Artificial Intelligence-Driven Cargo Optimization Service for Logistics

Solomiia Fedushko¹, Yuriy Syerov^{1,2} and Michal Gregus¹

¹ Department of Information Management and Business Systems, Faculty of Management, Comenius University Bratislava, Odbojárov 10, 820 05 Bratislava, Slovakia

² Social Communication and Information Activity Department, Lviv Polytechnic National University, Bandera 12, 79000 Lviv, Ukraine

Abstract

The development of forwarding services is becoming more and more dynamic every day. Observing this trend, we should look for solutions that will provide customers with comfort, speed, and quality of service. This article introduced an intelligent forwarder model that runs as a web application. The solution model and methods used to implement the system and the algorithm that uses elements of artificial intelligence are presented. In logistics, a freight forwarder is a nonvessel operating common carrier that helps coordinate shipments and ensure the smooth transportation of goods from producers or manufacturers to end-users, markets, or final distribution points for corporations or individuals. Software that is based on artificial intelligence can make the forwarder's work more effective. The particle swarm optimization algorithm is a heuristic optimization method inspired by the collective behavior of social agents observed in nature, such as bird flocking, bee swarming, and fish schooling. In this paper, we propose a Cargo Optimizer service that is an expert system supporting determining the most profitable sets of orders using AI - more precisely, it will be the particle swarm optimization algorithm. In the scope of these tests, we did some actions on the level of the graphical user interface to test the connectivity of the whole system. Tests prevent unexpected operations and make the system safe and optimal. The comparison will consist of comparing the algorithms in terms of speed and quality of the data obtained.

Keywords

Artificial intelligence, cargo optimization, logistics, particle swarm optimization, freight forwarding

1. Introduction

Logistics [1] is currently one of the fastest-growing industries. It involves planning and implementing the flow of products from the point of production to the point of sale or consumption. Freight forwarders coordinate the supply chain so that cargo is transported optimally. They find employment in many areas of the economy, including trade, services, and shipping. It is a very responsible job, with no room for error. In a situation where such a mistake occurs, it can be very costly for the shipping company [2, 3]. Many transport companies are regularly established around the world. They are entering any city where the squares are a typical sight. However, it is not only the domain of cities because in the surrounding areas, and even in many smaller towns, the view of various transport companies is seen, not to mention the hundreds of trucks encountered on the roads daily.

Regarding shipping companies, the increase in their number results from the demand for shipping services. If a given country's economy is developing at a fast pace, the consequence of this is an increasing number of production plants and warehouses. It is connected with the need to transport the goods they produce to various destinations, both within and outside of this country [4]. The IT industry has a more substantial impact on shipping services every day.

CEUR Workshop Proceedings (CEUR-WS.org)

ProfIT AI 2023: 3rd International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2023), November 20–22, 2023, Waterloo, Canada

Solomiia.s.fedushko@lpnu.ua (S. Fedushko); yurii.o.sierov@lpnu.ua (Yu. Syerov); michal.gregus@fm.uniba.sk (M. Gregus)

 ^{0000-0001-7548-5856 (}S. Fedushko); 0000-0002-5293-4791 (Yu. Syerov); 0000-0002-8156-8962 (M. Gregus)
 © 000 0002-0002-8156-8962 (M. Gregus)
 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Programmers create software [5] that allows you to replace the activities that the forwarder does daily. Among other things, software based on artificial intelligence can make the forwarder's work more effective. In computer science [6-8], artificial intelligence (AI) [9-11] represents a significant area of exploration. Its focus lies in developing computer systems capable of emulating human-like cognitive processes, including learning, reasoning, and self-correction.

An Application Programming Interface (API) [12] serves as a software intermediary that facilitates communication between two applications. Traditionally, a freight forwarder [13-16] acts as an intermediary connecting shippers or importers with carriers. In a broader context, a freight forwarder, also known as a nonvessel operating common carrier, assists in coordinating shipments and ensuring the smooth transportation of goods from producers or manufacturers to end-users, markets, or final distribution points for corporations or individuals.

The Knapsack problem [17] involves optimizing the packing of a knapsack with integer volume F using objects from K different classes to maximize profit. Objects from class k, where k = 1, ..., K, consume integer units of knapsack volume and produce a profit rk. If the volume of the knapsack F is an integer multiple of the volumes bk, k = 1, ..., K, a straightforward solution is to fill the knapsack with objects from the class k with the highest profit-to-volume ratio rk/bk. Dynamic programming can address cases where the knapsack volume ratio is not an integer multiple of object volumes with a time complexity of O(FK).

The particle swarm optimization algorithm (PSO) [18] is a heuristic optimization method inspired by the collective behavior of social agents observed in nature, such as bird flocking, bee swarming, and fish schooling. The freight market [19] serves as the nexus where buyers and sellers of shipping services converge to negotiate deals. Various categorizations can be applied to this marketplace.

2. Related Work

The optimization of cargo logistics through artificial intelligence (AI) has drawn inspiration from various swarm intelligence models and collective decision-making mechanisms observed in natural systems. A diverse range of studies has explored the application of swarm-based algorithms in different contexts, providing valuable insights into the potential enhancements achievable through mimicking biological phenomena.

Okubo's seminal work [20] delves into the dynamic aspects of animal grouping, shedding light on swarms, schools, flocks, and herds. Although focused on biological systems, the principles of self-organization and coordinated behavior in groups offer conceptual foundations applicable to artificial systems, especially in optimizing logistics operations.

Schumann [21] extends this perspective by exploring the transition from swarm simulations to swarm intelligence. This transition highlights the relevance of understanding collective behaviors and adaptive strategies exhibited by swarms, paving the way for informed design choices in AI-driven cargo optimization services.

Karaboga et al. [22] present a comprehensive survey of the artificial bee colony (ABC) algorithm, showcasing its potential applications. While the ABC algorithm is rooted in the foraging behavior of honeybees, its adaptability and efficiency have inspired applications in optimization problems, including those encountered in logistics.

Ventocilla's work [23] introduces a swarm-based approach to area exploration and coverage inspired by pheromones and bird flocks. This research provides valuable insights into decentralized decision-making mechanisms that can be harnessed for efficient logistics resource allocation and route planning.

Pourpanah et al. [24] comprehensively review artificial fish swarm algorithms, emphasizing recent advances and practical applications. The piscine-inspired algorithms offer novel perspectives for optimizing cargo routing and resource allocation in logistics scenarios.

Bakar [25] explores the understanding of collective decision-making in natural swarm systems, presenting applications and challenges. This work contributes to the theoretical

foundations of swarm intelligence, offering a deeper comprehension of decision-making processes that can inform the design of AI-driven cargo optimization services.

The literature further includes contributions from Sadiku and Musa [26], Li and Clerc [27], and Zhang et al. [28], offering insights into multiple intelligences, swarm intelligence handbooks, and applications in various domains. These works provide a rich background for integrating swarm intelligence principles into AI-driven cargo optimization.

Finally, Abidin et al. [29] introduce swarming robotics and discuss emerging trends in application development. This work broadens the scope of swarm intelligence, suggesting potential synergies between AI-driven cargo optimization and robotic swarm systems for efficient logistics operations.

The analysis of these works contributes to a holistic understanding of swarm intelligence models, offering a diverse array of inspirations and methodologies applicable to developing an artificial intelligence-driven cargo optimization service for logistics.

3. Description of the use of swarm optimization algorithm in the Cargo Optimizer

Cargo Optimizer would be an expert system supporting determining the most optimal trailer loading configurations. It would fetch data from the Distance Matrix API from Google Maps and data from the customer. Customers using our service (let us assume that it is some expedition/transportation company) would send a request to the freight market containing filters used to orders (max and min weight, max and min volume, max and min cost, location, and search radius). After getting a list of orders, he can choose which to send to Cargo Optimizer.

The user has to add the maximum weight and volume of the trailer to the request with the order list to Cargo Optimizer [30]. That way, we will obtain two lists of the most profitable orders (best income/cost ratio) that can be done in one go. This service's logic would collect data from clients and external services to generate the most profitable sets of orders using AI. More precisely, it will be the particle swarm optimization algorithm. The PSO algorithm can be used to optimize the function of many variables, and in this case, it will be used to find a set of goods that will fill the cargo space most cost-effectively.



Figure 1: UML Component Diagram for Cargo Optimizer

Figure 1 shows the dependencies between the components. There is a Graphical User Interface mainly on the outside (on the customer side). It directly connects to the FreightMarketDataFetcher and our main component, Cargo Optimizer. The latter has a connection to the RoutingDataFetcher. The PSO algorithm adapted to the needs of the cargo optimizer will retrieve data on available orders at the input. Each particle will contain a drawn set of goods with attributes such as cost, weight, cubic capacity, distance in meters, and duration in minutes between depot and destination. The study contains the fitness value returned by the target function for this set of goods. By target function, we mean a function that will take the list of goods as an input parameter and then convert the profitability value based on their attributes and return it as fitness. Apart from the attributes listed above, each particle will have the best set of goods remembered and the related fitness. Furthermore, the best set of goods will be remembered for the entire swarm, which will be used in the optimization process to update the speed and position of the particles. The particles will be updated according to the assumptions of

the original PSO, and the condition for the algorithm to end is a specified number of iterations, after which two of the most profitable lists of orders will be returned.

A trailer with a maximum weight of W, a maximum volume of V, and a set of N elements $\{x_1, x_i, ..., x_N\}$, each element having a specific value c_i , weight w_i , and volume v_i .

maximize
$$\sum_{i=1}^{N} c_i x_i$$
, where $x_i = 0, 1$ and $i = 1, ..., n$ (1)

with constraints:
$$\sum_{i=1}^{N} w_i x_i \le W$$
 and $\sum_{i=1}^{N} v_i x_i \le V$ (2)

(3)

The function which is responsible for determining the value of an order has been determined as follows:

$$value = \frac{cost}{weight + volume + distance + duration}$$

The cost, weight, volume, distance, and duration at the beginning of the program are scaled to a range of [0,1], so each has the same meaning. Therefore, in this case, the optimization task is based on finding a list with a maximum value, which is the sum of the values of all elements in that list. To provide more freedom for the user, to return the list with the second highest value so that he can finally decide which list of orders suits him better.

3.1. PSO algorithm adapted to the Cargo Optimizer's diagram and pseudocode

Present a workflow diagram better to illustrate the logic operation in the Cargo Optimizer. Furthermore, it was decided to present the logic operation differently. The pseudocode that describes the contents of Figure 3.

```
Initialization phase:
User sends a request to the freight market and receives an order list
  User introduces order list and restrictions in the form of maximum weight and
loading cubic capacity
  Attributes of each order (cost, weight, volume, distance, duration) are normalized
to the range [0, 1]
  Creating particles and drawing the initial speed and order list for each of the
particles (position)
         The particle consists of:
                 a position vector equal in length to the number of all orders
             entered by the user, assuming binary values
                 - a speed vector equal in length to the number of all orders entered
             by the user, assuming floating point values from the range {<}V_{\text{min}} , V_{\text{max}}{>}
  Calculate the fitness for each particle
  Saving the drawn order list and the calculated fitness as the best local fitness
for each particle
  Selecting the best global solution from among all particles
  Proper operations:
  Until the specified number of cycles of the PSO algorithm is completed:
      Update of the speed of each particle
         Flags representing a given order in each particle are changed based on the
      calculated fitness function
      If found solution meets the set constraints:
                If found solution is better than the best global solution:
                       Update of the best global solution
                    Update of the best local solution
                Else if found solution is better than the second best global solution:
                    Update of the second best global solution
             Else if found solution is better than the best local solution:
                    Update of the best local solution
  Return two list of orders with the best fitness
```

Figure 2: Pseudocode for Cargo Optimizer



Figure 3: Workflow diagram for Cargo Optimizer

The pseudocode presented in Figure 3. contains instructions on how to use the PSO algorithm implemented for Cargo Optimizer. This is how we tried to explain its operation step by step. What is worth noting is the fact that the attributes of the order and the way in which the value of a given order differs from the original backpacking problem. Returning the two best-order lists is also our bonus.

3.2. Automatic Tests

The cargo optimizer, as an expert system, should be reliable. It is essential to prove how reliable, fault-proof and optimized it is. To achieve this, the system will be tested in many ways, both manually and automatically.

Automatic tests are vital to developing and deploying any modern software project. It was created using particular testing frameworks, usually written in the same programming language as the program. These testing frameworks allow us to make assertions regarding parts of code, which usually define some particular logic, e.g., computing average value or realization of sorting algorithm. We can compare the characteristics returned values by function and check them against our expectations. It is also possible to mock-test objects and monitor their internal behavior, such as the order of called functions or used arguments [31].

3.2.1. Unit Test

Unit testing is the most basic technique of all testing approaches. It tests tiny fragments of code, primarily single functions or methods that fulfill one purpose. Outside dependencies, such as web services [32, 33] or database connections, are avoided. If some outside dependencies are mixed with logic, they are replaced with mocked connections returning some fixed values [34].

RoutingDataFetcher



Figure 4: UML component diagram - between the components and the RoutingDataFetcher component marked with red color

Figure 4 shows the dependencies between the components and the RoutingDataFetcher component marked in red. We posted it to illustrate which component is currently being tested.

In the scope of unit tests for RoutingDataFetcher, we tested internal functionalities of this component, such as:

- Parsing location-related data obtained from an external API,
- parsing responses from external API,
- computation of distance in a straight line between two points on the surface of the earth,
- construction of addresses for requests to external API.

FreightMarketDataFetcher



Figure 5: Example figure caption - Dependencies between the components and the FreightMarketDataFetcher component marked with red color

Figure 5 shows the dependencies between the components and the FreightMarketDataFetcher component marked in red. We posted it to illustrate which component is currently being tested.

In the scope of unit tests for RoutingDataFetcher, we tested internal functionalities of this component, such as:

- the randomness of generated lists of orders,
- removal of diacritics,
- parsing contents of files with customer/location-related data.

CargoOptimizer



Figure 6: UML component diagram - dependencies between the components and the CargoOptimizer component marked in red.

Figure 6 shows the dependencies between the components and the CargoOptimizer component marked in red. We posted it to illustrate which component is currently being tested.

In the scope of unit tests for RoutingDataFetcher, we tested internal functionalities of this component, such as:

- realization of the PSO algorithm,
- Normalization of the input data.

3.2.2. Integration tests

The test goal is to execute the system to verify its behavior and reveal possible failures. The integration testing phase is performed to find errors in the unit interfaces [35] and systematically build up the entire software system structure [36 -38]. For example, in our system, one of the components is Cargo Optimizer, whose purpose is to implement a PSO algorithm to compute a list of the most profitable order sets.

In the case of this component, we will check whether it behaves as expected on the level of the enabled interface, but we will not check what exact processes are going on while producing results.

RoutingDataFetcher and Distance Matrix API



Figure 7: UML component diagram – dependencies between the components. The RoutingDataFetcher and Distance Matrix API are marked in red.

In Figure 7, we can see the dependencies between the components. RoutingDataFetcher and Distance Matrix API are marked red to show what part of the project was tested.

In the scope of integration tests for RoutingDataFetcher, we tested the following:

• connectivity between the internal service of our system (RoutingDataFetcher) and the external Distance Matrix API.

CargoOptimizer and RoutingDataFetcher



Figure 8: UML component diagram – dependencies between the components. The cargo optimizer and the routingDataFetcher are marked red.

In Figure 8, we can see the dependencies between the components. CargoOptimizer and RoutingDataFetcher are marked red to show what part of the project was tested.

In the scope of integration tests for RoutingDataFetcher, we tested the following:

• connectivity between internal services: CargoOptimizer and RoutingDataFetcher.

3.2.3. Performance tests

Performance tests are necessary when a system is based on performance-dependent functionalities. Successful products should not force customers to wait long for desired results or exceed some designated limit. In our system, it is strictly bound to the execution time of the PSO algorithm.

For different sets of inputs, we will require that results be produced within some designated time limit [39].



Figure 9: UML component diagram - Dependencies between the components and the CargoOptimizer component marked in red color

Figure 9 shows the dependencies between the components and the CargoOptimizer component marked in red. We posted it to illustrate which component is currently being tested.

In the scope of unit performance for RoutingDataFetcher, we tested the following:

- speed and scalability of the realization of the PSO algorithm for different input sizes,
- stability of the provided realization of the PSO algorithm.

3.2.4. End-to-end tests

End-to-end tests are used to check the functioning of the whole system, checking logic enabled for customers. In this case, all dependencies are included in the testing process: the system's internal components, read/write operations on files, connections to third-party data providers,

and whatever is needed for the provided system. Put bluntly, this kind of testing checks whether a provided product fulfills its purpose [40].



Figure 10: UML component diagram – dependencies between the components

Figure 10 shows the dependencies between the components. We mark the components tested during the E2E test with a red rectangle. In the scope of this test, we use a client in the form of a program that uses a testing framework instead of the graphical user interface to simplify the testing process.

In the scope of unit performance for RoutingDataFetcher, we tested the following.

• overall connectivity and correctness of the propagated data between system components.

4. Manual tests

These tests are conducted without using remarkable testing and frameworks, automating testing and making test cases repeatable. Usually, manual tests are performed by company or beta testers, which are used to simulate the behaviors of actual customers. It means that what is tested is the enabled user interface (mainly graphical user interface), its logic, reliability, and fault-proofness. An important part of manual testing is the evaluation of user experience [41] – User Experience is an area focused on maximizing the comfortability of interfaces enabled for end users [42].



Figure 11: UML component diagram - manual tests with a red rectangle

Figure 11 shows the dependencies between the components. We marked the components being tested during manual tests with a red rectangle.

In the scope of these tests, we performed some actions on the level of the graphical user interface to test the connectivity of the entire system and enabled the elements of the graphical interface.

Figure 12 shows a manual system test to check if a user can proceed to the next step without filling in the required data. Missing data will appear on the screen, so the test was successful.

	Transportation designer
୍ତ୍ତି Transportation designer	Insert filters
Grders	Insert filters and values, which will be used to search for orders
Reports	Add new fitter: 🚺 Minimal weight 👻 😜
abs Companies	Maximum weight boot Maximum weight boot Maximum weight doo Maximum value: doo Maximum value: doo Fields orders Fields orders Grade Value Client Route Volume Weight Fields orders Maximum value: doo doo 100 542 32 2354 Taine Ryls 21 135 562 So 22 So 22 So 23 128 136 600 75 128 600 600 75 128 600 600 75 128 600 600 75 128 600 600 75 128 600 600 75 128 600
	Insert restrictions
	Choose orders

Figure 12: The screenshot shows a manual test.



Figure 13: The screenshot shows the test result.

Figure 13 shows a screenshot of a manual test to retrieve information on the optimal order lists generated based on previously entered data.

The test was successful, and two best-order lists were displayed on the screen.

4.1. Queuing model

In this report, we also decided to present a queueing model. For analysis, we have selected the main component of our service, Cargo Optimizer, because it is the most important to our customers.

To calculate the queuing model factors that allow us to evaluate the operation of our service, we decided to simulate the operation and estimate the input data. M/M/c was selected as the queueing model. As for the arrival rate λ , we chose a value of 5 requests per second, the service rate μ equals 3, and the number of servers m of 2.

The calculation of coefficients is as follows [43]:

• utilization of the server (4):

$$\rho = \frac{\lambda}{m * \mu} = 0,833 \tag{4}$$

• profitability of an empty system (5):

$$P_0 = 1 / \left[\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{c! * (1-\rho)^2} \right] = 0,091$$
(5)

• mean number of customers in the queue (6):

$$L_{q} = \frac{P_{0} * \left(\frac{\lambda}{\mu}\right)^{m} * \rho}{m! * (1 - \rho)^{2}} = 3,788$$
(6)

• mean wait in the queue (7):

$$W_q = \frac{L_q}{\lambda} = 0,758\tag{7}$$

• mean wait in the system (8):

$$W = W_q + \frac{1}{\mu} = 1,091$$
(8)

• mean number of customers in the system (9):

$$L = \lambda * W = 5,455 \tag{9}$$

5. Conclusion

The intelligent forwarder system is created according to previously planned assumptions. The system communicates with external API's and processes data received from them. Communication occurs between the components that store data, those that process it, and those responsible for displaying it. An essential element of the system is an algorithm based on artificial intelligence. It is a particle swarm optimization algorithm whose task is to return the optimal list of orders based on the data entered by the user.

The system implementation also includes many tests. Tests help to prevent unexpected operations and make the system safe and optimal.

The Cargo Optimizer module, which uses elements of artificial intelligence and is the essential point of the system, also uses a queueing model. This is to improve the system performance for multiple users.

The system can be improved in the future by using a different algorithm for the cargo optimizer. A possible algorithm to use is a genetic algorithm, which operates similarly to the current algorithm. The two algorithms can then be compared based on their speed and the quality of data obtained.

Acknowledgments

This study is funded by EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under project No. 09I03-03-V01-000153.

References

- H. Neng Chiu, The integrated logistics management system: a framework and casestudy. International Journal of Physical Distribution & Logistics Management, 25(6), 4-22 (1995). https://doi.org/10.1108/09600039510093249
- [2] D. Lyridis, T. Fyrvik, G. Kapetanis, N. Ventikos, P. Anaxagorou, E. Uthaug, H. Psaraftis, Optimizing shipping company operations using business process modelling. Maritime Policy & Management, 32(4), 403-420 (2005). doi: 10.1080/03088830500300636
- [3] A. Szymonik, Logistics and Supply Chain Management. Technical University of Lodz Press, 358p. (2012).
- [4] A. Anggorowati, "The evaluation of freight forwarding business existences," Proceedings of the Conference on Global Research on Sustainable Transport (GROST 2017), Advances in Engineering Research (2017). doi: 10.2991/grost-17.2018.62.
- [5] G. Syntia, J. Fahira, D. Himawan, and Y. Keke, The Implementation of Delivery Order Online as an Effort of Operational Efficiency. J. Phys.: Conf. Ser., vol. 1573, p. 012031(2020). doi: 10.1088/1742-6596/1573/1/012031.
- [6] N. Taheri, S. Jamali, M. Esmaeili, "Achieving Performability and Reliability of Data Storage in the Internet of Things", International Journal of Engineering and Manufacturing (IJEM), Vol.12, No.1, pp. 12-28, 2022. DOI: 10.5815/ijem.2022.01.02
- [7] A. Tzanetos, G. Dounias, "A Comprehensive Survey on the Applications of Swarm Intelligence and Bio-Inspired Evolutionary Strategies", in: G.A. Tsihrintzis, L.C. Jain (Eds.), Machine Learning Paradigms: Advances in Deep Learning-Based Technological Applications, Springer, Cham, 2020: pp. 337–378. DOI: 10.1007/978-3-030-49724-8_15.
- [8] N. Shakhovska, K. Shakhovska, S. Fedushko, Some Aspects of the Method for Tourist Route Creation. Advances in Artificial Systems for Medicine and Education II. Advances in Intelligent Systems and Computing, vol 902. Springer. pp 527–537 (2020). https://doi.org/10.1007/978-3-030-12082-5_48
- [9] C. Blum, R. Groß, "Swarm Intelligence in Optimization and Robotics", Springer Handbook of Computational Intelligence, Springer, Berlin, Heidelberg, 2015: pp. 1291–1309. https://doi.org/10.1007/978-3-662-43505-2_66.
- [10] A. Rahman, A. Nayem, S. Siddik, "Non-Functional Requirements Classification Using Machine Learning Algorithms", International Journal of Intelligent Systems and Applications (IJISA), Vol.15, No.3, pp.56-69, 2023. DOI:10.5815/ijisa.2023.03.05
- [11] J. N. Kok, Artificial Intelligence : Definition, Trends, Techniques, and Cases. Artificial Intelligence, Vol. 1, p. 401. ISBN: 978-1-84826-575-2
- [12] What is an API? (Application Programming Interface), MuleSoft. https://www.mulesoft.com/resources/api/what-is-an-api (accessed Nov. 03, 2023).
- [13] S. Sarker, "Freight Forwarding Industry: Function, Economic Role and Future Market in Bangladesh." Bangladesh Maritime Journal (BMJ), 1 (1), 93-98 (2017).
- [14] L. Rosenberg, G. Willcox, "Artificial Swarm Intelligence", in: Y. Bi, R. Bhatia, S. Kapoor (Eds.), Intelligent Systems and Applications, Springer International Publishing, Cham, 2020: pp. 1054–1070. https://doi.org/10.1007/978-3-030-29516-5_79.
- [15] V. Singh, & G. S. Dhillon, "A Effective Decision Making Approach "Human Swarming with Artificial Swarm Intelligence". International Journal of Advanced Research in Computer Science, Special Issue, Vol. 8 Issue 4, 427-430 (2017). DOI: 10.26483/ijarcs.v8i4.4157
- [16] M. Yousif, A. Salim, W. Jummar, "A Robotic Path Planning by Using Crow Swarm Optimization Algorithm", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.7, No.1, pp. 20-25, 2021. DOI: 10.5815/ijmsc.2021.01.03

- [17] K. W. Ross and D. H. K. Tsang, "The stochastic knapsack problem," IEEE Transactions on Communications, vol. 37, no. 7, pp. 740–747, Jul. 1989, doi: 10.1109/26.31166.
- [18] F. Hembecker, H. Lopes, W. Godoy, Particle Swarm Optimization for the Multidimensional Knapsack Problem. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds) Adaptive and Natural Computing Algorithms. Lecture Notes in Computer Science, vol 4431. Springer, 358–365 (2007). doi: 10.1007/978-3-540-71618-1_40
- [19] The Economic of Shipping Freight Markets." https://lawexplores.com/the-economic-ofshipping-freight-markets (accessed Nov. 05, 2023).
- [20] A. Okubo, "Dynamical aspects of animal grouping: Swarms, schools, flocks, and herds", Advances in Biophysics. 22 (1986) 1–94. DOI: 10.1016/0065-227X(86)90003-1.
- [21] A. Schumann, "From Swarm Simulations to Swarm Intelligence", EAI Endorsed Transactions on Serious Games. 3 (2016) 461–468.
- [22] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, 'A comprehensive survey: artificial bee colony (ABC) algorithm and applications", Artif Intell Rev. 42 (2014) 21–57. https://doi.org/10.1007/s10462-012-9328-0.
- [23] E. Ventocilla, "Swarm-based Area Exploration and Coverage based on Pheromones and Bird Flocks", 2013. https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-212190 (accessed November 17, 2023).
- [24] F. Pourpanah, R. Wang, C.P. Lim, X.-Z. Wang, D. Yazdani, "A review of artificial fish swarm algorithms: recent advances and applications", Artif Intell Rev. 56 (2023) 1867–1903. https://doi.org/10.1007/s10462-022-10214-4.
- [25] M.A. Bakar, "Understanding of Collective Decision-Making in Natural Swarms System, Applications and Challenges", ASEAN Journal of Science and Engineering. 1 (2021) 161– 170. https://doi.org/10.17509/ajse.v1i3.39637.
- [26] M.N.O. Sadiku, S.M. Musa, "A Primer on Multiple Intelligences', Springer International Publishing, Cham, 2021. https://doi.org/10.1007/978-3-030-77584-1.
- [27] X. Li, M. Clerc, "Swarm Intelligence", in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Springe, 2019: pp. 353–384. DOI: 10.1007/978-3-319-91086-4_11.
- [28] Y. Zhang, P. Agarwal, V. Bhatnagar, S. Balochian, J. Yan, "Swarm Intelligence and Its Applications", The Scientific World Journal. 2013 (2013). DOI: 10.1155/2013/528069.
- [29] Z. Abidin, M. Arshad, U. Ngah, "An introduction to swarming robotics: application development trends", Artif Intell Rev. 43 (2015) 501-514. DOI: 10.1007/s10462-013-9397-8.
- [30] M. Kwapisz, D. Gierowski, B. Szewczyk, Cargo Optimizer Service documentation. Seminar work, Poland (2019).
- [31] D. Rafi, K. Moses, K. Petersen, and M. Mäntylä, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey", in 7th International Workshop on Automating Software Test, (2012), 36–42, doi: 10.1109/IWAST.2012.6228988.
- [32] Molnár, E., Molnár, R., Kryvinska, N. et al. Web intelligence in practice. J Serv Sci Res 6, 149–172 (2014). https://doi.org/10.1007/s12927-014-0006-4
- [33] O. Shlyakhetko, Y. Syerov, M. Gregus, Model of Smart Locating Application for Small Businesses. Electronics, 12, 1993 (2023). https://doi.org/10.3390/electronics12091993
- [34] B. Danglot, "Automatic Unit Test Amplification For DevOps", Université de Lille (2019), p. 160.
- [35] R. Shashidhar, B. N. Arunakumari, A. Manjunath, M. Roopa, "Indian Sign Language Recognition Using 2-D Convolution Neural Network and Graphical User Interface", International Journal of Image, Graphics and Signal Processing, Vol.14, No.2, pp. 61-73, 2022. DOI: 10.5815/ijigsp.2022.02.06
- [36] F. Basanieri and A. Bertolino, "A Practical approach to UML-based derivation of integration tests," p. 10.
- [37] J. Ahamed, D. Nandi, "A Decision-Making Technique for Software Architecture Design", International Journal of Mathematical Sciences and Computing, Vol.9, No.4, pp. 44-49, 2023. DOI:10.5815/ijmsc.2023.04.05

- [38] D. Pashchenko, "Early Formalization of AI-tools Usage in Software Engineering in Europe: Study of 2023", International Journal of Information Technology and Computer Science, Vol.15, No.6, pp.29-36, 2023. DOI:10.5815/ijitcs.2023.06.03
- [39] E. Weyuker, F. Vokolos, "Experience with performance testing of software systems: issues, an approach, and case study," IEEE Transactions on Software Engineering, vol. 26, no. 12, 1147-1156 (2000). doi: 10.1109/32.888628.
- [40] W. Tsai, X. Bai, R. Paul, W. Shao, and V. Agarwal, "End-to-end integration testing design," 25th Annual International Computer Software and Applications Conference. COMPSAC 2001, USA, 166-171 (2001). doi: 10.1109/CMPSAC.2001.960613.
- [41] J. Anderson, J. McRee, R. Wilson, and T. E. Team, Effective UI: The Art of Building Great User Experience in Software. O'Reilly Media, Inc., 2010.
- [42] O. Taipale, J. Kasurinen, K. Karhu, and K. Smolander, "Trade-off between automated and manual software testing," Int J. Syst. Assur. Eng. Manag, vol. 2, no. 2, pp. 114-125, June 2011, doi: 10.1007/s13198-011-0065-6.
- [43] S. Sundari and S. Srinivasan, "M/M/C queueing model for waiting time of customers in bank sectors," International Journal of Mathematical Sciences & Applications, Vol. 1, No. 3 (2011).