An Artificial Immune System Approach for Solving Chance-Constrained Capacitated Logistic Distribution Center Problem

# Introduction

The efficient operation of logistic distribution centers is a critical factor in the performance of modern supply chains, particularly when managing limited resources such as vehicle fleets and storage capacities. The Capacitated Logistic Distribution Center (CLDC) problem seeks to optimize the allocation of these resources to minimize costs and maintain service levels. However, the uncertainty in demand, supply, and other operational factors introduces additional complexity. To address this, a chance-constrained framework is often applied, ensuring that certain constraints, such as demand fulfillment, are satisfied with a specified probability, thus adding a probabilistic dimension to the problem. In the CLDC problem, decisions must be made about how to allocate logistics resources (e.g., warehouses, vehicles) to meet customer demand, subject to capacity constraints. When chance constraints are introduced, accounting for uncertainty in demand and supply, the problem becomes even more challenging. The NP-Hardness of the problem arises from the need to balance multiple factors, such as minimizing transportation costs, selecting optimal distribution centers, and ensuring that capacity limits are not exceeded, all while managing the uncertainty of future demands.

Solving the CLDC problem involves selecting which distribution centers to activate and determining the optimal allocation of resources to minimize costs. However, the addition of capacity constraints and stochastic elements (through chance constraints) makes the problem significantly more complex. The number of potential combinations grows exponentially as the problem size increases, making exact optimization very complex for large instances. This combinatorial nature, along with the probabilistic constraints, places the chance-constrained logistic distribution center problem (CCLDC) in the class of NP-Hard problems. Due to the NP-Hard nature of this problem, traditional optimization methods, such as mixed-integer programming, are only practical for small instances. For larger, real-world problems, metaheuristic algorithms have proven effective. Therefore, the selected metaheuristic to tackle this problem in this research is an artificial immune system algorithm (AIS) that mimics the clonal selection of our immune system. This paper begins by mathematically formulating the problem and developing the steps of the proposed AIS. The algorithm is then tested on a set of optimization problems. Subsequently, it is adapted by hybridizing with a heuristic approach that utilizes a priority vector to generate a feasible solution. The benchmarks for CCLDC are not available, therefore we developed using python a set of 20 problems to implement our approach. The main objective of this method paper is to formulate the capacitated logistic distribution center problem under uncertainty using chance-constrained programming, and to develop a novel hybridized artificial immune system algorithm for solving it.

# Mathematical Model

The chance-constrained capacitated logistic distribution center problem shows another form of uncertainty of the problem found in (Ayid et al., 2024). This problem assumes that demands are stochastic, following a normal distribution with known expected values and standard deviations. Consequently, the demand constraints are reformulated as probabilistic constraints that must be satisfied with a certain chance probability. The mathematical model of the problem can be formulated as follows:

Notations:

|  |  |
| --- | --- |
|  | The set of plants |
|  | The set of logistic distribution centers |
|  | The transportation for transferring the units form logistic distribution center to plant |
|  | The service cost of the logistic distribution center |
|  | The capacity of logistic distribution center |
|  | The demand at plant |

The mathematical model of the problem can be formulated as follows:

|  |  |  |
| --- | --- | --- |
|  | | () |
|  | |  |
|  |  | () |
|  |  | () |
|  |  | () |

The objective function (1) seeks to minimize the total costs of transportation and services. The constraints (2) ensure that production from each plant meets specified requirements. The chance constraints (3) dictate that production must not exceed the stochastic demands with a probability of , where are random variables with known means and standard deviations. The chance constraints must be converted to their equivalent deterministic in order to solve the problem as follows:

Let be the cumulative distribution function (CDF) of the standard normal distribution. Then the calculation of the probability can be done as follows:

This can be true only if

Thus, the equivalent deterministic constraints can be reformulated as follows:

|  |  |
| --- | --- |
|  | () |

# The Heuristic Procedure

The heuristic procedure proposed for generating feasible solutions is primarily based on generating two priority vectors: one for plants and another for logistic distribution centers. The components of these vectors lie within the interval . These vectors are arranged in decreasing order, which alters the indices of the plants and logistic distribution centers. The new arrangement reflects the selection priority of each, ranked from highest to lowest. The heuristic procedure then populates an matrix, where represents the number of logistic distribution centers, and represents the number of plants. The plant and logistic distribution center with the highest priorities are selected first. If the demand of the plant exceeds the capacity of the logistic distribution center, the selected plant is removed from the plant priority vector, and vice versa. The heuristic continues to run until all plant demands are satisfied. Table 1 and 2 show an example of generating the priority vectors for the plants and logistic distribution centers.

Table Plant priority vector example

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Plant index | 1 | 2 | 3 | 4 | 5 | 6 |
| Priority Vector | 0.63 | 0.47 | 0.94 | 0.42 | 0.83 | 0.41 |
| Arranged indices with respect to priority | 3 | 5 | 1 | 2 | 4 | 6 |

Table 2 Logistic distribution center vector example

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Distribution Center Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Priority Vector | 0.99 | 0.04 | 0.13 | 0.63 | 0.02 | 0.11 | 0.87 | 0.35 | 0.97 | 0.5 |
| Arranged indices with respect to priority | 1 | 9 | 7 | 4 | 10 | 8 | 3 | 6 | 2 | 5 |

# The Proposed Artificial Immune System Algorithm

The human immune system activates in response to foreign bodies, or antigens. Initially, it relies on antigen-presenting cells to identify and confront these intruders (Dasgupta et al., 2011). If the immune system fails to recognize the antigens, they proceed to the lymph nodes, where white blood cells, specifically T-cells and B-cells, are generated. T-cells play a crucial role in clonal selection (Qi et al., 2014), a process that inspired the proposed artificial immune system algorithm, while B-cells are involved in a complementary process known as negative selection (Nemazee, 2000). Before adapting the proposed AIS to solve the chance-constrained logistic distribution center problem, a concise literature review is presented as follows. De Castro and Von Zuben (2002) proposed an algorithm called CLONALG, inspired by the immune system's ability to learn and adapt. This algorithm could be used for solving various problems, such as machine learning and optimization. It worked by selecting the best solutions and improving them over time, similar to how the immune system fights off infections. Zakaraia et al. (2023) proposed an artificial immune system algorithm to solve the stochastic multi-manned assembly line balancing problem. Their approach aligns with our work, as both utilize AIS for solving combinatorial optimization problems under uncertainty through chance-constrained programming. Liu et al. (2023) introduced a method called Human-in-the-Loop continual learning classification method using an artificial immune system. It allowed the system to adjust its parameters based on human feedback for misclassified data, making predictions more accurate without needing to retrain.

Ninphet et al. (2024) developed a method to predict estrus in dairy cows using artificial intelligence. They used convolutional neural networks and improved their accuracy by optimizing the parameters with the Artificial Immune System algorithm. Zhang et al. (2024) proposed the rank-based multimodal immune algorithm to improve immune algorithms for many-objective optimization problems. They introduced a rank-based clone selection method, a dynamic age-based elimination mechanism, and a multimodal mutation strategy to address cloning issues and enhance search performance in high-dimensional spaces. Molina et al. (2024) applied an artificial immune system to an electronic nose for chocolate bar classification, offering an alternative to traditional methods. The AIS adjusts its parameters dynamically with minimal tuning, and improved the accuracy in the single-label and multi-label scenarios. Ahmed et al. (2024) introduced a new method called X3PAIS, which combines two strategies to improve the Artificial Immune System algorithm. X3PAIS was tested in different areas, such as gear systems and wastewater treatment, showing its strength and flexibility.

The proposed algorithm in this paper mimics the clonal selection of T-cells, referred in the paper as antibodies, since they represent the priority vectors that represent the population. The affinity process of the clonal selection plays the role of measuring how does the paratope (receptor) of the T-cell match with the epitope of the foreign body. So, the affinity herein is the value of the objective function (1). As previously mentioned, using the priority vectors in the heuristic procedure leads to provide a solution based the priority values. Therefore, in the proposed AIS, modifying the components of these priority vectors leads to provide new solutions. Thus, each solution of the population is to be cloned for mutation to change its position in the solution space using equation (6).

|  |  |
| --- | --- |
|  | () |

Each antibody in the population has two priority vectors: one for the plants and another for the logistic distribution centers, as shown in equation (6) with the indices and . The number of components varies based on the number of plants and distribution centers. The learning rate is adjusted during the cloning process to generate new solutions, either near the current solution or close to the best solution found. The parameters of the algorithm are the number of iterations (), the number of cloned solutions (), population size (), and the learning rate (), which will be adjusted using linear decay function (7).

|  |  |
| --- | --- |
|  | () |

The proposed heuristic procedure in section 3 is used to find the associated solution and its evaluation for each antibody in the population. The proposed AIS algorithm for solving the problem can be summarized as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **AIS Algorithm for CCLDSP** | | | |
|  | Initiate algorithm parameters (, ,) | | |
|  | Generate initial population with antibodies, and evaluate them using the proposed heuristic | | |
|  | Set the best antibody in the population with evaluation | | |
|  | Set | | |
|  | While do: | | |
|  |  | Calculate | |
|  |  | For each antibody in population: | |
|  |  |  | Calculate using equation (7) |
|  |  |  | Clone antibody with antibodies and mutate the using equation (6) |
|  |  |  | Use the proposed heuristic to generate the evaluation of each antibody of the antibodies |
|  |  |  | If then and |
|  |  | Generate new population with antibodies | |
|  |  |  | |
|  | Return and | | |

The proposed algorithm is coded using python, and the source code is found in the GitHub repository <https://github.com/MZakaraia/AIS_Algorithm>.

# Experimental Design

The proposed AIS has three parameters that can be tuned to optimize its response and efficiency. These parameters are , , and . To find the selected parameter levels for the experimental design, a large range for each parameter is selected, and a Latin hypercube sampling (LHS) is generated with 30 samples. Each trail of LHS is replicated 10 times, and the mean and relative standard deviation (RSD) are calculated. Both the means and RSDs are normalized, and the proposed response is the average between the normalized value of means and RSDs. The selected high range for is from 10 to 50 iterations, is from 10 to 60 antibodies, and is 10 to 60 antibodies. According to these trails, the heatmap correlation matrix in Figure 1 shows weak positive correlation between parameters ensuring that the selected ranges are suitable for design of experiments. parameter shows high positive correlation with response indicating that it is the most influence parameter. This recommends focusing on covering low and high number of iterations in experimental design, while focusing on the low number of solutions on both population and cloned antibodies. The next step is to use Taguchi’s method to find the optimized parameter levels.



Figure Heatmap correlation matrix of the parameter levels and response

The selected parameter levels are as follows: , , and . A full factorial design for this setup would require trials for 30 replicates. To reduce the experimental effort, Taguchi’s method can be applied, requiring only 480 trials. This approach uses an orthogonal array to accommodate the total degrees of freedom, which equals 3 for each parameter and 1 for the overall mean. The experimental design is applied to a high-dimensional problem involving 15 distribution centers and 10 plants. This setup ensures that the algorithm's settings are adaptable to problems of any dimension. Figure 2 illustrates the main effects plot, revealing the optimized parameter levels: , , and .

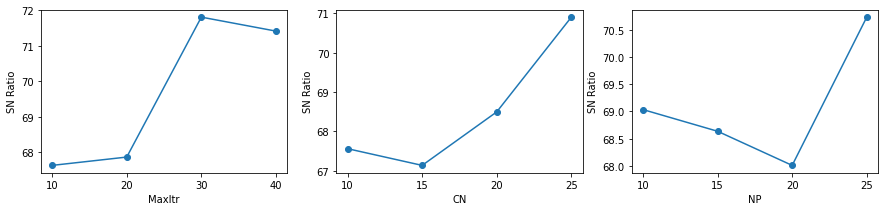


Figure Main effects plot

# Computational Results

The computational results for solving 12 generated problems are available in the GitHub repository: https://github.com/MZakaraia/AIS\_Algorithm. The problems vary in size: 5×5 (5 distribution centers and 5 plants), 5×10, 10×10, and 15×10. The chance probability used in the computational results 0.6, 0.8, and 0.9. The computational results show high accuracy in terms of robustness, since the relative standard deviation values are all approximately zeros.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Problem** |  | |  | |  | |
| **Mean** | **RSD** | **Mean** | **RSD** | **Mean** | **RSD** |
| 10X10\_01 | 125426.90 | 0.012 | 126489.10 | 0.016 | 127003.30 | 0.016 |
| 10X10\_02 | 119022.00 | 0.025 | 118996.90 | 0.020 | 119343.40 | 0.020 |
| 10X10\_03 | 134665.80 | 0.010 | 134873.60 | 0.008 | 135099.10 | 0.013 |
| 15X10\_01 | 61564.60 | 0.035 | 61059.40 | 0.040 | 60334.30 | 0.055 |
| 15X10\_02 | 59193.30 | 0.023 | 58690.70 | 0.051 | 59472.60 | 0.039 |
| 15X10\_03 | 68199.60 | 0.040 | 70541.60 | 0.035 | 67518.30 | 0.042 |
| 5X10\_01 | 111000.50 | 0.013 | 111206.70 | 0.015 | 111146.70 | 0.011 |
| 5X10\_02 | 117032.30 | 0.012 | 117072.60 | 0.012 | 117348.10 | 0.011 |
| 5X10\_03 | 126470.00 | 0.015 | 127066.90 | 0.013 | 127335.20 | 0.010 |
| 5X5\_01 | 17058.50 | 0.006 | 17070.00 | 0.007 | 17072.70 | 0.006 |
| 5X5\_02 | 18614.40 | 0.004 | 18599.60 | 0.004 | 18585.00 | 0.004 |
| 5X5\_03 | 16435.50 | 0.002 | 16441.10 | 0.002 | 16428.20 | 0.001 |