JOURNAL	International Academy Journal Web of Scholar
p-ISSN	2518-167X
e-ISSN	2518-1688
PUBLISHER	RS Global Sp. z O.O., Poland

ARTICLE TITLE	JUMPING FROG METHOD FOR OPTIMAL CLASSIFICATIONS
AUTHOR(S)	Kozin I. V., Selyutin E. K., Polyuga S. I.
ARTICLE INFO	Kozin I. V., Selyutin E. K., Polyuga S. I. (2021) Jumping Frog Method for Optimal Classifications. International Academy Journal Web of Scholar. 2(52). doi: 10.31435/rsglobal_wos/30042021/7519
DOI	https://doi.org/10.31435/rsglobal_wos/30042021/7519
RECEIVED	22 January 2021
ACCEPTED	16 March 2021
PUBLISHED	22 March 2021
LICENSE	This work is licensed under a **Creative Commons Attribution 4.0 International License.**

JUMPING FROG METHOD FOR OPTIMAL CLASSIFICATIONS

Kozin I. V.,
Doctor of Physical and Mathematical Sciences, Professor, Zaporizhzhia National University, Ukraine
Selyutin E. K.,
postgraduate student, Zaporizhzhia National University, Ukraine
Polyuga S. I.,
Ph.D., Zaporizhzhia Regional Institute of Postgraduate Pedagogical Education, Ukraine

DOI: https://doi.org/10.31435/rsglobal_wos/30042021/7519

ARTICLE INFO

Received: 22 January 2021
Accepted: 16 March 2021
Published: 22 March 2021

KEYWORDS

optimal classification problem, classification, evolutionary algorithm, ant colony algorithm, mixed jumping frog algorithm.

ABSTRACT

In the article the problem of finding optimal classifications on a finite set is investigated. It is shown that the problem of finding an optimal classification is generated by a tolerance relation on a finite set. It is also reduced to an optimization problem on a set of permutations. It is proposed a modification of the mixed jumping frogs to find suboptimal solutions of the problem of classification.

Citation: Kozin I. V., Selyutin E. K., Polyuga S. I. (2021) Jumping Frog Method for Optimal Classifications. *International Academy Journal Web of Scholar.* 2(52). doi: 10.31435/rsglobal_wos/30042021/7519

Introduction. Classification is a powerful scientific method. The classification problem arises in almost all areas of knowledge when analyzing research results, when designing and forecasting, when assessing and making decisions. Often having a simple formulation, the classification problem turns out to be quite complex and ambiguous. Moreover, sometimes when trying to classify, interesting paradoxes arise associated with the unification of fundamentally different objects into one class.

The solution to the classification problem, as a rule, includes a significant proportion of subjectivity, individual assessments, fuzzy, informal conclusions. Often the priorities of the decision maker (DM) influence the solution of this problem. This leads to the construction of fundamentally different classifications based on the same primary information. Especially often this situation arises in those areas of knowledge in which it is impossible to use numerical estimates in the classification of objects and phenomena, due to which there is a need for fuzzy assessments, the use of the concepts “similar”.

Formulation of the problem. The aim of this work is to construct metaheuristics for finding a suboptimal classification defined by a tolerance relation on a finite set. This approach allows one to construct partitions close to optimal sets in accordance with the relation of “proximity” of elements. Moreover, this relationship of proximity is not transitive. The proposed algorithms can find wide application in applied problems related to the problem of object classification by a number of attributes. Such problems often arise in the economic, social and technical sciences.

Solution method and analysis of the results. From the point of view of mathematics, the classification problem can be considered from different positions. The main one is the set-theoretic approach when constructing a classification. However, in practice, it turns out that this approach is good only post factum, that is, for clarifying and formally describing an already constructed classification.

The most widespread up to now are statistical classification models, which allow grouping objects according to the results of statistical data analysis [1, 2, 3]. Metric algorithms use the formalization of the concept of similarity between objects and the hypothesis of compactness [2, 3, 4]. There is another principle: the so-called logical classification algorithms. This approach is based on the principle of inductive inference of logical laws or induction of rules [5, 6, 7]. Classification models are becoming more widespread is they are based on the tools of the fuzzy sets theory. A comparatively new direction is classification models based on integral mathematics. An interesting direction is the use of artificial intelligence methods for solving classification problems. An overview of existing recognition methods is given in the monograph [6].

Statement of the classification problem on a finite set

By a partition of a finite set *X* we mean a set of its nonempty subsets $X_1, X_2, ..., X_n$ such that:

1) $\bigcup_{i=1}^{n} X_i = X$;

2) $\forall i, j,\ i \neq j \quad X_i \cap X_j = \varnothing, \quad i, j = 1, 2, ..., n$.

The classification problem on a finite set consists in finding a partition that has some given properties. A set partition defines the canonical equivalence relation associated with this partition. Namely: two elements are considered equivalent if they belong to the same split element. On the other hand, it is easy to show that any equivalence relation on a finite set determines its partition into classes of elements equivalent to each other.

Recall that an equivalence relation on a set *X* is a binary relation $" \sim "$ with the following properties:

1) reflexivity: $\forall x \in X \quad x \sim x$;

2) symmetrically $\forall x, y \in X \quad x \sim y \Rightarrow y \sim x$;

3) transitive: $\forall x, y, z \in X \quad x \sim y, y \sim z \Rightarrow x \sim z$.

Let us present a simple algorithm [8], which allows for a given equivalence relation to construct the corresponding partition of the set *X* into classes of equivalent elements.

Step 0. An arbitrary ordering (numbering) of elements of the set *X*: is chosen $x_1, x_2, ..., x_N \in X$. Here is $N = |X|$. A set of representatives of equivalence classes is determined, which is empty at the initial stage of the algorithm. The set of equivalence classes is also empty.

….

Step *i*. The next element *x of the* ordered sequence of elements of the set *X* is selected and sequentially compared with the set of representatives of already defined equivalence classes. If this element is equivalent to a representative of the class X_k, then it is placed in the class X_k. If it is not equivalent to any of the elements of the set of representatives of the classes, then the element is entered into the set of representatives and defines a new class of equivalence.

The algorithm ends when all elements have been viewed and categorized. The result of the algorithm is a set of representatives of different classes and a set of classes of equivalent elements.

It follows from the transitivity of the equivalence relation that the set of classes obtained as a result of the operation of the algorithm does not depend on the initial ordering of the elements of the set *X* (Step 0). Another ordering can only change the sequence of equivalence classes and the set of representatives. The above algorithm for finding equivalence classes and a set of representatives will be called linear.

Let us note one feature of the linear algorithm. It can be applied not only to an equivalence relation, but also to any binary relation. However, if the relation is not transitive, then the result of the algorithm will already significantly depend on the choice of the initial ordering of the elements.

The relation of tolerance and classification based on the concept of proximity of elements.

Most of the existing classifications in applied sciences are not built on the basis of the equivalence relation, but on the basis of another binary relation - the tolerance relation. Tolerance relation is a reflexive and symmetric relation "$\approx$" on the set *X*, that is, a relation that is determined by the following properties:

1. $\forall x \in X \quad x \approx x$;

2. $\forall x, y \in X \quad x \approx y \Rightarrow y \approx x$.

A typical example of such a relationship is the relation of approximate equality on a set of numbers. In practice, the attitude of tolerance appears in the form of a relationship between objects, which is described by the words “similar”, “close”.

If the tolerance relation “$\approx$” is defined on a finite set *X*, then we can apply a linear algorithm for class allocation and obtain a classification on this set. However, in contrast to the classifications that are based on equivalence relations, classification, constructed on the basis of tolerance relationship, depends essentially on the choice of the initial ordering of the elements of *X*. Different ways of ordering elements can lead to fundamentally different classifications.

Optimality criterion for classifications.

There are many approaches to determining the optimal classification. Informally, a classification is optimal if the elements within the classes are “close enough” to each other, and the classes themselves are “far enough” from each other. Let's consider one of these approaches.

A closeness measure on a finite set *X is a* function $p: X \times X \to R_+$ with the following properties:

1) $\forall x, y \in X \quad p(x,y) \geq 0$ moreover $p(x,y) = 0 \Leftrightarrow x = y$;

2) $\forall x, y \in X \quad p(x,y) = p(y,x)$.

In particular, the distance between points in metric space can serve as a measure of proximity.

Let a positive number be given $\varepsilon > 0$. We will say that the elements $x, y \in X$ are in proximity (close to each other), if $p(x,y) \leq \varepsilon$. This ratio is the ratio of tolerance and, as mentioned above, gives rise to many different classifications, which are defined by the selected ordering on the set *X*. We will call such classifications -classifications. The linear partitioning algorithm changes slightly. Namely: at step *i* there is a class (among the constructed ones), the representative of which is closest to the analyzed element. If the measure of proximity between this representative and the element in question is less than or equal to the value, then the element is added to the class. Otherwise, the element in question becomes a representative of the new class.

The distance between two non-empty disjoint subsets is $A, B \subseteq X$ defined as a function $p(A,B) = \min\limits_{x \in A, y \in B} p(x,y)$.

Let a linear order $"\prec"$ of elements on the set *X* be given, determined by some permutation $s \in S_n$. Let us denote the equivalence classes $X_1, X_2, ..., X_n$ for the ε-classification generated by this order. As a criterion for optimality of classification, we will choose a function $F(s) = \min\limits_{i,j,i \neq j} p(X_i, X_j)$. Then the condition for the optimality of the ε-classification will be the condition

$$F(s) \xrightarrow[s \in S_n]{} \max$$

Such a task is computationally complex [9]. However, the algorithm itself for finding a partition by a given order relation has a polynomial complexity.

Thus, the optimal classification of the search problem is reduced to the problem of finding *an optimal permutation* of the elements of *the X*. This allows us to propose a number of metaheuristics for finding suboptimal solutions to the classification problem [10]. Consider two such metaheuristics, the effectiveness of which has been confirmed by a large number of applications.

Permutation evolutionary algorithm.

The standard scheme of the evolutionary permutation algorithm is used. Let us briefly describe the principle of operation of such an algorithm [10]. The set of all permutations of *n* elements is chosen as the base set of solutions S_n. At the initial step, a set of solutions is constructed using the initial population operator $Y_0 \subseteq S_n$. At each next step, it is assumed that a certain set of permutations is given the current population. At the first step, this is a set $Y = Y_0$. For each of the elements of the set *Y*, the value of the selection criterion is calculated, which in this case is a covering mapping of the original problem. Then, using the selection operator in the current population *Y*, a set of pairs $U = (u_1, u_2, ..., u_n)$ and

$V=(v_1,v_2,...,v_n)$ is selected for crossover operation. A crossover operator $Cross(U,V)$ is applied to each pair, and then a mutation operator is applied to the result. Permutation – descendant is constructed as follows: sequences *U* and *V are* scanned from the beginning. At the *k* -th step, the smallest of the first elements of the sequences is selected and added to the new permutation – descendant. This element is then removed from the two parent sequences. For instance,

$$Cross((2,4,7,6,1,3,5,8),(3,8,1,5,4,2,6,7)) = (2,3,4,7,6,1,3,5,8).$$

The mutation operator *M* performs a random transposition (replacement of two elements) in a permutation with a given probability $\alpha \in (0,1)$.

In this way many elements are found the descendants of $\tilde{Y}$. The evolution operator is applied to the intermediate population $Y \cup \tilde{Y}$, which is the union of the current population and a set of descendants, which selects a new current population on this set. The evolution process is repeated until the condition for stopping the evolutionary algorithm is satisfied. The solution of the original problem is restored from the found permutation.

Mixed jumping frogs method.

The algorithm of the method of mixed jumping frogs is simple to understand and implement, has a small number of parameters, and has been successfully used to solve combinatorial and continuous optimization problems [5,6].

The essence of the jumping frog algorithm for finding the optimal permutation is reduced to the following sequence of steps.

Step 1. Initialize the initial frog population as a set of points in the permutation space with Kendall's metric S_n.

Step 2. Calculate the value of the optimality criterion for each permutation from the initial population.

Step 3. Arrange the solutions in descending order of the optimality criterion value.

Step 4. Divide virtual frogs (solutions) into memplex blocks in such a way that the first virtual frog in the sorted list falls into the first memplex, the second is entered into the second memplex, etc.

Step 5. Find the best s_{k1} and worst s_{k2} solution in each memplex $k \in \{1,2,...,K\}$.

Step 6. Try to improve the position of the worst virtual frog by randomly moving it in the direction of the best frog $s = Cross(s_{k2}, s_{k1})$.

Step 7. If the previous operation does not improve the solution, then try to improve the position of the worst virtual frog by moving it towards the globally better frog $s = Cross(s_{k2}, s_{11})$.

Step 8. If the last operation does not improve the position of the virtual frog, then instead of it, randomly create a new frog in the search area – a permutation.

Step 9. Combine virtual frogs of all memplexes into one group.

Step 10. If the conditions for the completion of the algorithm are not met, then go to Step 3.

Step 11. The last globally best virtual frog corresponds to a suboptimal problem solution.

Let us now describe this algorithm formally, taking into account the parameters.

The method parameters are as follows:

1) the number of classes of frogs $Q \ (Q \geq 2)$;

2) the number of elements *r* in each class (it is assumed that the sizes of the classes are the same and $r \geq 2$);

3) the maximum number of steps *K of the* algorithm;

4) the number *D of the* best frogs in the class, and $0 < D < r$.

In accordance with the specified parameters, the size *N* of the frog population (the set of feasible solutions) is determined by the formula $N = Qr$. In the initial step of the algorithm creates the initial population of frogs by generating random permutations $s^j = (i_{j1}, i_{j2},..,i_{jn})$, $j = 1,2,...,N$.

The best permutation of the vertices in terms of the goal function is selected, which defines the permutation $s^* = (i_1, i_2,...,i_n)$, and the value of the objective function is calculated $F(x^*)$ on this permutation:

Step *k* ($1 \le k \le K$). The set is ordered $P^{(k-1)}$ by the value of the objective function, that is $F(s^k) \ge F(s^{k-1}), \quad k = 2,3,...,N-1$. The population $P^{(k-1)}$ is divided into *Q* classes of the same cardinality *r*

$$P_q^{(k-1)} = \{s^{(qi)} \mid s^{(qi)} = x^j, \ j = q + (i-1)Q, \ i = 1,2,..,r, \ q = 1,2,...,q\}.$$

The best solution $s^* = s^1$ is determined by the value of the objective function for the entire population. In each class $P_q^{(k-1)}$, the "best" $s^{(q1)}$ and "worst" $s^{(qr)}$ are determined by the value of the objective function of the solution. In each class $P_q^{(k-1)}$, the positions (sequences of traversing the vertices of the graph) of frogs change with numbers from $D+1$ to *r*. For each value of the index $i \in \{D+1, 2, ..., r\}$ a new position of the *i*- th frog (the sequence of traversing the vertices) in the class with number *q* is determined according to the following rule: a random permutation s^c is calculated from the interval between the permutations $s^{(q1)}$ and $s^{(qr)}$ in the Kendall metric.

The permutation on the segment between $s^{(q1)}$ and $s^{(qr)}$ is built according to the rule: sequences $s^{(q1)}$ and $s^{(qr)}$ are viewed from left to right. At the next step, the smallest of the first elements of the sequences is selected and added to the new permutation. Then this element is removed from the permutations $s^{(q1)}$ and $s^{(qr)}$. For example, applying this operation to the permutations (2, 4, 7, 6, 1, 3, 5, 8) and (5, 8, 1, 3, 4, 2, 6, 7) gives the permutation (2, 4, 5, 7, 6, 1, 3, 8).

If $F(s^c) < F(s^{(qi)})$, then we assume $s^{(qi)} = s^c$. If $F(s^c) \ge F(s^{(qi)})$, then a random permutation s^c is chosen in the segment between $s^{(qr)}$ and s^*. If $F(s^c) < F(s^{(qi)})$ then we assume $s^{(qi)} = s^c$. Otherwise, we choose a randomly generated permutation $s^{(qi)}$.

We assume $P^{(k)} = \bigcup_{q=1}^{Q} P_q^{(k-1)}$ and go to the next step of the algorithm.

The algorithm ends when the specified number of steps has been completed. The current permutation s^* determined at the last step is taken as the optimal solution to the problem.

Note that description is, the above algorithm s Resch was the problem of finding the optimal permutations of *n* elements in the set of all permutations with the objective function $F(s)$ which is defined on the set of permutations. In this case, the specific type of the objective function does not matter. Therefore, the above algorithm can be used to find suboptimal solutions to optimization problems on a set of permutations with arbitrary objective functions.

Numerical experiment. The weights of the graph edges were chosen randomly in the range [1,100]. These weights were considered as a measure of proximity for the respective vertices. The set of vertices of the graph was considered as a set of elements to be classified. A positive number was chosen randomly in the interval [0,1]. Linear orders (permutations) on the set of vertices were not adjusted using the Fisher-Yates shuffle algorithm [17].

The problems were solved using a local search algorithm, a random search method, evolutionary algorithms, and the method of mixed jumping frogs.

The comparison of algorithms was carried out in the following directions:

Record is the number of problems in a series where the algorithm turned out to be the best among the tested. Bord rating is the sum of the number of points scored on each problem in the series. For the first place in comparison, 5 points were assigned, for the second 4, for the third 3.

The results of the algorithm comparison are presented on Table 1.

Table 1. Results of applying different approaches.

Series (number of vertices)	Number of tasks	Algorithm search loc.		Random Search		Evolutionary algorithm		Jumping frog method	
		Record	Rating	Record	Rating	Record	Rating	Record	Rating
A 50	100	44	286	88	468	100	500	100	500
B 100	100	ten	221	0	348	100	500	100	500
H 500	100	0	212	0	280	94	394	100	500
D 1000	100	0	118	0	201	92	392	100	500

Conclusions. In this article, a method for finding optimal β-classifications based on two well-known metaheuristics was considered. A numerical experiment showed good results of the proposed algorithms in comparison with local and random search. This approach can be transferred practically without changes to other types of classifications, which are based on the concept of proximity of elements.

In addition to the metaheuristics proposed in the work, any other metaheuristics applicable to optimization problems on fragmentary structures can be considered in a similar way [10].

REFERENCES

1. Ayvazyan S.A., Buchstaber V.M., Enyukov I.S. and Meshalkin L.D. (1989) Applied Statistics: Classification and Dimension Reduction. Reference edition.
2. Durand B. (1977) Cluster analysis.
3. Kim. J., Myuler C.R. and Klekka U.R. (1989) Factor, discriminant and cluster analysis.
4. Weintsweig M.N. (1973) Algorithm of teaching pattern recognition “bark”.
5. Dyulicheva Y.Y. (2002) Decision tree reduction strategy (overview) // Tavrichesky Bulletin of Informatics and Mathematics.
6. Zhuravlev Y., Ryazanov V.V. and Semko O.V. (2006) Software system. Practical applications.
7. Lbov G.S. (1981) Methods of processing different types of experimental data.
8. Perepelitsa V., Kozin I. and Tereshcenko E. (2012) Classification tasks and knowledge formation.
9. Gary M. (1982) Computing machines and intractable problems.
10. Kozin I.V., Maksyshko N.K., Perepelitsa V.A. (2017) Fragmentary Structures in Discrete Optimization Problems / Cybernetics and Systems Analysis. - Volume 53, Issue 6. - P. 931-936. DOI: https://doi.org/10.1007/s10559-017-9995-6.
11. Kozin I.V. (2015) Evolutionary algorithm for optimal classification / Artificial Intelligence. - No. 3-4 (69-70). - P. 98–104.
12. Dorigo M. (1992) Optimization, Learning, and Natural Algorithms.
13. Shtovba S.D. (2005) Ant algorithms: theory and application. Programming.
14. Karpenko A.P. (2014) Modern search engine optimization algorithms. Algorithms inspired by nature: textbook for universities.
15. Narimani M.R. (2011) A New Modified Shuffle Frog Leaping Algorithm for NonSmooth Economic Dispath / World Applied Sciences Journal. 2011. - P. 803–814.
16. Beasley J.E. (1990) OR-Library: distributing test problems by electronic mail / Journal of the Operational Research Society. - No. 41. - P. 1069-1072.
17. Fisher R.A. and Yates F. (1948) Statistical tables for biological, agricultural and medical research.