Turkish Journal of Computer and Mathematics Education Vol.12 No. 6 (2021), 1181-1188

Research Article

Balanced Academic Curriculum: Looking For An Optimal Solution With Metaheuristics And Functional Programming

[1]José Miguel Rubio, [2]Cristian Vidal-Silva, [3]Luis Carter, [4]Miguel Tupac-Yupanqui

[1]Ingeniería Informática, Facultad de Ingeniería, Ciencia y Tecnología,
Universidad Bernardo O'Higgins,
Santiago, Chile
[2]Departamento de Administración, Facultad de Economía y Administración,
Universidad Católica del Norte,
Antofagasta, Chile
[3]Docente de Ingeniería Civil Industrial, Facultad de Ingeniería,
Universidad Autónoma de Chile
Talca, Chile
[4]EAP Ingeniería de Sistemas e Informática,
Universidad Continental,
Huancayo, Perú

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: The curriculum design is quite a challenge in the academy, mainly because it requires an adequate distribution of content for the development of the expected professional competencies regarding the available time, the necessary academic load, and their gradual progress in the higher educational institutions. Considering the above, the main objective of this work is to present and exemplify a computational solution to minimize the cost of designing curriculum plans using bio-inspired algorithms to automate and reduce errors in such a process. Specifically, the purpose of this research focuses on solving the Curriculum Mesh Balancing (BACP) problem through metaheuristic optimization based on the behavior or algorithm of fireflies and the use of functional programming in the Haskell lang curricular meshes, rolling of curricular meshes, metaheuristics; firefly algorithm, functional programming in Haskell programming language. The firefly algorithm will be applied to a set of test instances to demonstrate its effectiveness. According to the obtained results, this proposal allows the efficient gathering of solutions to the problem under study.

Keywords: Academic curriculum; rolling of academic curriculum; metaheuristics; fireflyalgorithm; functional programming; Haskell.

1. Introduction

Rubio et al. [Rubio et al. (2019)] described that the curricular meshes are not only essential to differentiate the same careers from 2 or more institutions. These are an instrument that contains the knowledge and learning objectives that each student must achieve in a specific career designed by teachers and professors to meet future professionals' training needs, who should be able to solve common problems in their area of formation. Designing a curricular mesh is a challenge, not only because they must train a complete professional but also because students must complete it in a certain period. Addressing the same subjects many times could require additional time for their full development [Glatthorn et al. (2018)]. For these reasons, a balance must exist in curricular meshes regarding the student's probability of success. Too many branches (courses or modules) in an academic period could cause future professionals' failure; then, the academic load should be similar in each period and the minimum possible within the broad spectrum of skills and knowledge that students could acquire [Rubio et al. (2019)].

The Balanced Academic Curriculum Problem (BACP) consists of assigning courses to periods of teaching that satisfy the prerequisites and balance the load of the students in terms of credits and number of courses [Chiarandini et al., (2012)]. Castro and Manzano [Castro and Manzano, (2001)] presented the BACP together with proposing and developing a linear programming model considering the following entities and restrictions: 1). Non-optional courses (courses with their credits according to the academic curriculum). 2). According to the curriculum, academic periods correspond to a fixed number of time intervals (each academic period includes courses to teach). 3). The maximum academic load allowed (maximum number of credits and courses for each academic period). 4). Minimum academic load allowed (minimum number of credits and courses for each academic period). 5). Approval requirements for each course (students must take and approve some courses before others). The prerequisites and subsequent courses allow the generation of ordered pairs of courses. 6). A balanced distribution of the curriculum, that is, the number of credits in each academic period should be similar, ideally the same. Thus, the BACP planning horizon divides a career in academic years and each of them into

periods to take the courses. The problem is to find an assignment of courses to academic periods that satisfy certain load limits and prerequisites [Rubio et al., (2018)]. This paper considers BACP as an optimization problem.

The work of [Hnich et al., (2001)] defines a structural and behavioral model for the BACP problem, while the work of [Castro and Manzano, (2001)] treats it as a problem of restrictions. In recent years, various proposals and reviews of algorithmic solutions for constrained and optimization problems have been developed with a bio-inspired approach, usually specializations or optimizations of genetic algorithms [Yang, (2010)], [Cabrera-Guerrero et al., (2012)], [Kar, (2016)], [Lanza-Gutiérrez et al., (2016)], [García et al., (2019)], [Soto et al., (2019)]. The Firefly Algorithm (FA) [Yang, (2010)] is an example of such solutions whose basis of operation is the blinking patterns and behavior of fireflies in search of a potentially optimal solution.

Considering the impact of a balanced academic load on the students' performance and success, the objective of this work is to present a solution to minimize the cost of designing balanced curricular plans through the application of bio-inspired algorithms to automate and reduce errors in the design of a balanced academic curriculum [Rubio et al., (2018)]. That is, this work looks to find an assignment of courses for each academic period that optimally comply with all the entities and restrictions mentioned. Thus, this work proposes a BACP optimization solution using a bio-inspired optimization metaheuristic based on the algorithmic attraction behavior of fireflies [Yang, (2010)] implemented in the functional programming language Haskell [Hutton, (2016)], [Thompson, (1999)], [Bird, (2014)].

This work organizes as follows: Section 2 presents the Haskell programming language and the logic programming paradigm. Section 3 describes the methodology this work follows. Section 4 shows the main results. Section 5 mentions pros and cons of our work. Section 6 concludes this research.

2. Haskell and Logic Programming

According to [Hutton, (2016)], Haskell is a multipurpose, functionally pure, and polymorphically typed programming language. Haskell uses the lambda calculus. As a functional programming language, in Haskell, the primary or main constructor is the function. Language has its origins in the observations of Haskell Curry and his intellectual descendants for their work in mathematical logic [Hugh, (1989)].

Structured object-oriented programming languages are examples of imperative programming languages; that is, programming languages that define algorithmic solutions as a sequence of steps for executing them in the specified order. One of Haskell's remarkable properties is, due to its functional nature, which allows the definition of functions and expressions for their execution through their evaluation [Hutton, (2016)], [Hugh, (1989)]. Thus, Haskell is an example of a high-level and general-purpose programming language.

Figure 1 shows the Haskell and Java code of the function to, given an integer, determine its factorial number. Note that the Haskell code is analogous to the mathematical definition or declaration of the factorial function. In contrast, the Java procedural code requires to indicate step by step each of the function's actions to obtain results. The following are the unique properties of Haskell solutions: 1. Brevity - Programs tend to be much shorter; 2. Simplicity: functional programs are usually simpler to understand; 3. No memory errors: strongly-typed solutions do not present type mismatches or memory errors; 4. Reuse: Haskell solutions have minimal restrictions to facilitate reuse; 5. High abstraction: functional programming languages use the declaration of functions, which allows reaching high levels of procedural abstraction; that is, Haskell does not rely on variable and object state changes like traditional imperative programming; 6. Memory management: Haskell does not require explicit dynamic memory management, such as in traditional imperative programming languages [Bird, (2014)], [Bird, (2014)].

```
fac :: Int -> Int
fac 0 = 1
fac n | n > 0 = n * fac (n-1)
```

a. Haskell code

```
int fac(int n){
        if (n==0)
          return 1;
        else
          return n*fac(n-1);
}
```

b. Java code

Fig. 1. Haskell and Java code example of factorial function.

According to [Hutton, (2016)], functional solutions, according to the inherent properties of functional programming, require high levels of use of both memory and computing, which required high-level computing systems in the previous decade. However, due to the advances and current availability and accessibility of high-performance computing systems, functional programming solutions are much more accessible and viable today.

Applying the FA firefly algorithm to an optimization problem requires to consider the brightness proportional to the objective function's value. In genetic algorithms, the brightness can be defined in the same sense as the objective function. Since the firefly's attractiveness or luminosity is proportional to the light emanating, Eq. (1) defines this attraction.

$$\beta = \beta_{\{0\}\epsilon}^{\{\lambda r^{\{2\}}\}}. \qquad (1)$$

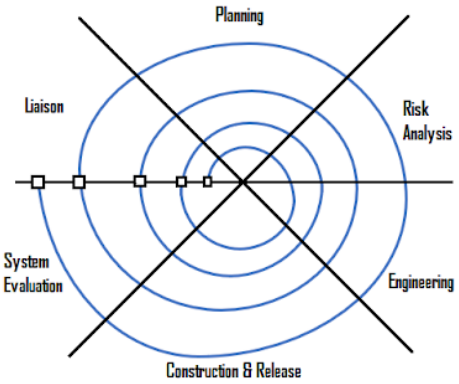

Fig. 2. Fireflies algorithm FA.

Figure 2 shows a general firefly algorithm scheme from [Durkota, (2011)] whose objective function is defined based on the objective of the problem. It is also necessary to define the initial light intensity and initialize the firefly parameters, the population size n, and the maximum number of generations (MaxGeneration). The main idea to solve the BACP proposed in this work is based on representing the problem through a binary arrangement for the evidence that it is beneficial to offer the solution—the firefly algorithm for this purpose is directly adaptable such and as presented in the next section.

3. Methodology

Fig. 3. Simplified spiral software development methodology.

This research consists of developing a bio-inspired algorithm for the optimization of curricular meshes in

Objective function f(x), $x=(x_1,\ldots\ldots,x_D)^T$
Initialize positions of fireflies x_i (i=1,2,……,M)
Calculate Light intensities by $I_i=f(x_i)$
while (t< MaxGeneration t_{max}) **do**
  for i = 1 to M, all M fireflies **do**
    for j = 1 to M, all M fireflies **do**
      if $I_i > I_i$ **then**
        Move firefly i toward j by Eq.(8)
      end if
    end for j
  end for i
  Evaluate new solutions $f(x_i)$
  Rank the fireflies and find the current global best g_*
end while

several stages through a modular production. Hence, we chose the spiral development methodology for its greater flexibility and ease for developing incremental solutions and evolutionary programs of software engineering [Sommerville, (2010)], that is, with the gradual and evolutionary application of stages of analysis, design, implementation, and testing of defined functionalities and modules. Figure 3 illustrates this development methodology. In the context of this work, we apply the incremental spiral model in the development of four phases of prototyping: discovery, coding, validation, and optimization, with gradual incorporation of constraints of the problem together with tests to determine the validity of the minimum viable product under development and then in production.

For the choice of solution heuristics, we use the design methodology proposed by [Bird. (2010)] that consists of using identities between functions to simplify algebraic expressions in functional programming and to optimize the implementation of functions in Haskell. We optimize the implementation of functions by using rules of inference to modify their complexity and thus designing a global repair heuristic strategy to improve the satisfaction of precedence constraints in the BACP.

This work adapts the elements presented by Rubio et al. [Rubio et al., (2018)]. The first step was to implement the original mathematical model of Castro and Manzano [Castro and Manzano, (2001)] in the Haskell programming language. We consider the precedence restriction and the academic load restriction to the set of initial solutions (initial population). Then, we apply optimization through the firefly algorithm incorporating the other restrictions of the problem.

In the algorithmic operation of the solution, first, a precedence graph is read, and then it eliminates the courses of the last period to obtain a new graph. We carry this process iteratively until getting a set of courses without precedence restrictions among themselves (elimination of precedence by blocks), to ensure that the courses in the final set do not have precedence conflicts. In this solution, functions are used that operate on lists of the form (c, p_1, …, p_n]) where [p_i] with i in [1, ..., n] that represents the list of courses preceding pi to the course c. For example, Figure 4 illustrates the used data structure for the BACP8 instance, that is, for an eight-semester curriculum.

Figure 5 presents part of the Haskell solution of this work to generate new values in a BACP matrix, specifically, for the modification of values according to Yang's firefly algorithm [Yang, (2010)]. Note that each functional definition in figure 5 presents comments (at least two consecutive hyphens precede Haskell comments).

```
graphBACP8 = [ (46, [32]), (45, [38]), (40, [36]),
               (39, [30]), (38, [36]), (37, [30],
               (36, [28]), (35, [28]), (34, [30]),
               (33, [26]), (32, [26]), (29, [24]),
               (28, [22]), (27, [21]), (26, [14]),
               (24, [16]), (21, [15]), (20, [14]),
               (19, [14]), (18, [3]), (17, [7]),
               (16, [10, 11]), (15, [9]), (14, [9]),
               (12, [8, 11]), (11, [5, 6]), (10, [5]),
               (8, [2, 6]), (7, [1]) ]
```

Fig. 4. Definition in Haskell of the correlation matrix between courses and its prerequisites.

```
-------------------------------------------
-- Function that modifies values a ccording to the Fireflies
-- algorithm defined by Xing-She Yang.
-------------------------------------------

newVal x i j = 0.5*expf+(fromIntegral x)+0.5*((randNum1 i j)-0.5)
where expf = exp(-18*2*(fromIntegral (abs (j-i))))

-------------------------------------------
-- Function to obtain the hyperbolic tangent
-------------------------------------------

tanh2 x = (exp (2*(sqrt (x^2))-1))/(exp (2*(sqrt (x^2))+1))

-------------------------------------------

-- Function to obtain the final values a fter comparing the obtained
-- values o f the hyperbolic tangent with respect to a random
-- number
-------------------------------------------

endVal x i j
| (randNum2 i j) < (tanh2 x) = 1
| otherwise                  = 0

-------------------------------------------
-- Function that applies endVal to a row of i elements where row j
-- is varied
-------------------------------------------

transfVal x i j = endVal (tanh2 (newVal x i j)) i j

-------------------------------------------
-- Function that enumerates the previous function for j between 1
-- and n to generate the complete modification of the matrix of i --
-- rows and j columns
-------------------------------------------

transfRow xs m = [transfVal (xs!!i) (i+1) m |
                  i <- [0..((length xs)-1)]]

-------------------------------------------
-- Function that applies transfRow to each matrix of a population
-- of matrices
-------------------------------------------

transfMat xss = [transfRow (xss!!j) (j+1) |
                 j <- [0..((length xss)-1)]]
```

Fig. 5. Haskell functions for generating new values in a correlation matrix of courses and prerequisites.

4. Results

When applying the solution functions proposal in the balanced academic load optimization problems of eight, ten and twelve semesters, the results obtained are similar to those obtained in both the works by Rubio et al. [Rubio et al., (2018)], and Rubio et al. [Rubio et al. (2019)], where the performance considering 50 executions is usually oscillating.

Still, it allows obtaining the expected values of minimum academic load for each case. That can be seen in Figure 6 that illustrates the results after 50 runs of the solution implemented in Haskell for BACP8, BACP10, and BACP12 instances, respectively. Each run considered 500 generations of fireflies (iterations). However, in all the cases studied, the algorithm converges to the optimal solution very early (before 50 iterations). For example, in BACP8, it begins with a load of 18 credits, to descend rapidly after two iterations to the minimum load of 17. For the case of BACP10, it starts at 16 credits, and after two iterations, it reaches the value of 15, to finally, after 38 iterations, it reaches the minimum load value of 14.

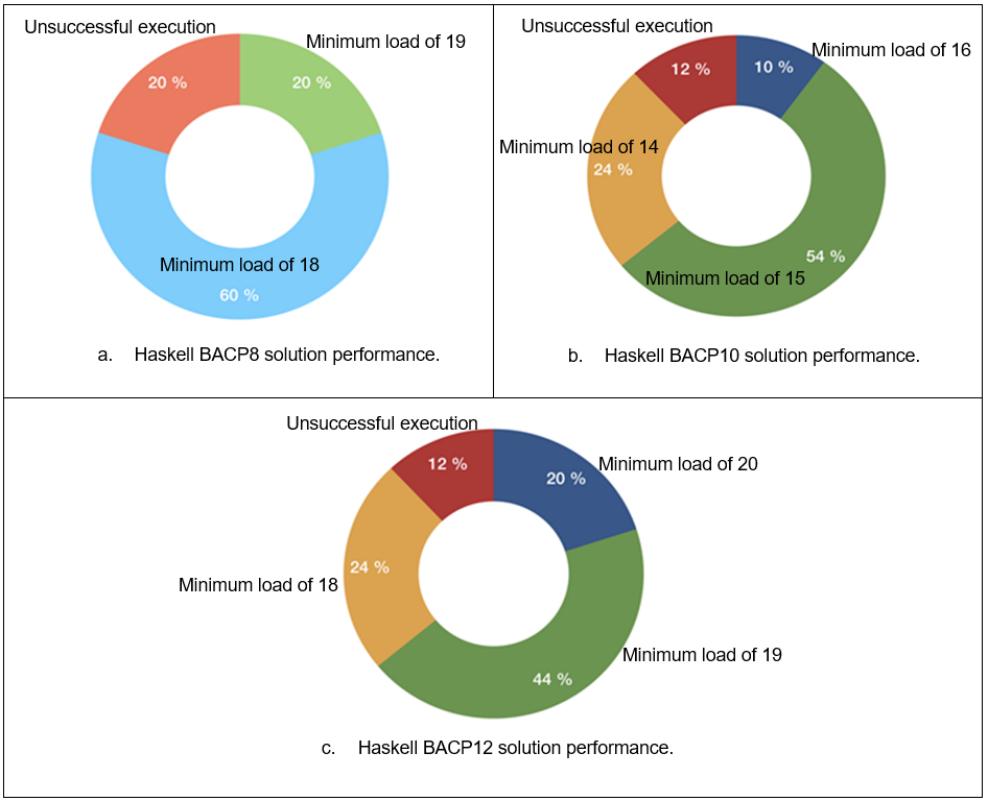

Fig. 6. Performance of Haskell BACP8, BACP10 and BACP12 solutions.

For BACP12, it starts at 20 credits, and after two iterations, it goes down to 19, getting after only ten generations, the optimal load of 18.The Haskell solution's great advantage is its functional nature for the definition of declarations of what one wants to obtain as a solution, not the imperative sequence of steps with the definition of variables and changes in their states to achieve the objective of the solution. Also, as exemplified by [Rubio et al. (2019)], [Chiarandini et al., (2012)], Haskell solutions even present adequate performance and usually some improvements compared to the same solutions in other programming languages such as C++ and Java.

5. Discussion

Since Haskell's random number generator is static and many of these numbers remain in memory once generated, their randomness is still difficult to control. Thus, it is challenging to develop a fully optimized program that solves the problem completely robust. We consider creating a tensor of random values in future developments to iteratively modify its dimensions depending on the size of the population, the random generation, and the execution of the solution.

Haskell is a language whose programming structure has the potential to optimize many tasks related to the calculation of long elements in a list. However, to reach a point where the type of structure of the problem can minimize the computational time until it becomes linear, we need to work with array structures. Solutions to work into those arrays can naturally run-in minimum processing cores in the machine. Precisely, the firefly algorithm solution of this work, with matrices, seeks to achieve this optimality. As indicated in the Results section, Haskell allows us to obtain a solution with a declarative character, to achieve the optimal results, even with greater efficiency in development and performance than solutions in other traditional programming languages. We implemented this work as part of an internal research project at a Chilean university.

6. Conclusions

This work presents an algorithmic solution to the BACP problem in the Haskell language based on the functional programming paradigm to demonstrate this approach's theoretical and practical feasibility. Thus, this work's greatest contribution is to present the viability of proposing functional solutions for optimization problems with restrictions such as the generation of a balanced academic load for study plans in higher

education institutions. All the tests carried out showed that it is possible to find solutions to the analyzed cases of CSPLib. In all test cases, the solutions' quality is satisfactory and allows them to obtain the best-known solution for each instance. Currently, we work to demonstrate the effectiveness of this approach to solve real cases of the problem and socialize the simplicity of applying functional solutions to optimization problems with a declarative nature.

References

1. Bird, R. (2010). *Pearls of Functional Algorithm Design*, 1st edn, Cambridge University Press, USA.
2. Bird, R. (2014). *Thinking Functionally with Haskell*, Cam-bridge University Press.
3. Cabrera-Guerrero, G.; Cabrera, E.; Soto, R.;León, J. M.; Crawford, B.; Paredes, F. (2012). *A hybrid approach using an artificial bee algorithm with mixed integer programming applied to a large-scale capacitated facility location problem*, Mathematical Problems in Engineering, vol.2012, 12.
4. Castro,C.; Manzano, S. (2001). *Variable and value ordering when solving balanced academic curriculum problems*, CoRR, vol. cs.PL/0110007.
5. Chiarandini, M.; Di Gaspero, L.; Gualandi, S.; Schaerf, A. (2012). *The balanced academic curriculum problem revisited*,Journal of Heuristics, vol. 18, no. 1, pp. 119–148.
6. Durkota, K. (2011). *Implementation of a discrete firefly algorithm for the qap problem within the sage framework*, in BachelorThesis, Czech Technical University.
7. García, J.; Crawford, B.; Soto, R.; Astorga, G. (2019). *A clustering algorithm applied to the binarization of swarm intelligence continuous metaheuristics*, Swarm and Evolutionary Computation, vol. 44, pp. 646 – 664.
8. Glatthorn,A. A.; Floyd, B.; Bruce W. (2018). *Curriculum Leadership: Strategies for Development and Implementation*, 5th edn, Los Angeles, USA: Sage.
9. Hnich,B.;Kiziltan, Z.; Walsh, T. (2002). *Modelling a balanced academic curriculum problem*, proceedings of CP-AI-OR-2002, pp. 121–131.
10. Hughes, J. (1989).*Why functional programming matters*, The Computer Journal, Oxford University Press, vol. 32, no. 2, pp. 98–107.
11. Hutton, G. (2016). *Programming in Haskell*, 2nd edn, Cambridge University Press, USA.
12. Kar, A. K. (2016).*Bio-inspired computing - a review of algorithms and scope of applications*, Expert Systems with Applications, vol. 59, pp. 20 – 32.
13. Lanza-Gutiérrez, J.; Crawford, B.; Soto, R.; Berríos, N.;Gómez-Pulido, J. A.;Paredes, F. (2016). *Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization*, Expert Systems with Applications, vol. 70, 11 2016.
14. Rubio, J. M.; Vidal-Silva, C. L.; Soto, R.; Madariaga, E.; Johnson, F.; Carter, L. (2019). *Applying fireflyalgorithm to solve the problem of balancing curricula*, International Journal of Advanced ComputerScience and Applications, vol. 10, no. 1.
15. Rubio, J. M.; Soto, R.; Jorquera, H.; Aguilar, J.; Vidal, C. (2018). *Solving the balanced academic curriculum problem using firefly algorithm*, Ingeniare, RevistaChilena de Ingeniería, vol. 26, pp. 102–112, 11.
16. Sommerville, I. (2010). *Software Engineering*, 9th edn, Addison-Wesley Publishing Company, USA.
17. Soto, R.; Crawford, B.; Aste-Toledo, A.; Fuente-Mella, H.; Castro, C.; Paredes, F.; Olivares, R.; Castillo, O. (2019). *Solving the manufacturing cell design problem through binary cat swarm optimization with dynamic mixture ratios*, Computational Intelligence and Neuroscience, vol. 2019.
18. Thompson, S. (1999). *The Haskell: The Craft of Functional Programming*, 2nd edn, Addison-Wesley Longman Publishing Co., Inc., USA.
19. Yang, X. S. (2010).*Firefly Algorithm, Lévy Flights and Global Optimization*, in Bramer M., Ellis R., Petridis M. (eds) Research and Development in Intelligent Systems XXVI, Springer, London.
20. AZADEH, YALDA, and B. E. H. N. A. M. MOHAMMADI-IVATLOO. "OPTIMAL HEAT AND POWER DISPATCH IN CO-GENERATION SYSTEMS USING FIREFLY ALGORITHM." TJPRC: International Journal of Power Systems & Microelectronics (TJPRC: IJPSM) 2.1 (2016) 77-86
21. MURAYAMA, TAKU. "LITERACY AND CURRICULUM IN THE UNITED STATES: A COMPARATIVE STUDY." International Journal of Educational Science and Research (IJESR) 6.1 (2016) 13-20
22. ARIGUSMAN, ANGGI. "DOMINANT FACTORS OF CURRICULUM INNOVATION: ENGLISH TEACHERS'BELIEFS." International Journal of English and Literature (IJEL) 8.3 (2018) 95-106
23. Ely, Luisita L. "Mastery Learning of Chemistry Competencies through the Spiral Progression Approach in Curriculum." International Journal of Educational Science and Research (IJESR) 9.5 (2019): 9-28

24. EZE, KENNETH O., CHRISTIAN S. UGWUANYI, and CHINEDU IO OKEKE. "Extent of the Upper Basic Education French Language Curriculum Content-Delivery with Technologies in Nigerian Secondary Schools." International Journal of Mechanical and Production Engineering Research and Development(IJMPERD) 10.4 (2020): 311-318.
25. Kulkarni, Shraddha. "Inclusion of Corporate Social Responsibility Practices as a Part of Curriculum@ Indira School of Business Studies, India and its Impact Over the Awareness & Sense of Responsibility of Students towards Community." International Journal of Business and General Management (IJBGM) 6.3 (2017): 1-18.