agriculture

MDPI

Article

Time–Cost–Quality Trade-Off in a Broiler Production Project Using Meta-Heuristic Algorithms: A Case Study

Erfan Khosravani Moghadam [1], Mohammad Sharifi [1,*], Shahin Rafiee [1] and Young Ki Chang [2,*]

[1] Department of Agricultural Machinery Engineering, Faculty of Agricultural Engineering & Technology, College of Agriculture & Natural Resources, University of Tehran, Karaj 31587-77871, Iran; khosravanierfan@ut.ac.ir (E.K.M.); shahinrafiee@ut.ac.ir (S.R.)

[2] Department of Engineering, Faculty of Agriculture, Dalhousie University, Truro, NS B2N 5E3, Canada

* Correspondence: M.sharifi@ut.ac.ir (M.S.); YoungChang@dal.ca (Y.K.C.); Tel.: +98-912-551-3254 (M.S.); +1-902-893-6715 (Y.K.C.)

Received: 18 November 2019; Accepted: 18 December 2019; Published: 20 December 2019

Abstract: The global production of broiler meat was forecasted to be 97.8 MT in 2019. The cost fluctuations create risks in production. In order to have an effective management system, process uncertainty must be taken into account. This approach considers the process as an interval with fuzzy numbers and, for managing the risks, it uses the variable α, a parameter determined by the manager in an interval between 0 and 1. Then two algorithms, namely the multi-objective imperialist competitive algorithm (MOICA) and multi-objective particle swarm optimization (MOPSO), were compared and applied. Since the process of production has many activities and each activity has possible options, the process does not have a unique solution. Therefore, the objective function and its assigned weights in terms of time, cost, and quality can be applied to select the best solution from those obtained. A vast amount of uncertainty can be observed, and effective management necessitates dealing with these uncertainty issues. The MOPSO algorithm showed better performance than the MOICA algorithm in this problem. Based on fuzzy logic and influenced by the uncertainty condition ($\alpha = 0$), time, cost, and quality in the MOPSO and the MOICA algorithms were 1793.8 h, \$260,571.7, and 46.66%, and 1792.5 h, \$260,585.7, and 51.19%, respectively.

Keywords: broiler; MOICA; MOPSO; fuzzy logic; optimization

1. Introduction

The poultry sector, the fastest expanding livestock sector, is the leader in global meat production growth in response to rising demand for this healthy, high protein, and low-fat type of meat [1]. Poultry is the second most consumed meat after pork in the world (14.99 and 16.02 kg/capita/year, respectively) and is expected to take over the leading position in the future. High costs of production, high food exchange coefficient in Iran in comparison with developed countries, high mortality of chickens during the growing period, and fluctuations in the price of the animals' feed make, as for other processes, an effective control and management system necessary. Furthermore, to accelerate a process, some of the process activities can be completed faster than normal by either spending less time on them or more money to cover the expenses of additional resources allotted to them. Thus, accelerating a process increases its cost.

Generally, several methods of time and cost optimization are categorized as heuristic, mathematical, and meta-heuristic. Added to dimensions and complexity matters, the possibility of solving common optimum or fast calculating methods in a suitable time has been reduced, and as a result, receiving the

whole set of optimum solutions in such a situation is very difficult. The researchers' attention were attracted to the successful expansion of meta-heuristic optimum algorithms in solving the optimum single-goal issues.

The Pareto front is an arc (for two dimension) or surface (for more than two dimension) that contains solutions representing all optimal trade-off possibilities of the objectives. There is no feasible solution in the exploration space that improves each of objectives without degrading at least one of the others at the same time according to any solutions on the Pareto front. Hence, finding the Pareto front of these non-dominated solutions is the goal of any multi-objective algorithm. Evolutionary search techniques have been applied by multi-objective evolutionary algorithms (MOEAs) to face finding the Pareto front, and evolutionary algorithms (EAs) could explore various parts of the Pareto front simultaneously. The EAs are appropriate in multi-objective problems (MOPs) if many dimensions are usually utilized. Genetic algorithms (GAs) are some of the important EAs, which have been applied in many MOEAs containing a multi-objective genetic algorithm (MOGA) [2,3], non-dominated sorting genetic algorithm (NSGA) [4], fast elitist non-dominated sorting genetic algorithm (NSGA-II) [5–7], etc. Numerical results of different studies show that NSGA-II is better than other GA based MOEAs [6]. On the other hand, owing to the promising results presented by particle swarm optimization (PSO) in single objective optimization problems [8], other studies have shown that using swarm intelligence improves MOEAs [9–11]. Utilizing PSO and fast non-dominated sorting, multi-objective particle swarm optimization (MOPSO) was proposed by Parsopoulos and Vrahatis (2002) [12] and Coello and Lechuga (2002) [13]. It has been used for multi-objective problems and has shown its efficiency in many studies [14–17]. Additionally, the imperialist competition algorithm (ICA) is one of the new EAs that was introduced by Atashpaz-Gargari and Lucas (2007) [18]. This algorithm has been successfully used for many single-objective problems such as counting job scheduling [19], design of thermal systems [20], design of linear induction motors [21], design of skeletal structures [22], solving global optimization [23], etc. Since ICA has shown good result in single-objective problems, researchers have improved the ICA for multi-objective problems. Compared to PSO, ICA has shown its great capability in finding a near-optimum solution and escaping the local optima. The multi-objective imperialist competitive algorithm (MOICA) is a novel multi-objective evolutionary algorithm that was proposed based on the ICA [24].

In the above-mentioned research, the results were computed considering the condition of complete certainty; however, this is not the case in the real world. The present study includes uncertainty using the fuzzy logic theory during the problem-solving process. The theory of fuzzy sets was introduced in 1965 by Lotfi Zadeh [25]. The application of the fuzzy sets theory is its capability to present ambiguous issues so that the manager can take advantage of the definition of uncertain and inaccurate data and thus reduce risks in using fuzzy data [26].

According to the previous studies for finding a near optimal solution, MOPSO and MOICA have shown good performance. Therefore, MOPSO and MOICA have been designed to solve problems with continuous range for comparing the result in this study. Additionally, it should be noted that the problems presented in this research have integer number ranges, so with changes in this algorithm, they can be applied for integer numbers that are explained in the following part.

2. Materials and Methods

2.1. Problem Definition

The experiments were carried out in Borujerd County, which is situated 100 km northeast of Khorramabad, the capital of Lorestan province, in the west of the Islamic Republic of Iran. The county is located between 33°36′ N 48°27′ E to 34°6′ N 49°27′ E, with an area of 1606 km^2 (Figure 1). Burojerd City is located approximately 1670 m above sea level and has a moderate climate with cold winters. The maximum and minimum temperatures were 38 °C and −18 °C in July and January, respectively,

with the mean value annual temperature of about 14.6 °C. Average annual precipitation is about 480 mm, mostly decreasing in the spring and the winter [27].

Figure 1. Map of the Borujerd County in the Iran.

In this study, the life span of thirty thousand chickens from eggs of broiler breeder hens to the slaughterhouse are summarized in fourteen activities. Each activity has some options for resource utilization and the goal is finding the optimal/near optimal ways of project completion in the search space of whole possible combinations of these resource utilization options to activities. Considering the situation of the implementation of activities, on average, each activity has 4 different executive approaches. Each approach itself contains time, cost, and quality in the shape of the individual triangular fuzzy number (Figure 2).

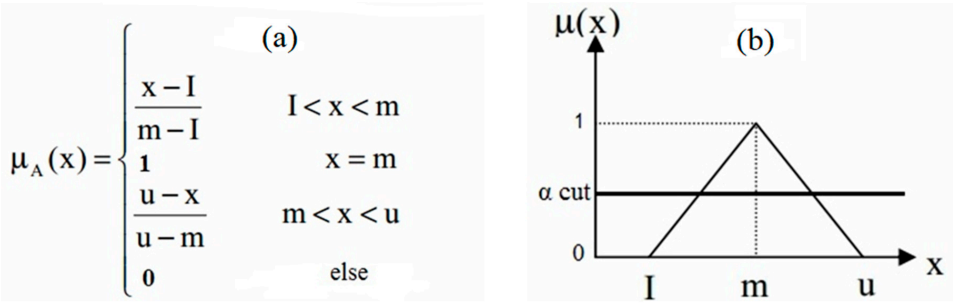

Figure 2. Explanation of the triangular fuzzy numbers. (**a**) The equation of triangular fuzzy numbers; (**b**) Triangular fuzzy numbers on the graph and how it is cut by α fuzzy cut.

2.2. *Fuzzy Theory*

In this section, some basic definitions of fuzzy theory that was defined by Kaufmann, Gupta and Qi, and Passino and Yurkovich [28–30], are presented. The characteristic function γA of a crisp set $A \subseteq X$ assigns a value of either 0 or 1 to each member in X. A fuzzy function can be generalized to a function μA that is called the membership function, and the set $A = \{(A, \mu A(x): x \in X\}$ defined by $\mu A(x)$ for each $x \in X$ is called a fuzzy set. A fuzzy set A defined on the set of real numbers R is said to be a fuzzy number. A fuzzy number is a generalization of a regular, real number. It refers to a connected set of possible values, where each possible value has its own weight between 0 and 1. A fuzzy number is thus a special case of a convex, normalized fuzzy set of the real line.

A triangular fuzzy number is one of the fuzzy numbers that is represented with three points as follows:

A = (I, M, U): this representation is interpreted as a membership function. These parameters represent the smallest possible value (I), the most promising value (M), and the largest possible value (U).

Different α cuts are defined by the manager between 0 and 1, where when α cut decreases to zero, uncertainty and risk increase, whereas when it increases to one, uncertainty and risk decrease. When defining α cut, it is crucial to consider also the fluctuations and the risk prediction for the production cost. For each different α, different solutions for cost, time, and quality are produced and, consequently, the objective function is calculated.

In this study, the number of completion conditions of the broiler production process was the act of rearranging the members of a set into a sequence or order; it was represented by m^n, where 'm' represents the number of different approaches for each activity and 'n' represents the number of activities in the production process (Table 1), and each activity could be done by four different approaches, as explained in Section 2.1. It was therefore the aim to find the best condition (smallest amount of cost and time and highest quality) among 4^{14} possibilities. Table 2 illustrates the sample of approaches for activity 7 that explains that activity 7 has four approaches. As mentioned, each approach has three fuzzy numbers for each of the total time, total cost, and total quality.

Table 1. Activities.

Number of Activity	Name of Activity
1	Buy and transport eggs from broiler breeders to incubation house and put them into the trays and fumigate them
2	Put trays into the setter and hatcher for hatching
3	Count broiler chicks and put them into the chick boxes
4	Wash and fumigate tray setter and hatcher
5	Wash and fumigate poultry house
6	Buy and transport broiler chicks from incubation house to poultry house
7	Grow up broiler chicks in the poultry house
8	When broilers reach market weight, they are caught and put into transport crates
9	Buy and transport poultry from poultry house to slaughterhouses
10	Slaughter, pick feathers, and eviscerate
11	Wash poultry and put them into spin chiller
12	Hang out poultry and transport them to a cold room
13	Pack and weigh
14	Buy poultry and transport them to the emporium

Table 2. Sample of approaches for "Grow up broiler chicks in the poultry house (activity 7)".

	Fuzzy Number	Total Cost of Feed for Broiler ($)	Total Cost of Electricity ($)	Total Cost of Vaccination ($)	Total Cost of Antibiotics ($)	Total Number of Labor	Total Cost of Food for Labor ($)	Total Cost of Labor ($)	Total Cost of Fuel ($)	Total Time (day)	Total Cost ($)	Total Quality
Approach 1	1	68,125	109.375	2812.5	3750	6	843.75	2812.5	1898.438	1080	80,351.56	0.7
	2	68,750	125	3750	4687.5	6	1125	2812.5	2531.25	1200	83,781.25	0.4
	3	69,375	140.625	4687.5	5625	6	1546.875	2812.5	3403.125	1320	87,590.63	0.2
Approach 2	1	68,125	109.375	2812.5	3750	9	1265.625	4218.75	1898.438	1080	82,179.69	0.7
	2	68,750	125	3750	4687.5	9	1687.5	4218.75	2531.25	1200	85,750	0.45
	3	69,375	140.625	4687.5	5625	9	2320.313	4218.75	3403.125	1320	89,770.31	0.3
Approach 3	1	68,125	109.375	2812.5	3750	12	1687.5	5625	1898.438	1080	84,007.81	0.8
	2	68,750	125	3750	4687.5	12	2250	5625	2531.25	1200	87,718.75	0.5
	3	69,375	140.625	4687.5	5625	12	3093.75	5625	3403.125	1320	91,950	0.5
Approach 4	1	68,125	109.375	2812.5	3750	15	2109.375	7031.25	1898.438	1080	85,835.94	0.9
	2	68,750	125	3750	4687.5	15	2812.5	7031.25	2531.25	1200	89,687.5	0.6
	3	69,375	140.625	4687.5	5625	15	3867.188	7031.25	3403.125	1320	94,129.69	0.7

2.3. Depicting Activities Network Diagram

After defining the activities of the production process, their interrelationships were identified to construct a network diagram (Figure 3). A network diagram is a chart in which the activities appear ordered and sequenced on a chronological basis, according to their interconnections. To design the diagram, the "activity on node (AON)" [31] method was used, as shown in Figure 3. According to this method, activities are shown on nodes and vectors indicate their chronological succession. It is obvious that "wash and fumigate poultry house (activity 5)" is independent of incubation house (activities 1–3); therefore, activity 5 is connected from the start to activity 6. Furthermore "Wash and fumigate tray setter and hatcher (activity 4)" must be done after activity 3 and it is independent of activities 6–13.

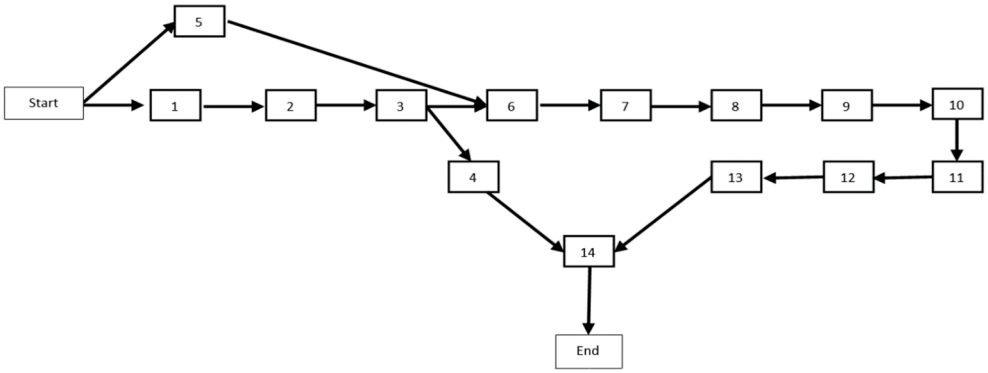

Figure 3. Activity-on-node network of broiler production process.

2.4. MOPSO Algorithm: A Proposal to Problem Solving

The algorithms used in this study were performed by MATLAB 2014a software (MathWorks, Natick, MA, USA). In the MOPSO algorithm, each solution for the problem identified is equal to one particle. The proposed algorithm was as follows: for each particle, considering the number of activities, 14 numbers were defined randomly between 0 and 1 (Figure 4). As the MOPSO works by a continuous range of variables, the solutions must be made randomly in a continuous range, and then they must be changed to an integer range for calculation of the objective function using Equation (1). Therefore, it is important to consider the existing constraints of the defined number of approaches for each activity. In Equation (1), the selected number between 0 and 1 was converted into an integer number in order that a random method could be selected for each activity.

$$\text{int}_i = \min[(1 + R_i \times M_i), M_i] \tag{1}$$

where i is activity number; int is integer number for each activity; R is randomized number between (0,1); and M is number of methods for each activity. After determining the approach of performing each activity, time, cost, and quality for each approach of performing activities were gathered in the fuzzy form. Then each particle has a fuzzy time, fuzzy cost, and fuzzy quality. The defuzzification (non-fuzzy) is made by using the gravity center method for each particle's time, cost, and quality [32]. Defuzzification is the process of obtaining a single number from the output of the aggregated fuzzy set. It is used to transfer fuzzy inference and results in a crisp output. In other words, defuzzification is realized by a decision-making algorithm that selects the best crisp value based on a fuzzy set. There are several forms of defuzzification including center of gravity (COG), mean of maximum (MOM), and center average methods, the most popular of which is the (COG) method. The COG method returns the value of the center of the area under the curve (Equation (2)).

$$u_{COG} = \frac{\sum_i \mu_c(x_i) x_i}{\sum_i \mu_c(x_i)} \tag{2}$$

where u_{COG} is crisp value, x_i is a point in the horizontal axis (i = 1, 2, 3, …), and $\mu_c(x_i)$ is the membership value of the resulting conclusion set according to (Figure 2).

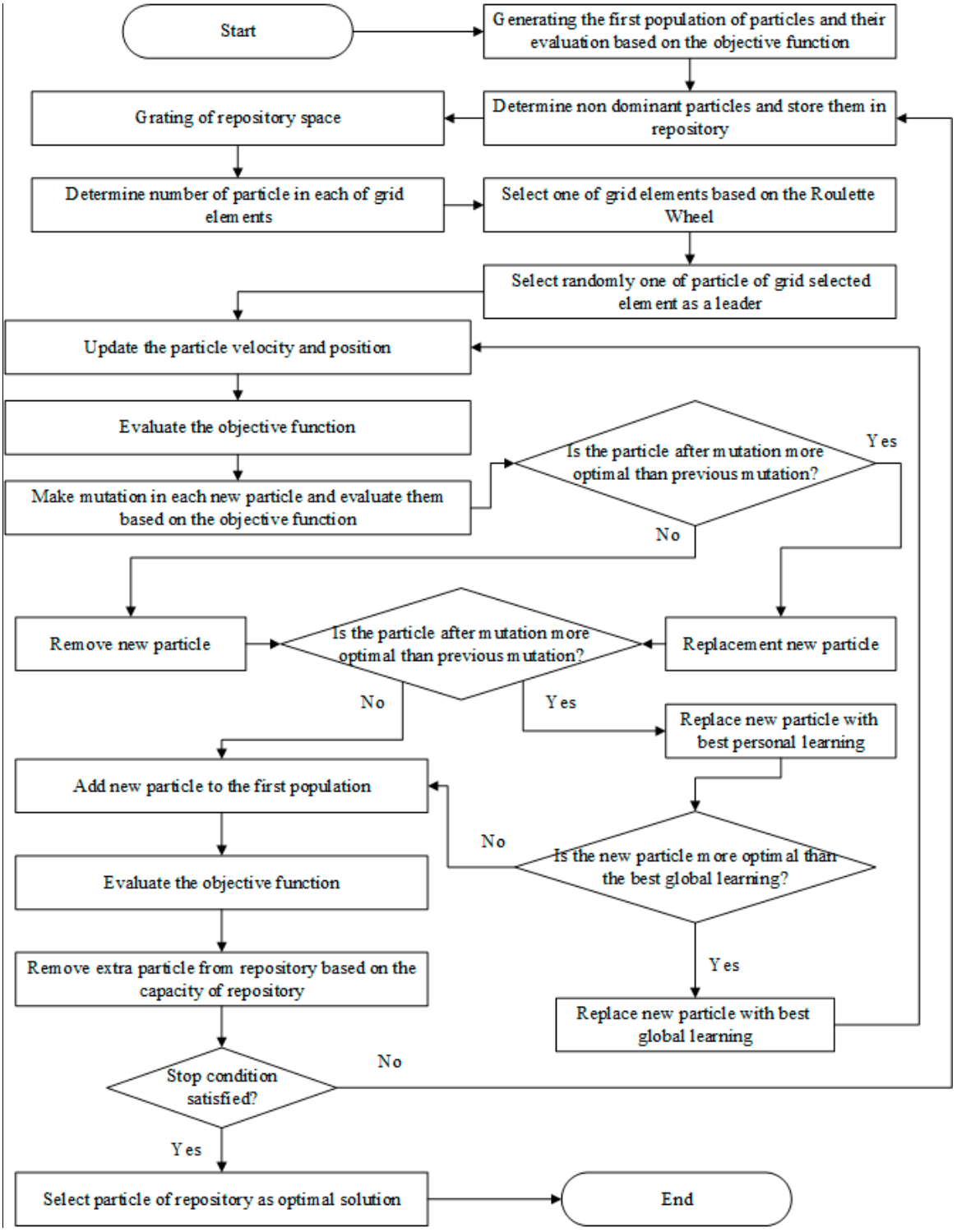

Figure 4. Flowchart of the proposed multi-objective particle swarm optimization.

In the next stage, considering the 3-dimensionality of objective function space, the solutions must be determined that in all three dimensions are not overcome against the other solutions in Equation (3).

$$\text{Dominated} = all([x_1, x_2, x_3] <= [y_1, y_2, y_3])\&\&any([x_1, x_2, x_3] < [y_1, y_2, y_3]) \quad (3)$$

where x_1, x_2, and x_3 are dimensions of time, cost, and quality for x activity, respectively, and y_1, y_2, and y_3 are dimensions of time, cost, and quality for y activity, respectively. So far, the objective function was defined, and it was mentioned how to calculate the objective function for each solution (particle) and compare them with each other to determine non-dominated solutions (particles). The non-dominated particles are reserved in a repository. In this paper, velocity update for a particular particle is done by selecting one of the non-dominated particles as a leader. Thereby, an initially grid must be performed because particles have no preference for each other to being selected as the leader. In this way, every dimension of the problem, the space between the biggest and smallest member of the repository, is determined and inflated in a certain extent by an inflation rate (in order that the biggest and smallest member of the repository enter into the grid as well). The determined grid number in the algorithm parameters was considered according to the following: this inflated space was divided equally, and the high and low limit for each part of the grid was determined, as seen in Figure 5.

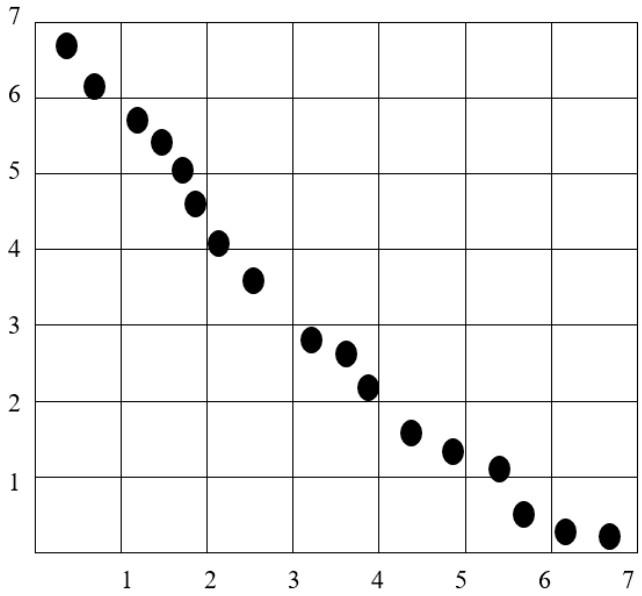

Figure 5. Graphical representation of grid.

In the next stage, it was determined which member of the repository belongs to which part of the grid and then each part's particle number was determined. More particles in one part's grid means a higher density of particles, and its selection probability is low when they are selected by roulette wheel (known as fitness proportionate selection) to associate a probability of selection.

This probability is defined by Equation (4); then one part of the grid with a roulette wheel, randomly one particle of that part, is selected as a leader and then for each particle, a leader is selected by the previous method. In the following, a velocity is defined for each particle. Velocity is zero at the beginning of the algorithm and updated in every iteration, and the particle is placed in a new position (Equation (5)).

$$P = EXP(-nG \times \beta) \quad (4)$$

where P is selection probability; nG is number of particles in each grid part; and β is selection pressure.

$$Vel(i) = W \times Vel(i) + C_1 \times (BestP(i) - Pop(i)) + C_2 \times (LeaderP - Pop(i)) \quad (5)$$

where $Vel(i)$ is velocity of particle(i); W is inertia weight; C_1 is personal learning coefficient; $BestP(i)$ is the best particle of each iteration; $Pop(i)$ is every particle of the population; $LeaderP$ is the best particle until the current iteration; and C_2 is the global learning coefficient. To increase exploration in the algorithm, a mutation was created in new position of each particle; considering the mutation rate, some activities (variables) were selected randomly in each particle and the approach's number of those activities was changed randomly with respect to its constraint. In the case that the mutation created the more optimal particle, it was accepted. Otherwise, the algorithm was continued by the non-mutated particle. It is worth mentioning that the mutation rate is reduced by an increase of the algorithm iteration according to Equation (6).

$$PM = \frac{1-(i-1)}{(i_{MAX}-1)^{\frac{1}{Mu}}} \tag{6}$$

where PM is percentage of mutation; i is number of iterations; and Mu is the mutation rate. The new position of each particle was evaluated according to the objective function. If the objective function's value of the new position of the particle is better than the best particle (global memory) of all particles, it is substituted for the best particle (global memory) of all particles. Then, the non-dominated particles are identified and added to existing particles in the repository. The non-dominated particles in the repository are separated and the grid stages are performed again.

If the number of particles within the repository is more than the repository capacity, the additional particles must be removed from the repository. Obviously, particles with the least preference for becoming a leader must be removed from the repository. To do this, initially using Equation (7), for each part of grid, a probability is considered such that every part of the grid that has more particles, using a roulette wheel, has a higher selection chance and then one of the particles of that part of grid is eliminated at random. On the contrary, when every part of the grid has fewer particles, it has a lower selection chance by the roulette wheel because the particles should support all the searching space and the chance of selecting particles for removal should be increased in the crowded part of grid. This stage was repeated until the number of existing particles in the repository was equal to the repository capacity.

$$P = EXP(nG \times l) \tag{7}$$

where P is probability of selection in each grid; nG is the number of particles in each grid part; and l is selection pressure. Each member of the repository (best particles) was a possible position in the implementation of the production process of a broiler. In this case, the best particle was selected from existing particles in the repository based on our considered coefficients of time, cost, and quality (regarding the importance of factors like time, cost, and quality for each manager, the coefficients can be different).

In this study, the relative weight of these criteria such as time, cost, and quality were considered as 0.34, 0.34, and 0.3, respectively (Equation (8)). The objective function coefficients of Equation (8) are determined according to the manager's opinion and considering the existing work conditions. These coefficients determine which factors (cost, time, and quality), taking work conditions into account, must have higher importance.

$$OF = (WT \times T) + (WC \times C) + (WQ/Q) \tag{8}$$

where OF is objective function; WT is weight of time; T is total time of process; WC is weight of cost; C is total cost of process; WQ is weight of quality; and Q is total quality of process. This procedure was continued until there was no considerable changes in the function utility, according to the factor coefficients. Other parameters were needed for this algorithm, which are shown in Table 3. Parameters were obtained through trial and error for obtaining the best solution along with logical running time.

Table 3. Multi-objective particle swarm optimization parameters.

Parameters	Values
Population Size	50
Repository Size	50
Maximum Total Number of Iterations	40
Mutation Percentage	40
Inertia Weight	0.5
Personal Learning Coefficient	1
Global Learning Coefficient	1.5
Number of Grids per Dimension	5
Inflation Rate	0.2
Mutation Rate	0.3

2.5. MOICA Algorithm: A Proposal for Problem Solving

This algorithm was tested using MATLAB 2014a software (MathWorks, Natick, MA, USA). As Figure 6 shows, MOICA begins with creating a country. Each country has an initial population of possible solutions for this problem and contains 14 variables, where each variable has different approaches when considering constraints. Every country is either an imperialist or a colony. Imperialists are assigned to the most powerful countries. Every imperialist makes empires by taking control of colonies. In the following, the imperialists attract all their colonies and create revolution in the imperialist and colonies (like the mutation in the genetic algorithm).

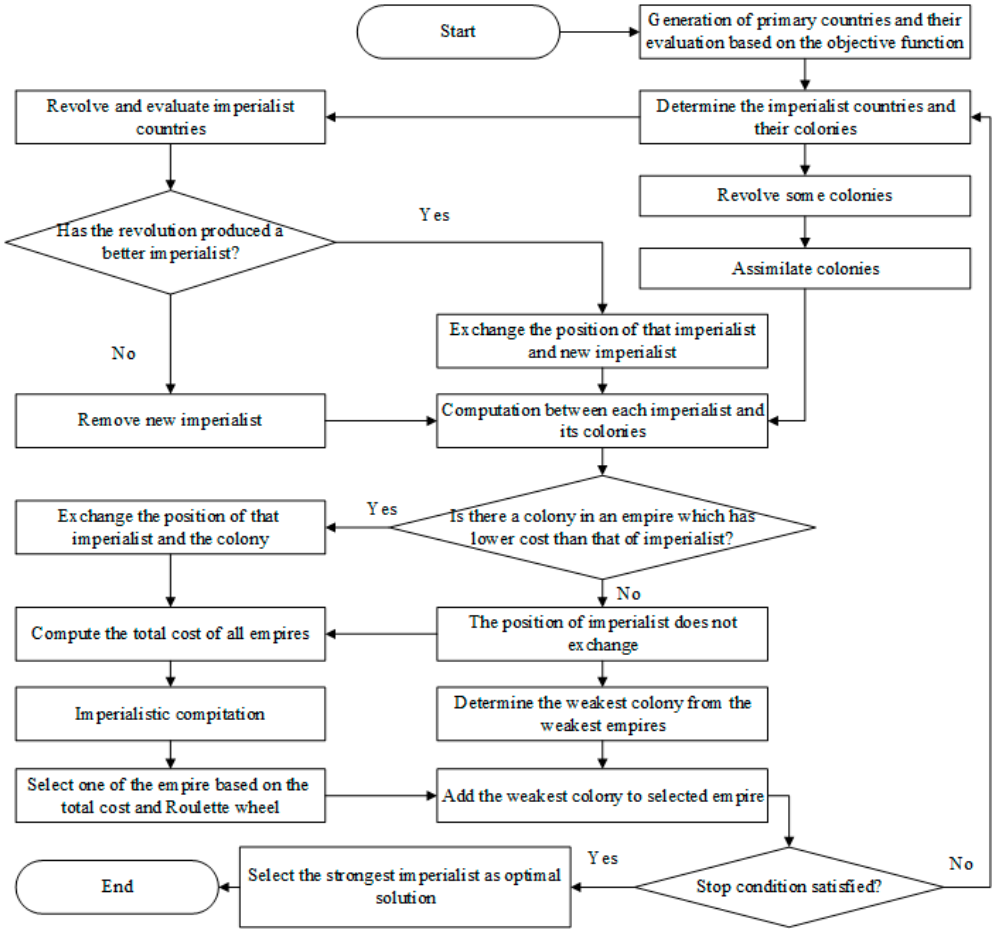

Figure 6. Flowchart of the purposed multi-objective imperialist competitive algorithm.

The basis of this algorithm is the competition among empires. In this process, all colonies move from weak empires to the more powerful ones. At the end, MOICA converges to one empire that

remains. In this state, the best solution of the optimization problem is calculated when the objective function's value is the same for all colonies and the corresponding imperialists. Like MOPSO, at first the MOICA algorithm selected 14 numbers randomly between 0 and 1 for each country (Equation (1)). The numbers were converted into integers in order that a random approach's number was selected for each activity. After determining the approaches of performing for each activity, time, cost, and quality for each approach of the production process activities were gathered in the fuzzy form. Then each country had a fuzzy time, fuzzy cost, and fuzzy quality. In this stage defuzzification (non-fuzzy), was made by using the gravity center method for each country's time, cost, and quality.

In the next stage, to sort countries, considering the 3-dimensionality of the objective function space, the solutions must be ranked, and the crowd distance of each country must be computed. The country set that is in the same rank constitutes the Pareto front surface (used in the multi-objective genetic algorithm). It is worthwhile mentioning that the more the crowding distance of a country is, the emptier the area of the objective function shows, as determined using Equation (9).

$$d_i = \sum_{j=1}^{D} \frac{f_{i-1}^j + f_{i+1}^j}{f_{\max}^j - f_{\min}^j} \tag{9}$$

where d is crowding distance; i is member of population; D is number of objectives; and f is objective function value. In that case, the countries were sorted based on the ranking (in ascending order) and then they were sorted based on the crowding distance (in descending order).

In the next step, the best countries were separated and selected as imperialists, and the remaining countries were assigned to imperialists by roulette wheel. Each imperialist that had the fewest components in sorting had an increased chance for selection by roulette wheel and the taking of more countries (Equation (4)). After determining and assigning countries to each empire, the countries in each empire moved towards an imperial country (belongs to each empire) with the assimilation coefficient determined using Equation (10) (Figure 7).

$$A(i) = Col(i) + C \times R \times (Imp(i) - Col(i)) \tag{10}$$

where *Col* is Colony(i) (country(i)); C is assimilation coefficient; R is random number; and *Imp* is Imperialist (i). In the next stage, considering the revolution percent, in the countries of each empire a revolution was made by random and in view of the revolution rate, some of the variables were selected randomly and the approach of those variables were changed. Like country as colony, the revolution could also be created in the imperialist countries; if the revolution is not destructive, a more optimal country is produced than the previous imperial country, and the new country is substituted for the previous imperialist country. In the following, new countries were formed by assimilation and revolution, then they competed with their imperialist country and if the imperialist countries were defeated, they were replaced. Then empires had their turned to compete. For competition between empires, Equation (11) must be used for an entire empire, and a value is defined to empires that compare them with each other.

$$Val(j) = Imp(j) + Z \times \frac{\sum_{i=1}^{nCol(j)} Imp(j)Col(i)}{nCol(j)} \tag{11}$$

where *Val* is value of the jth empire; *Imp* is the imperialist of the jth empire; Z is the colony's (country's) mean objective function's value coefficient; *Col* is ith colony of the jth empire; and *nCol* is the number of colonies for the jth empire. Like MOPSO, to sort empires, crowd distance was used. After sorting empires, the first and last empire were identified as the most powerful and weakest empire, respectively. In the following, weakest countries were sorted according to the previous method and the weakest country of the weakest empire was selected, as seen in Figure 8.

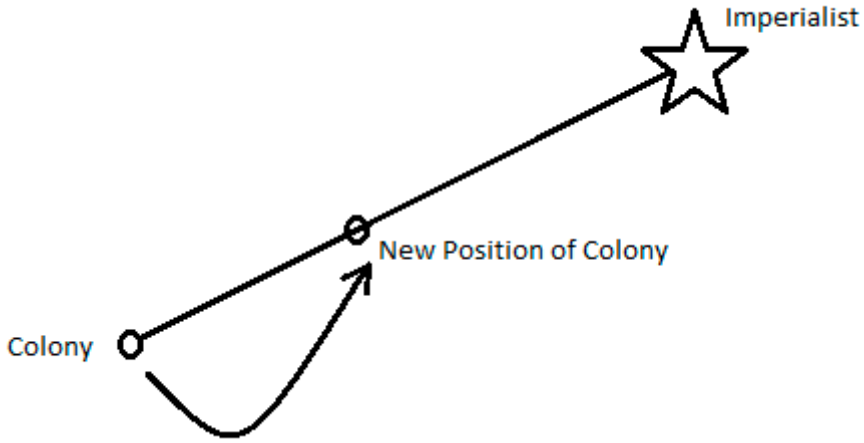

Figure 7. Assimilation policy: all colonies move toward their corresponding imperialists.

Figure 8. Imperialistic competition.

In the next stage, the weakest country was eliminated from the weakest empire and added to one of the remaining empires selected by the roulette wheel (every empire that has the fewest components in sorting has an increased chance for selection by roulette wheel and taking the weakest country, Equation (4)).

This trend caused some of the empires to be eliminated while performing the algorithm and finally, the imperial countries remained in the empires, which showed us the problem solutions. Parameters used in this algorithm are presented in Table 4. Parameters were obtained through trial and error for obtaining the best solution along with logical running time.

Table 4. Multi-objective imperialist competitive algorithm parameters.

Parameters	Values
Population Size	50
Number of Empires/Imperialists	15
Maximum Total Number of Iterations	40
Revolution Percentage	80
Assimilation Coefficient (C)	1.5
Revolution Rate	0.02
Colonies Mean Cost Coefficient (Z)	0.5

A counter in the objective function was applied to measure the reference numbers to objective function in both algorithms in Table 5.

Table 5. Number of functional evaluation (NFE) values for each algorithm.

Algorithm	Number of Functional Evaluation
Multi-objective particle swarm optimization	4050
Multi-objective imperialist competitive algorithm	3450

3. Results and Discussion

For reaching absolute solution in this study, around 270 million possible combinations that contained 14 activities with an average of four subcontracting alternatives have been checked. Project performance such as project time, cost, and quality changed with each possible combination. Searching in the large solution space of a possible solution was improved by the present multi-objective optimization model. The proposed methods can reduce the large solution space by finding a non-dominated solution over successive generations.

In this research, the two algorithms MOPSO and MIOCA were used to trade off between cost, time, and quality of the broiler production process. To compare the two algorithms in solving the problem, the initial population and iteration number in both algorithms were assumed equal; these two algorithms were used in this study by using Equation (1).

Tables 6 and 7 show the best solutions for each amount of fuzzy cut based on the lowest objective function. As the tables show, the more (α) increased, the lower the objective function amount was because the uncertainty and risk decreased. The best solutions are given in Tables 6 and 7, which show subcontracting plans and project objectives for each of the solutions. Furthermore, these tables show notable solutions with the optimum values for the three objectives in the Pareto-optimal front and the solution of the best compromise. The results in Tables 6 and 7 indicate that the ($\alpha = 1$) exhibits the minimum objective function, and the ($\alpha = 0$) exhibits the maximum objective function. These results make it clear that when (α) is one, the time and cost are the lowest amount and quality is the highest amount due to the fact that time and cost are increased, and quality is decreased by decreasing the (α). This assertion is concluded by center of gravity defuzzification. When fuzzy cut is moved to the base of the fuzzy triangle number ($\alpha \rightarrow 0$) by the manager, the output number of defuzzification is increased. The objective function is increased by increasing the time and cost and decreasing the quality based on Equation (8). According to the results, the amount of objective function in MOPSO is lower than that in MOICA for all fuzzy cut (α); therefore, MOPSO had better performance than MOICA for this problem.

If the contractor is allowed to increase cost by maximizing the project quality, then he/she should select the solution that has maximum quality value and that minimizes time, but that does not globally minimize the cost. In this case, he/she should give a higher weight to quality and a lower weight to time and cost in the objective function. If a contractor would like to minimize the project cost, then he/she needs to select a solution to achieve the minimum cost value by incurring additional duration with low project quality. In this case, he/she should give a higher weight to cost and a lower weight to time and quality in the objective function. Furthermore, if a contractor has a priority for minimizing project time, he/she should give a higher weight to time and a lower weight to cost and quality in the objective function.

According to Figures 9 and 10, in algorithm MOPSO, lesser amounts were obtained in equal iterations for the objective function, and the algorithm MOPSO had a better search ability to find the optimal solution for the concerned problem than the algorithm MOICA. According to Table 5, the number of functional evaluation (NFE) in MOPSO was lower than that in MOICA, although the numbers of the first population and its iteration in both algorithms were equal. Additionally, considering the convergence of algorithms and the achievement of a constant value, MOPSO showed better performance in contrast to MOICA. Figures 9 and 10 show that under different (α), MOPSO achieved a constant value in fewer iteration numbers. This showed higher efficiency of the MOPSO than MOICA.

Table 6. Results acquired in multi-objective imperialist competitive algorithm.

α [a]	Time (Hour)	Cost ($)	Quality	Objective Function [b]	Optimal Subcontracting Alternative Plan [sp]													
0	1792.531	260,585.7	0.511905	89,208.76	5	1	3	4	1	3	1	1	1	1	1	2	5	3
0.3	1786.534	260,613.1	0.526101	89,216.02	3	1	4	3	4	2	1	3	1	3	1	3	3	3
0.5	1789.515	260,496.1	0.55	89,177.28	5	1	5	4	1	5	1	1	1	1	1	4	4	3
0.8	1786.47	260,479.7	0.542857	89,170.67	4	2	5	4	4	2	1	2	1	1	1	2	4	3
1	1793.611	260,421.5	0.478854	89,153.27	5	1	4	4	1	1	1	1	1	2	1	1	2	3

[a] amount of fuzzy cut. [b] minimization functions that determinate the best solution from all of non-dominated solution. [sp] each solution indicates an identified subcontracting plan for each of the 14 activities.

Table 7. Results acquired in multi-objective particle swarm optimization.

α [a]	Time (Hour)	Cost ($)	Quality	Objective Function [b]	Optimal Subcontracting Alternative Plan [sp]													
0	1793.836	260,571.7	0.466667	89,204.42	5	1	5	1	1	1	1	1	1	1	1	2	1	3
0.3	1782.457	260,553.3	0.502283	89,194.31	5	1	2	2	1	4	1	2	1	4	1	1	2	3
0.5	1790.515	260,485.1	0.494048	89,173.86	5	1	5	1	1	4	1	1	1	1	1	1	1	3
0.8	1792.315	260,438.6	0.521705	89,158.68	5	1	5	4	1	2	1	1	1	1	1	4	5	3
1	1793.889	260,392.5	0.485714	89,143.5	5	1	3	4	2	1	1	1	1	2	1	4	2	3

[a] amount of fuzzy cut. [b] minimization functions that determinate the best solution from all of non-dominated solution. [sp] each solution indicates an identified subcontracting plan for each of the 14 activities.

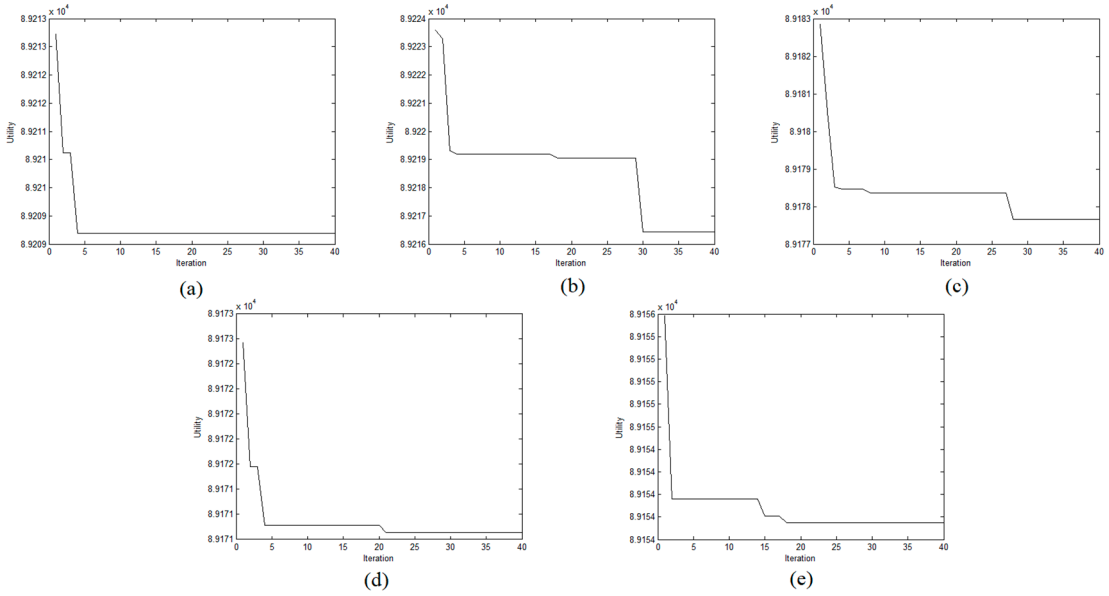

Figure 9. The best answer for objective function during the various iterations in multi-objective imperialist competitive algorithm. (**a**) Result of fuzzy cut (α = 0); (**b**) Result of fuzzy cut (α = 0.3); (**c**) Result of fuzzy cut (α = 0.5); (**d**) Result of fuzzy cut (α = 0.8); (**e**) Result of fuzzy cut (α = 1).

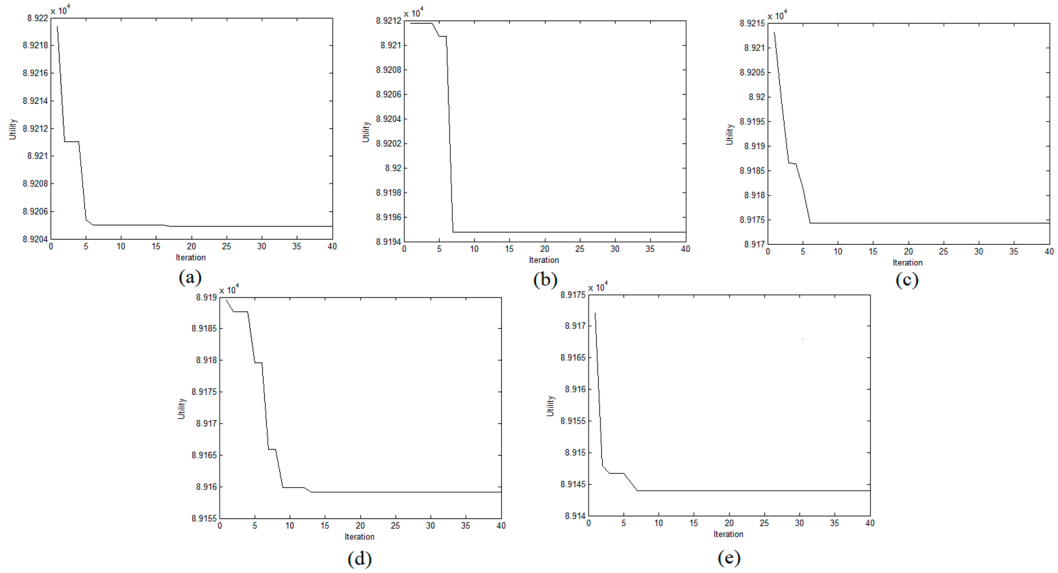

Figure 10. The best answer for objective function during the various iterations in multi-objective particle swarm optimization. (**a**) Result of fuzzy cut (α = 0); (**b**) Result of fuzzy cut (α = 0.3); (**c**) Result of fuzzy cut (α = 0.5); (**d**) Result of fuzzy cut (α = 0.8); (**e**) Result of fuzzy cut (α = 1).

As was mentioned before, among all the solutions obtained from the algorithms, one of the solutions that, in view of the utility function according to Equation (8), had the least amount was selected for performing the activities, and so each activity was carried out via a method that led to an optimum general answer.

Mungle et al. [33] identified 25 solutions for the same benchmark problem using fuzzy clustering-based optimization using the MOGA, and the best compromise solution was also identified. In order to demonstrate that the currently proposed methodology can generate more efficient schedules for construction projects, the reported results from Mungle et al. [33] were ranked based on the evidential reasoning (ER) approach. The weights of the attributes were assumed to be equal in order to

make the comparison valid. The best solutions identified by them [33] were the three solutions with the highest utility scores using the ER approach, along with their respective attribute values.

Rahimi et al. [34] used multi objective particle swarm optimization for a discrete time, cost, and quality trade-off problem and compared it to the genetic algorithm. They reported that the performance of PSO rather than GA in the same condition was better, and that PSO was more efficient. Their result was the same with this research in terms of better performance of PSO.

Zhang and Xing [35] focused on solving the fuzzy time, cost, and quality trade-off (TCQT) problem through a fuzzy-multi-objective particle swarm optimization (FMOPSO) based on a fuzzy multi-attribute utility methodology, in which constrained fuzzy arithmetic operations are adopted to reflect the inequality constraints among the involved fuzzy numbers. The efficiency or the convergence performance of the FMOPSO was compared with a GA method that also adopts the fuzzy multi-attribute utility and constrained fuzzy arithmetic proposed in their research. Their resulting comparison showed that the proposed FMOPSO can solve the TCQT problem with almost the same efficiency as the GA method based on the same fuzzy multi-attribute utility methodology and the constrained fuzzy arithmetic.

4. Conclusions

This study showed that MOPSO's performance was better than that of MOICA in the integer domain. Under the uncertainty conditions ($\alpha = 0$), the time, cost, and quality were computed as 1793.8 h, \$260,571.7, and 46.66%, respectively, by MOPSO. The MOICA results were 1792.5 h, \$260,585.7, and 51.19%, respectively. The risk in the production process was managed by adjusting the alpha value according to the working conditions. Based on the method that is determined for each activity by algorithms, the least time and cost and highest quality, as much as possible, are obtained.

Author Contributions: Methodology, E.K.M. and M.S.; software, E.K.M. and S.R.; validation, E.K.M. and Y.K.C.; formal analysis, E.K.M.; investigation, E.K.M.; writing—original draft preparation, E.K.M. and Y.K.C.; writing—review and editing, E.K.M. and Y.K.C.; supervision, M.S.; project administration, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The financial support provided by the University of Tehran, Iran, is duly acknowledged. I also want to express my deep appreciation of all Mr. Payam Hatami's efforts made to help me in this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. OECD/FAO. *OECD-FAO Agricultural Outlook 2018–2027*; OECD Publishing, Paris/Food and Agriculture Organization of the United Nations: Rome, Italy; Available online: https://doi.org/10.1787/agr_outlook-2018 (accessed on 18 November 2019).
2. Hakimi-Asiabar, M.; Ghodsypour, S.H.; Kerachian, R. Multi-objective genetic local search algorithm using Kohonen's neural map. *Comput. Ind. Eng.* **2009**, *56*, 1566–1576. [CrossRef]
3. Ko, C.H.; Wang, S.F. Precast production scheduling using multi-objective genetic algorithms. *Expert Syst. Appl.* **2011**, *38*, 8293–8302. [CrossRef]
4. Guria, C.; Bhattacharya, P.K.; Gupta, S.K. Multi-objective optimization of reverse osmosis desalination units using different adaptations of the non-dominated sorting genetic algorithm (NSGA). *Comput. Chem. Eng.* **2005**, *29*, 1977–1995. [CrossRef]
5. Jia, J.; Chen, J.; Chang, G.; Tan, Z. Energy efficient coverage control in wireless sensor networks based on multi-objective genetic algorithm. *Comput. Math. Appl.* **2009**, *57*, 1756–1766. [CrossRef]
6. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
7. Ramesh, S.; Kannan, S.; Baskar, S. Application of modified NSGA-II algorithm to multi-objective reactive power planning. *Appl. Soft Comput. J.* **2012**, *12*, 741–753. [CrossRef]

8. Altinoz, O.T.; Yilmaz, A.E. Particle Swarm Optimization with Parameter Dependency Walls and its sample application to the microstrip-like interconnect line design. *AEU Int. J. Electron. Commun.* **2012**, *66*, 107–114. [CrossRef]
9. Liu, H.; Abraham, A.; Clerc, M. Chaotic dynamic characteristics in swarm intelligence. *Appl. Soft Comput. J.* **2007**, *7*, 1019–1026. [CrossRef]
10. Sundar, S.; Singh, A. A swarm intelligence approach to the early/tardy scheduling problem. *Swarm Evol. Comput.* **2012**, *4*, 25–32. [CrossRef]
11. Chen, C.Y.; Chang, K.C.; Ho, S.H. Improved framework for particle swarm optimization: Swarm intelligence with diversity-guided random walking. *Expert Syst. Appl.* **2011**, *38*, 12214–12220. [CrossRef]
12. Parsopoulos, K.E.; Vrahatis, M.N. Recent approaches to global optimization problems through Particle Swarm Optimization. *Nat. Comput.* **2002**, *1*, 235–306. [CrossRef]
13. Coello, C.C.; Lechuga, M.S. MOPSO: A proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1051–1056.
14. Ali, H.; Shahzad, W.; Khan, F.A. Energy-efficient clustering in mobile ad-hoc networks using multi-objective particle swarm optimization. *Appl. Soft Comput. J.* **2012**, *12*, 1913–1928. [CrossRef]
15. Liu, Y.; Niu, B. A multi-objective particle swarm optimization based on decomposition. *Commun. Comput. Inf. Sci.* **2013**, *375*, 200–205.
16. Moslemi, H.; Zandieh, M. Comparisons of some improving strategies on MOPSO for multi-objective (r, Q) inventory system. *Expert Syst. Appl.* **2011**, *38*, 12051–12057. [CrossRef]
17. Mason, K.; Duggan, J.; Howley, E. Multi-objective dynamic economic emission dispatch using particle swarm optimisation variants. *Neurocomputing* **2017**, *270*, 188–197. [CrossRef]
18. Atashpaz-Gargari, E.; Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In Proceedings of the 2007 IEEE congress on evolutionary computation, Piscataway, NJ, USA, 25–28 September 2007; pp. 4661–4667.
19. Nazari-Shirkouhi, S.; Eivazy, H.; Ghodsi, R.; Rezaie, K.; Atashpaz-Gargari, E. Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm. *Expert Syst. Appl.* **2010**, *37*, 7615–7626. [CrossRef]
20. Yousefi, M.; Mohammadi, H. Second law based optimization of a plate fin heat exchanger using Imperialist Competitive Algorithm. *Int. J. Phys. Sci.* **2011**, *6*, 4749–4759.
21. Lucas, C.; Nasiri-Gheidari, Z.; Tootoonchian, F. Application of an imperialist competitive algorithm to the design of a linear induction motor. *Energy Convers. Manag.* **2010**, *51*, 1407–1411. [CrossRef]
22. Kaveh, A.; Talatahari, S. Optimum design of skeletal structures using imperialist competitive algorithm. *Comput. Struct.* **2010**, *88*, 1220–1229. [CrossRef]
23. Talatahari, S.; Azar, B.F.; Sheikholeslami, R.; Gandomi, A.H. Imperialist competitive algorithm combined with chaos for global optimization. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 1312–1319. [CrossRef]
24. Enayatifar, R.; Yousefi, M.; Abdullah, A.H.; Darus, A.N. MOICA: A novel multi-objective approach based on imperialist competitive algorithm. *Appl. Math. Comput.* **2013**, *219*, 8829–8841. [CrossRef]
25. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
26. Ostadi, B.; Daloie, R.M.; Sepehri, M.M. A combined modelling of fuzzy logic and Time-Driven Activity-based Costing (TDABC) for hospital services costing under uncertainty. *J. Biomed. Inform.* **2019**, *89*, 11–28. [CrossRef]
27. *Statistical Year Book*; Statistical Center of Iran: Khorramabad, Iran, 2016.
28. Kaufmann, A. Theory of expertons and fuzzy logic. *Fuzzy Sets Syst.* **1988**, *28*, 295–304. [CrossRef]
29. Gupta, M.M.; Qi, J. Theory of T-norms and fuzzy inference methods. *Fuzzy Sets Syst.* **1991**, *40*, 431–450. [CrossRef]
30. Passino, K.M.; Yurkovich, S. Fuzzy control. In *The Control Systems Handbook: Control System Advanced Methods*, 2nd ed.; Springer: Dordrecht, The Netherlands, 2010; pp. 1225–1252.
31. Ventresca, M.; Harrison, K.R.; Ombuki-Berman, B.M. The bi-objective critical node detection problem. *Eur. J. Oper. Res.* **2018**, *265*, 895–908. [CrossRef]

32. Ab Mutalib, S.M.; Ramli, N.; Mohamad, D.; Mohammed, N. A New Method to Forecast TAIEX Based on Fuzzy Time Series with Trapezoidal Fuzzy Numbers and Center of Gravity Similarity Measure Approach. In *Proceedings of the Third International Conference on Computing, Mathematics and Statistics (iCMS2017), Langkawi, Malaysia, November 2017*; Springer: Singapore, 2019; pp. 489–495.
33. Mungle, S.; Benyoucef, L.; Son, Y.J.; Tiwari, M.K. A fuzzy clustering-based genetic algorithm approach for time-cost-quality trade-off problems: A case study of highway construction project. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1953–1966. [CrossRef]
34. Rahimi, M.; Iranmanesh, H. Multi Objective Particle Swarm Optimization for a Discrete Time, Cost and Quality Trade-off Problem. *World Appl. Sci. J.* **2008**, *4*, 270–276.
35. Zhang, H.; Xing, F. Fuzzy-multi-objective particle swarm optimization for time-cost-quality tradeoff in construction. *Autom. Constr.* **2010**, *19*, 1067–1075. [CrossRef]