Learning Network Flow Based on Rough Set Flow Graphs and ACO Clustering in Distributed Cognitive Environments

Arkadiusz Lewicki[1]
Assiatant Profesoor
[1]University of Information Technology and Management
Rzeszow, Poland
alewicki@wsiz.rzeszow.pl

Eugene Eberbach[2,1]
Senior Research Scientist
[2]Leaders in Machine Learning and Robotics Engineering
Seekonk, MA, USA
eeberbach@gmail.com

ABSTRACT

This paper presents the use of a modified collective behavior strategy of ant colonies to find approximate sets in the multi-objective optimization problem. The currently used methods search for non-dominated solutions, which takes place directly on the basis of definitions in the previously generated finite set of admissible ratings, searching in the space of goals by analyzing active constraints, solving optimization tasks in terms of all subsequent individual optimization criteria and adopting optimization criteria in order to form a substitute criterion of optimization in the form of a combination of linear criteria with appropriately selected weighting factors. However, these methods are ineffective in many cases. Therefore, the authors of the article propose a new approach based on the use of rough sets flow graphs to control the strategy of communicating artificial ants in distributed cognitive environments. The use of this approach allows to maximize the number of generated solutions and finding non-dominated solutions for the multiple objectives.

KEYWORDS

Cognitive systems services, Network Flow algorithms, Rough Sets flow graphs, ACO clustering

1 INTRODUCTION

There is a lot of confusion among current IT practitioners about what "cognition" is. Many use it loosely to promote their solutions and one of the aims of this workshop is to define more precisely so that we can let software engineering practices to evolve and incorporate cognitive practices in their methodology.

The most accepted definition of cognition is the ability to process information though perception (stimuli that we receive through our different senses), knowledge acquired through experience, and our subjective characteristics that allow us to integrate all of this information to evaluate and interpret our world. In other words, cognition is the ability that we have to assimilate and process the information that we receive from different sources (perception, experience, beliefs, ...) to convert them into knowledge. Cognition includes different cognitive processes, like learning, attention, memory, language, reasoning, decision making, etc., which form part of our intellectual development and experience.

In other words, **cognition** is "the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses". It encompasses processes such as knowledge, attention, memory and working memory, judgment and evaluation, reasoning and computation, problem solving and decision making, comprehension and production of language. Human cognition is conscious and unconscious, concrete or abstract, as well as intuitive (like knowledge of a language) and conceptual (like a model of a language). Cognitive processes use existing knowledge and generate new knowledge.

The semantic network of knowledge representation systems has been studied in various paradigms. More dynamic models of semantic networks have been created and tested with neural network experiments based on computational systems such as latent semantic analysis (LSA), Bayesian analysis, and multidimensional factor analysis. The semantics (meaning) of words is studied by all the disciplines of cognitive science.

Cognitive services in the form of vision, speech, language and knowledge capabilities are provided by multiple companies, for example IBM Watson Intelligence, Microsoft Azure, Amazon Alexa, and Google Cloud Platform.

We would like to steer the discussion to more on what is cognition with respect to current information technology practices and how can we improve our software systems to become more cognitive and sentient. Application of clustering to discern hidden data insights is just one small narrow path of applying cognition. Meta-processing and meta-algorithms seem to provide another component. Can we have a broader path to make our software systems "Cognitive" and what does that mean? Hopefully, a partial answer will be provided by this workshop.

In our paper, we concentrate on one aspect of cognitive processes, represented by clustering from Ant Colony Optimization (ACO). To do the clustering the ants use approximation measure derived from rough sets flow graphs and flow graphs have the direct roots in network flow algorithms searching for the optimal flow in the distributed network using some kind of flow performance measure. In our case, we consider multiple performance measures solving multi-objective optimization.

The paper is organized as follows. In section 2 network flow algorithms and their rich applications are discussed. We concentrate on one such application, i.e, the Project Selection Problem. In section 3, basic fundamentals of rough sets are presented, including information systems, lower and upper approximations, flow graphs, and certainty factors needed for ACO clusterization from section 4. Section 5 outlines the implementation, and section 6 contains the results and conclusions.

2 NETWORK FLOW ALGORITHMS AND THEIR APPLICATIONS

Network flow emerged as a cohesive subject through the work of Ford and Fulkerson [3,4]. The initial motivation for network flow problems comes from the issue of traffic in a network, however we will see that they have applications in a surprisingly diverse set of areas.

Network Flows algorithms, similar like dynamic programming, have very colorful roots [3]. A quite recently declassified U.S. Air Force report shows that in the original motivating application for minimum cuts - the network was a map of rail lines in the Soviet Union, and the goal was to disrupt transportation through it during time of cold war (i.e., solving the Network Connectivity Problem) [5].

Maximum Flow and Minimum Cut are two very rich algorithmic problems. They are cornerstone in combinatorial optimization and they provide beautiful mathematical duality.

The Maximum-Flow/Minimal Cut Problem is solved by the Ford-Fulkerson Algorithm [6,7,11].

It is interesting that many rich and diverse applications, can be reduced/solved basically by the same Ford-Fulkerson algorithm. Such applications include: *Data mining, Open-pit mining, Project Selection, Airline scheduling, Bipartite matching, Baseball elimination, Image segmentation, Disjoint paths, Network connectivity, Network reliability, Distributed computing, Egalitarian stable matching, Security of statistical data, Network intrusion detection, Multi-camera scene reconstruction, and many more applications.*

Flow network is an abstraction for material flowing through the edges of the directed graph with no parallel edges G = (V, E), with V set of vertices, and E set of edges. There are two distinguished nodes: s is the source and t is the sink. Each edge has the maximum capacity, and c(e) denotes capacity of edge e.

In particular, an s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$. We look for the min-cut partition separating s and t. It turns that min-cut is equivalent to max-flow in the network.

This exactly expresses the Ford-Fulkerson theorem [6,7]:

Theorem 2.1 (Ford-Fulkerson) The value of max flow is equal to the value of min cut.

The Ford-Fulkerson algorithm uses augmented paths and residual graphs in its construction, and is really quite simple - done is successive iterations of while loop (for more details the reader is advised to look at [2,6,7,11]}. What is not at all clear is whether its central loop terminates, and whether the flow returned is a maximum flow. The answers to both of these questions turn out to be fairly subtle.

Let's assume that n denote the number of vertices in G, and m denote the number of edges in G. Let C denote the sum of capacities of edges incident to s. Let's assume that all capacities are integers between 1 and C. Of course, flows cannot exceed capacities.

With such assumptions the Ford-Fulkerson terminates and runs in $O(nmC) = O(nm2^{\log C})$ time. As we see the generic Ford-Fulkerson algorithm (i.e., selecting randomly augmenting paths) is not polynomial as the function of n, m and logC.

Choosing good augmenting paths is crucial:

- Some choices (e.g., random like in a generic Ford-Fulkerson algorithm) lead to exponential algorithms;
- Clever choices (e.g,, Max-bottleneck capacity, Sufficiently large bottleneck/Capacity scaling running in $O(m^2 \log C)$ time, Fewest number of edges) lead to polynomial algorithms;
- If we drop the requirement that capacities are integers, i.e., they are irrational real numbers, then the algorithm is not guaranteed even to terminate!

Thus good selecting of augmenting paths and having integer capacities are crucial for successful implementation of the Ford-Fulkerson algorithm and its rich applications.

In the next parts of the paper, we will concentrate on one such application – the Project Selection Problem.

2.1 Project Selection Problem

Large (and small) companies are constantly faced with a balancing act between projects that can yield revenue, and the expenses needed for activities that support these projects. We consider projects with prerequisites. The special case of project selection is the Open-Pit Mining Problem with the goal to determine the most profitable set of blocks ("projects") to extract, subject to the precedence relation ("prerequisites").

Let's consider the set P of possible projects. Project v has an associated profit that can be positive or negative:

– some projects generate money, e.g., create interactive e-commerce interface, redesign web page

– others cost money, e.g., upgrade computers, get site license

Let' s define the set of prerequisites E. If $(v, w) \in E$, can't do project v and unless also do project w.

A subset of projects $A \subseteq P$ is *feasible* if the prerequisite of every project in A also belongs to A.

Definition 2.1 The Project Selection Problem is to select a feasible set of projects with maximum profit.

Project Selection Problem can be reduced to Min cut – Max flow Ford-Fulkerson algorithm [11].

Project Selection reduced to Min cut formulation:

Given a bipartite graph with sets of projects L giving (positive) profits and set R projects bringing costs (negative profits).

- Assign capacity ∞ to all prerequisite edges.
- Add edge (s, v) with capacity p_v if $p_v > 0$.
- Add edge (v, t) with capacity $-p_v$ if $p_v < 0$.

- For notational convenience, define $p_s = p_t = 0$.

2.2 Multi-Objective Optimization

Multi-objective optimization (also known as multi-objective programming, or vector optimization, or multicriteria optimization, multiattribute optimization or Pareto optimization) is an area of *multiple criteria decision making*, that is concerned with *mathematical optimization problems* involving more than one *objective function* to be optimized simultaneously.

A multi-objective optimization problem is an optimization problem that involves multiple objective functions. In mathematical terms, a multi-objective optimization problem can be formulated as

$$\min(f_1(x), f_2(x), \ldots, f_k(x))$$

where $x \in X$, the integer $k \geq 2$ is the number of objectives and the set X is the feasible set of decision vectors. The feasible set is typically defined by some constraint functions. It is possible either to use an aggregating function converting multi-objective optimization to single-objective optimization, or keeping the vector of objectives functions. It is an open research problem which of these approaches is better. Both have advantages and disadvantages [3,4]. In this paper, we consider the vector of objectives.

In many areas, optimal decisions must be made taking into account compromises between two or more conflicting goals. Therefore, problems with multi-objective optimization [5,10] are much more common than problems with single-criterion optimization. As one of many examples, we can mention the planning of transport services with consideration of transport costs, storage costs or costs related to delays in deliveries. Finding a solution for which all the criteria will be optimal is often not possible. If there is no possibility to find a better solution due to at least one criterion without deterioration of the other, then such a solution is called non-dominated (Pareto-optimal). Otherwise, the solution is dominated. These solutions are presented in Fig. 1.

The nadir (the worst) solution contains all the upper bound of each objective in the Pareto-optimal set. The utopian solution represents. The utopian solution is a vector whose components are slightly smaller than that of the ideal objective vector.
For the task of maximizing the set *k* of objective functions (1):

$$f(x) = (f_1(x), f_2(x), \ldots f_k(x)) \qquad (1)$$

we say that the solution *x* is dominated by solution y if for each objective function f_i , i=1,2,3,...,k:

$$f_i(x) \leq f_i(y) \qquad (2)$$

For the task of minimizing the set *k* of objective functions (1) we say that the solution *x* is dominated by solution y if for each objective function f_i we have (3):

$$f_i(x) \geq f_i(y) \qquad (3)$$

Otherwise, *x* is a non-dominated (Pareto-optimal) solution.

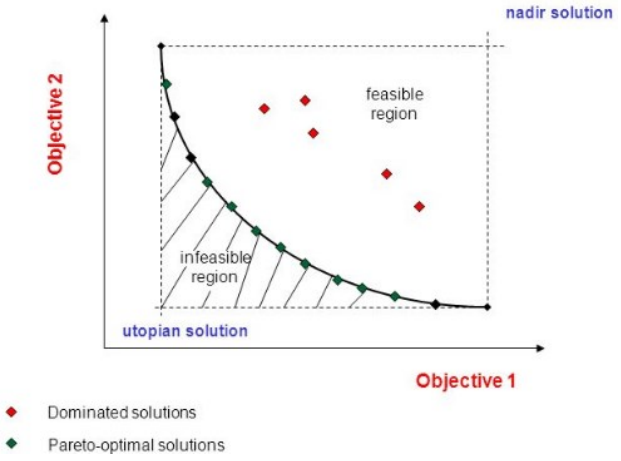

Fig.1. Pareto-optimal solutions and dominated solutions in multi-objective optimization problem.

Although there are deterministic methods (like for example: searching for non-dominated solutions, which takes place directly on the basis of definitions in the previously generated finite set of admissible ratings, searching in the space of goals by analyzing active constraints, solving optimization tasks in terms of all subsequent individual optimization criteria and adopting optimization criteria in order to form a substitute criterion of optimization in the form of a combination of linear criteria with appropriately selected weighting factors) that enable finding a set of optimal solutions in the Pareto sense, they are not suitable for most practical applications due to the very high computational cost. However, the search time for a collection of non-dominated solutions can be significantly reduced by using heuristic methods. These methods include genetic algorithms, simulated annealing or tabu-search. An example of such an approach can be also ACO algorithms.

The ACO algorithms can use one or more pheromone traces and use one or more matrices with heuristic information. Therefore, they can provide a set of non-dominated solutions. In this article, we propose the use ant colonies with approach based on the use of rough sets flow graphs to control the strategy communicating artificial agents.

3 ROUGH SETS AND FLOW GRAPHS

Rough sets, meaning approximation sets, were introduced by *Zdzislaw Pawlak* from Warsaw Univ. of Technology in 1982 as an extension of classical set theory to reason about imprecise or incomplete data [16]. Lofi Zadeh's fuzzy sets allow partial membership to deal with gradual changes or uncertainties. Rough sets, on the other hand, allow multiple membership to deal with indiscernibility.

The primary application domains of rough sets have been in symbolic approaches such as data and decision analysis, data mining, databases, expert systems, machine learning, and control. The assumption that objects can be "seen" only through the information available about them leads to the view that

knowledge has granular structure. Due to granularity of knowledge some objects characterized by the same information are indiscernible (similar) in view of the available information about them. The *indiscernibility relation* generated in this way is the mathematical basis of rough set theory.

Suppose we are given two finite, non-empty sets U and A, where U is the universe of objects, and A - a set of attributes. The pair (U,A) is called an information table or information system. With every attribute a in A we associate a set V_a, of its values, called the domain of a.

The indiscernibility relation I(B) on U, where B is any subset of A, forms *equivalence classes* of objects, is defined as follows

x I(B) y iff a(x) = a(y) for every a ∈ B

The equivalence class, determined by B, partitions U into equivalence classes. The block of partition U/B, containing x will be denoted by B(x).

B-lower and B-upper approximations of X: for every subset X of U we define its

- *B-lower approximation* $B_*(X) = \{x \text{ in } U : B(x) \subseteq X\}$
- *B-upper approximation* $B^*(X) = \{x \text{ in } U : B(x) \cap X \neq \emptyset\}$

The set $BN_B(X)=B^*(X)-B_*(X)$ is called the *B-boundary region* of X

- If the boundary region of X is the empty set, i.e., $BN_B(X)$ is empty, then the set X is *crisp (exact)* with respect to B;
- If $BN_B(X)$ is nonempty, the set X is *rough (inexact)* with respect to B.

Rough set *flow graphs* have been defined by Z. Pawlak [17] as a tool for reasoning from data.

A **flow graph** is a directed acyclic finite graph G = (V, E, f) , where V is a set of nodes, E is a set of edges, and f: B → [0, 1} is a flow function. Note that compared to network flow graphs, flows are normalized to [0, 1], and capacities are fixed to 1.

A set of solutions in the multi-objective optimization problem is a set of objects, decisions, candidates, variants or actions that are analyzed in the decision-making process. This set can be defined in advance or can be modified during the decision procedure. Therefore subsequent criteria will be separate subsets (for example, V1, V2, V3) – Fig. 2. For example, for the problem of project management, the set V1 will represent the goals, the set V2 will represent requirements and the set V3 will represent selection criteria. Note that multiple sources and sinks can be converted easily to a single source and single sink like we have in section 2..

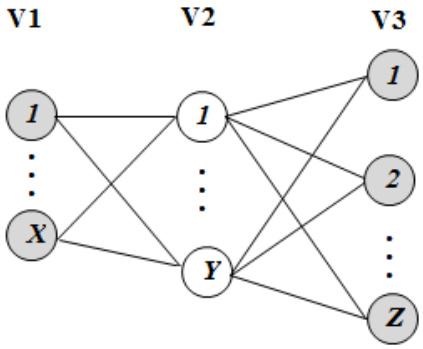

Fig.2. A N-partite graph representing the multi-objective optimization problem (for N=3).

The presented graph G(V1,V2,V3,E) will contain associations mapped by edges *E*, incidental to their vertices *V*. Therefore, the expected solution will be the association with maximum power and having maximum weight. In this case we can implement the reference method of Ford-Fulkerson [6,7]. This directed acyclic finite graph is also a flow graph G=(N,B,δ), where N represents a set of nodes, B is a set of directed branches, and δ is a flow function. It means that $B \subseteq NxN$, $\delta : B \rightarrow R+$, where R+ represents the set of non-negative reals. We can define the basic concepts of flow graphs for sets of all inputs (4) of a graph G and outputs (5) of a graph G:

$$I(G) = \{x \in N : I(x) = \Phi\} \quad (4)$$

$$O(G) = \{x \in N : 0(x) = \Phi\} \quad (5)$$

for (x, y) ⊆ B, where x is input of y and y is an output of x.

These concepts are an inflow (6) and outflow (7) for the graph G:

$$\delta_+(x) = \sum_{y \in I(x)} \delta(x, y) \quad (6)$$

$$\delta_-(x) = \sum_{y \in O(x)} \delta(x, y) \quad (7)$$

For the whole flow graph G we have respectively formula (8) and (9):

$$\delta_+(G) = \sum_{x \in I(G)} \delta_-(x) \quad (8)$$

$$\delta_-(G) = \sum_{x \in O(G)} \delta_+(x) \quad (9)$$

The throughflow of G is defined as (10):

$$\delta(G) = \delta_+(G) = \delta_-(G) \quad (10)$$

When considering the task of multi-criteria optimization and flow networks, we carry out the process of normalizing the undirected graph to the form of a directed graph (Fig. 3).

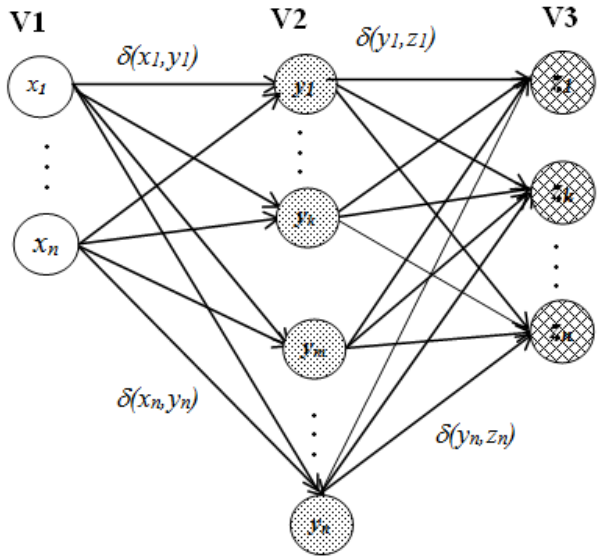

Fig.3. Flow graph with directed branches.

A normalized flow is defined by χ called the strength of (x,y). This indicator (11) represents the ratio of throughflow of the edge to the total flow:

$$\chi(x,y)=\frac{\delta(x,y)}{\delta(G)} \qquad (11)$$

Therefore, for flow graph with directed branches and a normalized inflow has a form (12):

$$\chi_{+}(x)=\frac{\delta_{+}(x)}{\delta(G)}=\sum_{y\in I(x)}\delta(x,y) \qquad (12)$$

and a normalized inflow has a form (13):

$$\chi_{-}(x)=\frac{\delta_{-}(x)}{\delta(G)}=\sum_{y\in O(x)}\delta(x,y) \qquad (13)$$

The throughflow of G is then defined as (14):

$$\chi(G)=\chi_{+}(G)=\chi_{-}(G) \qquad (14)$$

With each edge (x, y) of a flow graph, we may also associate its certainly (15):

$$cert(x,y)=\frac{\chi(x,y)}{\chi(x)} \qquad (15)$$

and the coverage factor (16):

$$\mathrm{cov}(x,y)=\frac{\chi(x,y)}{\chi(y)} \qquad (16)$$

Regarding to the fact that the solutions of the multi-criteria problem can be many, each of them can be equivalent or preferred to the other. Therefore, we can define the existence of two thresholds: e (equivalence) and p (preferences). Below the equivalence threshold, the variants will be equivalent, and above the preference threshold - one will be preferred to the other. The criterion function in this two-threshold model can be defined (17):

$$f_i(x)\geq f_i=\begin{cases} x|y \text{ if } |f_i(x)-f_i(y)|\leq e(f_i(y)) \\ x^{*}y \text{ if } f_i(y)+p(f_i(y))\geq f_i(x)>f_i(y)+q(f_i(y)) \\ x \text{ st } y \text{ if } f_i(x)>f_i(x)+q(f_i(y)) \end{cases} \qquad (17)$$

where $x^{*}y$ means that x is preferred to y, and x st y means a small advantage of one variant over the other, while x|y means any choice of x or y. The function proposed by the authors fits in the theory of rough sets.

4 ACO CLUSTERIZATION

Evolutionary Computing consists of four main subareas:

- Genetic Algorithms (GA)
- Evolution Strategies (ES)
- Evolutionary Programming (EP)
- Genetic Programming (GP)

Related Subareas include:

- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)
- Co-evolution
- Artificial Immune Systems (AIS)
- Evolutionary robotics
- Evolvable hardware
- Classifier Systems
- Artificial Evolutionary Neural Networks (EANN)
- Evolutionary multi-objective optimization
- Artificial Life (ALife)
- DNA-based computing

Ant Colony is a relatively new subarea of EC targeting multi-agent systems and with its own Ant Colony Optimization (ACO) evolutionary algorithm (see, e.g., [1,8]). In our paper, we will utilize ACO for clusterization. Ants will use traditionally traces of artificial pheromone for their navigation. The ACO algorithm created by Marco Dorigo [1,8] assumes that we have an available set of representations of artificial ants. For the graph (G) representation of the available space of solutions, each ant is placed in a randomly selected vertex point. From this vertex, the ant then selects the point (with the probability P) to which it travels. Probability P is a function (18) of the distance to the target and the size of the pheromone trace left on the edge E, connecting it with this point.

$$P_{xy}(t)=\begin{cases}\dfrac{[\tau_{xy}]^{\alpha}\cdot[\eta_{xy}]^{\beta}}{\sum_{h\in\Omega}[\tau_{xh}]^{\alpha}[\eta_{xh}]^{\beta}} & for \quad y\in\Omega \\ 0 \quad for \quad y\notin\Omega\end{cases} \qquad (18)$$

where τ_{xy} is the intensity of the pheromone trace located on the edge E (x, y), η_{xy} is the value of the local criterion function related to the visibility of y from the point x, α is a parameter that allows

controlling the relative importance of the intensity of the pheromone mark, β is a parameter that allows to control the relative importance of the visibility of the next point, and Ω is a collection of vertices that have not yet been visited by ant. each ant has memory (in the form of a list of recent solutions - TABU list). It is a collection of vertices already visited. The intensity of the pheromone in the real environment decreases after some time, until its total disappearance. Therefore, in the algorithm the amount of pheromone is reduced by a certain factor in each iteration. The mentioned dependencies show that in the ACO algorithm we have the ability to control the α and βparameters, which are the relative importance of the intensity and visibility of the pheromone trace.

An improved version of this approach [8] assumes the modification of the process of applying the pheromone footprint in order to minimize the cost of calculations by eliminating the exhibitor. It is called Ant Colony Optmization (ACO). Local updating of the trace is associated not only with its strengthening, but also with evaporation (19):

$$\tau_{xy}(t+1) = (1-\rho)\cdot\tau_{xy}(t) + \rho\cdot\tau_0 \qquad (19)$$

where ρ means the pheromone evaporation coefficient and represents values (0; 1], and τ_0 is the initial intensity of the pheromone at the edges of the graph G. After completing the full graph transition cycle for all ants, global pheromone evaporation occurs at all edges of the graph, and then the trace of attractiveness is updated. In the problem of clustering, the ant moves to the next point of the solution space, raises (20) or drops (21) the object. It is done according to the rules:

$$P_{pick} = 1 - \frac{1-T}{1+T} \qquad (20)$$

$$P_{drop} = \frac{1-T}{1+T} \qquad (21)$$

$$T = e^{-c.D} \qquad (22)$$

where c in formula (22) is a parameter for controlling convergence and D means density from the range (0,1).

The solution the optimization problem using the ACO algorithm has already been considered in various areas [12,13,14,15]. however, the problem of multi-objective optimization has not been solved in satisfactory manner yet.

5 EXPERIMENTAL STUDY

In the proposed modification of the algorithm we have set all places of the square grid G on which objects are scattered. Each object (o) on the grid G can be described by two coordinates: x and y. The neighborhood n(o) of o(x,y) is given (23):

$$n(o) = \{o'(x', y')\} \mid\in P : abs(x' - x) \le r \text{ and } abs(y' - y) \le r\} \qquad (23)$$

where r is a radius of perception of ants, and abs denotes the absolute value. For each object o_d designated to pick up from or drop at the place on the grid we have local function f, which is calculated as (24):

$$f(o_d) = \sum_{i=1}^{n-1} cer(n^i, n^{i+1}) \qquad (24)$$

where n^i is the node in the i-th subset of graph G, and n^{i+1} is the node in the (i+1) subset of graph G.

This function is based on the aggregation of certainty factors of branches belonging to the path in graph G. According to local function f, picking (25) and dropping (26) strategy can be expressed by threshold formulas:

$$p_{pick}(o) = \begin{cases} 1 & if \quad f(o) \le \delta_p \\ \dfrac{1}{(1-\delta_p{}^2)}(f(o)-1)^2 & otherwise \end{cases} \qquad (25)$$

$$p_{drop}(o) = \begin{cases} 1 & if \quad f(o) \ge \delta_d \\ \dfrac{1}{\delta_d{}^2} f(o)^2 & otherwise \end{cases} \qquad (26)$$

where δ_p is threshold for picking operation, δ_d is threshold for dropping operation. Both parameters are selected experimentally. The research carried out by the authors showed that the proposed modification of the ACS algorithm works well for well-structured data sets. For example, the UCI repository data collection can be that case (Fig. 4).

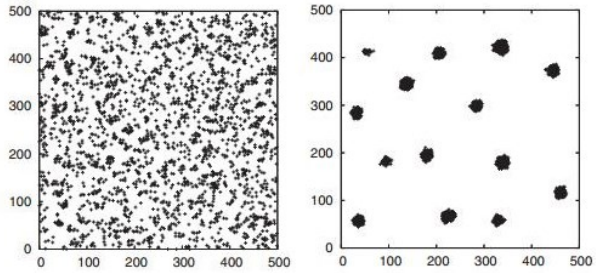

Fig.4. The ACS clustering using "user knowledge modeling set" from UCI Machine Learning Repository.

However, a strategy for dividing objects into subsets for unstructured sets should be developed in subsequent research works. Next study should also lead to determine a valid method for define coefficients thresholds for picking and dropping operations.

6 CONCLUSIONS AND FUTURE WORK

In summary, we have performed both a theoretical and experimental study in this paper. The implementation part is still in the process of the development and evaluation.

We considered many problems from various areas: optimal network flow algorithms, multi-objective optimization with set of

flows, rough sets providing an approximate measure for ant colony optimization clustering.

Regarding future work, it would be worth to check and compare implementation of classical network flow algorithms (e.g., for problem selection problems) with the ACO clustering approach regarding both the speed and precision. It would be also interesting to compare the Pareto optimality vector of objectives approach with an agglomerating function transferring a multi-objective optimization to a single-objective optimization.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for their comments.

REFERENCES

[1] E. Bonabeau., M. Dorigo. and G. Theraulaz, 1999. Swarm Intelligence: From Natural to Artificial Systems, Oxford Univ. Press.

[2] T.H. Cormen, Ch E. Leiserson, R.L. Rivest and C. Stein, 2004. Introduction to Algorithms. The MIT Press, 2nd ed.

[3] E. Eberbach, 2005. $-Calculus of Bounded Rational Agents: Flexible Optimization as Search under Bounded Resources in Interactive Systems, Fundamenta Informaticae, vol.69, no.1-2, pp.47-102.

[4] E. Eberbach, 2007. The $-Calculus Process Algebra for Problem Solving: A Paradigmatic Shift in Handling Hard Computational Problems, Theoretical Computer Science, vol.383, no.2-3, pp.200-243 (doi: dx.doi.org/10.1016/j.tcs.2007.04.012).

[5] T. Erfani, S. V. Utyuzhnikov, 2011. *Directed Search Domain: A Method for Even Generation of Pareto Frontier in Multiobjective Optimization*, Journal of Engineering Optimization.

[6] L.R. Ford, 1956. Network Flow Theory. *RAND Corporation Technical Report P-923.*

[7] L.R. Ford and D.R. Fulkerson, 1962. Flows in Networks. Princeton University Press.

[8] J. Handl, J. Knowles, M. Dorigo. 2006. *Ant-based clustering and topographic mapping*. Artificial Life 12(1), 35.

[9] G. T. Heineman, G. Pollice, S. Selkow. 2008. *Network Flow Algorithms. Algorithms in a Nutshell*, Oreilly Media.

[10] M. Kaisa. 1999. *Nonlinear Multiobjective Optimization.* Springer.

[11] J. Kleinberg and E. Tardos, 2006. Algorithm Design, Pearson/Addison-Wesley.

[12] A. Lewicki, K. Pancerz. R. Tadeusiewicz. 2013. *The Use of Strategies of Normalized Correlation in the Ant-Based Clustering Algorithm.* Lecture Notes in Computer Science, Springer-Verlag, Berlin.

[13] K. Pancerz, A. Lewicki, R. Tadeusiewicz, J. Gomula. 2011. *Ant Based Clustering of MMPI Data - An Experimental Study.* Lecture Notes in Artificial Intelligence, Vol. 6954.

[14] K. Pancerz, A. Lewicki, R. Tadeusiewicz, J. Warchoł. 2012. *Rough Set Flow Graphs and Ant Based Clustering in Classification of Distributed Periodic Biosignals.* International Workshop on Concurency, Specification and Programming, Vol 2.

[15] K. Pancerz, A. Lewicki, R. Tadeusiewicz, J. Warchoł. 2013. *Ant Based Clustering in Delta Episode Information Systems Based on Temporal Rough Set Flow Graphs.* Fundamenta Informaticae, Vol. 128 (1-2), IOS Press, Amsterdam 11, pp.341-356.

[16] Z. Pawlak, 1982. Rough Sets, Int. J. Computer and Information Sci. 11, pp.341-356.

[17] Z. Pawlak, 2005. Flow Graphs and Data Mining, in: Transactions on Rough Sets III (J. Peters, A. Skowron, eds), Springer-Verlag, pp.1-36.