solution modeling and encoding, recombination operators, selection operators and fitness calculation. In the end, the conducted experiment and the configuration parameters will be described.

5.1 System Design

The proposed approach in this paper, combines different heuristics of population initialization, selection methods, recombination operators, to develop a hybrid approach based on GA, to optimize generation of personalized learning paths according to level difficulty and relation degree of courseware concepts. The whole process in the system is presented visually through a flowchart shown in the Figure 4. As you can see in this figure, the process begins with defining methods for population initialization, selection and reproduction, as we have implemented different methods, so that in the end we have results to compare depending on the methods used.

This process is iterative, and the number of iterations is defined as the configuration parameter before the algorithm starts working. We will discuss the configuration parameters and their values later in a different section. In each iteration the evaluation of the optimal solution of the current iteration (generation) is done and then this optimal solution of the iteration is compared with the preliminary solution, which represents the best solution found so far. In case the current solution is better than the previous one, then the current solution becomes the best solution so far. Thus, the approach offers elitism, as it always maintains the best solution found so far.

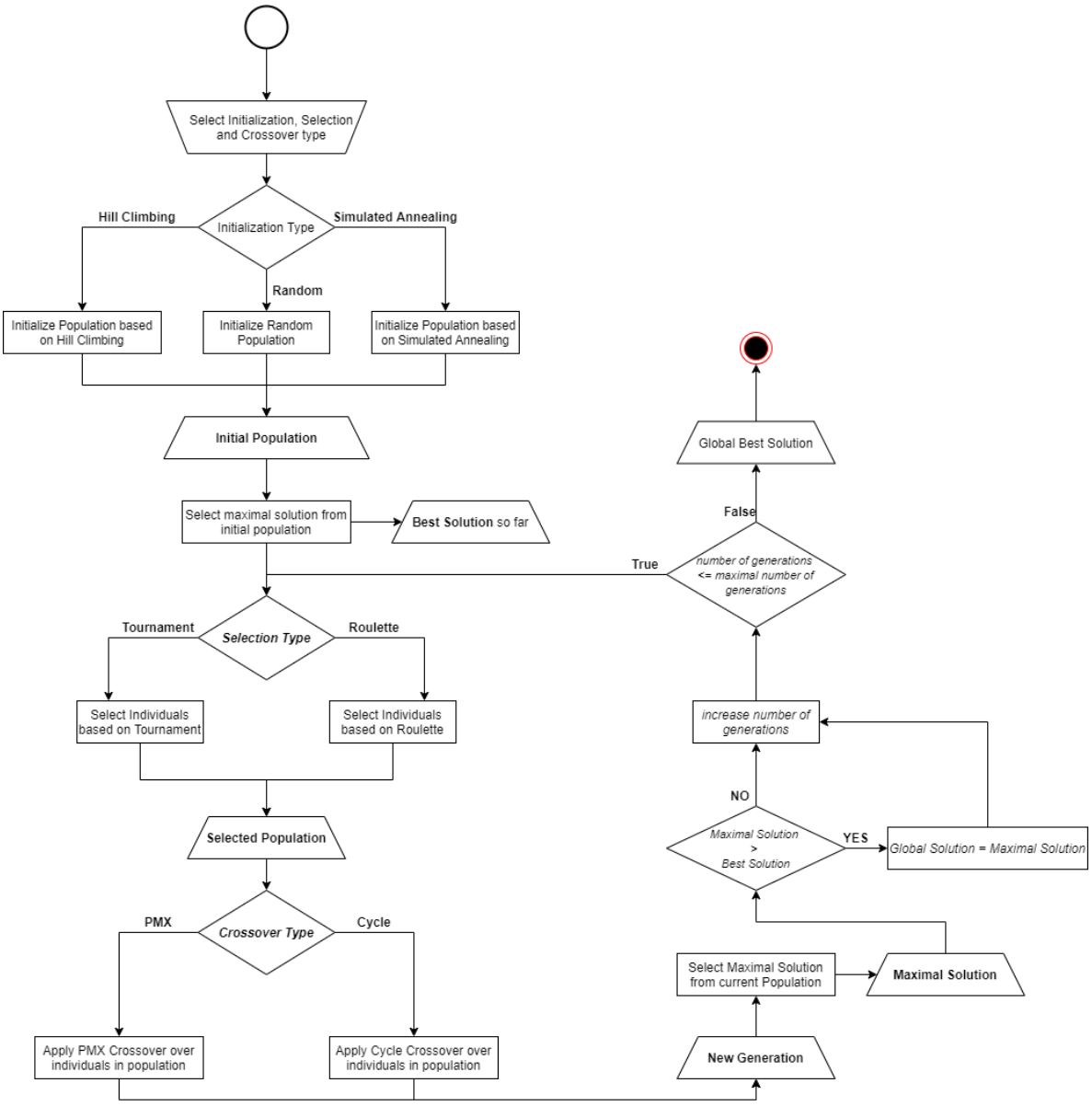

Figure 4 - System Flowchart

Once the number of predefined iterations reaches the maximum, the solution found by switching to all cycles will be the output of the system. This solution will represent the optimal learning path for a student to go through.

5.2 Solution Modeling and Encoding

To model an individual solution to the system, we used integer-based encoding because this way of encoding is easier to apply then recombination and selection operators. An integer

number is assigned to each concept from 0 to n - 1, where n is the maximum number of courseware concepts. We developed a class, that presents a learning path (solution), and it has an ArrayList object, where each position contains an integer value, which is the ID of a specific concept. The object containing the ArrayList will be used to model the learning path for the GA, and it will be used to form the population. Figure 5 contains a visual description of the solution modeling, where in each cell is the ID number that determines each concept in a courseware.

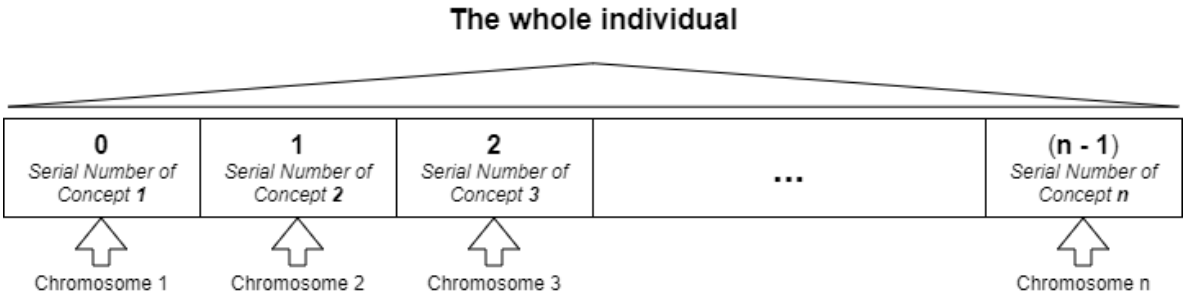

Figure 5 - Solution Encoding

The structure of the solution presented on Figure 5 determines a sequence of interrelated concepts, and as such is evaluated by an objective function, to judge its quality based on the level of difficulty of each concept and the relation degree with the next concept.

5.3 Recombination Operators

Since GAs apply techniques for combining the chromosomes of different individuals, with the sole purpose of creating young individuals that are likely to have higher quality than current individuals, we have developed these techniques to use in the system: ***SWAP Mutation*** – This is a technique used for mutation, which makes chromosome changes in a single individual. This is a simple technique to change an individual's genetic material and is applied through these two simple steps:

1. Select two positions of chromosomes,
2. Swap their values.

The figure below shows the swap mutation application process, which consists of these two steps above.

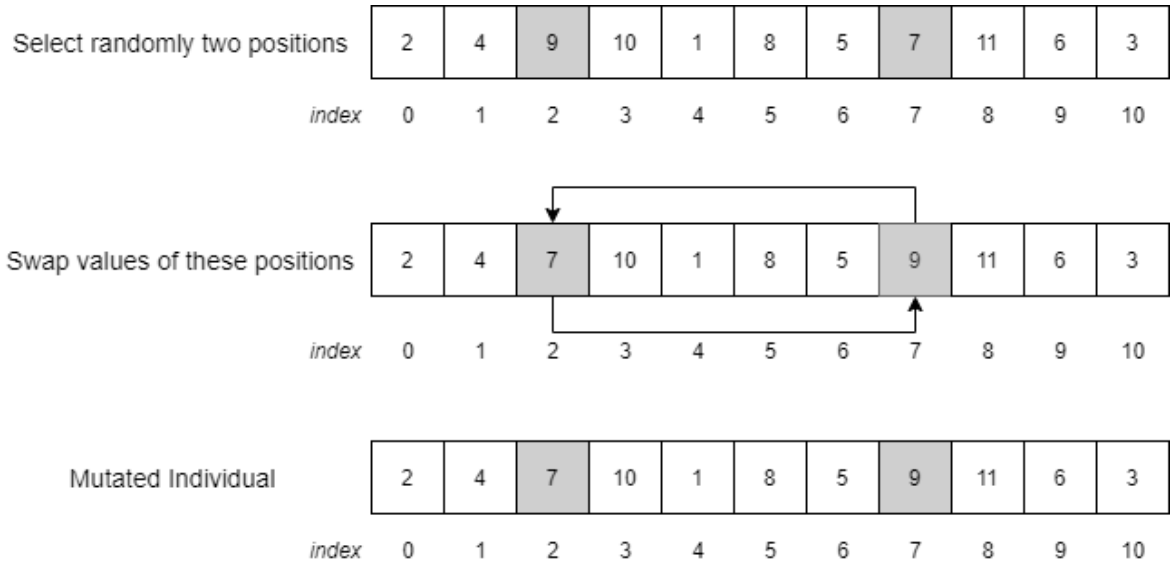

Figure 6 - SWAP Mutation

While the code developed for the application of SWAP mutation in the algorithm is shown in the code snippet below.

```
public LearningPath SWAPmutate(LearningPath lp) {
    Random random = new Random();
    float mutate = random.nextFloat();
    if (mutate < this.mutationRate) {
        List<Integer> mutatedLearningPat = lp.getSolution();
        int firstIndex = random.nextInt(this.learningPathSize)
        int secondIndex = random.nextInt(this.learningPathSize)
        Collections.swap(mutatedLearningPath, firstIndex, secondIndex);
        return new LearningPath(mutatedLearningPath);
    } else {
        return lp;
    }
}
```

Code Snippet 1 - SWAP Mutation

While as crossover operators, we have developed PMX and CYCLE crossover, as the most advanced techniques for applying the crossover to individuals of the current population to create new individuals.

PMX – Partially-Mapped Crossover for some problems offers better performance than most other crossover techniques. It builds an offspring by choosing a subsequence from one parent and preserving the order and position of as many alleles as possible from the other parent. The subsequence is selected by choosing two random cut points, which serve as boundaries for the swapping operations. The steps we have followed to perform this operation are:

1. Randomly select a subsequence from parent 1 and copy them direct to the child. Keep note the indexes of this segment.
2. Select each value which is not already been copy to the child looking in the same segment positions in parent 2.
    a. For each of these values:
        i. Note the index of this value in parent 2. From parent 1 in the same position locate this value.
        ii. Locate the same value from parent 2.
        iii. If the index of this value in parent 2 is part of the subsequence, using this value go to step i.
        iv. Else insert step a's value into the child to this position.
3. Copy the remaining positions from parent 2 to the child if there is any.

We use this function by sending two solutions from the population, which are selected randomly and are known as parents, and then in these parents the process described above is applied, thus creating two new solutions, which are placed in the population and are known as children. Figure 7 visually presents this process and visually describes the steps we have followed to achieve the generation of child solutions.

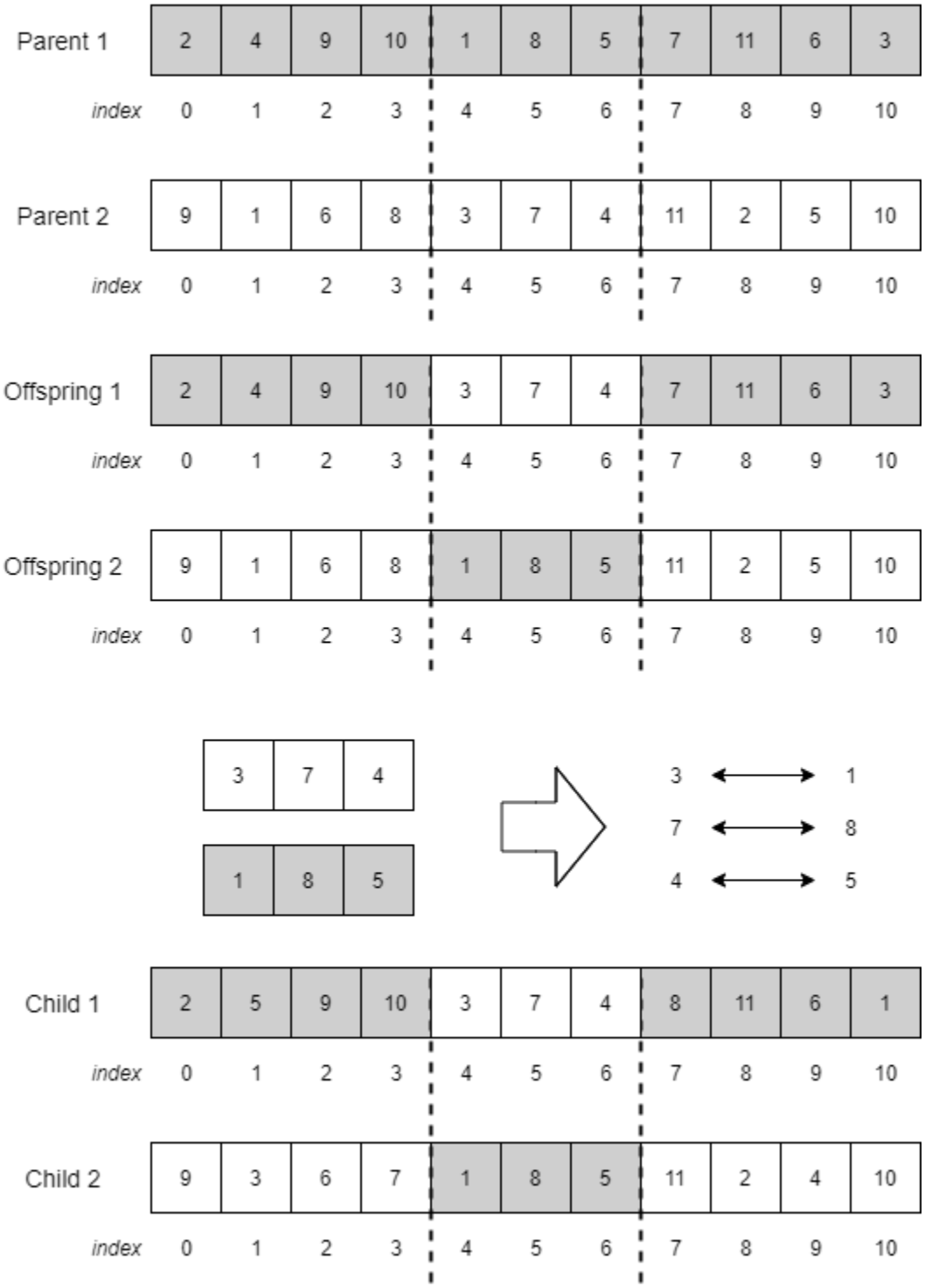

Figure 7 - PMX Crossover

Below is also the code developed for the recombination of solutions through PMX.

```
public List<LearningPath> PMXcrossover(List<LearningPath> parents) {

    Random random = new Random();
    int breakpoint = random.nextInt(this.learningPathSize);
    List<LearningPath> children = new ArrayList();
    List<Integer> p1 = new ArrayList(parents.get(0).getSolution());
    List<Integer> p2 = new ArrayList(parents.get(1).getSolution());
    int i;
    int newVal;

    for (i = 0; i < breakpoint; ++i) {
        newVal = (Integer) p2.get(i);
        Collections.swap(p1, p1.indexOf(newVal), i);
    }

    children.add(new LearningPath(p1));
    List<Integer> parents1Genome = parents.get(0).getSolution();

    for (i = breakpoint; i < this.learningPathSize; ++i) {
        newVal = parents1Genome.get(i);
        Collections.swap(p2, p2.indexOf(newVal), i);
    }

    children.add(new LearningPath(p2));
    return children;

}
```

Code Snippet 2 - PMX Crossover

CYCLE – Cycle Crossover is the other genetic operator we have used in our approach. It attempts to create children from the parents where every position is occupied by a corresponding chromosome from one of the parents. As the name itself indicates, this operator is based on closed loops formed by elements from two individuals selected by the population. To implement cycle crossover into our proposed algorithm, we have followed these steps:

1. Create a cycle of chromosomes from first parent, in the following way:
    a. Start with the first chromosome of first parent.
    b. Check at the chromosome with at the same position in second parent.
    c. Go to the position with the same chromosome in the first parent.
    d. Add this chromosome to the cycle.

e. Repeat steps b to d until arrival at the first chromosome of first parent.

2. Put the chromosomes of the cycle in the first child on the positions they have in the first parent.
3. Take next cycle from the second parent.

These steps are also presented visually in Figure 8, whereas the code we developed to use this operator in the system is shown in Code Snippet 3.

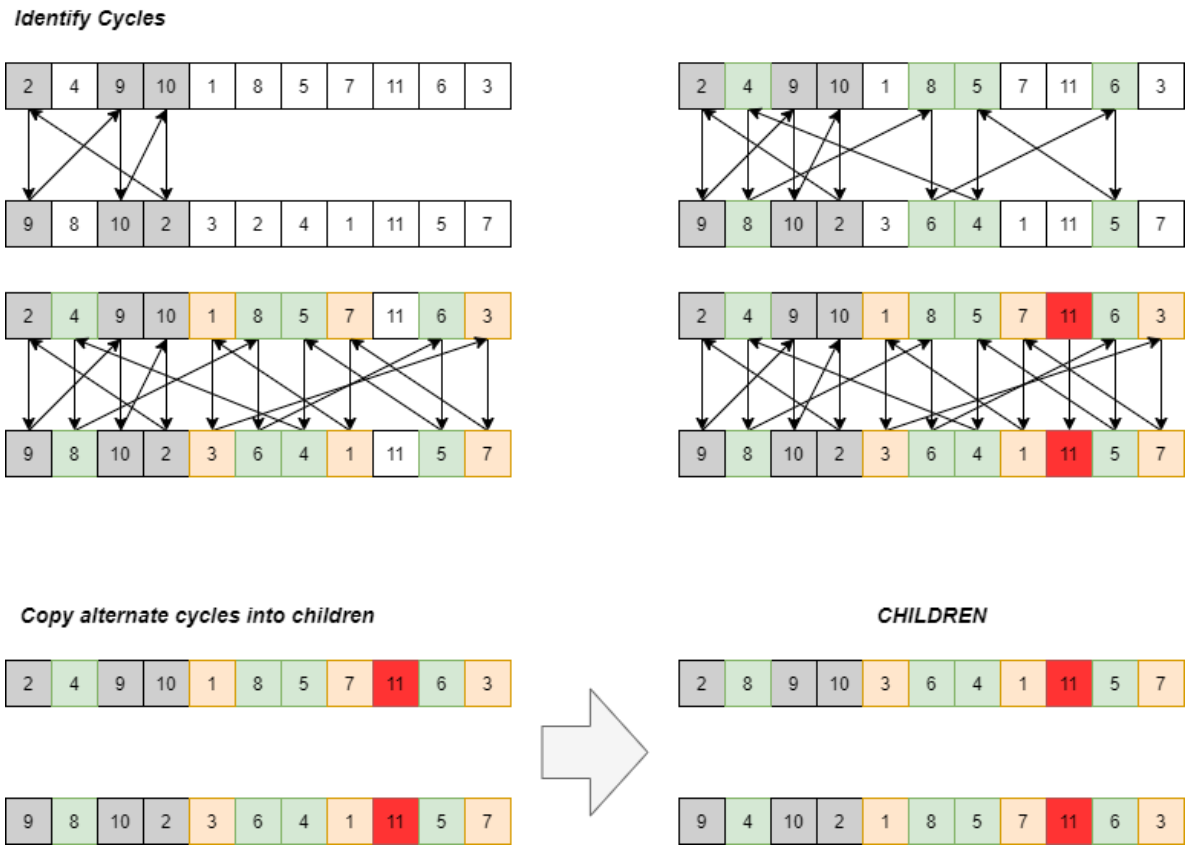

Figure 8 - CYCLE Crossover

The process of finding the cycles is described in the upper part of the figure and at the end each formed cycle is filled with a color to distinguish it from the other cycles. Meanwhile, the bottom part illustrates the creation of young individuals (children) by placing the parts of the cycle in the positions they have in the first parent, while the next cycle is copied from the second parent.

```
public List<LearningPath> CYCLECcrossover(List<LearningPath> parents){

    List<Integer> parent1=new ArrayList(parents.get(0).getSolution());
    List<Integer> parent2=new ArrayList(parents.get(1).getSolution());
    LearningPath lp1 = new LearningPath();
    LearningPath lp2 = new LearningPath();
    final int length = parent1.size();
    List<LearningPath> children = new ArrayList<LearningPath>(2);
    List<Integer> child1 = parent1;
    List<Integer> child2 = parent2;
    Set<Integer> visitedIndices = new HashSet<Integer>(length);
    List<Integer> indices = new ArrayList<Integer>(length);
    int idx = 0;
    int cycle = 1;

    while (visited.size() < length) {
        indices.add(idx);
        int item = parent2.get(idx);
        idx = parent1.indexOf(item);
        while (idx != indices.get(0)) {
            indices.add(idx);
            item = parent2.get(idx);
            idx = parent1.indexOf(item);
        }
        if (cycle++ % 2 != 0) {
            for (int i : indices) {
                int tmp = child1.get(i);
                child1.set(i, child2.get(i));
                child2.set(i, tmp);
            }
        }

        visited.addAll(indices);
        idx = (indices.get(0) + 1) % length;

        while(visited.contains(idx) && visited.size() < length){
            idx++;
            if (idx >= length) {
                idx = 0;
            }
        }
        indices.clear();
        lp1.setSolution(child1);
        lp2.setSolution(child2);
    }

    children.add(lp1);
    children.add(lp2);
    return children;
}
```

Code Snippet 3 - CYCLE Crossover

5.4 Selection Policies

This is a process of selecting individuals from population as parents to recombine and create off-springs for the next generation. It is very crucial for the convergence rate of GA and fittest parents drive to better solutions. For a high success of GA maintaining good diversity over population is extremely crucial, and one undesirable condition is premature convergence. Regarding selection methods to select learning paths from the population, then apply recombination operators to them, for the experimental phase we have developed these policies:

Tournament Selection – this method selects the fittest candidates from the current generation and passes them to the next generation. The selection of some potential candidates happens randomly, then the candidate with the highest fitness is selected from them. The steps we have followed to build this selection method are:

1. Select k elements from the population.
2. Select the best individual from the k elements in step 1.

Implementation of Tournament Selection into our algorithm is like this:

```
public LearningPath tournamentSelection(List<LearningPath> pop){
    List<LearningPath> selected = pickNRandomElements(pop, this.tS);
    return (LearningPath) Collections.max(selected);
}
```

Code Snippet 4 - Tournament Selection

The best individual selected becomes parent, on which the recombination operators will be applied. In case we need to have two parents to apply the recombination, as in the case of crossover, then the process is repeated.

Roulette Selection – the other method we used to select individuals from population, is Roulette Selection. The process followed to apply this selection approach is described in chapter 4, while our implementation is presented in Code Snippet 5. This function calculates the sum of all fitness values, then for each candidate in the population determines the probability of being selected. In the end the partial sum of individuals in population is calculated and the individual for which partial sum exceeds total sum is chosen.

```
public LearningPath rouletteSelection(List<LearningPath> pop){
    double tS = 0.0;
    LearningPath lp = null;
    for (int i = 0; i < pop.size(); i++) {
        totalSum += pop.get(i).getFitness();
    }
    for (int i = 0; i < pop.size(); i++) {
        pop.get(i).setLearningPathProb(tS, pop.get(i).getFitness());
    }
    double partialSum = 0;
    double roulette = (double) (Math.random());
    for (int i = 0; i < pop.size(); i++) {
        partialSum += pop.get(i).learningPathProbability;
        if (partialSum >= roulette) {
            lp = pop.get(i);
            break;
        }
    }
    return lp;
}
```

Code Snippet 5 - Roulette Selection

5.5 Fitness Function Selection

To generate personalized learning path for individual learners based on their pretest results, difficulty level, courseware concepts relation degrees, rating and concept granularity are considered in our method. To evaluate how fit and close a generated solution is to the

optimum solution of personalized learning path generation problem, we mathematically modeled the fitness function as follows:

$$f = \sum_{i=2}^{n} \left((g_i / r_i) * (1 - w) * rd_{(i-1)i} + w * (1 - d_i) \right) \quad (1)$$

where f is the fitness function used by GA to evaluate each solution, g_i is the concept duration or granularity, r_i is the concept rating, telling how useful was the particular concept for that particular learner, $rd_{(i-1)i}$ represents the (i - 1)th courseware concept relation degree with the ith concept in the learning path, d_i is the difficulty parameter of the ith concept, n stands for the total number of concepts that build a courseware, while w is an adjustable weight. The code written to model the fitness function formulated above is as follows. RDM stands for Relation Degree Matrix, which is a two-dimensional array containing relation degrees of each courseware concept with others.

```
private double calculateFitness() {

    double fitness = 0;

    for (int i = 1; i < solution.size(); i++) {

        Concept c = course.getConcept(solution.get(i));

        Concept pC = course.getConcept(solution.get(i - 1));

        fitness += (c.getGranularity() / c.getRating()) *
               (w * RDM[c.getID()][pC.getID()]) + ((1 - w) * c.getDifficulty());

    }

    return (double)Math.round(fitness * 100) / 100;

}
```

Code Snippet 6 - Fitness Function

5.6 Population Initialization

There are several approaches and heuristic to initialize the population used by GA to start the process of finding optimal solutions from it. The simplest and most used approach in GA

applications for population initialization is to randomly generate individuals and then put them in the population until a limited number of defined population size is reached. This is a dumb idea, but for most problems it works very well. Another way to initialize the population, is starting the process with solutions that have been previously evaluated and have a higher affinity than randomly generated solutions. For the experimental phase we have deployed three ways for population initialization, randomly, Hill Climbing and Simulated Annealing. We will compare these approaches in the end to test which one is providing better starting input for personalized learning paths. The code to randomly initialize population is given below and as can be seen it is very simple. The first method is used to instantize learning paths according to the population size, whereas the second method is used internally by LearningPath class model to create a list of integers, which represents the path of concepts to follow.

```
public List<LearningPath> initialPopulation(){
    List<LearningPath> population = new ArrayList();
    for (int i = 0; i < this.populationSize; ++i){
        population.add(new LearningPath());
    }
    return population;
}

private List<Integer> randomLearningPath(){
    List<Integer> result = new ArrayList<Integer>();
    for (int i = 0; i < numberOfConcepts; i++){
        result.add(i);
    }
    Collections.shuffle(result);
    return result;
}
```

Code Snippet 7 - Randomly Population Initialization

Hill Climbing – is a local search algorithm which continuously moves towards of increasing fitness value to find the peak of the mountain that also is the best solution for the problem. The condition to terminate is when it reaches a peak where no neighbor has a higher value. The process we followed to build this algorithm and use it into our approach is:

1. Generate a randomly solution.
2. Loop until the number of predefined iterations is reached.
3. For each iteration get the neighbor of the randomly initialized solution.
    a. If the neighbor fitness value is better than the randomly solution, then replace it with the neighbor solution.
    b. Else go to the step 2.
4. Add the remaining solution from above steps to the population.
5. Repeat all the steps above until we reach the population size.

The steps above are also reflected in the code below, while the part to find the neighbor of a particular solution is given in Code Snippet 7.

```
public List<LearningPath> population(){

    List<LearningPath> initPop = new ArrayList<LearningPath>();

    for (int i = 0; i < ConfigurationParameters.populationSize; i++){

        LearningPath lP = new LearningPath(randomLearningPath());

        LearningPath nLP;

        for (int j = 0; j < iterationsBeforeMaxima; j++) {

            nLP = getNeighbor(lP, numberOfConcepts);

            if (nLP.getFitness() > lP.getFitness()) {
                lP = nLP;
            }
        }

        initPop.add(lP);
    }

    return initPop;
}
```

Code Snippet 8 - Hill Climbing based Population Initialization

```
private LearningPath getNeighbor(LearningPath lp, int nrOfConcepts){

    int x1 = 0, x2 = 0;
    boolean cond1 = (x1 == x2);
    boolean cond2 = (x1 >= lp.getSolution().size());
    boolean cond3 = (x2 >= lp.getSolution().size());

    while (cond1 || cond2 || cond3) {
        x1 = (int) (Math.random() * nrOfConcepts);
        x2 = (int) (Math.random() * nrOfConcepts);
    }

    int number1 = lp.getSolution().get(x1);
    int number2 = lp.getSolution().get(x2);
    lp.getSolution().set(x1, number1);
    lp.getSolution().set(x2, number2);

    return lp;
}
```

Code Snippet 9 - Neighbor Generation from Current Solution

Simulated Annealing – is an algorithm used for optimizing parameters in a model by imitating the Physical Annealing process. It is almost the same as Hill Climbing, but instead of picking the best move, it picks a random one and if the selected move improves the solution, then it is accepted, if not the algorithm makes the move anyway with a probability less than 1. This probability decreases exponentially as bad as the movement is. The mathematical model to calculate the acceptance probability is:

$$P = exp(\frac{e - e_n}{T}) \quad (2)$$

Where P is the acceptance probability, e is the current energy, e_n is the new energy, whereas T is the parameter for controlling the annealing process. Implementation of this function is given below:

```
public static double accProbability(double e, double nE, double T){

    if (e < nE) {
        return 1.0;
    }
    return Math.exp((e - nE) / T);
}
```

Code Snippet 10 - Acceptance Probability Function

5.6 Data Collection and Preparation

The dataset used for the development of the experiment is taken in the XML format from the author of paper [25], which consists of learning objects within the Java course. From these data we extract a number of detailed information with respect to each learning object, such as: title, description, next learning object, relation degree with the next learning object, duration or granularity, difficulty value, rating by individual learner, overall rating, and other information, but the information that the objective function of the algorithm will accept as input for evaluating a solution are granularity, individual rating, difficulty, and relation degree of learning objects.

Since the algorithm needs the relation degree values for each learning object with the others, then these values are prepared in an excel sheet, from where the algorithm initially reads and stores them into a two-dimensional array. Figure 10 presents a screenshot of these values from the excel spreadsheet.

r i,j	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
C0	1	0.985	0.982	0.864	0.731	0.829	0.816	0.716	0.501	0.373	0.294	0.683	0.674
C1	0.985	1	0.23	0.411	0.275	0.497	0.943	0.387	0.669	0.185	0.912	0.547	0.537
C2	0.982	0.23	1	0.75	0.487	0.215	0.699	0.156	0.714	0.755	0.354	0.503	0.393
C3	0.864	0.411	0.75	1	0.702	0.38	0.148	0.618	0.144	0.237	0.487	0.849	0.649
C4	0.731	0.275	0.487	0.702	1	0.433	0.119	0.243	0.904	0.653	0.204	0.093	0.431
C5	0.829	0.497	0.215	0.38	0.433	1	0.369	0.95	0.399	0.438	0.855	0.954	0.701
C6	0.816	0.943	0.699	0.148	0.119	0.369	1	0.521	0.687	0.72	0.957	0.422	0.583
C7	0.716	0.387	0.156	0.618	0.243	0.95	0.521	1	0.348	0.949	0.665	0.921	0.559
C8	0.501	0.669	0.714	0.144	0.904	0.399	0.687	0.348	1	0.476	0.537	0.785	0.496
C9	0.373	0.185	0.755	0.237	0.653	0.438	0.72	0.949	0.476	1	0.692	0.256	0.762
C10	0.294	0.912	0.354	0.487	0.204	0.855	0.957	0.665	0.537	0.692	1	0.591	0.814
C11	0.683	0.547	0.503	0.849	0.093	0.954	0.422	0.921	0.785	0.256	0.591	1	0.907
C12	0.674	0.537	0.393	0.649	0.431	0.701	0.583	0.559	0.496	0.762	0.814	0.907	1

Figure 9 - Learning Object Relation Degrees

5.7 Experiment and Configuration Parameters

Once all the definitions for building a GA have been achieved, designed, and implemented, and on the other hand the data has been collected and prepared for use, we managed to develop the experimental part according to the description of our approach shown in the

chapters above using Java programming language and Intellij IDE. The purpose of this experiment is to evaluate the built-in approach and analyze the performance of possible combinations of genetic operators, selection and initialization methods. In addition, another reason for the experimental phase is to show the possibility that such an approach can be used in providing personalized learning paths, to increase the performance and efficiency of students on individual basis. The project structure after the final development of the algorithm is shown in Figure 11.

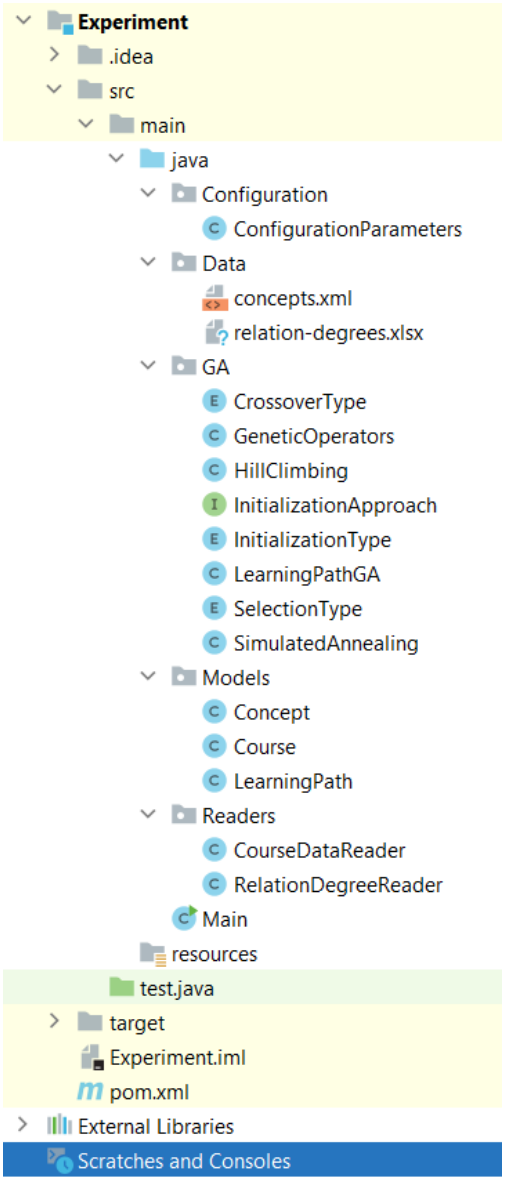

Figure 10 - Project Structure

In the following, a brief description of each part of the project structure will be given.

Configuration Parameters

- *ConfigurationParameters* – Java class containing configurations needed for the GA.

Data

- *concepts.xml* – learning objects information, such as: granularity, difficulty, and rating.
- *relation-degrees.xlsx* – relation degrees of each learning object with others.

GA

- *CrossoverType* – Java enumeration type to determine which crossover method will be applied.
- *GeneticOperators* – implementation of genetic operators.
- *HillClimbing* – population initialization implementation based on Hill Climbing.
- *InitializationApproach* – Java interface to define which method signature will each initialization approach override.
- *InitializationType* – Java enumeration type to specify which population initialization method will be used.
- *LearningPathGA* – Java class used to connect all other objects and maintain the GA process flow.
- *SelectionType* – Java enumeration type to define which selection policy will be used.
- *SimulatedAnnealing* – population initialization implementation based on Simulated Annealing.

Models

- *Concept* – Java class to model a single concept or learning object.
- *Course* – Java class to model the entire course.
- *LearningPath* – Java class to model a sequence of concepts and calculate its fitness.

Readers

- *CourseDataReader* – Java class to read data from xml format and create a course with all information needed.
- *RelationDegreeReader* – Java class to read relation degrees of each learning object and create a two-dimensional array.

✓ ***Main*** – Java class containing the entry point method where the program starts running.

Defining and changing the working parameters of the algorithm can be done in *ConfigurationParameters* class, whereas the configuration used to evaluate the performance of possible combinations is given in the table below.

Table 1 - Configuration Parameters

Parameter	Value	Usage
Number of generations	150	*LearningPathGA*
Population Size	20	*LearningPathGA, HillClimbing, SimulatedAnnealing*
Reproduction Size	10	*LearningPathGA*
Tournament Size	8	*LearningPathGA*
Mutation Rate	0.1	*LearningPathGA, GeneticOperators*
Crossover Rate	0.9	*LearningPathGA, GeneticOperators*
Target Fitness	500	*LearningPathGA*
Adjustable Weight	0.5963	*LearningPath*

These values are set according to the guidelines for achieving the best performance of the genetic algorithm and based on the goal of optimizing the problem raised. This configuration can be changed, and a new configuration can be used, depending on the specifics and features on which the evaluation of a learning path is made. Based on our objective function as well as the methods developed for recombination, solution selection and population initialization, this configuration is extremely suitable for our case.

6. RESULTS AND DISCUSSION

This chapter provides the simulation results and observations obtained while performing the experiment of using our proposed GA based approach on the data presented in chapter 5. The course unit “Control Flow and Conditionals” from “Introduction to Java Programming” course, is used to generate personalized learning paths, including different levels of difficulty, granularity, rating, and relation degrees between course concepts. The learning path for this course unit, that a student would follow in the traditional way of lecturing would be listed as in the table below, and its fitness value is 41.68.

Table 2 - Learning Path Sequence

ID	Title	Difficulty	Granularity	Rating	Relation Degree
0	Introducing Control Flow	1	5	6	0.985
1	Decision Making	1.65	10	7	0.982
2	If Statement	2.15	12.5	6.5	0.864
3	Variable Scope	2.35	15	5.5	0.731
4	Else Statement	2.95	15	7	0.829
5	Else If	3.15	13	8	0.816
6	Multiple Else Ifs	3.95	18	6.5	0.716
7	Boolean Expressions	2.75	8	7.5	0.501
8	Logical Operators	2.15	10	7	0.373
9	Logical Operators Practice	2.25	7.5	8	0.294
10	Nested If Statements	4.55	20	8.5	0.683
11	Switch Statement	4.15	20	8.5	0.674
12	Conclusion	1.85	14	9	-

Assume that the student initially before starting to attend this unit performs a pretest and answers incorrectly in all learning objectives, and Figure 10 illustrates the concept relation degrees of corresponding concept that student gives incorrect answers, then based on the

relation degrees table and additional data related to a learning object extracted from XML format, the modeled algorithm can construct high quality personalized paths according to the designed objective function. Improving the quality of learning paths occurs throughout the evolution of generations, but to observe the trend of quality improvement we will compare all possible combinations of recombination operators, selection techniques and initialization approaches, that we have developed as part of this experiment. The results of each combination used from the algorithm is presented in the table below, together with best performing learning sequence, first fitness value found in the first generation and best fitness value after reaching 150 generations as defined in algorithm configuration parameters.

Table 3 – Results of Possible Combinations

Selection Policy	Population Initialization Approach	Crossover Type	First Fitness	Last Fitness	Learning Path Sequence
Tournament	Random	PMX	41.59	45.14	[0, 2, 9, 7, 12, 1, 6, 10, 5, 11, 3, 4, 8]
Tournament	Random	CYCLE	41.63	45.47	[0, 2, 3, 4, 8, 1, 6, 10, 12, 11, 5, 7, 9]
Tournament	Hill Climbing	PMX	42.13	44.73	[0, 2, 8, 6, 12, 11, 3, 4, 9, 7, 5, 10, 1]
Tournament	Hill Climbing	CYCLE	42.17	45.34	[0, 1, 6, 10, 12, 5, 7, 11, 3, 2, 8, 4, 9]
Tournament	Simulated Annealing	PMX	43.44	44.9	[0, 3, 2, 8, 4, 1, 6, 9, 7, 11, 5, 10, 12]
Tournament	Simulated Annealing	CYCLE	43.89	44.71	[0, 4, 8, 2, 6, 1, 10, 5, 11, 7, 9, 12, 3]
Roulette	Random	PMX	38.04	44.48	[1, 2, 3, 0, 9, 11, 5, 8, 4, 7, 12, 10, 6]
Roulette	Random	CYCLE	38.4	43.23	[8, 7, 1, 2, 3, 5, 4, 10, 11, 12, 0, 9, 6]

Roulette	Hill Climbing	PMX	38.98	43.72	[7, 0, 1, 8, 3, 12, 5, 9, 6, 11, 10, 2, 4]
Roulette	Hill Climbing	CYCLE	40.66	44.99	[0, 2, 9, 7, 12, 8, 4, 3, 11, 5, 10, 6, 1]
Roulette	Simulated Annealing	PMX	40.47	43.9	[3, 9, 10, 1, 0, 12, 5, 6, 7, 11, 8, 2, 4]
Roulette	Simulated Annealing	CYCLE	38.83	44.29	[4, 1, 10, 5, 9, 11, 8, 2, 6, 7, 0, 3, 12]

From the results presented in Table 3, we note that the combination that has resulted in the learning path with the poorest quality (43.23) is roulette selection with random initialization and cycle crossover, while the combination that has resulted in the solution with the highest quality (45.47) is tournament selection with random initialization and cycle crossover. What can we claim from these results compared to the quality of the traditional learning path, is that even the quality of the weakest learning path generated by our GA approach is better than the quality of the traditional learning path, with a difference of 3.59%, while the best solution generated compared to the traditional learning path is better for 8.34%. In the following sections, we will present the results obtained by comparing different techniques of population initialization, selection policies and crossover methods that we have used in our approach. We will first present the results of the comparison between randomly population initialization and using other heuristics, such as Hill Climbing and Simulated Annealing.

6.1 Results based on Population Initialization Approach

In this section we will discuss the results and evolution of the learning path using as a starting generation a population created by random solutions, as opposed to the use of a population generated through known heuristics, such as Hill Climbing and Simulated Annealing. As we can conclude from Figure 12, in the first generation created, SA provided the most qualitative solution, then continued with HC and followed by filling the generation with random solutions. In the first iterations of the algorithm this trend continued, then changing the course

of evolution of solutions from the generation created after iteration of 51 using the HC technique. Upon reaching the final iteration, which was also the termination condition, the quality of the best solution in the last generation was almost the same in all the techniques used. From these results we can conclude that the use of population initialization techniques affects only the first generation, because only at that moment the approach of population initialization is applied, then the evolution of the solution quality depends on other factors, such as selection policies and recombination operators.

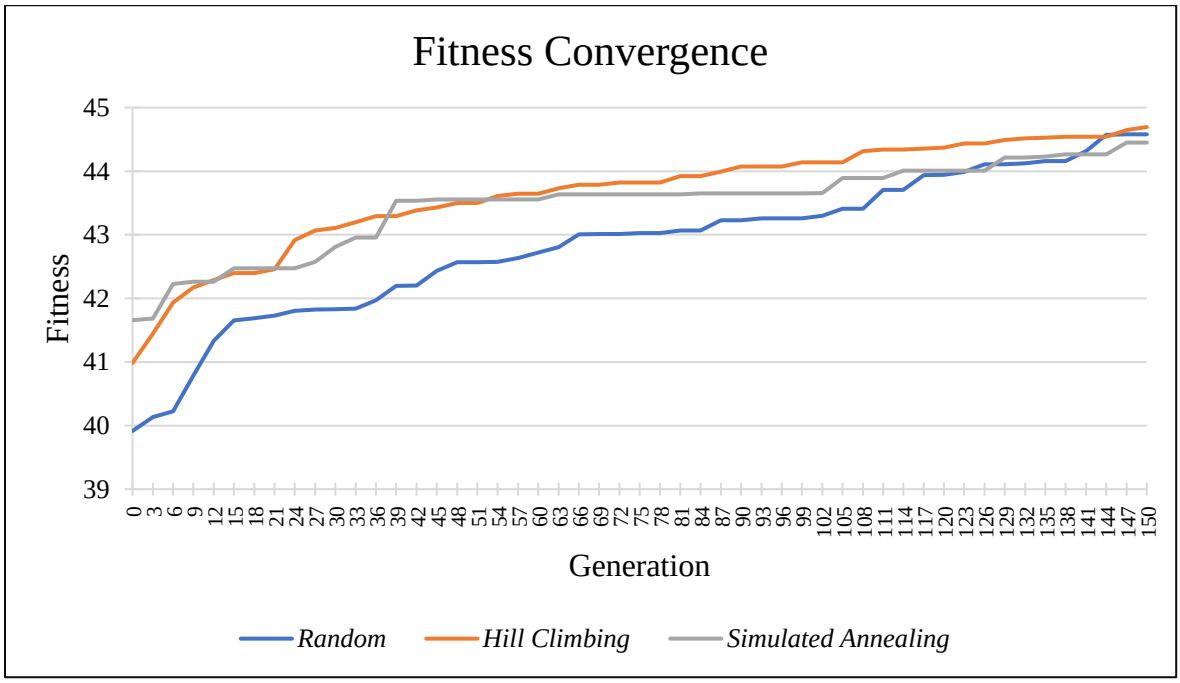

Figure 11 - Path's Fitness Convergence by Population Initialization Approach

6.2 Results based on Selection Policy

To see the impact of the selection policies of individuals from the current generation and then their use to create the new generation, we have analyzed the performance of the two policies developed in our approach. From the graph below we can observe that at the initial development stage, Tournament Selection performed much better than Roulette Selection. In the first generation the best solution from using TS has had a fitness value of 42.74, while RS produced the best solution with fitness value of 39.23. An interesting pattern that can be

noticed is that TS after the iteration of 60, remains in the plateau, making the evolution of fitness value for the best solutions in the next generations to be minor. While the other selection technique after each iteration tends to improve the evolution of the solution by increasing the value of fitness after each new generation until the last iteration. Although this upward trend of RS seems promising, in the end it still fails to produce better solutions compared to TS. In the last iteration, the solution found using TS has a fitness value of 45.04, while the use of RS has provided a solution with fitness of 44.10.

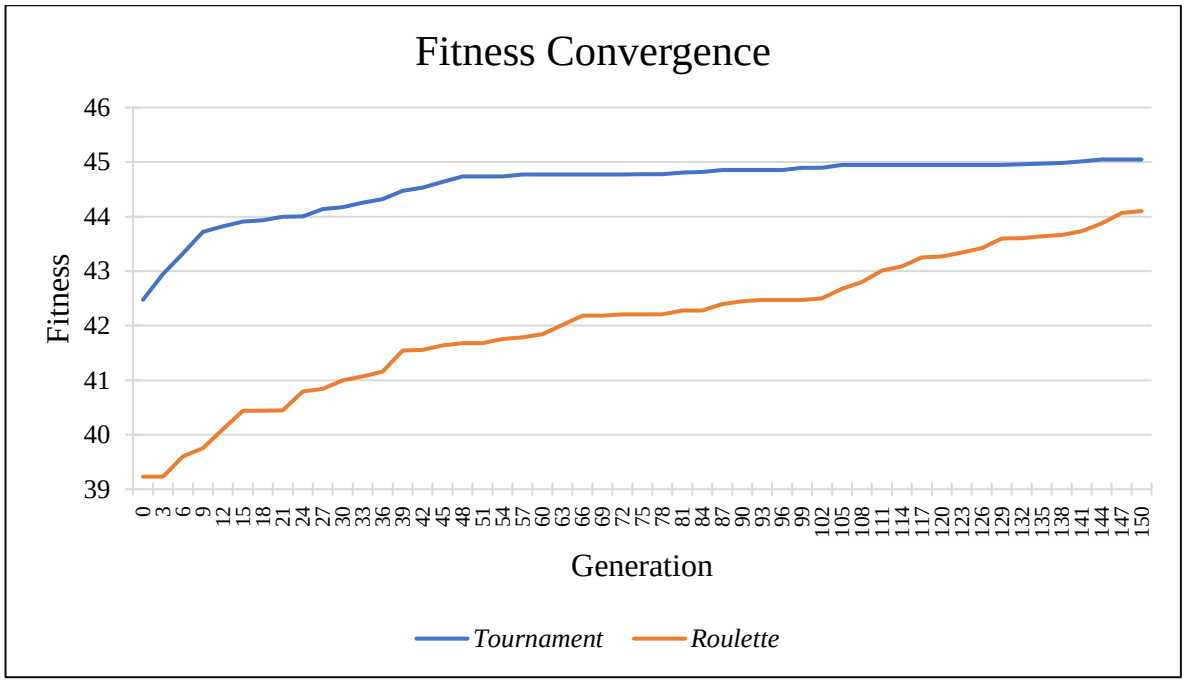

Figure 12 - Path's Fitness Convergence by Selection Policies

6.2 Results based on Crossover Type

From the results of comparing the performance of the implemented crossover types, we can see that in the first generations, both types produced solutions with approximate fitness value. With the increase in the number of iterations and the creation of new generations, cycle crossover has shown higher potential in chromosome combination and higher degree of search space exploitation, thus providing more potential solutions for use in creating new generations. However, as we can conclude from the visualization of the results in Figure 14,

although cycle crossover has shown better performance for a small nuance throughout generation lifecycle, still in the last iteration the use of both techniques has provided valuable learning pathways, and their fitness value is almost the same.

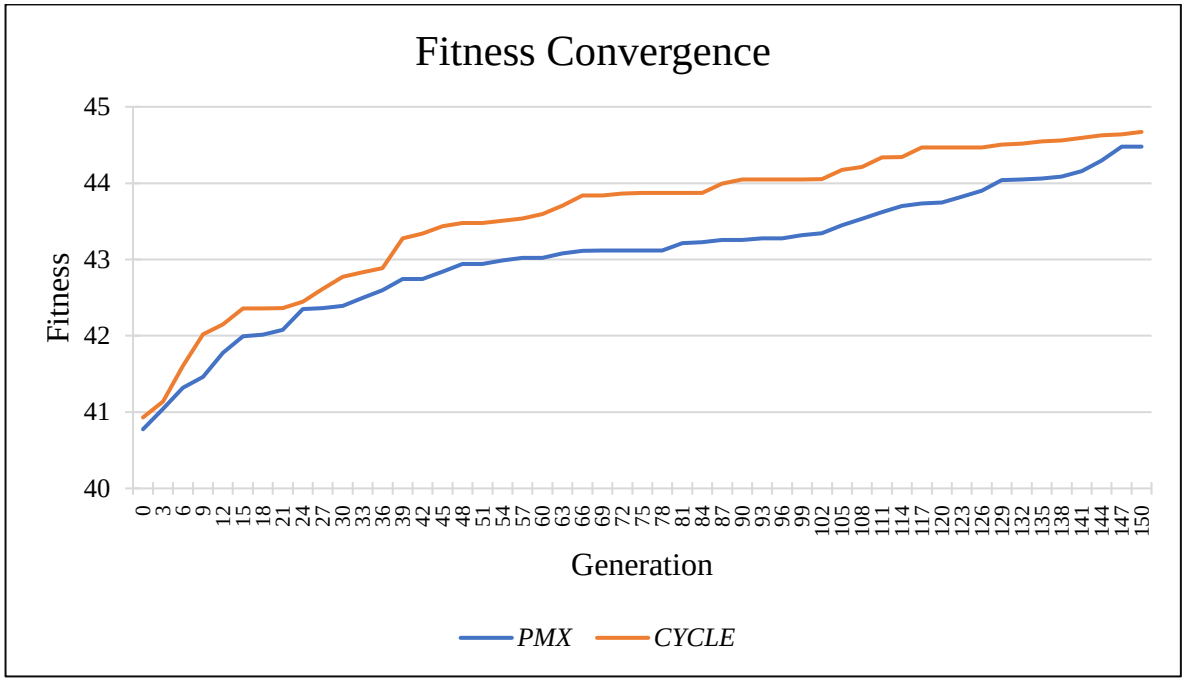

Figure 13 - Path's Fitness Convergence by Crossover Methods

7. CONCLUSION

In this thesis, we introduced a new genetic algorithm hybrid approach that was able to find and generate personalized learning trajectories considering course difficulty level, duration, rating, and relation degrees between learning objects. From the results of the experimental phase, we can conclude that the proposed algorithm is able to find better solutions compared to the traditional path, which is uniformly modeled for all students. We can also conclude that the most optimal solution was generated by using tournament selection with random population initialization and cycle crossover, while the combination with the weakest score was roulette selection with random initialization and cycle crossover. Also, from the results achieved, we can observe that the objective function presented in this paper corresponds to the problem of composing personalized learning paths, as it shows convergence with increasing number of iterations. From the individual comparison of population initialization approaches, we can say that Simulated Annealing, as one of the most advanced techniques for addressing the problem of population initialization has contributed on the creation of first generations with solutions closer to the optimum. While from the selection policies with the best performance has resulted Tournament Selection, although after a certain number of iterations somehow stalled on the plateau, then continuing with the provision of solutions with small improvements in quality. Regarding the comparison of crossover methods, between PMX and cycle, there was no very distinct difference in their performance, resulting in the crossover cycle being dominant for a very small nuance.

This thesis makes three crucial contributions:

1. Review of the preliminary literature on the use of genetic algorithm in the composition of personalized learning paths based on the learner's profile and the attributes of the course topics.
2. Modeling and developing a hybrid approach based on genetic algorithm of the course topics, duration, rating, and the relation degree between topics.
3. Comparison of performance from the development of different techniques for population initialization, selection and recombination.

In conclusion, this approach can be extended including additional attributes related to the learner profile, such as prior knowledge background and modeling of an objective function that includes these additional attributes. Another possibility of improvement is the provision of new techniques for population initialization, selection of individuals to create new generations and various methods for combining genetic material, as well as changing configuration parameters, such as increasing the number of generations, population size, and tuning the probability for recombination operators.

REFERENCES

[1] H. J. Jih, “The impact of learners' pathways on learning performance in multimedia Computer Aided Learning,” *Journal of Network and Computer Applications*, vol. 19, no. 4, pp. 367–380, 1996.

[2] P. Karampiperis and D. Sampson, “Adaptive Learning Resources Sequencing in Educational Hypermedia Systems,” *Educational Technologies and society*, vol. 8, no. 40, pp. 128–147, Oct. 2005.

[3] J. M. Gascuena, A. Fernandez-Caballero, and P. Gonzalez, “Domain Ontology for Personalized E-Learning in Educational Systems,” *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*.

[4] L. de-Marcos, C. Pages, J.-J. Martinez, and J.-A. Gutierrez, “Competency-Based Learning Object Sequencing Using Particle Swarms,” *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, 2007.

[5] H. Kwasnicka, D. Szul, U. Markowska-Kaczmar, and P. B. Myszkowski, “Learning Assistant - Personalizing Learning Paths in e-Learning Environments,” *2008 7th Computer Information Systems and Industrial Management Applications*, 2008.

[6] Y. J. Yang and C. Wu, “An attribute-based ant colony system for adaptive learning object recommendation,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 3034–3047, 2009.

[7] V. Kumar, J. Nesbit, and K. Han, “Rating learning object quality with distributed Bayesian belief networks: the why and the how,” *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, 2005.

[8] M. Neil, N. Fenton, and M. Tailor, “Using Bayesian Networks to Model Expected and Unexpected Operational Losses,” *Risk Analysis*, vol. 25, no. 4, pp. 963–972, 2005.

[9] D. Heckerman, “A Tutorial on Learning with Bayesian Networks,” *Learning in Graphical Models*, pp. 301–354, 1998.

[10] A. E. Eiben and J. E. Smith, “Introduction to Evolutionary Computing,” *Natural Computing Series*, 2003.

[11] K. Jebari, A. E. Moujahid, A. Bouroumi, and A. Ettouhami, “Genetic algorithms for online remedial education based on competency approach,” *2011 International Conference on Multimedia Computing and Systems*, pp. 1–6, 2011.

[12] M. HUANG, H. HUANG, and M. CHEN, “Constructing a personalized e-learning system based on genetic algorithm and case-based reasoning approach,” *Expert Systems with Applications*, vol. 33, no. 3, pp. 551–564, 2007.

[13] C.-M. Chen, “Intelligent web-based learning system with personalized learning path guidance,” *Computers & Education*, vol. 51, no. 2, pp. 787–814, 2008.

[14] M. Bhaskar, M. Das, T. Chithralekha, and S. Sivasatya, “Genetic Algorithm Based Adaptive Learning Scheme Generation For Context Aware E-Learning,” *International Journal on Computer Science and Engineering (IJCSE)*, vol. 2, no. 4, pp. 1271–1279, 2010.

[15] J. Clement, “Model based learning as a key research area for science education,” *International Journal of Science Education*, vol. 22, no. 9, pp. 1041–1053, 2000.

[16] L. de-Marcos, J. J. Martinez, J. A. Gutierrez, R. Barchino, J. R. Hilera, S. Oton, and J. M. Gutierrez, “Genetic algorithms for courseware engineering,” *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 7, 2011.

[17] C. M. Hong, C. M. Chen, and M. H. Chang, “Personalized Learning Path Generation Approach for Web-based Learning,” *4th WSEAS International Conference on E-ACTIVITIE*, pp. 62–68, 2005.

[18] V. V. Zaporozhko, I. P. Bolodurina, and D. I. Parfenov, “A genetic-algorithm approach for forming individual educational trajectories for listeners of online courses,” *In Proceedings of Russian Federation & Europe Multidiciplinary Symposium on Computer Science and ICT*, 2018.

[19] M. Bellafkih, “Adaptive E-learning using Genetic Algorithms,” *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 7, 2010.

[20] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*. Springer Verlag, 2006.

[21] J. H. Holland, *Adaptation in Natural and Artificial System*. Ann Arbor: The University of Michigan Press, 1975.

[22] C. Darwin, *The Origin of Species*. John Murray, 1859.

[23] T. P. Hong, “Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process,” *Applied Intelligence*, vol. 16, pp. 7–17, 2002.

[24] D. Greenhalgh and S. Marshall, “Convergence Criteria for Genetic Algorithms,” *SIAM Journal on Computing*, vol. 30, no. 1, pp. 269–282, 2000.

[25] K. Pireva and P. Kefalas, “A Recommender System Based on Hierarchical Clustering for Cloud e-Learning,” *Intelligent Distributed Computing XI*, pp. 235–245, 2017.