Solving the vaguely defined assignment problems

Miha Moškon

***Abstract*—**Assignment problems are defined with two sets of inputs, i.e. set of resources and set of demands. Assignment of each resource to each demand has its own cost. Exactly one resource has to be assigned to each of the demands in such way, that maximal cost of the assignment is minimal when comparing to other assignments. Hungarian algorithm (also known as Kuhn-Munkres algorithm) is able to find an optimal solution of assignment problems in polynomial time, but is only able to solve assignment problems with precisely defined demands and resources. This 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. Formalization of their properties 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 article presents an extension of the Hungarian algorithm with the introduction of fuzzy logic methods – *fuzzy Hungarian algorithm*. Vaguely defined resources and demands can be easily described with fuzzy values which present an input to fuzzy Hungarian algorithm. The extended version of the algorithm is therefore 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.

Basic version of the Hungarian algorithm which was firstly presented by Harold Kuhn is presented in this article. Its extension with fuzzy logic methods is described and its usage on an example of vaguely defined assignment problem is demonstrated. Its benefits were also justified by the comparison of the results between the basic version of Hungarian algorithm and the fuzzy version of Hungarian algorithm on the same problem.

***Keywords*—**Hungarian algorithm, fuzzy logic, assignment problems, fuzzy Hungarian algorithm, optimal resource assignment

I. 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 [1,2,3]. 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 maximal cost in the assignment is minimal when comparing to the maximal cost of any other valid assignment. 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 the usage of Hungarian algorithm [4,5,6]. The original version of this algorithm is only able to solve exactly defined assignment problems, where each demand and each resource is described with exactly defined properties. 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 properties of demands, or by imprecisely defined properties of resources or even both. It is well known that fuzzy logic can successfully cope with inexactly, vaguely defined data [7,8,9,10,11]. Hungarian algorithm can thus be extended with fuzzy logic 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 [4,6]. 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. Method proposed is demonstrated on an example of vaguely defined assignment problem where different people with certain qualifications and properties have to be assigned to different tasks regarding their demands. Usage of Fuzzy Hungarian algorithm was justified by the comparison of the results between the basic Hungarian algorithm and its extended (i.e. fuzzy) version on the same problem, which is presented in section Experiments.

Miha Moškon is with the Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia.
(e-mail: miha.moskon@fri.uni-lj.si)

II. Methods

A. 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).

	d_1	d_2	$\dots$	d_m
r_1	c_{11}	c_{12}	$\dots$	c_{1m}
r_2	c_{21}	c_{22}	$\dots$	c_{2m}
.	.	.	$\dots$	.
.	.	.	$\dots$	.
.	.	.	$\dots$	.
r_n	c_{n1}	c_{n2}	$\dots$	c_{nm}

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.

Exactly one resource has to be assigned to each of the demands and each of the resources can be chosen at most once. Specific assignment A_k can be presented by an m-tuple

$$A_k = (k_1, k_2, \dots, k_m), \tag{1}$$

where k_i present an index of resource assigned to demand i. Assignment is valid if following conditions are fulfilled

$$k_i \neq k_j; \forall i \neq j, \\ i, j \in \{0,1,\dots,n\}. \tag{2}$$

All the values in the m-tuple present an index of a certain resource and all the values in the m-tuple differ among each other.

We can calculate the cost function $c(A_k)$ of a specific assignment A_k as the maximal cost in the assignment made, which is defined by the following equation

$$c(A_k) = \max_{i\in\{1,2,\dots,m\}} c_{k_i i}, \tag{3}$$

where c_{ij} presents the cost of the assignment of resource r_i to the demand d_j.

Optimality is achieved when an assignment with a minimal value of cost function cannot be found (i.e. no other assignment with lower cost exists). In order to find such solution with naive approach, i.e. exhaustive search of the solution space (brute force solution), one has to calculate the cost function of all possible assignments. The time complexity of such solution would be

$$\frac{n!}{(n-m)!}, \tag{4}$$

where n is the number of available resources and m is the number of demands that need to be satisfied. The necessary condition for the existence of the solution is that the number of resources is higher or equal to the number of demands

$$n >> m. \tag{5}$$

The time complexity of brute force approach therefore equals

$$T(m,n) = O(n!). \tag{6}$$

Size of the space that has to be investigated drastically increases with the number of given demands and consecutively with the number of available resources. The brute force solution is therefore useless for the majority of real life problems.

B. Hungarian algorithm

Hungarian algorithm [4,5,6] is a combinatorial optimization algorithm which identifies the optimal assignment in polynomial time:

$$T(m,n) = O(n^3). \tag{7}$$

Comparing to brute force approach optimal assignment can be found much faster with Hungaran algorithm and could therefore be used for arbitrary problem sizes (*m, n >> 0*). In fact, it was shown that it is possible to find optimal assignment with improved Hungarian algorithm even faster [12]

$$T(m,n) = O(n^{2.22} r^{0.11}). \tag{8}$$

For the reasons of simplicity only its basic version is presented here. The extension of the Hungarian algorithm to the Fuzzy Hungarian algorithm could on the other hand be made on the improved (i.e. faster) version of the algorithm.

We can describe the basic version of Hungarian algorithm with seven steps which are executed sequentially. Some steps are iterated more than once. In order to describe the algorithm following symbols will be used.

Maximal cost presents the maximal cost defined in the cost matrix and can be calculated as follows

$$c_{\max} = \max_{i\in\{1,2,\dots,n\}, j\in\{1,2,\dots,m\}} c_{ij}. \tag{9}$$

Minimal resource cost presents the minimal cost of a certain resource *i; i ∈ {1,2,...,n}* – i.e. minimal element in the row *i* of the cost matrix:

$$c_{i,\min} = \min_{j\in\{1,2,\dots,m\}} c_{ij}. \tag{10}$$

Minimal demand cost presents the minimal cost of a certain demand *j; j ∈ {1,2,...,m}* – i.e. minimal element in the column *j* of the cost matrix.

$$c_{\min,j} = \min_{i\in\{1,2,\dots,n\}} c_{ij}. \tag{11}$$

Hungarian algorithm can be thus described in the following way:

1. Transfer the cost matrix of size *n* x *m* to square matrix with the introduction of new fictional demands if number of resources is bigger than number of

demands. Fictional demands are introduced with the addition of new columns in the cost matrix. Costs of fictional demands are the same for all the resources and must be larger than maximal cost ($c_{\max}$) in the matrix.

2. Subtract the minimal element of each row i ($c_{i,\min}$) from each element in the same row.

3. Subtract the minimal element of each column j ($c_{\min,j}$) from all elements in the same column.

4. Select rows and columns across which you draw lines, in a way that all the zeros in the cost matrix 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. If none such element cannot be found in step 5, go to step 6. Otherwise go back to step 4.

6. Assign resources to demands starting in the top row. Assign a resource only when there is only one zero in a row (only when a unique assignment can be made). As you make an assignment delete the row and the 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 have reached the bottom of the matrix, proceed to next step (proceed to step 7).

7. Assign resources to demands starting in the leftmost column. Assign a resource only when there is only one zero in a row (only when a unique assignment can be made). 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.

Algorithm described is performed in $O(n^3)$ time.

C. Fuzzy Logic

Fuzzy logic was introduced by Zadeh in 1965 [7] 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

$$\mu_s(x) = \begin{cases} 0; x \notin S \\ 1; x \in S \end{cases}. \tag{12}$$

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:

$$\mu_s(x) \in \{0,1\}. \tag{13}$$

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]:

$$\mu_s(x) \in [0,1]. \tag{14}$$

Fuzzy variables can be defined over fuzzy sets. Each fuzzy variable can partially belong to several fuzzy sets. Let us 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
Fig. **1**).

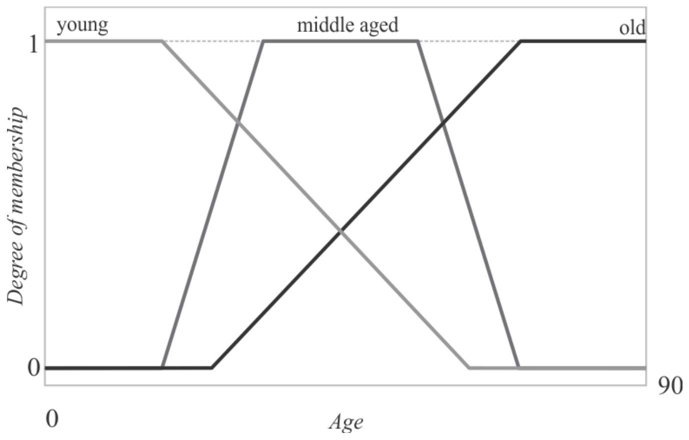

Fig. 1 Fuzzy variable age defined over fuzzy sets *young*, *middle aged* and *old*.

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 (*precedents*) 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:

rule 1: if *age* is *young* then *suitability* is *false*

rule 2: if *age middle aged* then *suitability* is *true*

rule 3: 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 variables to sets defined over that variable are calculated (in our example membership to fuzzy sets *false* and *true*).

In many cases output fuzzy values 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 *Center of Gravity* method is used which calculates the center of gravity of calculated fuzzy variable as its crisp value. Following equation is used

$$\frac{\int_0^\infty x \cdot \mu_s(x)dx}{\int_0^\infty \mu_s(x)dx}, \tag{15}$$

where $\mu_S(x)$ presents the membership element x to set S [7,8]. The method calculates the Center of Gravity of the surface described by the fuzzy value [13].

D. 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.

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) Resources formalization

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

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.

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 deffuzified 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.

	d_1	d_2	…	d_m
r_1	s_{11}	s_{12}	…	s_{1m}
r_2	s_{21}	s_{22}	…	s_{2m}
.	.	.	…	.
.	.	.	…	.
.	.	.	…	.
r_n	s_{n1}	s_{n2}	…	s_{nm}

Table 2 Suitability matrix of fuzzy assignment problem, where s_{ij} presents the crisp suitability of resource r_i to the demand d_j.

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:

$$\begin{gathered} c_{ij} = M - s_{ij}, \\ \forall i \in \aleph, 1 \le i \le n, \\ \forall j \in \aleph, 1 \le i \le m, \end{gathered} \tag{16}$$

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. The algorithm therefore finds an optimal solution regarding the following criteria:

$$s(A_k) = \min_{i\in\{1,2,\ldots,m\}} s_{k,i}, \tag{17}$$

where s_{ij} presents the suitability of the assignment of resource r_i to the demand d_j. Optimality is achieved when an assignment with a maximal value of optimality criteria is achieved (i.e. no other assignment with higher value of suitability exists).

III. Demonstration of Fuzzy Hungarian algorithm

The behavior of developed algorithm will be demonstrated on an example of vaguely defined assignment problem, where different people have to be assigned to different tasks. Each person is defined with set of properties which describe the capabilities and attributes he or she has. Each task is defined with demands that have to be fulfilled by the person who is assigned to it. Suitabilities of each person to each task can be calculated regarding the properties of each person and properties of each task.

1) Demands formalization

We presume that each demand is defined with the following properties:

- *age*: descriptional parameter that is vaguely defined with three possible values, i.e. *young, middle aged* and *old*.
- *education*: seven possible levels (from 0 to 6) according to International Standard Classification of Education (*ISCED*),
- *physical condition*: descriptional parameter that is vaguely defined with three possible values, i.e. *bad*, *average* and *good*.
- *driving license*: five different categories (A,B,C and D). One can be in several categories.

Let us presume that we have to find the optimal assignment for the following demands:

- d_1: *young* person with *B* driving license in a *good physical condition*.
- d_2: person with at least 4th *degree of education*.
- d_3: *middle aged* person in *average physical condition* with at least 5th *degree of education*.

Demands can be presented in a tabular format (see Table 3).

	age	*education*	*physical condition*	*driving license*
d_1	*young*	-	*good*	*B*
d_2	-	*4th*	-	-
d_3	*middle aged*	*5th*	*average*	-

Table 3 Formalization of demands that have to be satisfied. Certain cell is empty if demand is not defined for a certain parameter.

2) Resources formalization

Resources are defined with the following properties:

- *age*: person's age in years.
- *education*: seven possible levels (from 0 to 6) according to International Standard Classification of Education (*ISCED*),
- *physical condition*: physical condition evaluated from 0 (bad) to 1 (good).
- *driving license*: five different categories (A,B,C and D). One can be in several categories.

Let us presume that we have the resources which are presented in Table 4 at our disposal.

	age	*education*	*physical condition*	*driving license*
r_1	*29*	*2nd*	*0.9*	*AB*
r_2	*45*	*5th*	*0.4*	*ABD*
r_3	*28*	*5th*	*1*	*AB*

Table 4 Formalization of resources at our disposal.

3) Formation of Fuzzy Rules

a) Fuzzy variables

Fuzzy variables that will be dealt fuzzily within the fuzzy rule set have to be defined. In our example we are dealing with two vaguely defined terms, i.e. *age* and *physical condition*. Other parameters, i.e. *education* and *driving license* can be dealt without fuzzy logic.

Fuzzy variable *age* can be defined over three fuzzy sets, i.e. *young*, *middle aged* and *old*. We can define it using the trapezoidal parametric notation [8] as

- *young* $(0,0,25,50)_p$,
- *middle aged* $(25,50,50,75)_p$,
- *old* $(50,75,100,100)_p$.

Defined fuzzy variable *age* is also presented in Figure 2.

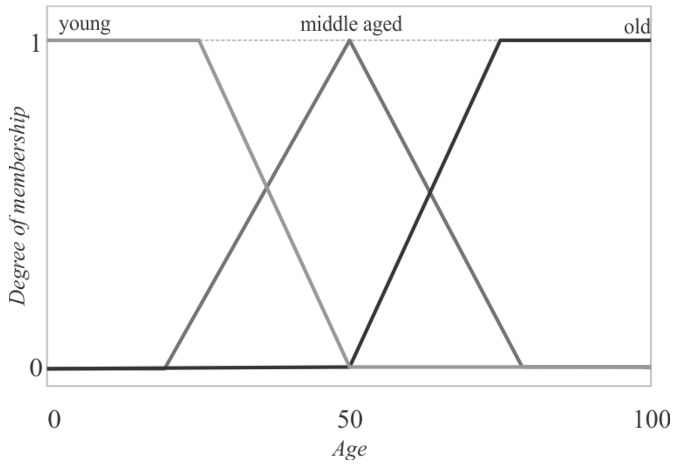

Fig. 2 Fuzzy variable *Age* used in the demonstration of Fuzzy Hungarian algorithm.

Similarly fuzzy variable *pc (physical condition)* can be defined as

- *bad* $(0,0,0.25,0.5)_p$,
- *average* $(0.25,0.5,0.5,0.75)_p$,
- *good* $(0.5,0.75,1,1)_p$.

Defined fuzzy variable *physical condition* is also presented in Figure 3.

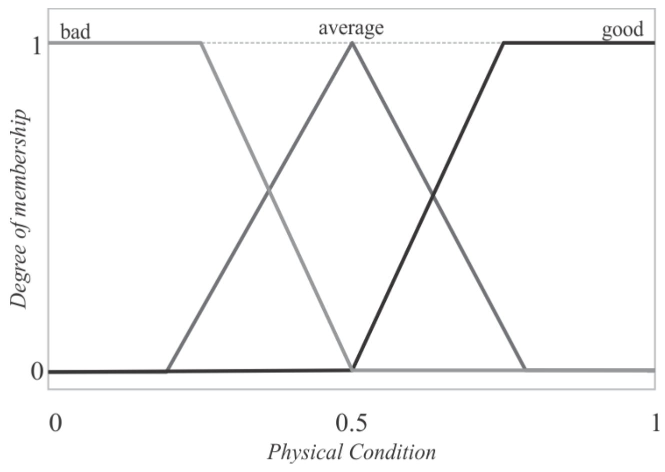

Fig. 3 Fuzzy variable *physical condition* used in the demonstration of Fuzzy Hungarian algorithm.

In order to calculate the suitabilities fuzzy variable *suitability* is defined as

- *false* $(-1,0,0,1)_p$,
- *true* $(0,1,1,2)_p$.

Defined fuzzy variable *suitability* is also presented in Figure 3.

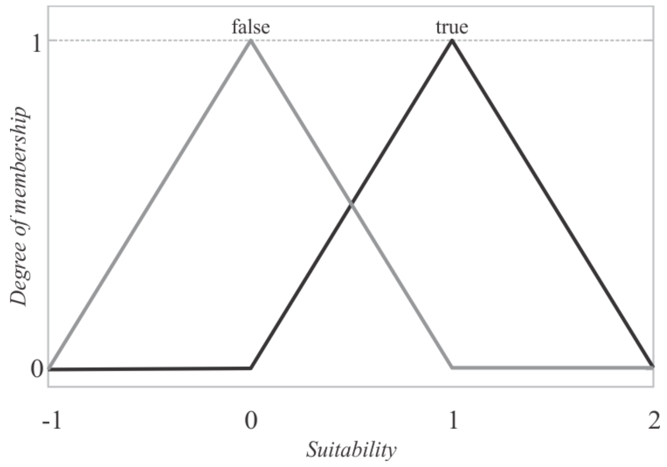

Fig. 4 Fuzzy variable *suitability* used in the demonstration of Fuzzy Hungarian algorithm.

b) Fuzzy rules

Regarding the defined fuzzy variables following fuzzy rule set can be defined:

rule 1: if *age(demand*) is *young* and *age(resource*) is *young* then *suitability* is *true*

rule 2: if *age(demand*) is *young* and *age(resource*) is *middle aged* then *suitability* is *false*

rule 3: if *age(demand*) is *young* and *age(resource*) is *old* then *suitability* is *false*

rule 4: if *age(demand*) is *middle aged* and *age(resource*) is *young* then *suitability* is *false*

rule 5: if *age(demand*) is *middle aged* and *age(resource*) is *middle aged* then *suitability* is *true*

rule 6: if *age(demand*) is *middle aged* and *age(resource*) is *old* then *suitability* is *false*

rule 7: if *age(demand*) is *old* and *age(resource*) is *young* then *suitability* is *false*

rule 8: if *age(demand*) is *old* and *age(resource*) is *middle aged* then *suitability* is *true*

rule 9: if *age(demand*) is *old* and *age(resource*) is *old* then *suitability* is *true*

rule 10: if *pc(demand*) is *bad* and *pc(resource*) is *bad* then *suitability* is *true*

rule 11: if *pc(demand*) is *bad* and *pc(resource*) is *average* then *suitability* is *false*

rule 12: if *pc(demand*) is *bad* and *pc(resource*) is *good* then *suitability* is *false*

rule 13: if *pc(demand*) is *average* and *pc(resource*) is *bad* then *suitability* is *false*

rule 14: if *pc(demand*) is *average* and *pc(resource*) is *average* then *suitability* is *true*

rule 15: if *pc(demand*) is *average* and *pc(resource*) is *good* then *suitability* is *false*

rule 16: if *pc(demand*) is *good* and *pc(resource*) is *bad* then *suitability* is *false*

rule 17: if *pc(demand*) is *good* and *pc(resource*) is *average* then *suitability* is *true*

rule 18: if *pc(demand*) is *good* and *pc(resource*) is *good* then *suitability* is *true*

Note that *pc* stands for *physical condition* and that operator *(demand)* presents the value of fuzzy variable belonging to demand and operator *(resource)* value of fuzzy variable belonging to resource.

4) Evaluating the suitabilities

Suitabilities of each resource to each demand are calculated in the next step of the algorithm with the evaluation of fuzzy rule set regarding the parameters of each demand and each resource. Before the fuzzy rule set is evaluated it is verified if certain resource fulfills all crisp conditions, i.e. *education* and *driving license*, of a certain demand. If crisp conditions are fulfilled the value of fuzzy *suitability* is calculated regarding the defined fuzzy rules. Crisp conditions have to be fulfilled in the following way:

- Education of resource must be at least as high as the education required in the demand.
- Driving license required in the demand must be included in the driving license of the resource.

If crisp conditions are not fulfilled *suitability* of the resource to the demand is set to 0 (resource is not suitable for chosen demand). Otherwise fuzzy rules are evaluated in order to calculate the suitabilities. Resource parameters that are dealt fuzzily have to be fuzzified before the evaluation of the rules. In order to use calculated suitabilities in further steps of the algorithm fuzzy *suitability* must be deffuzified. Fuzzy values are converted to their crisp equivalents with the aid of *Center of Gravity* (COG) method (see Equation 15).

5) *Suitability Matrix Construction*

Suitability matrix can be constructed with the evaluation of suitabilities of each resource r_i to each demand d_j. Fuzzy variable *suitability* is defined in such way, that crisp suitability is always in the interval [0,1], where higher values can be interpreted as higher suitabilities. The suitability matrix for our example is presented in Table 5.

	d_1	d_2	d_3
r_1	*0.848*	*0*	*0*
r_2	*0.192*	*1*	*0.674*
r_3	*0.887*	*1*	*0.113*

Table 5 Suitability matrix of our example.

6) *Cost Matrix Construction*

In order to use the calculated data as inputs to Hungarian algorithm, cost matrix has to be calculated using the Equation 16, where *M* equals 1 in our example. The cost matrix of our example is presented in Table 6.

	d_1	d_2	d_3
r_1	*0.152*	*1*	*1*
r_2	*0.808*	*0*	*0.326*
r_3	*0.113*	*0*	*0.887*

Table 6 Cost matrix of our example.

Higher suitabilities give lower costs and vice versa.

7) *Finding an optimal solution*

Using the algorithm described in section *Hungarian algorithm* optimal solution can be found regarding the cost matrix presented in Table 6. Optimal assignment in the meaning of optimal costs (see Equation 3) and consequently in the meaning of optimal suitabilities is as follows:

- resource r_1 is assigned to demand d_1,
- resource r_3 is assigned to demand d_2,
- resource r_2 is assigned to demand d_3.

Optimal assignment of our example marked in the cost matrix is presented in Table 7.

	d_1	d_2	d_3
r_1	*0.152*	*1*	*1*
r_2	*0.808*	*0*	*0.326*
r_3	*0.113*	*0*	*0.887*

Table 6 Optimal assignment of our example marked in the cost matrix.

Optimal assignment can also be presented by the following 3-tuple (see Equation 1)

$$A = (1,3,2). \tag{18}$$

Cost of the assignment can be calculated according to Equation 3. Cost of the assignment therefore equals $c(A_k) = 0.326$. According to Equation 17 suitability of the assignment can be calculated. Assignment suitability therefore equals $s(A_k) = 0.674$. One can calculate the costs of other assignments (respectively suitabilities) and reach the conclusion that no assignment with lower cost (respectively higher suitability) exists. Solution found is therefore optimal.

IV. Experiments

The behavior of our method was further tested in details 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.

V. 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. Extended algorithm was also demonstrated on an example of vaguely defined assignment problem where resources were presented with people and demands with tasks that have to be fulfilled. Algorithm was further 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

[1] S. Martello R., *Assignment Problems*.: Society for Industrial and Applied Mathematics, 2009.
[2] J. Aguilar, R. Sumoza, "A Multiagent Grid metascheduler, "*WSEAS Transactions on Computers*, vol. 8, pp. 1388-1397, 2009.
[3] C. M. Ribeiro, "Problem of Assignment Cells to Switches in a Cellular Mobile Network via Beam Search Method," *Wseas Transactions on Communications*, vol. 9, pp. 11-21, 2010.
[4] H. W. Kuhn, "The Hungarian Method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83-97, 1955.
[5] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, pp. 32-38, 1957.
[6] H. W. Kuhn, Springer Berlin Heidelberg, 2010, ch. The Hungarian Method for the Assignment Problem, pp. 29-47.
[7] L. A. Zadeh, "Fuzzy Sets," Journal of Information and Control, vol. 8, pp. 338-353, 1965.
[8] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*.: Academic Press, 1980.
[9] T. J. Ross, *Fuzzy Logic with Engineering Applications*, 2nd ed.: Wiley, 2004.
[10] M. Hanss, *Applied Fuzzy Arithmetic: An Introduction with Engineering Applications*.: Springer, 2005.
[11] G. O. Tirian, C. B. Pinca, D. Cristea, M. Topor, "Applications of Fuzzy Logic in Continuous Casting," *WSEAS Transactions on Systems and Control*, vol. 5, pp. 133-142, 2010.
[12] M. B. Wright, "Speeding up the Hungarian al gorithm," *Computers and Operations Research*, vol. 17, pp. 95-96, 1990.
[13] C. T. Chi, "Self-Equilibrium Control on a Dynamic Bicycle Ride," *WSEAS Transactions on Systems and Control*, vol. 2, pp. 527-536, 2007.

Miha Moškon is a doctoral student and a teaching assistant at the University of Ljubljana, Faculty of Computer and Information science. He received his BSc degree in Computer Science from the Faculty of Computer and Information Science, University of Ljubljana in 2007. His research interests include fuzzy logic and unconventional computing.