|  |  |
| --- | --- |
| 202000797 | محمد علاء الدين حسين هلال |
| 202000698 | ماريان معوض عزيز |
| 202000169 | انطوانيت عماد صالح |
| 202000689 | مارفلين يوسف انور |
| 202000713 | مايكل صفوت نجيب |
| 202000705 | مازن السيد كامل السيد |

Solving Faculty’s Time Table

Scheduling Problem Using Differential

Evolution

CONTENTS:

**(1) Project idea in details**

**(2) Main functionalities**

**(3) Similar applications in the market**

**(4) An initial literature review of Academic publications (papers) relevant to the idea (5) the Dataset employed (preferably a publicly available dataset)**

**(6) Details of the algorithm(s)/approach(es) that will be used.**

**---------------------------------------------------------**

**(1) Project idea in details**

**This work implements the principle of an differential evolution**

**algorithm to solve the timetabling problem to get a random and full**

**optimal timetable with the ability to generate a multi-solution timetable**

**for each stage in the collage.**

**The major idea is to generate course timetables automatically**

**while discovering the area of constraints to get an optimal and flexible**

**schedule with no redundancy through the change of a viable course**

**timetable.**

**• What is the objective of this Project?**

**-Department's course scheduling problem involves assigning**

**courses, timeslots, and rooms to faculty members. However, due to the**

**large number of constraints that must be handled, searching for an**

**optimal solution for course scheduling problem is considered to be a**

**complex and a time-consuming task**

**•What are the main components?**

**-Classes – Instructors – Time Intervals – Room – Courses –**

**population - Departments**

**•What is the scope of this project?**

**-The project is a software application that many Institutions,**

**businesses and some companies may actually need. This is a simple case**

**of an allocation problem.**

**The project involves developing a program that can schedule time table**

**effectively for school. The prototype of this work should be followed by**

**the development of DE Algorithm**

**(2).Main Functionalities:**

**1. Taking instructor’s name and the courses and the time**

**intervals**

**2. The system should be able to take input about rooms and**

**store it**

**3. Generate the table without any overlapping lecture time with**

**another lecture**

**(3)Similar applications in the market:**

**◘ My Study Life**

**My Study Life, a cross-platform scheduler for students, lecturers,**

**and teachers, is designed for making your student life more**

**organized and manageable. It enables you to store lessons,**

**homework, and exams. As you can store the information in the**

**cloud, it is easy for you to retrieve on any device irrespective of**

**where you are.**

**◘ Timetable**

**The timetable is one of the most downloaded apps as it enables**

**you to easily create and save schedules, and sync them across**

**many devices. . The intuitive app helps you to manage your**

**school/university life in an organized manner.**

**◘ Handy Timetable**

**Handy Timetable app enables you to make two types of**

**schedules: one for the school and one for your studies. It is also**

**easy to create assignment and memos schedules with this**

**timetable app on Android devices.**

**◘ Asctimetables**

**aSc TimeTables is a tool that helps you prepare a school**

**schedule within minutes. It's a very handy tool since creating a**

**complete schedule for a school could be quite complicated,**

**particularly when you are dealing with multiple classes and**

**different teachers. & Instructors**

**(4) An initial literature review of Academic publications (papers) relevant to the idea**

**(paper1) "**[**https://www.researchgate.net/publication/261450836\_A\_Differential\_Evolution\_Algorithm\_for\_the\_University\_course\_timetabling\_problem**](https://www.researchgate.net/publication/261450836_A_Differential_Evolution_Algorithm_for_the_University_course_timetabling_problem)**"**

**I. INTRODUCTION**

**The CTTP is expressed as the minimization of the objective function f(x): x=(x1,…,xn)t, where x is the initial feasible solution to which neighbourhood structures are applied in order to generate new feasible solutions. The objective function, f(x), is minimized by an optimum solution vector x\*=(x1,…,xn)t where f(x\*) < f(x) for all x belonging to a feasible domain. Existing meta-heuristic optimization algorithms designed to solve the CTTP can be categorized as single-solution and population-based approaches. Surveys on meta-heuristic techniques are abundant in the literature (a typical recent review can be found in [1]). Examples of single-solution approaches are: tabu search [3, 4], graph coloring heuristics [4], great deluge [5.6.7], simulated annealing [8, 9], and variable neighbourhood search [10]. Population-based approaches include: genetic algorithms [11, 12], ant colony systems [13], memetic algorithms, evolutionary algorithms [14, 15], etc. Socha et al. [13] employed a local search and ant based algorithms, tested on eleven problems. Kostuch and Socha [16] investigated a statistical model in predicting the difficulty of timetabling problems particularly on the competition datasets. In 2007, Abdullah et al. [17] developed an iterative improvement algorithm with composite neighbourhood structures and later combined this algorithm with a mutation operator. McMullan [7] applied a two phased approach utilizing an adaptive construction heuristic and an extended version of the Great Deluge Algorithm. Landa-Silva and Obit employed a nonlinear great and deluge on the same instances [6]. Interested readers are referred to Lewis [1] for a comprehensive survey of the university timetabling approaches in recent years. In addition, other related papers on Enrolment-Based course timetabling problems can be found in Jat and Yang [18], and Pongcharoen et al. [19].**

**II. PROBLEM DESCRIPTION The enrolment-based course timetabling problem considered in this work was initially defined by the Metaheuristics Network1. This problem was discussed as an assignment of lecture events to timeslots and rooms according to a variety of hard and soft constraints. The problem description that is employed in this paper is adapted from the description presented in [13]. The problem involves scheduling 100-400 courses into a timetable with 45 timeslots corresponding to 5 days of 9 hours each, whilst satisfying room features and room capacity constraints. They are divided into three categories: small, medium and large. We deal with 11 instances: 5 small, 5 medium and 1 large. The characteristics which define the categories are given in Table I. TABLE I. THE DESCRIPTION OF THE ENROLMENT-BASED INSTANCE The problem has: • A set of N courses, e = {e1,…,eN} • 45 timeslots • A set of R rooms • A set of F room features**

**• A set of M students. This problem includes four hard constraints and three soft constraints as follows: Hard constraints: • Event conflict i.e. no student can be assigned to more than one course at the same time (coded as H1). • Room features i.e. the room should satisfy the features required by the event (coded as H2) • Room capacity i.e. the number of students attending the event should be less than or equal to the capacity of the room (coded as H3). • Room occupancy i.e. no more than one event is allowed at a timeslot in each room (coded as H4). Soft constraints: • Event in the last timeslot i.e. a student shall not have to sit a course that is scheduled in the last timeslot of the day (coded as S1) • Two consecutive events i.e. a student shall not have more than 2 consecutive events (coded as S2). • One event a day i.e. a student shall not have to sit a single course on a day (coded as S3). Hard constraints act an inviolable requirement. A timetable which meets the hard constraints is recognised as a feasible solution. The main objective is to minimise the violation of the soft constraints in a feasible solution that later represents the quality of the obtained solution. A solution consists of an ordered list of length N where the position corresponds to the events i.e. position i corresponds to event ei for i = 1,…,N. The values for each position are a number in between 0 to 44 corresponding to the timeslot index, and 0 to R-1 correspond to the room index. For example, a timeslot vector is given as (0,17,30,…,10) and a room vector is given as (4,3,0,…,3) means that event e1 is scheduled in timeslot 0 in room 4. Event e2 is scheduled in timeslot 17 in room 3 and finally event eN is scheduled in timeslot 10 in room 3. The objective function for the problem is defined in the formula below. min 321 sss ++∑ (1) where S1, S2 and S3 represent the relevant soft constraints. III. THE ALGORITHM The proposed algorithm consists of two phases i.e. build a feasible initial population using a constructive heuristic; and an improvement algorithm with an aim to optimise the violation of the soft constraints while maintaining the feasibility of the solutions. A. Neighbourhood Structure The different neighbourhood structures and their explanation can be outlined as follows: N1: Choose a single course at random and move to a feasible timeslot that can generate the lowest penalty cost.**

**N2: Select two courses at random from the same room (the room is randomly selected) and swap timeslots. B. Constructive Heuristic A least saturation degree is used to generate initial solutions (the population) which start with an empty timetable [7]. The events with less rooms available and more likely to be difficult to be scheduled will be attempted first without taking into consideration the violation of any soft constraints, until the hard constraints are met. This process is carried out in the first phase. If a feasible solution is found, the algorithm stops. Otherwise, phase 2 is executed. In the second phase, neighbourhood moves (N1 and/or N2) are applied with the aim to change infeasible solution to feasible one. N1 is applied for a certain number of iterations. If a feasible solution is met, then the algorithm stops. Otherwise the algorithm continues by applying a N2 neighbourhood structure for a certain number of iterations. We have no proof that this constructive heuristic is guaranteed to find a feasible solution for a given instance. However, in this experiment, the solutions were made feasible before the improvement algorithm is applied. C. Improvement Algorithm: Differential Evolution Algorithm DE is a basic algorithm of the population that employed crossover, mutation and selection operators as in genetic algorithms. The main difference in obtaining better solutions is that genetic algorithms rely on the crossover operation, while in the DE algorithm it is based on the mutation operation. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search towards the potential regions in the search space. This population is then improved by applying mutation, crossover and selection operators. The main steps of the differential evolution algorithm**

**The algorithm starts with the randomly selecting two parents (p1, p2) from the population. The mutation operator carried out on selected parent timetables (p1, p2) as one of the neighbourhood structures (as listed in Section III) selected randomly will be applied to both parent timetables (p1, p2) separately, to generate a new parent timeslots (p1\*, p2\*). Then, a crossover operator is defined: taking the two parent timetables (p1\*, p2\*) and exchanging two timeslots selected randomly between p1\*and p2\*, and allocating rooms to events in each time slot. Two new offspring will be generated. These offspring solutions are called child1 and child2. Then the best among the new offspring will be selected by using the evaluation and selection operator. The quality of the new**

**In most cases small, the way the algorithm explores the search space clearly indicates that further improvement is achieved. In addition, less complexity of small datasets allows the technique to explore a number of non-improvement moves and achieve a zero evaluation. The algorithm is also effective for the medium and large datasets, in the graphs clearly indicating that a further reduction of more than 60% in the cost evaluation can be achieved. We believe that the algorithm can fast when it relies on the mutation operation rather than the crossover operation. V. CONCLUSION AND FUTURE WORK This paper presents an effective Differential Evolution Algorithm (DE) for the university course timetabling problems, showing that evolutionary computation can deal successfully with the problem. The use of the effective Differential Evolution Algorithm (DE) is an important element of its high performance. There are two advantages from using DE; first, fast convergence, and second, a few control parameters are use. Preliminary comparisons indicate that this algorithm is competitive with other approaches in the literature. Future research will be aimed to test this algorithm on the International Timetabling Competition datasets (ITC2007). REFERENCES [1] R Lewis. A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum 30 (1), 167-190 (2008). [2] EK Burke, G Kendall and E Soubeiga, A tabu search hyperheuristic for timetabling and rostering. Journal of Heuristics 9(6), 451-470 (2003). [3] R Alvarez-Valdes, E Crespo and JM Tamarit. Design and implementation of a course scheduling systems using tabu search: Production, manufacturing and logistics. European Journal of Operational Research, 137, pp 512-523 (2002). [4] EK Burke, B McCollum, A Meisels, S Petrovic and R Qu, A Graph-Based Hyper Heuristic for Educational Timetabling Problems, European Journal of Operational Research 176(1), 177-192 (2007). [5] S Abdullah, K Shaker, B McCollum & P McMullan. Incorporating great deluge with Kempe chain neighbourhood structure for the enrolment**

**based course timetabling problem. RSKT 2010, In LNAI: Vol. 6401. Springer, pp 70–77, 2010. [6] D Landa-Silva and JH Obit. Great Deluge with Nonlinear Decay Rate for Solving Course Timetabling Problems. Proceedings of the 2008 IEEE Conference on Intelligent Systems (IS 2008), IEEE Press, 8.11-8.18 (2008) [7] P McMullan. An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, Lecture Notes in Computer Science, Springer, Vol 4487, pp538-545 (2007) [8] S Abdullah, K Shaker, B McCollum, P McMullan. Dual Sequence Simulated Annealing with Round-Robin Approach for University Course Timetabling. EVOCOP 2010, LNCS: 6022, Springer-Berlin /Heidelberg, pp 1–10, 2010. [9] R Bai, EK. Burke, G Kendall and B McCullum. “A Simulated Annealing Hyper-heuristic for University Course Timetabling Problem”. (Abstract) PATAT '06, Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, ISBN 80-210-3726-1, pp345-350 (2006). [10] S Abdullah, EK Burke and B McCollum, An investigation of variable neighbourhood search for university course timetabling. The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), 413-427 (2005). [11] S Abdullah and H Turabieh, Generating university course timetable using genetic algorithms and local search. The Third 2008 International Conference on Convergence and Hybrid Information Technology ICCIT, vol. I, 254-260 (2008). [12] R Lewis and B Paechter. New crossover operators for timetabling with evolutionary algorithms. Proceedings of the 5th International Conference on Recent Advances in Soft Computing (ed. Lotfi), UK, December 16th-18th, pp 189-194 (2004). [13] K Socha, J Knowles and M Samples, A max-min ant system for the university course timetabling problem. Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002), Springer Lecture Notes in Computer Science Volume 2463, 1-13 (2002). [14] Jat, S. N., & Yang, S. (2008). A memetic algorithm for the university course timetabling problem. Proc. of the 20th IEEE international conference. tools with artificial intelligent. (pp. 427–433). [15] S Abdullah, EK Burke and B McCollum, A Hybrid Evolutionary Approach to the University Course Timetabling Problem. IEEE Congres on Evolutionary Computation, ISBN: 1-4244-1340-0, 1764-1768 (2007). [16] P Kostuch and K Socha, Hardness Prediction for the University Course Timetabling Problem, Proceedings of the Evolutionary Computation in Combinatorial Optimization (EvoCOP 2004), Coimbra, Portugal, April 5-7, 2004, Springer Lecture Notes in Computer Science Volume 3004, 135-144 (2004). [17] S Abdullah, EK Burke and B McCollum, Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling. In: Metaheuristics Progress in Complex Systems Optimization, Springer, 153-169 (2007). [18] Jat, S. N. and Yang, S. 2010. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. Journal of Scheduling pp.1-21. [19] P. Pongcharoen, W. Promtet, P. Yenradee and C. Hicks, “Stochastic optimisation timetabling tool for university course scheduling,” International Journal of Production Economics 112 (2): 903 – 918, special Section on RFID: Technology, Applications, and Impact on Business Operations, 2008. [20] R Storn, “Differential Evolution, A Simple and Efficient Heuristic Strategy for Global Optimization over Continuous Spaces” Journal of Global Optimization, Vol. 11, Dordrecht, pp. 341-359, 1997. [21] H Turabieh,S Abdullah, B McCollum. Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem. RSKT 2009. LNCS, vol. 5589, pp. 497–504. Springer, Heidelberg, 2009. [22] M Al-Betar, A. Khader and I. Yi Liao, “A Harmony Search with Multi-pitch Adjusting Rate for the University Course Timetabling,” In: Z.W. Geem: Recent Advances in Harmony Search Algorithm, SCI 270, pp. 147–161. Springer, Heidelberg, 2010.**

**( paper 2)**

**"**[**https://sci-hub.hkvisa.net/10.1109/scis-isis.2012.6505416**](https://sci-hub.hkvisa.net/10.1109/scis-isis.2012.6505416)**"**

**The timetabling problem is a kind of combinatorial**

**optimizations. However, it is very difficult to be solved the**

**timetabling problem from the enormous total number of**

**combination and the complexity of the limitation condition.**

**Many articles about the timetabling problems are reported, but**

**the evaluation varies environment even if a feasible solution**

**was provided. In general, the faculty members make the**

**timetable based on their knowledge and past experience using**

**vast time. There are no exceptions in Osaka International**

**University (OIU) authors working, faculty members must**

**consider several elements（i.e., number of class and number of**

**classroom, whether or not a compulsory subject, etc.）.The**

**main purpose of this study is the development of timetabling**

**algorithm for OIU. However, it is difficult to design and**

**solving the problem with many constraints. If we create**

**timetable for all grades and all departments, it is necessary to**

**consider the following issues as constraints:**

**A. Inflexible constraint**

**1. There are 5 classes a day from Monday to Friday.**

**2. Each class is opened once a week.**

**3. Some subjects open two classes continuously.**

**4. All classes must be placed in the timetable.**

**5. The same teacher must be not place at the same time.**

**6. The same class must be not place at the same time.**

**7. There is the class that opening time is fixed like a**

**seminar.**

**8. Problem of facilities (whether using a computer room**

**or a projector)**

**9. A conference day (each class of the exclusive teacher**

**can be place only 2 classes in the morning on**

**Wednesday, because the afternoon is a conference**

**time.)**

**10. Externship day (as for each exclusive teacher takes**

**externship day in a week)**

**11. The number of the student attending a class and size**

**of the classroom (because the required subjects have**

**many student attending a class, it is necessary to**

**assign a large classroom)**

**12. The class to affect graduation requirements like a**

**required subject locates for 2-4 periods on timetable.**

**B. flexible constraint**

**1. The classes in the required subject do not open in the**

**same time as possible.**

**2. The class of the same teacher does not leave the**

**periods as much as possible (after the first period not**

**including lesson until fifth period on time table)**

**3. On a duty day of the same part-time teacher opens on**

**the same day as possible.**

**These may be the most phenomena to be common in other**

**universities. In addition, these constraints are divided into**

**limitation (inflexible constraint) that must be satisfied and**

**limitation (flexible constraint) that must be not need to**

**necessarily keep. These constraints are important elements to**

**consider in OIU that when faculty members create a timetable.**

**As described above, it is difficult to propose the solution**

**method that satisfies all of these conditions. So, in this paper,**

**we design the reduced model and propose a solution method by**

**Differential Evolution (DE).**

**DE is one of the evolution algorithm that proposed by Storn**

**and Price (1995) [1], [2]. This algorithm is population-based**

**stochastic search method for optimization problem. The**

**advantages of DE are its simple structure, ease of use, speed**

**and robustness. Moreover, DE has few parameters that a user**

**must set from experience. Today, DE has become a very**

**popular evolutionary computation technique, but there are few**

**articles treating the timetabling by DE.**

**Its primary purpose of this research is to improve the work**

**efficiency and to reduce the burden of faculty members. In this**

**time, we only treat the reduced model, however we are going**

**to continue this research for large scale problems.**

**II. PROBLEM STATEMENT**

**A problem to deal with in this study is a timetabling**

**problem in consideration of the environment of the Osaka**

**International University (OIU). However, we design the**

**reduced model as a test model for DE solution. So, we define**

**the constraints that considered in this time as follows:**

**1. Created timetable is for first grade and for one**

**department.**

**2. There are 5 classes a day from Monday to Friday.**

**3. Each class is opened once a week.**

**4. All classes must be placed in the timetable.**

**5. The same teacher must be not place at the same time.**

**6. The same class must be not place at the same time.**

**7. There is the class that opening time is fixed like a**

**seminar.**

**8. The class to affect graduation requirements cannot**

**locate same section.**

**In addition, 30 subjects are used in experiment. When DE**

**assigns of each subject, if above constraints do not satisfied,**

**penalty value is given. DE searches the solution to**

**minimize the total penalty value.**

**III. SOLVING THE MODEL BY DE**

**Differential Evolution (DE) is one of the Evolutionary**

**Algorithms (EA) that proposed by Storn and Price (1995) [1],**

**[2]. This algorithm is population-based stochastic search**

**method for optimization problem. In DE, each variable’s value**

**is represented by a real number. The advantages of DE are its**

**simple structure, ease of use, speed and robustness. Moreover,**

**DE has few parameters that a user must set from experience. In**

**timetabling problem, several solution methods using EA**

**techniques have been proposed [3], [4], [5]. For example, there**

**is a solution by the GA, however this algorithm has**

**disadvantage that the chromosome will be long when we**

**convert timetable to chromosome. It means that the number of**

**combinations increases. Therefore, GA will not be able to fully**

**explore the solution space even after crossover and mutation.**

**In addition, the calculation time will increase, or the premature**

**convergence of evolution happens. On the other hand, there are**

**few articles treating the timetabling by DE.**

**So, in this research, we propose an effective solution**

**method by DE that is robust and able to fast convergence**

**compared with other evolutionary strategy**

**Here, DE is a population based search method which**

**utilizes NP variables as population of D dimensional parameter**

**vectors in each generation. The initial population is chosen**

**randomly if no information is available about the problem. In**

**the case of the available preliminary solution, the initial**

**population is often generated by adding normally distributed**

**random deviations to the preliminary solution. DE generates**

**new parameter vectors by adding the weighted difference**

**vector between two population members to third member. And,**

**the resulting vector yields a lower objective function value than**

**a predetermined population member; the newly generated**

**vector replaces the vector with which it was compared. In**

**addition, the best parameter vector is evaluated for every**

**generation. DE maintains two arrays, each of which holds a**

**population size NP and D dimensional, real-valued vectors.**

**The primary array holds the current vector population, while**

**the secondary array accumulates vectors that are selected for**

**the next generation. In each generation, NP competitions are**

**held to determine the composition of the next generation. In**

**mutation process, every pair of vectors (xa, xb) defines as a**

**differential vector: (xa-xb). When xa and xb are chosen randomly,**

**their weighted differential is used to perturb another randomly**

**chosen base vector xc. This mutation process can be**

**mathematically expressed as:**

**x xcc ( ) x xba F −+=′ (1)**

**The scaling factor F is a user supplied constant in the**

**optimal range between 0.5 and 1.0. In every generation, each**

**primary array vector xi is targeted for crossover with a vector**

**xc’ to produce a trial vector xt. This trial vector xt is the**

**offspring which generated from parent vectors xi and xc’. The**

**uniform crossover is used with a crossover rate (CR) which**

**actually represents the probability that the child vector inherits**

**the parameter values from the random vector. If trial vector xt**

**has better value compare with parent vector, parent vector is**

**replaced with child vector.**

**So far, several kinds of DE have been proposed, such as**

**“DE/best/1/bin” and “DE/rand/1/exp” are well known. This**

**notation method has rule as DE/base/num/cross. For example,**

**the part of “base” represents the selection method of the parent**

**and “num” part represents the number of differential vector**

**respectively. If “base” part is best, the algorithm selects**

**individual as a parent having a best fitness value. On the other**

**hand, if individual is chosen as a parent at randomly, this part**

**is describe “rand”. The last part of “cross” means crossover**

**method for generates offspring. For example, DE**

**/base/num/bin uses binomial crossover to replace the gene with**

**a certain probability, and DE /base/num/exp uses exponential**

**crossover to replace the gene based on probability decreasing**

**exponentially. The basic DE process is shown bellow.**

**Step1. Initial populations are generated.**

**Step2. DE evaluates the all individuals, if the termination**

**condition is satisfied calculation process is finished.**

**Step3. The trial vector is generated from mutation process.**

**Step4. The child vector is generated form crossover process**

**Step5. If trial vector has better value compare with parent**

**vector, parent vector is replaced with child vector.**

**Step6. If the termination condition is satisfied, calculation**

**process is finished, otherwise go back to Step 2.**

**Furthermore, we propose one of generating method of**

**initial population. First, we assign the class ID to each square**

**from 1 to total number of class ID. In this research, the total**

**number of classes and total number of squares are 30 and 50**

**respectively. So, 0 is input into 20 remaining squares (In other**

**words there is no assigned class). Next, random numbers (0-1)**

**that correspond to each class are generated. Third, these**

**random numbers are sorted in ascending order. At the same**

**time, the corresponding class ID is also sorted. In addition, this**

**random numbers are used when DE calculated differential**

**vector. We will show the image in Figure 3.**

**IV. NUMERICAL EXPERIMENTS**

**In numerical experiments, we compare 5 DE strategies as**

**follows.**

**1. DE/best/1/exp**

**2. DE/rand/1/exp**

**3. DE/best/2/exp**

**4. DE/rand/2/exp**

**5. DE/best/1/bin**

**In this time, we have set the parameters as follows:**

**･ Number of population: 100**

**･ Max generation: 1000**

**･ crossover rate: 0.5**

**･ Scaling factor F: 0.5**

**Table 2 shows the experimental results. All of DE that**

**including proposal method can solve the problem. And it was**

**little inferior in the standard deviation, but DE2 obtain a good**

**solution**

**(5) the Dataset employed (preferably a publicly available dataset):**

**The genetic algorithm is an effective way to solve many problems, and it depends mainly on the theory of evolution and natural selection.**

**It consists of some steps:**

* **Create random population**
* **Calculate fitness of each element in the population**
* **Select the elements which has the higher fitness value or the lower based on the problem and on the implementation.**
* **And we choose a small percentage of the elements which has lower fitness value.**
* **And we apply mutation function to change some genes to reach global minima or optima.**
* **After this steps we will apply the cross over function to create a new individuals has genes from the two parents which are selected first with higher fitness.**
* **We will calculate the fitness for the new population it will be better (highier or lower) than the original population.**

Use case Diagram:Diagram

Description automatically generatedFlowchart:Diagram

Description automatically generated