Evolutionary Grain-Mixing to Improve Profitability in Farming Winter Wheat

Md Asaduzzaman Noor and John W. Sheppard

Gianforte School of Computing, Montana State University, Bozeman MT-59717, USA
mdasaduzzaman.noor@student.montana.edu, john.sheppard@montana.edu

Abstract. This paper focuses on adapting and applying a genetic algorithm (GA) and differential evolution (DE) to solve the grain (wheat) mixing problem. The proposed algorithms explore a search space that aims at finding a quality mixing of wheat from grain bins that produce the maximum profit at a grain elevator. The experimental results demonstrate that mixing bins provide more profit than not mixing, and that the evolutionary approaches lead to consistently higher profits than the non-evolutionary methods.

Keywords: Grain Mixing · Blending Problem · Profit Maximization · Genetic Algorithm · Differential Evolution.

1 Introduction

Agriculture and agricultural products are essential in sustaining lives on the planet. To feed the population on the Earth efficiently, considerable planning is required. Issues, such as crop rotation or mixed cropping techniques, optimal seeding and fertilizing, proper irrigation, and efficient harvest and distribution are all required for agriculture to meet the needs of most people.

In this paper, we consider an important component of the food supply chain, referred to as grain mixing. The grain considered in our case is wheat that has been stored in different bins in preparation for being sent to a local grain elevator for dissemination. The farmers load the grain from these bins into trucks and sell the wheat at the nearest elevators/markets. The price they get from the elevators depends on the quality (protein level) of the wheat.

Several factors play an essential role when determining the profit from wheat production. One of the critical elements determining the price of a bushel of wheat is protein content, which is affected by several environmental factors, including temperature during the growing season, soil nitrogen levels, genetics, timing, and precipitation. Due to the resulting variation, protein content might change, not only from year to year but also from crop to crop. The technology is available to track the protein content in a bushel of wheat on site; however, it is expensive, making it inaccessible to several smaller farmers. Consequently, most wheat producers end up taking their harvest to the closest elevators and collecting whatever amount is paid.

The purpose of this project was two-fold. First, we investigated if mixing grain increases the profitability of the farmers. The general approach involved comparing a naïve strategy for delivering grain, as if no protein tracking were done, to that of using the protein content to determine a quality mix for delivery. Constrained evolutionary algorithms: specifically a Genetic Algorithm (GA) and Differential Evolution (DE), was used to optimize the mixing of the grain. These methods were also compared to a greedy mixing strategy to determine if the added complexity is beneficial. The final goal was to see if it is worth investing in the technology and infrastructure needed to track the protein level and mix the wheat on site.

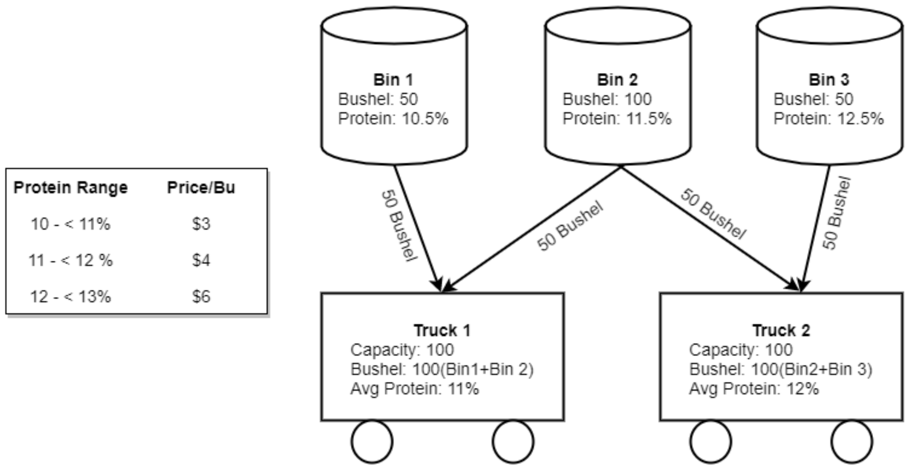
The remainder of this paper is structured as follows. In Section 2, we discuss work related to ours. In Section 3, we present the grain mixing problem. We then present the formal problem statement in Section 4. The data collected for this study is introduced in Section 5 followed by our methodology and experimental design in Sections 6 and 7 respectively. We present our results in Section 8, followed by a discussion of those results in Section 9. The paper ends with future work and our conclusions in Sections 10 and 11.

2 Related Work

The grain mixing problem studied in this paper relates to the classical blending optimization problem. The blending problem seeks to find blends of materials by mixing them to meet specific requirements that lower overall production cost. It is possible to solve the classical blending problems by linear programming (LP) methods if both the objective function and the constraints are linear [9]. There exist a few works that attempted to solve the decision version of the grain mixing problem (also known as wheat blending) with linear programming. Hayka and Cakmalki utilized LP methods capable of predicting the optimal wheat blend ratio for a targeted final quality to produce a bread making flour [6]. Haas used the simplex algorithm to find the optimum blend that satisfies the customer's specific solvent retention capacities (SRC) [5]. In our case, the objective function is non-linear and there is no specific targeted wheat quality, which makes it challenging to solve the problem by only using LP.

Mixed-integer linear programming (MILP) is often used to solve many real-world blending problems [14]; however, MILP is known to be NP-hard. Bilgen and Ozkarahan proposed a MILP model to optimize the cost for the wheat supply chain (blending, loading, transportation, and storage), where the model used a specific blending formula for the mixing [3]. MILP has also been used in the blending of oil [12], water [15], gasoline [8], and chemical fertilizer [2].

Xiang *et al.* proposed a Hybrid-Evolutionary method to solve the wheat blending problem in Australia [10]. Their method used a GA with a heuristic method and a liner-relaxed version of the simplex algorithm to solve the blending problem. Their work closely relates to our mixing problem; however, the problem formulation differs based on the US wheat market. In our problem, there is an added constraint on the capacity a truck can carry, and the farmers have the



Protein Range	Price/Bu
10 - < 11%	$3
11 - < 12 %	$4
12 - < 13%	$6

Fig. 1. Grain mixing example

choice of delivering wheat to multiple elevators. Evolutionary approaches have also been used for blending gasoline [4] and composite laminates [1].

Most of the work for the wheat blending problem in the US market has been done in the milling companies to produce client-specific bread making flour. To the best of our knowledge, our approach is the first to address the grain-mixing problem to increase the profitability of the farmers.

3 Grain-Mixing Problem

The objective of this study was to develop a method for farmers to maximize their profit when selling wheat at local grain elevators. When selling wheat, the price per bushel of grain varies based on a range of protein levels. For example, if the protein level is 10–11%, the price might be $3 per bushel where it might increase to $4/bu with protein levels of 12–13%. The overall profit of a truck depends on the number of bushels and the average protein level. At times, there are many cases where the protein level of a bin is short of reaching a higher price range; therefore, it might be possible to mix grain from a high-quality (in terms of protein level) bin with a low-quality bin to make a better overall profit.

Figure 1 illustrates the grain-mixing process. In the example, there are three bins with different numbers of bushels and protein levels. The table presents elevator prices for different quality grains. Without mixing, the profit obtained from all the bins would be $(50*\$3+100*\$4+50*\$6) = \$850$. However, if we mix grain as shown in the figure, the overall profit would be $(100*\$4+100*\$6) = \$1000$, which increases the profit by $150. There are other factors like mixing cost and delivery cost involved in the overall profit. Only the effect of protein content is shown for simplicity.

4 Problem Statement

The grain mixing problem aims to determine the optimal mix of wheat from storage bins to maximize profit when sold to a set of elevators. More formally, let $B = \{b_1, b_2, ..., b_n\}, E = \{e_1, e_2, ..., e_m\}$,and $T = \{t_1, t_2, ..., t_l\}$ represents the set of bins, elevators, and trucks respectively. Each bin contains a different number of bushels and protein levels represented by b_i^{Bu}, and b_i^{Pr} $\forall b_i \in B$ respectively. For this problem, the mixing of wheat is restricted to drawing from only two bins at a time based on the physical limitations of the farmers. Therefore, a loaded truck will only contain bushels from a maximum of two bins (or one bin if there is no wheat left in the second bin). For example, truck 1 may mix bushels from b_1 and b_5 but not from the combination of b_1, b_5, and b_7. The mixing ratio of a truck is represented by $\alpha \in [0, 1]$. A truck mixing bins b_1 and b_5 with $\alpha = 0.2$ will fill 20% of the truck with bushels from b_1 and 80% from b_5.

4.1 Cost Model

Let C denote the total number of bushels a truck can carry. If a truck t_i mixes grain from two bins b_j and b_k with a mixing ratio α, the total number of bushels in the truck is computed as follows.

$$t_i^{Bu} = b_{i,j}^{Bu} + b_{i,k}^{Bu}$$

where, $b_{i,j}^{Bu}$ represents $(C \times \alpha)$ bushels drawn from b_j to t_i and $b_{i,k}^{Bu}$ represents $(C \times (1-\alpha))$ bushels drawn from b_k to t_i. The remaining bushels of b_j^{Bu} and b_k^{Bu} is updated by subtracting the used amount respectively. Moreover, by mixing grain from two bins, the weighted average protein level of t_i is computed as

$$t_i^{Pr} = \alpha \cdot b_j^{Pr} + (1-\alpha) \cdot b_k^{Pr}.$$

Finally, the revenue of truck t_i depends on the elevator/market prices and the average protein level in the truck. The revenue of truck t_i going to elevator e_j is computed as

$$t_{i,j}^{Rev} = \left[e_j^{b\_price} + \left(\left\lfloor \frac{|t_i^{Pr} - e_j^{b\_Pr}|}{e_j^{up/down\_Pr}} \right\rfloor \right) \times e_j^{up/down\_price} \right] \times t_i^{Bu}$$

where, $e_j^{b\_price}$ is the base protein price of the elevator, t_i^{Pr} is the truck's protein level per unit bushel, $e_j^{b\_Pr}$ is the base protein level of the elevator, $e_j^{up/down\_Pr}$ is the up or down protein level, $e_j^{up/down\_price}$ is the up or down price, and t_i^{Bu} is the total number of bushels in the truck.

To calculate the profit of a truck t_i, the associated mixing cost and delivery cost need to be deducted from the final revenue of the truck. The mixing cost $(t_i^{mix\_cost})$ is associated with the mixing difficulty of the two bins used in the truck, and the delivery cost depends on the cost to deliver the grain to the elevators. The delivery cost is represented as $t_{i,j}^{del\_cost}$ for t_i going to the elevator e_j. The individual profit for truck t_i is then calculated as

$$t_i^{Profit} = \max_j \{t_{i,j}^{Rev} - (t_i^{mix\_cost} + t_{i,j}^{del\_cost})\} \tag{1}$$

A maximum of l trucks are loaded until there are no bushels left in the respective bins. A truck may contain a single bin if there are no bushels left to fill the second bin. In that case, the truck protein level would be the same as the single bin and there will be no mixing cost. Finally, the objective is to maximize the total profit from l trucks given by

$$\text{maximize} \left(\sum_{i=1}^{l} t_i^{Profit} \right) \tag{2}$$

The constraints involved in the optimization problem are as follows:

- For a truck t_i with $b_{i,j}^{Bu}$ bushels drawn from b_j and $b_{i,k}^{Bu}$ bushels drawn from b_k, the total bushels of the truck cannot exceed maximum capacity C.
$$t_i^{Bu} = b_{i,j}^{Bu} + b_{i,k}^{Bu} \leq C \qquad \forall t_i \in T, \forall j,k \in B$$
- The bushels from bin b_j loaded in multiple trucks cannot exceed the total bushels in that bin.
$$\sum_{i=1}^{l} b_{i,j}^{Bu} \leq b_j^{Bu} \qquad \forall b_j \in B$$
- For all trucks and bins,
$$\begin{aligned} t_i^{Bu} &\geq 0 \qquad \forall t_i \in T \\ b_j^{Bu} &\geq 0 \qquad \forall b_j \in B \end{aligned}$$

4.2 Hypothesis

Our goal is to find the best set of bin pairs and the mixing ratios α for l trucks that yield maximum profit. Several optimization techniques will be considered to assess the solution quality. The *NoMix* algorithm will be used to obtain the baseline profit when the trucks contain grain from a single bin to assess if it is better to mix bins in the first place. The *GreedyMix* algorithm will mix grain based on a lookup profit table by taking all the pairwise combinations of bins and α and greedily fill the trucks based on the maximum profit combination (see Section 6.2 for more details). We also apply a genetic algorithm (GA) [7] and differential evolution algorithm (DE) [16] as alternative optimization methods.

We hypothesize that the *GreedyMix* algorithm will yield a higher profit than the *NoMix* algorithm as the expectation is that mixing bins provide a better solution. We hypothesize further that the GA and DE algorithms will yield higher profit than the *GreedyMix* as they provide a more comprehensive search of the search space. For the proposed GA and DE algorithm, both algorithms have been adapted to fit the problem definition (constraint validation), and we hypothesize that both algorithms will yield similar solution quality.

Table 1. Wheat distribution across 16 bins

Year	**Tot Bush**	**Max Bu/Bin**	**Min Bu/Bin**	**Avg Bu/Bin**
2016	114284.8	14836.8	1712.7	7142.8
2017	112417.5	14836.8	2985.3	7026.1
Year	**Tot Prot**	**Max Prot/Bin**	**Min Prot/Bin**	**Avg Prot/Bin**
2016	191.3	13.3	10.1	12.0
2017	190.1	13.8	10.1	11.9

Table 2. Market price for different elevators

Year	**Market**	**BasePrice**	**BaseProtein**	**upPrice**	**upProtein**	**downPrice**	**downProtein**
2016	1	3.32	11.25	0.03	0.75	-0.06	0.75
	2	3.84	11.75	0.25	0.50	-0.30	0.50
	3	3.54	12.00	0.50	0.60	-0.40	0.30
2017	1	4.42	11.50	0.05	0.50	-0.10	0.50
	2	4.47	12.00	0.25	0.50	-0.30	0.50
	3	4.39	12.20	0.50	0.60	-0.40	0.30

5 Dataset

The dataset used in this project was collected from a local Montana farmer who tracks the protein level of his wheat. The data collected is for wheat harvested in 2016 and 2017. All of the wheat was distributed among various bins. For our problem, a bin entry includes the bin id, the average protein level of wheat in that bin, the number of bushels stored, and the site number where the wheat was harvested. Our farmer used 16 bins with different numbers of bushels and protein levels in each bin (Table 1).

The data used also includes a list of elevators that provide the market price for selling wheat. An entry in the elevator list contains the base protein level, the price for the base protein level, the payment added to the base price for higher protein level (upPrice), and the payment deducted from the base price for lower protein level (downPrice). We collected information on three elevators for both years. The market price for different elevators is shown in Table 2.

Figure 2 illustrates the cost function for elevator 3 for year 2017. The orange dot shows the base protein level and base protein price for that elevator. As shown, the price of protein changes as a step function, and the step size differs based on the upProtein (or downProtein) levels from the base protein level.

Multiple trucks, each with a fixed capacity of 8000 bushels, were used to carry the wheat to the elevators. A delivery cost was charged for a fully loaded truck, ranging from \$960 to \$2000 based on the distance between a bin site and an elevator. Moreover, mixing the wheat from multiple bins to change the protein level incurs a mixing cost, ranging from \$8 to \$800. The mixing cost also depends on the site number of the bins; mixing two bins from the same site is less expensive than mixing bins from different sites.

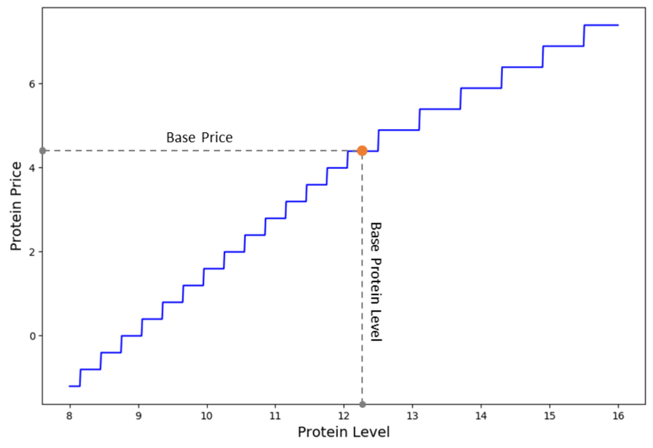


Fig. 2. Cost function of elevator 3 for 2017 dataset

6 Methodology

6.1 No Mixing

The *NoMix* algorithm calculates the overall profit without any bin mixing. The algorithm takes one bin at a time and fills the trucks with the wheat from that bin. If there is not enough wheat to load a truck fully, the algorithm checks if the partially filled truck provides a positive profit. If the profit of a partially filled truck is positive, it takes the truck to the elevator. If not, the grain in that truck is discarded without penalty, and the truck is freed up for use on another bin. The algorithm fills up to l trucks, and the total profit is calculated using the sum in Equation 2.

6.2 Greedy Mix

The *GreedyMix* algorithm first generates a profit table by taking all of the possible combinations of two bins and the mixing ratio α. The value of α is selected from a discrete set of values in increments of 0.1: $\{0.1, 0.2, ..., 0.9\}$ to generate a finite combination set. When generating the lookup table, we compute the profit of a fully loaded truck using Equation 1 based on the selection of two bins and the mixing ratio. The dataset contains a total of 16 bins so there are $\binom{16}{2} = 240$ ways to select two bins maintaining the order. Then there exist 9 discrete values for α which provides a total combination of $240 \times 9 = 2160$ unique truck entries in the profit table. Notice that the combinations $((1, 2), \alpha = 0.5)$ and $((2, 1), \alpha = 0.5)$ provide a different profit since delivery cost only depends on the distance between the second bin and the elevator. Therefore, the table contains an entry for 2160 mixing combinations.

To find a solution, the algorithm sorts the profit table in descending order of mixing profit and traverses it from the top. For any particular combination, if both bins have enough grain left to fill a new truck at a given α, the algorithm fills the truck and records the profit. Otherwise, it partially fills the truck with the remaining grain from both bins, updates the new mixing ratio, and includes it in the solution if the truck provides a positive profit.

After the first sweep through the profit table, if there exists a bin with sufficient grain remaining to make a profit (where other bins do not have any bushels left to make a combination), a new truck with id 0 is filled with the remaining grain from that bin. Finally, the overall profit of l trucks is obtained using the sum in Equation 2.

6.3 Genetic Algorithm

For this experiment, we introduced a novel permutation-based GA representation to apply to the grain mixing problem. We use the permutation to determine how to fill the trucks so as to ensure the constraints remain satisfied. Here, we discuss our specific implementation of the GA.

Population: An individual in the population of the GA algorithm contains a list of candidate trucks. The number of available trucks is set to be sufficiently large to transport all of the grain, and each candidate truck contains a random entry from the profit table (total 2160 mixing combination) defined for the *GreedyMix* algorithm. Therefore, for an individual, the gene contains the tuple $(truck\_id, bin\_pair, \alpha)$, and the list of genes (candidate trucks) represents the chromosome. Each gene in the chromosome represents a unique tuple (a $truck\_id$ is going to appear exactly once), which ensures a feasible solution in the search space. Finally, to evaluate the fitness of an individual, the candidate truck list is used in the *IndividualMix* method. The *IndividualMix* method traverse the candidate trucks in order and load trucks fully/partially based on grain availability. For an individual I with list $candidateTrucks$, the fitness is determined as follows.

$$fitness(I) = IndividualMix()$$

Note that the candidate trucks only provide the bin-pair combination with the mixing ratio; however, the *IndividualMix* generates the final solutions by assigning bushels based on the bin-pair combinations without violating any constraints. *IndividualMix* skips a particular combination in the candidate trucks if both the bins are already empty and then move to the next combination. Therefore, for a candidate truck size of m, only l trucks are used in the final solution $(l < m)$ returned by the *IndividualMix* method.

Selection: The GA uses tournament selection [11] to select parents from the current population to generate new offspring for the next generation. A tournament consists of s randomly selected individuals from the population, and the individual with the highest fitness (tournament winner) is sent to the mating pool to generate new offspring.

Crossover: After selecting the parents, the GA generates offspring using a variant on the ordered crossover (OX) operator [13], which ensures that the new offspring contains a valid permutation of unique genes in its chromosome. To

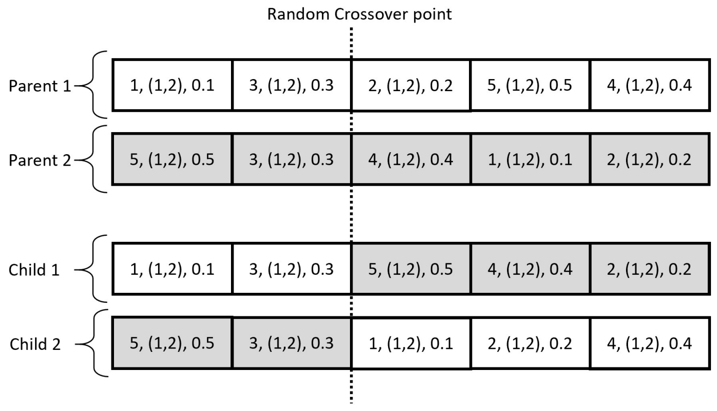


Fig. 3. Crossover operator example

generate the offspring from two parents, a random crossover point is selected from parent 1. Then the offspring copies genes from parent 1 up to the crossover point. Then the offspring sweeps through the genes of parent 2 and copies the gene if it is not already in the gene sequence. The process stops when the sequence is full. Figure 3 shows an example of the crossover operator with five trucks in a pair of chromosomes. The fitness of the new offspring differs due to different truck order in the chromosome.

Mutation: Our mutation operator takes a gene sequence (bin-pair combinations) and randomly swaps the genes of an individual depending on the mutation probability. The operator replaces a gene of a randomly chosen individual with an entry from the profit table that is not already present in the gene sequence.

6.4 Differential Evolution (DE)

The computational steps for our implementation of DE resemble that of the GA so as to limit the differences in performance to the mechanics of the algorithm alone. The initialization of the population and the fitness function used in DE is the same as the GA. The major difference lies in the *selection*, *crossover*, and *mutation* operators.

Selection: The selection operator selects the better individual by comparing the fitness of the trial vector $u_{i,g}$, and the target vector $x_{i,g}$. The selected vector replaces the current vector individual for the next generation as follows.

$$x_{i,g+1} = \begin{cases} u_{i,g}, & \text{if } f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g}, & \text{otherwise} \end{cases}$$

Crossover: The crossover operator closely follows the standard DE crossover operator. After the mutation phase, the crossover operation is applied on the target vector $x_{i,g}$ and the mutant vector $v_{i,g}$ to obtain the offspring/trial vector

$u_{i,g}$. The following is a common crossover technique known as binomial crossover to obtain the trial vector $u_{i,g}$.

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j}^g, & \text{otherwise} \end{cases}$$

where CR represents the crossover rate: a user defined parameter ranges in $[0, 1)$. j_{rand} represents a random value in the vector dimension $[1, D]$. The trial vector $u_{i,g}$ takes the j^{th} real-value of either the target vector or mutant vector based on the condition specified above. Our implementation uses this process to generate a new trial vector; however, the trial vector first copies the chromosome of the mutant vector $v_{i,g}$ and replaces the gene with the target vector $x_{i,g}$ if it is not already present in the trial vector and $(rand(0,1) \leq CR$ or $j = j_{rand})$ condition is satisfied, as in the OX operator. Here, CR represents the probability of the crossover rate.

Mutation: The algorithm uses the "DE/Best/1" strategy to create the mutant vector. The first donor vector represents the best individual in the current generation, the second donor vector is a random individual selected from the current generation, and the third donor vector is selected randomly from the profit table to add diversity in the population. All of the donor vectors are mutually exclusive.

The chromosome representation is not real-valued and therefore the standard mutation equation had to be changed to adapt to the problem representation. The new mutant vector copies genes from the three donor vectors based on the mutation factor F. In this case, the mutant vector copies genes from the first donor vector (generation best) with probability F and from either the second or third donor vector with probability $(1-F)$. It excludes genes if they already exist in the gene sequence to maintain a valid permutation. The process is similar to the GA crossover (Figure 3), but instead of two parents, the operator uses the three donor vectors to create a mutant individual.

7 Experimental Design

Twelve additional datasets along with the two real datasets were created to evaluate the performance of all algorithms. Among the twelve datasets, two datasets were created by flipping the elevator prices of the real dataset (using 2017 elevator data with 2016 bin data and vice-versa). The remaining ten datasets were created by randomly varying the number of bushels and the protein levels of each bin in the original datasets. Therefore, there are five simulated datasets for the year 2016 and five for the year 2017.

The results of the *NoMix* and *GreedyMix* algorithms are deterministic; however, GA and DE are stochastic. Therefore, to evaluate the performance of the stochastic algorithms, we ran 10 experiments for each dataset and recorded the average overall profit.

The GA and DE implementations create random individuals by taking a small subset of the profit table entry for the initial population. The intuition is

Table 3. Parameter settings for GA and DE algorithms.

Algo	**Population**	**#Iteration**	**Tournament**	**Mutation**
GA	100	500	5	0.2
Algo	**Population**	**#Iteration**	**CR**	**Mutation**
DE	100	500	0.9	0.5

that for a small subset of the profit table, the *IndividualMix* method returns a different result than the *GreedyMix* algorithm based on the few choices of bin combinations, and the goal of the GA and DE algorithms is to find the best sequence from the profit table for which the profit is maximum. Therefore, the subset size (candidate trucks) plays an important role in the overall solution. The subset has to be selected in a way so that it contains enough bin combinations to empty all the bushels of the bins. If the subset is too small, then it may not contain an entry for a particular bin; therefore, all of the bushels of that bin may be unused in the solution. If the subset is too large, then it will perform like the *GreedyMix* algorithm with all the choices of filling a truck.

To select the subset size, random individuals were created with subset size ranges from 50 to 150, increasing by 10 on each experiment. Then the solution returned from the individual was checked to see if they collected all of the wheat available. For a subset size of 100, more than 90% of the individuals returned solutions collecting all of the wheat; therefore, the number of candidate trucks was fixed to 100. Even if an individual fails to utilize all the bushels and excludes a significant portion of the wheat, the intuition is that the individual will be replaced by others with higher fitness. And sometimes excluding a small portion of wheat might increase the overall profit because the cost of using an extra truck for that small amount might decrease the overall profit.

The hyperparameters used in the GA and DE algorithms were tuned manually. Table 3 shows all the parameter values used in the GA and DE algorithms for all test cases. These sets of parameters provided the highest average profit for the real datasets.

To evaluate the effectiveness of the evolutionary algorithms further, we also introduced a *Random* algorithm that creates 100 random individuals from the profit table and returns the solution from the best individual. The *Random* algorithm helps us assess if the evolutionary algorithms are exploring the search-space efficiently to provide a better solution than random search. Due to stochastic nature, the *Random* algorithm was also run 10 times, and the average overall profit was recorded.

8 Results

Table 4 shows the overall profit obtained from the different algorithms for each of the test cases studied. The notation R_year denotes the real dataset for respective years and RF_year denotes the flipped version of the real dataset. A^*_year

Table 4. Algorithm performance (*Profit* in thousands of dollars).

Dataset	**GA**		**DE**		**Random**		**Greedy**		**NoMix**	
	Profit	TR	Profit	TR	Profit	TR	Profit	TR	Profit	TR
R_2016	**435.8**	19	435.4	19	428.9	19	430.5	20	424.5	18
R_2017	496.2	18	**497.0**	18	491.2	18	491.6	20	487.1	19
RF_2016	**506.2**	19	505.9	19	501.9	19	499.9	21	499.7	18
RF_2017	**428.9**	18	427.8	18	421.2	18	423.1	19	418.0	19
A1_2016	415.7	19	**416.0**	19	407.6	19	394.5	19	395.4	20
A2_2016	562.0	22	**562.4**	22	550.8	22	533.1	23	545.4	23
A3_2016	**591.1**	24	589.0	27	569.8	26	543.9	31	570.0	25
A4_2016	**573.9**	24	571.0	26	558.0	25	520.4	29	539.9	25
A5_2016	533.8	25	**534.0**	25	517.6	25	496.9	23	512.2	24
A1_2017	**621.5**	22	617.9	24	599.2	23	581.7	20	608.9	23
A2_2017	**537.0**	20	536.8	21	522.0	20	515.8	18	520.6	22
A3_2017	**676.0**	25	672.5	26	664.3	26	655.8	25	643.9	24
A4_2017	450.5	16	**452.5**	17	443.5	16	435.6	16	430.1	18
A5_2017	**685.4**	24	**685.4**	25	672.4	24	656.0	25	663.5	23

denotes the artificial dataset for respective years. The stochastic algorithm's results are the average of 10 runs. The column "Profit" shows the average best profit in thousands of US dollars, and "TR" shows the average number of trucks used in the solution. The bold values represent the best profit obtained for a dataset, and the underlined values represent a substantial profit increase (more than $5000) comparing with the base value (*NoMix* results)[1]. Unfortunately, we do not have the cost associated with the production (harvesting) that would be necessary to provide a more complete estimate of profit.

For all test cases, the evolutionary algorithms performed significantly better than all other methods. For real datasets (including the flipped versions), the *GreedyMix* and *Random* algorithms always performed better than the *NoMix*. However, for the artificial datasets, *GreedyMix* failed to outperform *NoMix* 80% of the time, and the *Random* algorithm failed to outperform *NoMix* 20% of the time. The performance of the evolutionary algorithms shows that grain mixing always provides a higher profit than no mixing at all.

Table 5 shows an example of one complete solution obtained from the GA algorithm for the R_2017 dataset. In the table, the column name "Bin_Pair" shows the pair of bins mixed to fill the truck, "Pair1_Bu" and "Pair2_Bu" show the number of bushels taken from each bin in the pair, "Protein" shows the weighted average protein level, "Load" gives the total amount of wheat loaded

[1] Note that statistical hypothesis testing was not done since *NoMix* and *GreedyMix* only yield a single, deterministic solution.

Table 5. Best solution from the GA algorithm for dataset R_2017.

Bin_Pair	**MixRatio**	**Pair1_Bu**	**Pair2_Bu**	**Protein**	**Load**	**Elevator**	**Profit**
(1, 2)	0.20	1600.0	6400.0	13.49	8000.0	3	41992.0
(15, 2)	0.77	2985.3	892.5	13.50	3877.8	3	20147.5
(9, 12)	0.89	6539.8	800.0	13.03	7339.8	2	35378.0
(3, 12)	0.80	6395.3	1600.0	12.57	7995.3	2	36538.5
(6, 12)	0.90	7200.0	800.0	12.52	8000.0	2	36240.0
(8, 12)	0.38	2400.0	3889.7	11.52	6289.7	2	27228.0
(11, 7)	0.74	4921.3	1712.7	11.85	6634.0	2	28658.8
(1, 4)	0.60	4800.0	3200.0	11.93	8000.0	2	34552.0
(1, 4)	0.23	1600.0	5325.5	11.57	6925.5	2	29911.2
(1, 5)	0.60	4800.0	3200.0	11.74	8000.0	2	34480.0
(1, 5)	0.46	2036.8	2400.0	11.54	4436.8	2	19122.6
(16, 6)	0.93	7200.0	503.7	11.48	7703.7	1	32124.3
(14, 5)	0.98	5600.0	113.4	12.11	5713.4	2	24110.4
(14, 8)	0.63	3057.4	1792.7	11.61	4850.1	2	20224.8
(13, 10)	0.68	5050.0	2400.0	10.60	7450.0	1	30470.5
10	1.00	4892.5	0.0	10.11	4892.5	1	19765.7
16	1.00	6309.0	0.0	11.40	6309.0	1	26308.5
Total					**112417.5**		**497252.8**

in the truck (max 8000) and "Elevator" identifies the elevator where the truck gets the highest profit.

All of the grain-mixing algorithms provide a similar solution format but with different bin-pair mixing. A complete solution shows a farmer how to load each truck with bushels from respective bins to accomplish the solution's overall profit.

9 Analysis and Discussion of Results

In most of the cases, the results obtained from the experiments supported the hypothesis. The proposed GA and DE algorithms yielded higher profits than the *NoMix* algorithm for all test cases, demonstrating their efficiency and effectiveness in grain mixing. The *GreedyMix* algorithm, however, did not appear to be an obvious choice as its profits degraded than that of the *NoMix* algorithm in case of the artificial datasets only.

At first glance, it may appear that the *GreedyMix* algorithm should provide the optimal results as it creates all possible combinations of mixes and greedily selects the mix based on the maximum profit. This claim might be true if there existed an infinite number of bushels in all of the bins. But, in the real world, our problem contains a different number of bushels for the different bins, and filling a truck greedily based on profit may not be always possible if there are no bushels left in a specific bin. Furthermore, the truck might be partially filled based on the remaining bushels of bins giving a lower profit than the actual one. Therefore, for all the test cases, *GreedyMix* not only failed to provide optimal

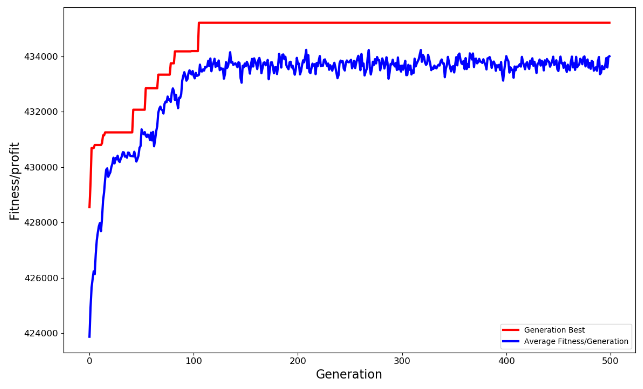


Fig. 4. Convergence example of the GA algorithm (R_2016)

results but also gave a lower profit (in the artificial test cases) than the *NoMix* algorithm. The evolutionary methods do suggest, however, that taking a sub-optimal bin-pair combination can yield a higher overall profit than the greedy optimal choice due to the varying bin sizes.

The example solution in Table 5 contains two trucks with a single bin (bin 10 and 16). These trucks suggest that after the first sweep in the candidate trucks list, grain remained in bin 10 and bin 16. This indicates that the candidate trucks do not have any truck entry with bin pair $(10, 16)$. Therefore, the solution loads the truck by separately taking bushels from each bin (no mixing cost incurred). Even without bin pair $(10, 16)$, this solution returned the best profit for the dataset.

To check the convergence of the GA and DE algorithms, the best individual in each generation, and the average fitness of the total population in each generation were tracked. If the generation best individual did not change for more than 100 consecutive iterations, the algorithm was determined to have converged to a solution. Figure 4 and figure 5 shows the convergence of the GA and DE algorithms respectively for a test run with R_2016 dataset.

The average population fitness in Figure 4 is a spiked curve because the crossover and mutation operator in the GA may create a new individual with a lower/higher fitness giving a lower/higher average profit. However, for DE, Figure 5 shows the average population is a smooth curve since DE always replaces a current individual with a better-fit individual and improving the average population fitness for each generation. The results obtained from the *Random* algorithm also suggest that the evolutionary algorithms are efficiently exploring the search-space for finding a better solution.

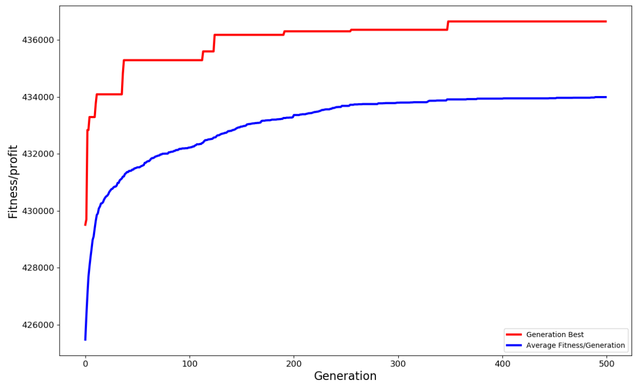


Fig. 5. Convergence example of the DE algorithm (R_2016)

10 Future Work

This work has considered several approaches to determining how to maximize the profit of wheat production by mixing different numbers of bushels to change the protein level contained in a truck being delivered to an elevator. The price of a bushel of wheat depends on its protein level as measured at the elevator. The idea is to see if mixing different protein levels can yield a mix such that the farmer will receive more profit than taking them separately. This work makes certain assumptions for the problem representation like mixing only two bins at a time and used a discrete mixing ratio α for mixing two bins. The effect of using more than two bins in the mixing and using a continuous value for mixing ratio $\alpha \in [0, 1]$ will be considered as future work. In addition, using different mutation operators for the DE algorithm and the crossover operator for the GA algorithm and their impact on the overall solution quality will be considered. It is also our intent to explore the suitability of comparing to an MILP model; however, the inherent nonlinearity of the objective function would seem to suggest optimization quality would be limited. Furthermore, a more realistic cost model that includes the production cost associated with harvesting, cost of the protein device and supporting infrastructure, and alternative representations of the model will be considered.

11 Conclusion

In this paper, we evaluated two different evolutionary algorithms—the genetic algorithm and differential evolution—to solve the grain (wheat) mixing problem. The algorithms explore a search space that aims at finding the best mix of wheat that will produce the maximum profit on a farm. This wheat is usually stored in grain bins and are subsequently delivered to elevators for sale. These elevators buy the wheat from the farmers before being sold to customers. The experimental results show that both the GA and DE algorithms find better solutions (increased profit) than no mixing, random mixing, and greedy mixing, which implies that the farmers should invest in the infrastructure of grain mixing as long as the profit exceeds the amortized cost of that infrastructure.

References

1. Adams, D.B., Watson, L.T., Gürdal, Z., Anderson-Cook, C.M.: Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness. Advances in Engineering Software **35**(1), 35 – 43 (2004)
2. Ashayeri, J., van Eijs, A., Nederstigt, P.: Blending modelling in a process manufacturing: A case study. European Journal of Operational Research **72**(3), 460 – 468 (1994)
3. Bilgen, B., Ozkarahan, I.: A mixed-integer linear programming model for bulk grain blending and shipping. International Journal of Production Economics **107**(2), 555 – 571 (2007)
4. Chen, X., Wang, N.: Optimization of short-time gasoline blending scheduling problem with a dna based hybrid genetic algorithm. Chemical Engineering and Processing: Process Intensification **49**(10), 1076 – 1083 (2010)
5. Haas, N.: Optimizing Wheat Blends for Customer Value Creation: A Special Case of Solvent Retention Capacity. MS Thesis, Kansas State University, USA (2011)
6. Hayta, M., Cakmalki, U.: Optimization of wheat blending to produce breadmaking flour. Journal of Food Process Engineering **24**, 179 – 192 (08 2001)
7. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
8. Jia, Z., Ierapetritou, M.: Mixed-integer linear programming model for gasoline blending and distribution scheduling. Industrial & Engineering Chemistry Research **42**(4), 825–835 (2003)
9. Karloff, H.: Linear Programming. Birkhauser Boston Inc., USA (1991)
10. Li, X., Bonyadi, M.r., Michalewicz, Z., Barone, L.: A hybrid evolutionary algorithm for wheat blending problem. TheScientificWorldJournal **2014**, 967254 (02 2014)
11. Miller, B., Goldberg, D.: Genetic algorithms, tournament selection, and the effects of noise. Complex System **9**, 193–212 (1995)
12. Moro, L.F.L., Pinto, J.M.: Mixed-integer programming approach for short-term crude oil scheduling. Industrial & Engineering Chemistry Research **43**(1), 85–94 (2004)
13. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the traveling salesman problem. In: Proceedings of the Second International Conference on Genetic Algorithms and Their Application. p. 224–230 (1987)
14. Pochet, Y., Wolsey, L.A.: Production Planning by Mixed Integer Programming. Springer Publishing Company, Incorporated, 1st edn. (2010)
15. Randall, D., Cleland, L., Kuehne, C.S., Link, G.W.B., Sheer, D.P.: Water supply planning simulation model using mixed-integer linear programming "engine";. Journal of Water Resources Planning and Management **123**(2), 116–124 (1997)
16. Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. rep., TR-95-012, International Computer Science Institute, Berkeley (1995)