ISSN 2007-9737

A Linear Genetic Programming Approach for the Internet Shopping Optimization Problem with Multiple Item Units (ISHOP-U)

Jazmin Del-Angel, Alejandro Santiago, Salvador Ibarra-Martínez,
José Antonio Castán-Rocha*, Mayra Guadalupe Treviño-Berrones

Universidad Autónoma de Tamaulipas,
Facultad de Ingeniería Tampico,
Mexico

a2233338031@alumnos.uat.edu.mx, aurelio.santiago@uat.edu.mx,
{sibarram,jcastan, mgtrevino}@docentes.uat.edu.mx

Abstract. Evolutionary computation (EC) is a broad field of artificial intelligence where evolutionary processes inspire algorithms, such as artificial immune systems, inspired by the evolution of acquired immune systems. The predominant approach in EC is Evolutionary Algorithms (EAs), inspired by the evolution of Darwin's natural species. A different approach is Evolutionary Programming (EP), which, instead of evolving individuals representing the problem decision variables (chromosomes), evolves programs, which code instructions, and executing those instructions generates a solution. Genetic Programming (GP) is an approach analogous to Genetic Algorithms (GAs), but it differs in that it works over programming instructions instead of decision variables. Although GP is an exciting approach, it is more complicated to implement due to the necessity of managing tree data structures. Linear Genetic Programming (LGP) is more straightforward than traditional GP, without the need for tree data structures. This chapter shows a proof of concept to implement LGP to evolve programs for the Internet Shopping Optimization Problem with multiple item Units (ISHOP-U), an NP-Hard optimization problem. Readers can easily implement the proposed approach and produce Linear Genetic Programming algorithms for other problems.

Keywords. ISHOP-U, evolutionary programming, linear genetic programming.

1 Introduction

Genetic Algorithms are the more expansive Evolutionary Algorithm (EA) approach, generally used for search and parameter optimization problems based on sexual reproduction and the principle of survival of the fittest. To solve a problem, we start from an initial set of individuals, called a population, generated randomly. Each of these individuals represents a possible solution to the problem.

These individuals will evolve through environmental selection and adapt their characteristics according to a fitness function, improving it after generations. The population evolves towards similar characteristics to achieve a particular goal or objective function. EAs have proved to be effective in addressing real-world complex optimization problems (e.g., [23, 24]).

The Artificial Immune Systems [7, 28, 30, 8, 27, 31, 11, 9, 1, 12, 19] is an example of an Evolutionary Computation algorithm not inspired by genetics evolution, inspired by the adaptation of biological immune systems [18]. The EA analogy is that individuals represent decision variables.

Another subfield of Evolutionary Computation is Evolutionary Programming [10] (EP), where the

Algorithm 1 Genetic algorithms standard framework

while $\neg$ Stop condition **do**
  $Parent \leftarrow Selection(Pop)$
  $offSpring \leftarrow Crossover(Parent)$
  $Mutation(offSpring)$
  $Pop \leftarrow Pop \cup offSpring$
  $EnviromentalSelection(Pop)$
end while

Algorithm 2 Matrix S translation to program instructions

$Program \leftarrow$ an empty list of instructions
for $i = 1$ to m **do**
  for $j = 1$ to n **do**
    if $S_{i,j} > 0$ **then**
      $Program \leftarrow$ add the instruction $s_{i,j}$ equals to $S_{i,j}$ matrix value
    end if
  end for
end for
return $Program$

term programs replace individuals, and a program means a set of instructions to be executed, as in a computer program. In other words, EAs explicitly evolve the problem decision variables, while EP evolves operations that would implicitly conduct to the decision variables.

A subfield inside EP is Genetic Programming [4, 22, 13, 14, 15, 16] (GP). GP shares the sexual reproduction inspiration with the Genetic Algorithms (GAs). Therefore, the evolutionary operators of selection [3], crossover [26, 25, 20, 2], and mutation [29] also exist in GP, although a distinctive characteristic of GP is that chromosomes are of variable length. In addition, GP has the clone operator.

A clone operator is necessary because the environmental selection (survival of the fitness) is regularly different than in GAs, as the flowchart in Figure 1 shows, not allowing overlap in the generations between offspring and parents. Since GP inspiration is to evolve instructions and not decision variables, we found the above to be the authentic difference between the GA and GP approaches and not their frameworks. Although Genetic Programming [17] entails greater implementation complexity due to the management of tree data structures. Using tree data structures in GP requires tree traversal techniques to evaluate the programs, and the trees produced could be unbalanced, adding complexity.

Linear Genetic Programming [5] (LGP) offers a simpler alternative to traditional GP, dispensing with tree data structures; LGP evaluates programs sequentially. LGP uses variable-length linear data structures as the linked list to represent the programs, executing instructions sequentially in the list order, closing the gap between GP and the regular computer programming languages.

The assembly language has served as the metaphor for LGP; programs have registers (variables), read-only registers (constants), and instructions consist of operators (e.g., $+$,$-$,$\times$,$\div$) and operands, e.g., $r_3 = r_1 \times c_2$ where the destination register r_3 stores the result of the multiplication of the register r_1 with the constant register c_2.

We consider the GA framework well endorsed by artificial intelligence researchers and tested; we don't see a restriction not to use it directly with GP programs, as in Algorithm 1, instead of the non-overlapping between parents and offspring approach from the orthodox GP framework in Figure 1. Therefore, this work follows the GA framework instead of the orthodox GP flowchart. Generally speaking, in the words of John R. Koza: "The best computer program that appeared in any generation, the best-so-far solution, is designated as the result of genetic programming" [13].

This chapter presents a proof of concept for implementing LGP and developing programs that address the Internet Shopping Optimization Problem with Multiple Item Units [21] (ISHOP-U), an NP-Hard optimization problem. The proposed approach uses the GA framework for GP. It maps LGP instructions to the classical decision's variables representation in the GAs and the decision's variables to LGP instructions. The remainder of the chapter sections are as follows: Section 2 our Linear Genetic Programming proposal for the ISHOP-U.

ISSN 2007-9737

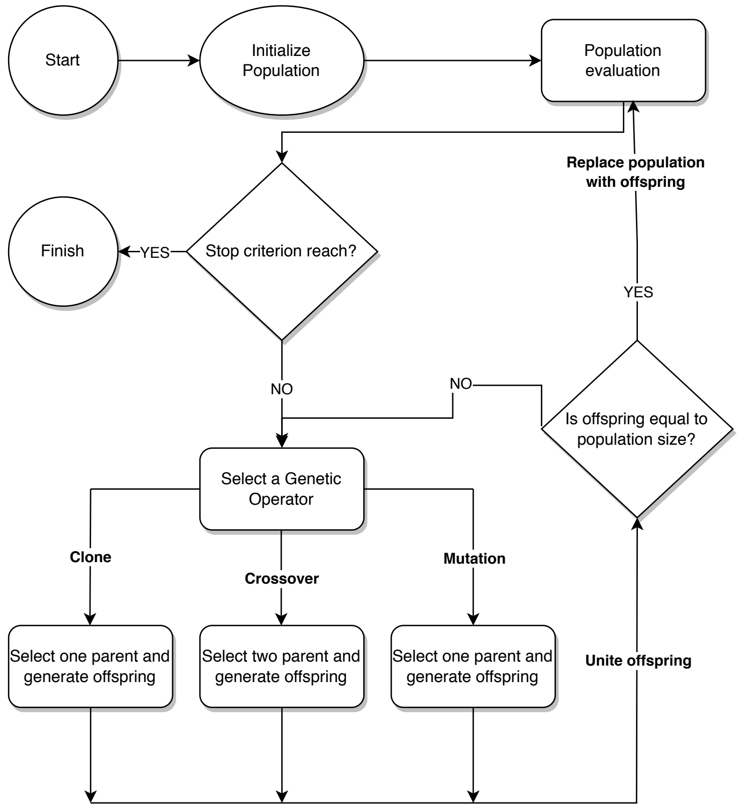

Fig. 1. Genetic programming orthodox flowchart

Section 3 is the experimental setup. Finally, Section 4 discusses and concludes the experimentation results.

2 Linear Genetic Programming for the ISHOP-U

This section describes our Linear Genetic Programming (LGP) proposal for the Internet Shopping Optimization Problem with multiple item Units (ISHOP-U). As the first attempt to apply LGP to the ISHOP-U, our approach is as simple as possible. Recalling the ISHOP-U formulation in [21] for a problem instance of m stores and n products, a candidate solution for the problem is a matrix S of size $m \times n$; we consider their components $s_{i,j}$ as variable registers in the LGP.

As constant registers, we consider the components $a_{i,j}$ of the product availability matrix A of size $m \times n$. The variable registers are all conditioned to be $s_{i,j} < a_{i,j}$, and the only operator implemented is assignation ($=$), e.g., $s_{i,j} = 5$, for a purchase of 5 units of the product j in the store i. The description of the implementation guidelines and evolutionary operators appears below.

2.1 Population Initialization

The first population of programs has a random initial length (number of instructions) between $[0, m \times n]$. Every instruction is filled with an assignation to random register $s_{i,j}$ with a random integer between $[0, a_{i,j}]$, e.g., $s_{1,2} = 5$ if and only if $a_{1,2} \geq 5$.

Table 1. Parameter setting for the LGP implementation

	LGP
Population size:	100
MaxEvaluations:	$25,000$
Selection:	Binary Tournament
Recombination:	$p_r = 1.0$
Mutation:	$p_m = 0.05$

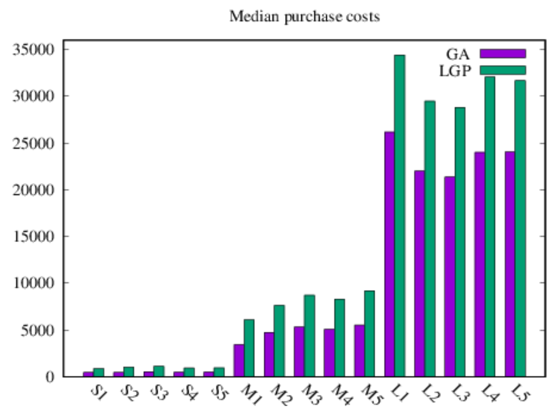

Fig. 2. Graph for the experimental results for the GA and LGP over the considered 15 instances

2.2 Crossover

Given two programs represented as linear vectors $\vec{x}$ and $\vec{y}$. The crossover operator copies in a single third vector $\vec{z}$ the individual instructions from $\vec{x}$ and $\vec{y}$. Later, the two new programs (child) inherit the instructions from $\vec{z}$. Every $\vec{z}$ instruction goes exclusively to the first or second child, with a uniform probability.

2.3 Mutation

The mutation operator uses a probability of mutation p_m to change every program instruction. Once an instruction is decided to be changed, their register $s_{i,j}$ is reassigned to a random integer value between $[0, a_{i,j}]$.

2.4 Repair

This LGP proposal for the ISHOP-U uses the same repair method for unfeasible solutions as the Genetic Algorithm from [21].

The above is possible by executing the program instructions to load the matrix S as an intermediate representation and then repairing S. Once repaired, S is translated to program instructions using the following Algorithm 2.

Algorithm 2 has the advantage of producing a compact program representation in $O(m \cdot n)$ complexity. Unnecessary instructions do not appear in the program, i.e., where $s_{i,j} = 0$ or duplicated register assignations. Slight modifying Algorithm 2 can produce instructions from decision variables with the same complexity. Figure 3 graphically shows the crossover, mutation, and repair process when performing over the decision variables.

2.5 Environmental Selection

The environmental selection is equivalent to the one found in traditional Genetic Algorithms (GAs). The current population and the offspring join to form a single population. Later, according to the fitness value of programs, only the N best programs survive to the next generation, where N is equal to the population size.

3 Experimental Configuration

This section gives the details to reproduce this chapter's experimentation. To evaluate our proposed Linear Genetic Programming (LGP) algorithm, we use the benchmark instances in [21] of 10 products with 25 stores (five small size instances), 25 products with 50 stores (five medium size instances), and 50 products with 100 stores (five large size instances).

Instances are available at https://github.com/AASantiago/ISHOP-U-Instances, and LGP source code is available at https://github.com/AASantiago/LGP-ISHOP-U/. This chapter LGP implementation uses the parameter settings from Table 1, a population size of 100, 25,000 as a maximum number of objective function evaluations, Binary Tournament as selection, 100% of crossover probability p_r and a mutation probability p_m of 5%. We reproduce the Genetic Algorithm (GA) described in [21], with their respective parameter settings for comparison purposes.

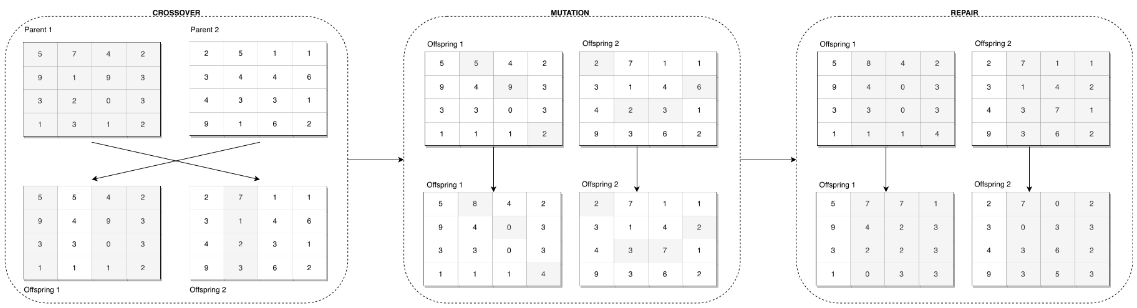

Fig. 3. Graphical representation of the evolutionary operators

Table 2. Experimental results for the GA and LGP over the considered 15 instances

Problem	GA	LGP
UniformS1	$447.2_0.0E0$	$863.3_3.1E1$
UniformS2	$447.2_0.0E0$	$1023.0_6.5E1$
UniformS3	$524.2_7.8E-1$	$1103.6_8.5E1$
UniformS4	$462.9_0.0E0$	$928.4_7.0E1$
UniformS5	$489.6_0.0E0$	$955.1_8.2E1$
UniformM1	$3429.3_7.9E1$	$6080.4_1.7E2$
UniformM2	$4721.1_1.2E2$	$7644.1_3.1E2$
UniformM3	$5301.3_1.1E2$	$8717.9_1.9E2$
UniformM4	$5068.5_1.1E2$	$8299.3_2.5E2$
UniformM5	$5509.3_1.8E2$	$9197.7_1.7E2$
UniformL1	$26161.2_5.8E2$	$34419.5_1.2E3$
UniformL2	$21999.1_4.3E2$	$29470.2_5.1E2$
UniformL3	$21361.1_3.8E2$	$28811.2_5.4E2$
UniformL4	$23991.8_4.7E2$	$32061.2_1.0E3$
UniformL5	$24036.8_4.0E2$	$31671.4_8.1E2$

Due to the stochastic nature of the algorithms, we perform 30 independent runs over every considered ISHOP-U instance. To validate the statistical significance difference between the GA and LGP, we perform the Wilcoxon signed rank test [6], with an $\alpha = 0.05$ for a 95% of significance.

4 Results

This section outlines and discusses the numerical experimental results from the 30 independent executions of both algorithms in comparison GA and the LGP.

The experimentation numerical results are in Table 2 in terms of the achieved median value and interquartile range (IQR), with the format $MEDIAN_{IQR}$ and IQR in scientific notation. The median results are graphically show in Figure 2.

The instance name nomenclature starts with the distribution of the prices in the instances Uniform, followed by the instance size, S small, M Medium, L large, followed by their identification number; for example, UniformS1 is the instance number one of small size with a uniform distribution in their prices.

According to the results computed by the Wilcoxon signed rank test, a p-value≤ 0.05 is found in every considered instance, finding differences with a statistical significance of 95% between the GA and the LGP. The above difference favors the GA with a better achieved median value in all the instances, in the demerit of the LGP.

Given the computed numerical results, the Genetic Algorithm (GA) in [21] outperforms this chapter's Linear Genetic Programming proposal with statistical significance in the fifteen instances with uniform distribution prices in [21].

However, the purpose of this chapter is to prove that the concept of using Linear Genetic Programming (LGP) is feasible for the Internet Shopping Optimization Problem with multiple item Units purpose achieved. We highlight that LGP uses the same repair method as the GA, which is possible through an intermediate representation (the original problem decision variables) for later return to the instructions representation once feasible.

Our approach reutilizes evolutionary algorithm operators in evolutionary programming and can be applied to other problems straightforwardly. For future research, we want to design and implement new crossover and mutation operators, using the proposed mapping approach between decision variables and instructions for LGP to improve the results.

Acknowledgments

Alejandro Santiago would like to thank CONAHCYT Mexico for the SNII salary award.

References

1. **Aickelin, U., Dasgupta, D., Feng, G. (2013).** Artificial immune systems. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer US, Boston, MA, pp. 187–211. DOI: 10.1007/978-1-4614-6940-7\7.

2. **Amador-Larrea, S., Quiroz-Castellanos, M., Hoyos-Rivera, G., Mezura-Montes, E. (2022).** An experimental study of grouping crossover operators for the bin packing problem. Computación y Sistemas, Vol. 26, No. 2, pp. 663–682. DOI: 10.13053/cys-26-2-4249.

3. **Arellano-Verdejo, J., Guzmán-Arenas, A., Godoy-Calderon, S., Barrón-Fernández, R. (2014).** Efficiently finding the optimum number of clusters in a dataset with a new hybrid cellular evolutionary algorithm. Computación y Sistemas, Vol. 18, No. 2, pp. 313–327.

4. **Banzhaf, W., Nordin, P., Keller, R. E., Francone, F. D. (1998).** Genetic programming: an introduction: on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers Inc. DOI: 10.5555/323917.

5. **Brameier, M., Banzhaf, W. (2007).** Linear genetic programming. Genetic and Evolutionary Computation, Springer US.

6. **Corder, G. W., Foreman, D. I. (2011).** Nonparametric statistics for non-statisticians: A step-by-step approach. Wiley. DOI: 10.1002/9781118165881.

7. **Dasgupta, D. (1999).** Artificial immune systems and their applications. Springer Berlin, Heidelberg.

8. **Dasgupta, D. (2006).** Advances in artificial immune systems. IEEE Computational Intelligence Magazine, Vol. 1, No. 4, pp. 40–49. DOI: 10.1109/MCI.2006.329705.

9. **Dasgupta, D., Yu, S., Nino, F. (2011).** Recent advances in artificial immune systems: Models and applications. Applied Soft Computing, Vol. 11, No. 2, pp. 1574–1587. DOI: 10.1016/j.asoc.2010.08.024.

10. **Eiben, A. E., Smith, J. E. (2015).** Introduction to evolutionary computing. Springer.

11. **Greensmith, J., Whitbrook, A., Aickelin, U. (2010).** Artificial immune systems. Handbook of Metaheuristics, pp. 421–448. DOI: 10.1007/978-1-4419-1665-5\14.

12. **Jenhani, I., Elouedi, Z. (2014).** Re-visiting the artificial immune recognition system: a survey and an improved version. Artificial Intelligence Review, Vol. 42, pp. 821–833. DOI: 10.1007/s10462-012-9360-0.

13. **Koza, J. R. (1992).** Genetic programming: On the programming of computers by means of natural selection. MIT Press.

14. **Koza, J. R. (1994).** Genetic programming II: Automatic Discovery of Reusable Programs. MIT Press.

15. **Koza, J. R. (1999).** Genetic programming III: Darwinian invention and problem solving. Morgan Kaufmann.

16. **Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., Lanza, G. (2005).** Genetic programming IV: Routine human-competitive machine intelligence, Vol. 5, Springer New York, NY.

17. **Langdon, W. B., Poli, R., McPhee, N. F., Koza, J. R. (2008).** Genetic programming:

An introduction and tutorial, with a survey of techniques and applications. Computational Intelligence: A Compendium, pp. 927–1028. DOI: 10.1007/978-3-540-78293-3_22.

18. **Monroy, R., Saab, R., Godínez, F. (2004).** On modelling an immune system. Computación y Sistemas, Vol. 7, No. 4, pp. 249–259.

19. **Nebili, W., Farou, B., Kouahla, Z., Seridi, H. (2021).** Revised artificial immune recognition system. IEEE Access, Vol. 9, pp. 167477–167488. DOI: 10.1109/ACCESS.2021.3133731.

20. **Nimbiwal, M., Vashishtha, J. (2021).** A novel hybrid grey wolf optimization algorithm using two-phase crossover approach for feature selection and classification. Computación y Sistemas, Vol. 25, No. 4, pp. 793–801. DOI: 10.13053/cys-25-4-3931.

21. **Ornelas, F., Santiago, A., Martínez, S. I., Ponce-Flores, M. P., Terán-Villanueva, J. D., Balderas, F., Rocha, J. A. C., García, A. H., Laria-Menchaca, J., Treviño-Berrones, M. G. (2022).** The internet shopping optimization problem with multiple item units (ISHOP-U): Formulation, instances, np-completeness, and evolutionary optimization. Mathematics, Vol. 10, No. 14. DOI: 10.3390/math10142513.

22. **Poli, R., Langdon, W. B., McPhee, N. F. (2008).** A field guide to genetic programming.

23. **Rivera, G., Cisneros, L., Sánchez-Solís, P., Rangel-Valdez, N., Rodas-Osollo, J. (2020).** Genetic algorithm for scheduling optimization considering heterogeneous containers: A real-world case study. Axioms, Vol. 9, No. 1, pp. 27. DOI: 10.3390/axioms9010027.

24. **Rivera, G., Cruz-Reyes, L., Fernandez, E., Gomez-Santillan, C., Rangel-Valdez, N., Coello, C. A. C. (2023).** An ACO-based hyper-heuristic for sequencing many-objective evolutionary algorithms that consider different ways to incorporate the DM's preferences. Swarm and Evolutionary Computation, Vol. 76, pp. 101211. DOI: 10.1016/j.swevo.2022.101211.

25. **Romero-Ruiz, E., Segura, C. (2018).** Memetic algorithm with hungarian matching based crossover and diversity preservation. Computación y Sistemas, Vol. 22, No. 2, pp. 347–361. DOI: 10.13053/cys-22-2-2951.

26. **Singh, D. R., Singh, M. K., Singh, T., Prasad, R. (2018).** Genetic algorithm for solving multiple traveling salesmen problem using a new crossover and population generation. Computación y Sistemas, Vol. 22, No. 2, pp. 491–503. DOI: 10.13053/cys-22-2-2956.

27. **Timmis, J., Hone, A., Stibor, T., Clark, E. (2008).** Theoretical advances in artificial immune systems. Theoretical Computer Science, Vol. 403, No. 1, pp. 11–32. DOI: 10.1016/j.tcs.2008.02.011.

28. **Timmis, J., Knight, T., de-Castro, L. N., Hart, E. (2004).** An overview of artificial immune systems. Computation in Cells and Tissues: Perspectives and Tools of Thought, Springer Berlin Heidelberg, pp. 51–91. DOI: 10.1007/978-3-662-06369-9\4.

29. **Torres-Soto, A., Ponce-de-León-Sentí, E. E., Hernández-Aguirre, A., Torres-Soto, M. D., Díaz-Díaz, E. (2010).** Un sistema evolutivo robusto para la síntesis de circuitos analógicos. Computación y Sistemas, Vol. 13, No. 4, pp. 409–421.

30. **Watkins, A., Timmis, J., Boggess, L. (2004).** Artificial immune recognition system (AIRS): An immune-inspired supervised learning algorithm. Genetic Programming and Evolvable Machines, Vol. 5, pp. 291–317. DOI: 10.1023/B:GENP.0000030197.83685.94.

31. **Zheng, J., Chen, Y., Zhang, W. (2010).** A survey of artificial immune applications. Artificial Intelligence Review, Vol. 34, No. 1, pp. 19–34. DOI: 10.1007/s10462-010-9159-9.

Article received on 28/02/2024; accepted on 15/05/2024.
**Corresponding author is José Antonio Castán Rocha.*