INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS
DOI: 10.46300/91013.2020.14.10 Volume 14, 2020

Distribution Problems, Their Modifications and Applications

MILOŠ ŠEDA, JAN ROUPEC
Institute of Automation and Computer Science
Brno University of Technology
Technická 2, 616 69 Brno, Czech Republic
seda@fme.vutbr.cz, roupec@fme.vutbr.cz

JINDŘIŠKA ŠEDOVÁ
Faculty of Economics and Administration
Masaryk University
Lipová 41a, 602 00 Brno, Czech Republic
jsedova@econ.muni.cz

***Abstract*—In this paper, we deal with well-known distribution problems and discuss their restrictions, extensions and modifications including a possible application in agriculture. We show that the transportation problem can be transformed to an assignment problem using special constraints, but because of NP-hardness it needs quite different methods of its solving. Another modification of the transportation problem, the crop problem, has an application in agriculture, but we must deal with uncertain data. We propose a genetic algorithm and fuzzy logic approach for solving these problems.**

***Keywords*—transportation; assignment problem; crop problem, PERT; heuristic; genetic algorithm; fuzzy number**

I. INTRODUCTION

The Hitchcock *transportation problem* with m sources (supply points, factories) and n destinations (demand points, clients) can be formulated using linear programming as follows [1]:

Minimise

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \ldots, m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, n \tag{3}$$

$$x_{ij} \geq 0, \quad i = 1, \ldots, m, \; j = 1, \ldots, n, \tag{4}$$

where
z is the total transportation cost,
c_{ij} is the unit shipping cost from source i to destination j,
x_{ij} is the number of units shipped from source i to destination j,
a_i is the supply of the source i,
b_j is the demand of destination j,
and only a single commodity is transported.

Since there is only one commodity, a d estination can receive its demand from more than one source. Therefore, the objective is to determine how much should be shipped from each source to each destination so as to minimise the total transportation cost.

If total commodity supply equals to total demand, the problem is said to be a *balanced transportation problem*:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \tag{5}$$

If (5) is not satisfied, then it becomes an *unbalanced transportation problem*.

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j \tag{6}$$

If total supply exceeds total demand, see (6), we can balance the problem by adding a dummy destination to absorb the excess supply. Shipments to this destination are assigned a cost of zero.

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j \tag{7}$$

If a transportation problem has a total supply that is strictly less than total demand, see (7), the problem has no feasible solution, because one or more demands cannot be satisfied. In such situations a penalty cost is often associated with unmet demand and the total penalty cost is desired to be minimal.

There are several methods for solving the balanced transportation problem as follows:

- The Northwest Corner Method
- The Least Cost Method
- Vogel's Approximation Method

We can also solve the transportation problem using specialised software tools, e.g. GAMS, LINDO, LINGO, or MS Excel Solver.

If it is possible to both ship into and out of the same point of the transport network, then we speak about a *transshipment* (or *transhipment*) *problem*. For the transshipment problem, you can ship from one supply point to another or from one demand point to another.

The Hitchcock formulation of the transportation problem may also be extended considering fixed charges associated with supply points (e. g. warehouses), means of transport, their capacity, cost of transport by vehicles to 1 km, which enables to determine the number of trips due to volume, transport in two levels: primary source – warehouses – destinations, admitting the possibility of direct transport from the primary source to destinations, etc.

Instead, we turn our attention to problems that seem to have nothing with transportation, but their formulation can be obtained from the basic model of the transportation problem.

II. Assignment Problem

Let us assume a balanced transportation problem in which all supplies and demands are equal to 1 and the same number of sources and destinations [2].

An example of this situation is an assignment of tasks to persons or jobs to machines. Additionally, we assume that each person must do on e and only one task, and each task must be done by only one person. The problem is to find an assignment which minimizes the total cost (or the total time) for processing all tasks.

Consider the following denotations:

n = number of persons = number of tasks,
x_{ij} represents the assignment of person i to task j, x_{ij}=1 if the assignment is done, and x_{ij} = 0 otherwise,
c_{ij} is the cost the assignment of person i to task j.

Hence we get the following model of the (*linear*) *assignment problem* (AP):

Minimise

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{8}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \dots, n \tag{9}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \dots, n \tag{10}$$

$$x_{ij} \in \{0,1\}, \quad i = 1, \dots, n,\ j = 1, \dots, n \tag{11}$$

We can see that:

(1) The assignment problem is a special case of the transportation problem where

$m = n$,
$a_i = 1,\ i = 1,2, \dots, m,$
$b_j = 1,\ j = 1,2, \dots, n,$
$x_{ij} \in \{0,1\},\ i = 1,2, \dots, m,\ j = 1,2, \dots, n.$

(2) The assignment problem can also be understood as a problem of finding a permutation

$$\begin{pmatrix} 1 & 2 & \dots & n \\ \pi_1 & \pi_2 & \dots & \pi_n \end{pmatrix} \tag{12}$$

where person i to task π_i is assigned, i=1, … , n, and

$$\sum_{i=1}^{n} c_{i,\pi_i} \to \min \tag{13}$$

Since there are n! permutations for n tasks, finding the solution for large instances is not reachable in a r easonable amount of time, and thus heuristic methods must be used. We present a genetic algorithm approach.

III. Genetic Algorithm for AP

The skeleton for GA can be described as follows [3]:

```
generate an initial population ;
evaluate fitness of individuals in the population ;
repeat
    select parents from the population;
    recombine (mate) parents to produce children ;
    evaluate fitness of the children ;
    replace some or all of the population by the children
until a satisfactory solution has been found ;
```

Since the principles of GAs are well-known, we will only deal with GA parameter settings for the problems to be studied. Now we describe the general settings [4], [5].

Individuals in the population (*chromosomes*) are represented as binary strings of length n, where a value of 0 or 1 at the i-th bit (*gene*) implies that x_i = 0 or 1 in the solution respectively.

The *population size* N is usually set between n and 2n. Many empirical results have shown that population sizes in the range [50, 200] work quite well for most problems.

Initial population is obtained by generating random strings of 0s and 1s in the following way: First, all bits in all strings are set to 0, and then, for each of the strings, randomly selected bits are set to 1 until the solutions (represented by strings) are feasible.

The *fitness function* corresponds to the objective function to be maximised or minimised.

There are three most commonly used methods of *selection* of two parent solution for *reproduction*: proportionate selection, ranking selection, and tournament selection. The tournament selection is perhaps the simplest and most efficient among these three methods. We use the *binary tournament selection* method where two individuals are chosen randomly from the population. The more fit individual is then allocated a

reproductive trial. In order to produce a child, two binary tournaments are held, each of which produces one parent.

The *recombination* is provided by the *uniform crossover* operator, which has a better recombination potential than do other crossover operators as the classical *one-point* and *two-point* crossover operators. The uniform crossover operator works by generating a random crossover mask B (using Bernoulli distribution) which can be represented as a binary string $B = b_1b_2b_3 \cdots b_{n-1}b_n$ where n is the length of the chromosome. Let P_1 and P_2 be the parent strings $P_1[1]$, **...** ,$P_1[n]$ and $P_2[1]$, **...** ,$P_2[n]$ respectively. Then the child solution is created by letting: $C[i] = P_1[i]$ if $b_i = 0$ and $C[i] = P_2[i]$ if $b_i = 1$. *Mutation* is applied to each child after crossover. It works by *inverting M* randomly chosen bits in a string where M is experimentally determined. We use a mutation rate of $5/n$ as a lower bound on the optimal mutation rate. It is equivalent to mutating five randomly chosen bits per string.

When v child solutions have been generated, the children will replace v members of the existing population to keep the population size constant, and the reproductive cycle will restart. As the replacement of the whole parent population does not guarantee that the best member of a population will survive into the next generation, it is better to use *steady-state* or *incremental replacement* which generates and replaces only a few members (typically 1 or 2) of the population during each generation. The *least-fit* member, or a randomly selected member with *below-average fitness*, are usually chosen for replacement.

Termination of a GA is usually controlled by specifying a maximum number of generations t_{max} or relative improvement of the best objective function value over generations. Since the optimal solution values for most problems are not known, we choose $t_{max} \leq 5000$.

In our implementation the population was set to 50 and the number of iterations to $10 \times n^2$.

Permutations of n tasks for the initial population are best to be generated randomly, e.g. by the following procedure:

```
Randomize;
for i := 1 to n do  { 1, 2, … , n }
    perm[i] := i;
for i := n downto 2 do
    begin  j := 1+Trunc(i*Random)
           x := perm[i]; perm[i] := perm[j];
           perm[j] := x
    end;
```

As to the crossover operation, we cannot use the traditional two-point crossover, because it would lead to infeasible solutions. If we change the middle parts of the parent chromosomes P_1=(1,10,7,2,8,9,4,6,5,3), P_2=(5,8,2,9,7,4,1,10,3,6) between the 4-th and 7-th position, then we would obtain offspring (1,10,7,9,7,4,1,6,5,3) and (5,8,2,2,8,9,4,10,3,6) that correspond to no permutations, because some jobs are duplicated or omitted. We used the so called crossover in a partially mapped representation where the genes in the middle part of one chromosome are ordered in its offspring by their occurrence in the second parent chromosome.

In the literature, slight modifications of these shift operations can be found, e.g. *1stSwap*, *FullSwap*, *DoubleCut, DoublePointShift* and *RightDoublePoint* [6], [7].

Genetic algorithms may be enhanced by other heuristic methods to perform local searches. A sophisticated approach was presented in [8].

First, define the *distance of two permutations* P_1, P_2. The most frequent definitions are the following

- *precedence distance* – computed as the number of pairs of jobs $\{i,j\}$ where i precedes j in P_1, but does not precede it in P_2,
- *positional distance* – given as the sum of differences of jobs on the same positions in both permutations.

This method combines the genetic algorithm with a local search and the Metropolis criterion used in simulated annealing. Its main modification is based on a special crossover, called *multistep crossover fusion* (MSXF). Let P_1 and P_2 be two parent permutations. On permutation P_1 a local search is applied that is, to a certain extent, influenced by permutation P_2, which serves a reference point. All permutations from a neighbourhood of P_1 are ordered in the ascending order by their distance from permutation P_2. With a probability inversely proportional to this sequence, a neighbour, P_s, is selected. If this neighbour solution gives a shorter schedule than the one computed from the permutation of jobs P_1, i.e. $C_{\max}(P_s) < C_{\max}(P_1)$, then P_s is accepted. In the opposite case, it is accepted with the probability of $e^{-(C_{\max}(P_s)-C_{\max}(P_1))/T}$ as in the Metropolis criterion. For simplicity, the temperature T is considered constant. The selection of neighbours is repeated until a neighbour is selected. The number of iterations is also set fixed. From all the accepted permutations, only the best one is included in the next iteration of the genetic algorithm.

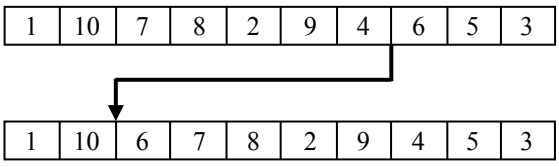


Fig. 1. Shift mutation

For mutation we considered three operators:

- *exchange mutation* (it exchanges two randomly selected positions in a permutation),
- *shift mutation* (it removes a value at one position and puts it at another position), see Fig. 1, and
- *mutation* inspired by well-known *Lin*-2-*Opt change operator* usually used for solving the travelling salesman problem [9]. Here first two elements are added to the permutation (into positions 0 and $|n|+1$) and then the same values are assigned to them to

simulate a cyclic tour. Two 'edges' (pairs of neighbour elements in permutation) are randomly chosen ((p_1, p_2) and (q_1, q_2) say), the inner elements p_2, q_1 are swapped and the elements between p_2 and q_1 are reversed.

The best results were achieved with the shift mutation.

IV. Crop Problem

Let us denote:

$p_1, \dots, p_m$ = grounds

$r_1, \dots, r_m$ = area of grounds

$k_1, \dots, k_n$ = crops

c_{ij}, $i = 1,2, \dots, m$, $j = 1,2, \dots, n$

= profit from 1 ha of ground sown by crop k_j

x_{ij} = number of hectares of ground p_i sown by crop k_j

TABLE I. Crop Problem

crops / grounds	k_1	k_2		k_n	area [ha]
p_1	c_{11}	c_{12}	…	c_{1n}	r_1
p_2	c_{21}	c_{22}	…	c_{2n}	r_2
…					…
p_m	c_{m1}	c_{m2}	…	c_{mn}	r_m

In the crop problem [10], [11] is to find the optimum sowing of areas by crops for given yields of crops (in quintals per hectare) and contractual purchase prices so as to maximise the total profit [12].

From Table I we get the following system of equations:

$$\begin{aligned} x_{11} + x_{12} + \dots + x_{1n} &\le r_1 \\ x_{21} + x_{22} + \dots + x_{2n} &\le r_2 \\ \dots & \\ x_{m1} + x_{m2} + \dots + x_{mn} &\le r_m \end{aligned}$$

$$x_{ij} \ge 0, \quad i = 1, 2, \dots, m, \quad j = 1, 2, \dots, n$$

$$z = c_{11}x_{11} + c_{12}x_{12} + \dots + c_{1n}x_{1n} + c_{21}x_{21} + c_{22}x_{22} + \dots + c_{2n}x_{2n} + \dots + c_{m1}x_{m1} + c_{m2}x_{m2} + \dots + c_{mn}x_{mn} \to \max$$

It can be expressed as follows:

Maximise

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{14}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le r_i, \quad i = 1, \dots, m \tag{15}$$

$$x_{ij} \ge 0, \quad i = 1, \dots, m, \; j = 1, \dots, n \tag{16}$$

If we require for each crop that were sown at a certain minimum area, then the task has become an example of maximisation version of the *generalized distribution problem*. It is included in Table II and the corresponding model follows:

TABLE II. Crop Problem with Minimum Requirements

crops / grounds	k_1	k_2		k_n	area [ha]
p_1	c_{11}	c_{12}	…	c_{1n}	r_1
p_2	c_{21}	c_{22}	…	c_{2n}	r_2
…					…
p_m	c_{m1}	c_{m2}	…	c_{mn}	r_m
minimum requirements for crop sowing area	d_1	d_2		d_n	

Maximise

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{17}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le r_i, \quad i = 1, \dots, m \tag{18}$$

$$\sum_{i=1}^{m} x_{ij} \ge d_j, \quad j = 1, \dots, n \tag{19}$$

$$x_{ij} \ge 0, \quad i = 1, \dots, m, \; j = 1, \dots, n \tag{20}$$

V. Crop Problem with Uncertain Yields

Of course, yields of crops are only estimated and in real conditions cannot be considered as deterministic.

This situation can be solved with techniques inspired by PERT.

Denote

a = estimate of the crop yields under the most favourable conditions

b = estimate of the crop yields under the least favourable conditions

m = most likely value for the crop yields

PERT requires the assumption that estimated parameter follows a beta distribution. Then its mean values may be approximated by the following equation:

$$y = \frac{a + 4m + b}{6} \tag{21}$$

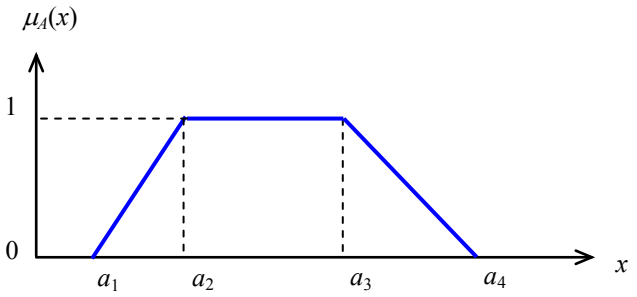
Since the beta distribution is not guaranteed, we propose a fuzzy approach.

Let us assume now that crop yields are given by fuzzy numbers [13].

A *fuzzy number A* is a fuzzy set represented by 4-tuple (a_1, a_2, a_3, a_4) and a piecewise continuous membership function with the following properties:

- $a_1 \le a_2 \le a_3 \le a_4$
- $\mu_{A(x)} = 0$ for $x \le a_1, x \ge a_4$
- $\mu_{A(x)} = 1$ for $a_2 \le x \le a_3$
- μ_A is increasing on $[a_1, a_2]$ and decreasing on $[a_3, a_4]$.

The fuzzy set defined by the membership function is an example of fuzzy number. In this paragraph we consider trapezoidal fuzzy numbers, see (22) and Fig. 2.



$$\Pi(x,a,b,c,d) = \begin{cases} 0 & \text{for } x < a_1 \\ \dfrac{x-a_1}{a_2-a_1} & \text{for } a_1 \le x \le a_2 \\ 1 & \text{for } a_2 < x \le a_3 \\ \dfrac{a_3-x}{a_4-a_3} & \text{for } a_3 < x \le a_4 \\ 0 & \text{for } x > a_4 \end{cases} \tag{22}$$

Fig. 2. Trapezoidal fuzzy number

The addition of fuzzy numbers can be derived using the extension principle and it is determined as follows:

$$X^F \oplus Y^F = (x_1, x_2, x_3, x_4) \oplus (y_1, y_2, y_3, y_4) = \\ = (x_1+y_1, x_2+y_2, x_3+y_3, x_4+y_4) \tag{23}$$

When the maximum operation would be derived in the same way, then its results may not be trapezoidal fuzzy numbers. Therefore we approximate this operation as follows.

$$\max(X^F, Y^F) = (\max(x_1, y_1), \max(x_2, y_2), \max(x_3, y_3), \max(x_4, y_4)) \tag{3.20}$$

To find a solution of the crop problem which maximises the total profit, we must compare fuzzy numbers in some way, which is a difficult problem. An *ordering relation* $\le$ can be defined e.g. as follows:

$$X^F \le Y^F \Leftrightarrow (x_1 \le y_1) \wedge (x_2 \le y_2) \wedge (x_3 \le y_3) \wedge (x_4 \le y_4) \tag{24}$$

However, this relation is not a complete ordering relation, as fuzzy numbers X^F, Y^F satisfying

$$(\exists\, i,j \in \{1,2,3,4\}): (x_i < y_i) \wedge (x_j > y_j) \tag{25}$$

are not comparable by $\le$.

It is evident that, for non-comparable fuzzy numbers X^F, Y^F, this fuzzy max operation results in a fuzzy number different from both of them. For example, for X^F =(4,9,12,16) and Y^F =(6,8,13,15), we get from (23) a fuzzy max (6,9,13,16) which differs from X^F and Y^F.

This problem can be solved by assigning a scalar value to each resulting fuzzy number and comparing these scalars.

We use the fuzzy ranking method described in [14], modified for the case of trapezoidal fuzzy numbers. This method uses inverse functions

$g_A^L : [0,1] \to [a_1, a_2]$ and $g_A^R : [0,1] \to [a_3, a_4]$ derived from functions $f_A^L : [a_1, a_2] \to [0,1]$ and $f_A^R : [a_3, a_4] \to [0,1]$, respectively.

From $y = \dfrac{x-a_1}{a_2-a_1}$ (increasing part) and $y = \dfrac{x-a_4}{a_3-a_4}$ (decreasing part) we can derive that

$$g_A^L = a_1 + (a_2 - a_1)y, \quad g_A^R = a_4 + (a_3 - a_4)y \tag{26}$$

The ranking function is defined as the distance between the *centroid point* (x_0, y_0) and the origin

$$R(A) = \sqrt{(x_0)^2 + (y_0)^2} \tag{27}$$

where

$$x_0 = \frac{\int_{\text{Supp } A} x\mu_A(x)dx}{\int_{\text{Supp } A} \mu_A(x)dx}, \quad y_0 = \frac{\int_0^1 yg_A^L dy + \int_0^1 yg_A^R dy}{\int_0^1 g_A^L dy + \int_0^1 g_A^R dy} \tag{28}$$

and Supp A is the support of A.

Fuzzy numbers A, B are then ranked by their ranking function values $R(A)$ and $R(B)$.

VI. Conclusions

In this paper we studied the well-known transportation problem and presented several modifications which are important in various application areas.

The assignment problem can also be derived from the linear transportation problem, but it cannot be solved for large instances using linear programming methods and heuristics must be used. We presented a g enetic algorithm (GA) approach and GA parameter settings.

Finally, we studied the crop problem, important in agriculture engineering, and generalised it for case of uncertain crop yields. Instead of traditional interval or PERT approach we propose a fuzzy algebra based on fuzzy numbers and their transformation to scalar values.

In the future, we foresee further tests with other stochastic heuristics, including more applications with special constraints in studied problems.

References

[1] M. B. Stroh, A Practical Guide to Transportation and Logistics. Dumont: Logistics Network, 2006.

[2] E. Cela, The Quadratic Assignment Problem: Theory and Algorithms. Berlin: Springer, 2010.

[3] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, 1996.

[4] M. Šeda, "Mathematical Models of Flow Shop and Job Shop Scheduling Problems." International Journal of Applied Mathematics and Computer Science, nol. 4, no. 4, pp. 241-246, 2007.

[5] M. Šeda, "Heuristic Set-Covering-Based Postprocessing for Improving the Quine-McCluskey Method." International Journal of Computational Intelligence, vol. 4, no. 2, pp. 139-143, 2007.

[6] D. E. Goldberg, The Design of Innovation (Genetic Algorithms and Evolutionary Computation). Dordrecht: Kluwer Academic Publishers, 2002.

[7] Z. Michalewicz and D.B. Fogel, How to Solve It: Modern Heuristics. Berlin:,Springer-Verlag, 2002.

[8] T. Yamada, C. R. Reeves, "Permutation Flowshop Scheduling by Genetic Local Search,". Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA 97. Glasgow, pp. 232-238, 1997

[9] G. Gutin, and A. P. Punnen (eds.). The Traveling Salesman Problem and Its Variations. Dordrecht: Kluwer Academic Publishers, 2002.

[10] N. H. Mitchell, Mathematical Applications in Agriculture, 2nd ed. London: Cengage Learning, 2011.

[11] J Thornley, and J. France, Mathematical Models in Agriculture, 2nd ed., Wallingford, Oxfordshire: CABI, 2006.

[12] A. Bechar, and G. Vitner, "Work Planning in Packing Houses of Flowers Mixed Farms to Increase the yield," Agricultural Engineering International: CIGR Journal, vol. 13, no. 3, 2011, 10 pp., manuscript no. 1901.

[13] V. Novák, Fuzzy Sets and their Applications. Bristol: Adam Hilger, 1989.

[14] C.-H Cheng, "A New Approach for Ranking Fuzzy Numbers by Distance Method," Fuzzy Sets and Systems, vol. 95, pp. 307-317, 1998.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US