Finding optimal solutions of vaguely defined assignment problems

MIHA MOŠKON
University of Ljubljana
Faculty of Computer and Information Science
Tržaška cesta 25, SI-1000 Ljubljana
SLOVENIA
miha.moskon@fri.uni-lj.si

Abstract: The article presents an extension of the Hungarian algorithm (also known as Kuhn-Munkres algorithm) which is used for solving the assignment problems in polynomial time. The fact that the original version of the algorithm is only able to solve the assignment problems with precisely defined inputs (i.e. demands and resources) presents a major problem in many real-life scenarios while the nature of these problems is such that inputs are commonly defined only vaguely (i.e. fuzzily). In order to solve them, their precise formalization is needed which is normally far from being a straightforward procedure and can present large costs in the meaning of time and money. Fuzzy logic on the other hand successfully copes with the processing of imprecise data.

The Hungarian algorithm was extended with the introduction of fuzzy logic methods in order to be able to efficiently solve vaguely defined assignment problems. The extended version of the algorithm (i.e. fuzzy Hungarian algorithm) is thus able to cope with vaguely defined assignment problems, can be used more efficiently (i.e. with no further formalization of vaguely defined terms) and in a wider scope of assignment problems than the basic approach.

Here we describe the basic version of the Hungarian algorithm which was firstly presented by Harold Kuhn. Its extension with the introduction of fuzzy methods is also described. Its usage is justified by the comparison of the results between its crisp (i.e. basic) and fuzzy (i.e. extended) version on the same problem.

Key–Words: Hungarian algorithm, fuzzy logic, assignment problems, fuzzy Hungarian algorithm, optimal resource assignment

1 Introduction

Cost minimization in many real-life problems is often associated with optimal resource assignment. The optimal assignment of available resources to specified demands is so called assignment problem [1]. Each assignment of a specific resource to a specific demand has its own cost. Appropriate resource from the set of resources available has to be assigned to each of the demands in the way that the cost of the whole assignment is minimal. The naive approach to solve this problem would be the exhaustive search of the solution space (i.e. brute force solution), but would lead us to exponential time complexity. The assignment could on the other hand be made in polynomial time with Hungarian algorithm [2, 3, 4]. The original version of this algorithm is only able to solve exactly defined assignment problems. On the other hand there is a lack of exact knowledge in many real-life scenarios. These problems are therefore often vaguely defined, i.e. by imprecisely defined demands, or by imprecisely defined resources or even both. It is well known that fuzzy logic can successfully cope with such data [5, 6, 7, 8]. Hungarian algorithm can thus be extended with fuzzy methods in order to successfully solve vaguely defined assignment problems.

Here we describe the structure of the assignment problems and the basic version of Hungarian algorithm which was firstly presented by Harold Kuhn [2]. The basics of fuzzy logic are also presented and the extension of Hungarian algorithm to Fuzzy Hungarian algorithm which successfully copes with vaguely defined assignment problems. Its usage is justified by the comparison of the results between the basic Hungarian algorithm and its extended (i.e. fuzzy) version on the same problem.
Table 1: Presentation of assignment problems with cost matrix, where \( c_{ij} \) presents the cost of the assignment of resource \( r_i \) to the demand \( d_j \).

<table>
<thead>
<tr>
<th></th>
<th>( d_1 )</th>
<th>( d_2 )</th>
<th>\ldots</th>
<th>( d_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1 )</td>
<td>( c_{11} )</td>
<td>( c_{12} )</td>
<td>\ldots</td>
<td>( c_{1m} )</td>
</tr>
<tr>
<td>( r_2 )</td>
<td>( c_{21} )</td>
<td>( c_{22} )</td>
<td>\ldots</td>
<td>( c_{2m} )</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\ddots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( r_n )</td>
<td>( c_{n1} )</td>
<td>( c_{n2} )</td>
<td>\ldots</td>
<td>( c_{nm} )</td>
</tr>
</tbody>
</table>

2 Methods

2.1 Assignment problem

Assignment problem [1] is a combinatorial optimization problem where \( n \) resources have to be assigned to \( m \) demands (\( n \geq m \)) and optimality is defined with minimal cost of the assignment. Each assignment of a certain resource to a certain demand has its own cost. We can present the assignment problem with so called cost matrix (see Table 1).

Exactly one resource has to be assigned to each of the demands and each of the resources can be chosen at most once. We can calculate the cost function of a specific assignment as the sum of all costs in the assignment made. Optimality is achieved when an assignment with a minimal value of cost function is found (i.e. no other assignment with lower cost exists).

2.2 Hungarian algorithm

Hungarian algorithm [2, 3, 4] is a combinatorial optimization algorithm which identifies the optimal assignment in \( O(n^3) \) time. In fact, it was shown that it is possible to find optimal assignment with improved Hungarian algorithm even faster [9], but for the reasons of simplicity only its basic version is presented here. We can describe it with the following steps:

1. Transfer the cost matrix to square matrix with the introduction of new fictional demands if number of resources is bigger than number of demands (costs of fictional demands must be larger than maximal cost in the matrix for all resources).
2. Subtract the minimal element of each row from all elements in the same row.
3. Subtract the minimal element of each column from all elements in the same column.
4. Select rows and columns across which you draw lines, in a way that all the zeros are covered and that number of lines is minimal.
5. Find a minimal element that is not covered by any line. Add its value to each element covered by both lines and subtract it from each element that is not covered by any line. Go back to step 4. If nothing was done in step 5, go to step 6.
6. Assign resources to demands starting in the top row. Assign a resource only when there is only one zero in a row. As you make an assignment delete a row and a column from which you have made it. If there is no such assignment possible, move to the next row. Stop when all assignments have been made. If you reached the bottom of the matrix, proceed to next step.
7. Assign resources to demands starting in the leftmost column. Assign a resource only when there is only one zero in a row. As you make an assignment delete a row and a column from which you have made it. If there is no such assignment possible, move to the next column. Stop when all assignments have been made. If no assignments were made in this step, choose a random zero value and make an assignment. Proceed to step 6.

2.3 Fuzzy Logic

Fuzzy logic was introduced by Zadeh in 1965 [5] and presents a generalization of classical, i.e. crisp logic. According to crisp logic certain element \((x)\) is a member or is not a member of a certain set \((S)\). We can describe its membership to the set with membership function, which is defined as

\[
\mu_S(x) = \begin{cases} 
0; & x \notin S \\
1; & x \in S 
\end{cases} \tag{1}
\]

Degree of membership of certain element to certain set which is calculated according to membership function can in crisp scenario have whether value 0 whether value 1

\[
\mu_S(x) \in \{0, 1\} \tag{2}
\]

In fuzzy logic on the other hand, certain element can be a partial member of a certain set. Degree of membership of certain element to certain set which is calculated according to fuzzy membership function can therefore have any value in the interval \([0, 1]\):

\[
\mu_S(x) \in [0, 1] \tag{3}
\]

Fuzzy variables can be defined over fuzzy sets. Each fuzzy variable can partially belong to several fuzzy sets. Lets presume that fuzzy variable \( age \) is
defined over the fuzzy sets young, middle aged and old. Boundaries among these sets are fuzzily defined, which means that certain person can for example be young with membership value 0.1, middle aged with membership value 0.7 and old with membership value 0.05 (see Figure 1).

![Figure 1: Fuzzy variable age defined over fuzzy sets young, middle aged and old.](image)

In order to process information with fuzzy logic fuzzy rule set is used, i.e. set of rules with fuzzy variables for their inputs and outputs. Values of input fuzzy variables (premises) therefore define values of output fuzzy variables (consequents) regarding the fuzzy rule set. An example of fuzzy rule set with fuzzy variable age as an input variable and fuzzy variable suitability as an output variable is as follows:

if age is young then suitability is false

if age is middle aged then suitability is true

if age is old then suitability is false

It is obvious that fuzzy rules can be constructed and understood very easily. With the evaluation of the rules, memberships of output fuzzy variable to sets defined over that variable are calculated (in our example membership to fuzzy sets false and true).

In many cases output fuzzy variables have to be converted to crisp values (i.e. defuzzified) in order to use them in other (crisp) segments of our system. Many methods can be used for that purpose. Usually Centre of Gravity method is used which calculated centre of gravity of calculated fuzzy variable as its crisp value. Following equation is used

$$
\frac{\int_0^\infty x \cdot \mu(x)dx}{\int_0^\infty \mu(x)dx}, \quad (4)
$$

where \( \mu_S(x) \) presents the membership element \( x \) to set \( S \) [5, 6].

### 2.4 Fuzzy Hungarian Algorithm

Basic version of Hungarian algorithm was extended with fuzzy logic methods in order to efficiently solve vaguely defined assignment problems. Fuzzy Hungarian algorithm finds a solution where suitabilities of each resource to each demand are optimal (i.e. maximal possible). We can describe the fuzzy Hungarian algorithm through five phases.

#### 2.4.1 Demands formalization

Each of the demands requires exactly one resource. Required resource characteristics are given with at least one demand property. Demands and required properties must therefore be identified. Exact definition of demands can be a very difficult or sometimes even impossible task to perform. On the other hand our algorithm supports vaguely - fuzzily defined demands and therefore overcomes this barrier. Algorithm also supports crisp demand property definition.

#### 2.4.2 Resources formalization

Resources and their properties also have to be specified. Method presented supports fuzzy and crisp specifications of each resource property.

#### 2.4.3 Formation of Fuzzy Rules

The suitability of each resource to each demand has to be calculated using the fuzzy rule set. Rules are established regarding fuzzy demands. Each property that is about to be dealt fuzzily has to be defined as a fuzzy variable. Fuzzy variables are included in fuzzy rules. Suitability of each resource to each demand can be calculated using the values of fuzzy properties of each resource as inputs to fuzzy rule set.

#### 2.4.4 Suitability Matrix Construction

Using the previously constructed fuzzy rule set fuzzy suitabilities of each resource to each demand can be calculated. These values have to be defuzzified in order to use them in a combination with Hungarian algorithm. We can construct a suitability matrix (i.e. matrix of suitabilities - see Table 2) using the defuzzified (crisp) values.

#### 2.4.5 Solution calculation

In order to solve the problem using the basic version of Hungarian algorithm cost matrix has to be calculated from the given suitability matrix. The calculation of cost matrix is straightforward and is performed using the following equation:
Table 2: Suitability matrix of fuzzy assignment problem, where $s_{ij}$ presents the crisp suitability of resource $r_i$ to the demand $d_j$.

\[
\begin{array}{cccc}
  & d_1 & d_2 & \cdots & d_m \\
 r_1 & s_{11} & s_{12} & \cdots & s_{1m} \\
r_2 & s_{21} & s_{22} & \cdots & s_{2m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
r_n & s_{n1} & s_{n2} & \cdots & s_{nm} \\
\end{array}
\]

\[c_{ij} = M - s_{ij},\quad (5)\]

\[\forall i : i \in \mathbb{R}, 1 \leq i \leq n,\]

\[\forall j : j \in \mathbb{R}, 1 \leq j \leq m,\]

where $M$ is maximal element value in the suitability matrix (in our example $M$ equals 1). Having a cost matrix, problem can be solved using the algorithm described in Section 2.2.

3 Experiments

The behavior of our method was tested on different training sets of Slovenian Army Engineer brigade urgent missions. The goal of the algorithm was to make a vehicle convoy formation where the algorithm had to decide which vehicles to use based on the chosen mission. Each mission therefore defined the demands, which were input data for the algorithm and were vaguely defined in many cases. Vehicles, which were at disposal, presented the resources that could be chosen for the mission. Vehicles were specified by several parameters which were mostly dependent on their technical characteristics. We were also dealing with vaguely defined parameters like purpose of the vehicle (urban, long distance or all-terrain vehicle). Algorithm was tested on several training sets of demands and several sets of available vehicles. The basic version of Hungarian algorithm was unable to make an appropriate assignment in many cases while its fuzzy version made an optimal assignment in all cases.

4 Conclusion

Hungarian algorithm was extended with fuzzy logic methods in order to be able to solve vaguely defined assignment problems without their exact formalization. The methods used in an extended version of the algorithm were described here. Algorithm was tested on the training set from real-life scenarios of Slovenian army. Comparison among results of its basic version was also made. Good results, especially in comparison with crisp Hungarian algorithm justified the usage of Fuzzy Hungarian algorithm in potential future applications.

References: