ITHEA

International Journal INFORMATION THEORIES & APPLICATIONS

2019 Volume 26 Number 3

International Journal

INFORMATION THEORIES & APPLICATIONS
Volume 26 / 2019, Number 3

International Journal “INFORMATION THEORIES & APPLICATIONS”
Vol. 26, Number 3, 2019

Edited by the **Institute of Information Theories and Applications FOI ITHEA**, Bulgaria, in collaboration with: University of Telecommunications and Posts, Bulgaria,
V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
Universidad Politécnica de Madrid, Spain,
Hasselt University, Belgium, University of Perugia, Italy,
Institute for Informatics and Automation Problems, NAS of the Republic of Armenia
St. Petersburg Institute of Informatics, RAS, Russia,

Printed in Bulgaria
Publisher ITHEA®
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: office@ithea.org
Technical editor: **Ina Markova**

ISSN 1310-0513 (printed) **ISSN 1313-0463 (online)**

Particle Swarm Optimization in Linear Optimization Problems

Nuria Gómez Blas, Luis Fernando de Mingo López, Juan Castellanos Peñuela

Abstract: *Swarm colonies reproduce social habits. Working together in a group to reach a predefined goal is a social behaviour occuring in nature. Linear optimization problems have been approached by different techniques based on natural models. In particular, Particles Swarm optimization is a meta-heuristic search technique that has proven to be effective when dealing with complex optimization problems. This paper presents and develops a new method based on different penalties strategies to solve complex problems. It focuses on the constraints and the election of the parameters to ensure successful results.*

Keywords: *Partile Swarm Optimization, Non Linear Optimization, Artificial Intelligence, Swarm Intelligence.*

ACM Classification Keywords: *F.1.1 Theory of Computation - Models of Computation, I.2.6 Artificial Intelligence.*

MSC: *68Q32 Computational learning theory, 68T05 Learning and adaptive systems.*

Introduction

Particles Optimization is a technique based on heuristics, which emulates the intelligence of social and biological organizations. It is mainly used when dealing with optimization problems. Swarms of individuals are moving freely within an area searching for food; when an individual finds it somehow communicates with its neighbors, so that the swarm will follow the individual. A model of individuals called particles was originally designed to optimize functions in a binary search space. Then, this model was modified for problems of unrestricted searching spaces with continuous variables. Currently it is being tested for applications of nonlinear optimization problems with constraints and multitarget problems. Moreover, there are some models which have been developed to use hybrid techniques combined with genetic algorithms and fuzzy logic, neural networks and others.

This paper presents a standard problem of optimization of particles and some of its variants, emphasizing the problem of nonlinear constrained optimization. It also uses different methods of penalty. This work shows examples of PSO algorithms implemented in C ++.

Particles optimization model

Particle swarm optimization (PSO) is a heuristic optimization technique, originally developed by James Kennedy and Russell C. Eberhart in 1995. PSO is considered to fall within the branch of evolutionary computing, which is commonly referred as swarm intelligence. Evolutionary Computation is a collection of methods that simulate evolution by using computers, including genetic programming, evolution strategies, Swarm Optimization, Artificial Life and other related fields. Evolutionary computation and genetic algorithms share the technique for generating populations in a stochastic way. Furthermore, this technique is used to generate new generations too.

The algorithm used by PSO is inspired by the collective work of swarms, specifically in the social organization of bird flocks and fish schools. These social organizations take on a collective behavior that comes from communication and cooperation among its members. In the algorithm, individuals of the population are treated either as particles or searching points, which have a position and velocity (the velocity vector indicates where it is headed). Then they move into an area of dimension n. The particle that gets the best assessment is the *leader*, in the sense that any particle in the swarm follows it. However any individual could change their orientation. In that case the leader passes leadership to the one that changed the orientation. The *leader* can either be global leader in the whole swarm or local in a part of it.

The emulation of these organized societies has been successful in discrete and continuous optimization problems. Initially, Kennedy and Eberhart developed a standard algorithm, which has undergone several improvements. This is a fairly simple algorithm and widely used today.

Particles move along the search space by using a combination of the best individual solution found by the particle and the best found by any of the neighboring particles. Performance of every particle is monitored.

The three main operations Kennedy and Eberhart [9] algorithms use are: To assess, to compare and to imitate. Evaluation is one of the most important features of some living organisms; they evaluate, learn and evolve. Besides, organisms analyze their neighbors and imitate only those who have better performance. In general, these three operations can be applied to simple social beings and to computer programs. This enable them for solving complex problems.

Considering that partciles reside in a space of dimension d , each particle i is related to three vectors: position: $x^((i)) = (x_1, x_2 \dots, x_d)$, the best position of its history: $p^((i)) = (p_1, p_2 \dots, p_d)$ and speed $v^((i)) = (v_1, v_2 \dots, v_d)$. Initially values related to particles are generated randomly, then these particles move around a search space by using a system of equations that is updated for every iteration. The intention is to find the best solution. Each particle moves to the more succesful neighbor, influencing the other particles. The algorithm updates the swarm at each step. It also changes the position and velocity of each particle and it applies the following rules:

$$v_d^{(i)} = v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{1}$$

$$x_d^{(i)} = x_d^{(i)} + v_d^{(i)} \tag{2}$$

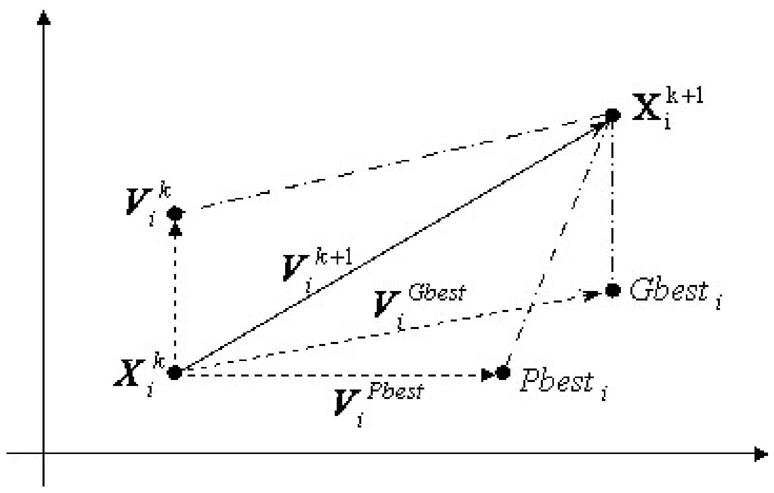


Figure 1: Original models of particles relations.

where $v_d^{(i)}$ is the d-th component of the velocity of particle i, $x_d^{(i)}$, the d-th component of vector position of the particle i, c_j, $(j = 1, 2)$ is a constant value (obtained from experimental results) and ϵ_1 and ϵ_2 are independent random numbers uniformly distributed in $[0.1]$. They are generated for every update, p_d is the d-th component of the improved performance of the particle in its history, and g_d is the d-th component of the particle with the best position found between neighboring particles, throughout its history. This neighborhood is defined according to the topology of the particle system. The factor c_1 is known as the factor of personal or cognitive learning and the factor c_2 as the social learning factor. Both factors have much influence on the rate of convergence of the optimization process.

Mathematical models developed for PSO vary according to how the particles interact with their neighbors, this is known as the topology of the system, which can be understood as the way the swarm of particles organizes itself. Early models of PSO used a Euclidean neighborhood. However the number of operations were high; in order to reduce the number of operations, a new mathematical model was developed. In this mathematical model neighborhoods were not related to the location of the particle; these are called local neighborhoods or *lbest models*. They can be also global or and *gbest models*, see figure (1). Originally *gbest* had better performance, but recent research has also shown good results in some problems when using the model *lbest* by adding some improvements to the algorithm.

In PSO, neighborhood is understood as the set of particles related to a given particle. This relationship influences the search capability and convergence. In the ring-type topology, each particle communicates only with n neighbors, $n/2$ on each side. Figure 2, the ring topology presented is the simplest: $n = 2$. This means that only two neighboring particles are evaluated. In the wheel topology all information is concentrated in a central particle. In the star type each particle is related to all the particles of the swarm.

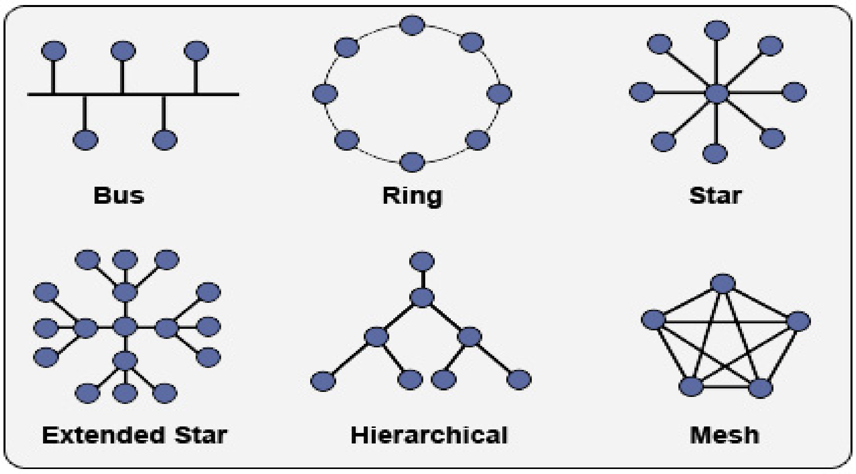


Figure 2: Different topologies.

The standard PSO algorithm is described as follows:

Algorithm

```
BEGIN
  Create initial population of particles $\{x^i\}$
  FOR each particle of the swarm DO
    IF $f(x) > f(p)$ THEN DO    $//f$ *fitness funtcion*
    FOR $d = 1$ TO $D$ DO
          $p_d = x_d$    $//p_d$ *It is the best so far*
    ENDFOR
   ENDDO
   $g = i$
   FOR $j \in J$    $//J$ *Set of indexes of neighboring particles*
    IF $f(p_j) > f(p_g)$ THEN $g = j$    $//g$ *is the index of the best particle in the neigborhood*
   ENDFOR
   FOR $d = 1$ TO $D$ Do
     $v_d(t) = v_d(t-1) + c_1\epsilon_1(p_d - x_d(t-1)) + c_2\epsilon_2(g_d - x_d(t-1))$ [1]
     $x_d = x_d + v_d$
   ENDFOR
```

[1] $v_d \in (-V_{max}, +V_{max})$

```
    ENDFOR
  END
```

It begins with a population of particles with position and velocity randomly assigned in the search space. With regard to the velocity of the particles, we consider a maximum value as restriction. This is called V_{max}. This is done because an explosion might happen, since the velocities could be increased quickly. The selection of the values V_{max} is difficult to determine. They will be chosen by trial and error. In very large spaces, large values are usually selected in order to ensure proper exploration. This is justified since a large inertia weight facilitates exploration in new areas in the global search space. On the other hand a small one facilitate the exploration in a local area. In the case of small spaces, small values are required to prevent the explosion. As for the size of the particle population, empirical results have shown that good results are not always obtained by increasing the number of particles of the population . Some examples have been influenced by that but others have not.

01 Model variants

Regarding the PSO algorithm, different variants have been developed. Most of them aimed at speeding up the convergence of it. In addition to the unconstrained optimization problem in discrete or continuous variable, the multitarget problem and the constrained problem have been addressed. We have also developed some hybrid optimization techniques. PSO technique has been tested with good results for training Artificial Neural Networks. When applying the method of Back Propagation, we are able to find appropriate weights that minimize an error function through a succession of iterations. Furthermore, by applying the PSO technique, the weights found are more efficient just by making small modifications to the algorithm. The new guidelines are aimed at avoiding PSO stagnation of the local optimal solutions.

Shi and Eberhart [13] proposed adjustments to the velocities of the particles by using a factor w called *inertial weight*. This factor utilizes the inertia of the particles in the process of friction when they are moving. This modification in the algorithm is done to control the search space. In order to do that it must change (3). The large inertia weight makes the global search easier; however small inertia weight does not improve local search. That is why was the initial value is greater than 1.0 to promote global exploration, and then gradually decreases to obtain more refined solutions. The algorithm decreases linearly at each iteration. Moreover, the use of inertial weight removes the restriction V_{max} on the velocity.

$$v_d^{(i)} = wv_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{3}$$

In each iteration, inertia weight decrease linearly through the following expression:

$$w = w_{max} - (w_{max} - w_{min})\frac{g}{G} \tag{4}$$

g is the index of the generation, G is the maximum number of iterations previously determined, w_{max} is a value greater than 1, and w_{min} a value under 0.5. This variation of the method has proven to accelerate convergence.

Clerc and Kennedy [4] obtain another variation in the speed calculation. A constriction factor χ is introduced with that purpose, This factor depends on the constants that are used when calculating speed and it affects to the formula (1) The aim is to avoid the explosion of velocity:

$$v_d^{(i)} = \chi[v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)})] \tag{5}$$

χ is:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \varphi = c_1 + c_2 = 4.1$$

The results are: $\chi = 0.729$ and c_1=c_2= 2.05. These parameters were obtained by performing several tests.

χ factor is similar to the inertial weight. This means that controlling the velocity with $V_(max)$ is not required when χ is used. Bratton and Kennedy [2], analyzed the stability of this algorithm by using these values and by following a comparative study of both PSO algorithms (inertial weight and χ factor). Both of them are mathematically equivalent, in particular the algorithm with constriction factor is a special case of the inertial weight. Moreover, Parsopoulos al [12], combined both for problems with constraints and they obtained equally good results in several tests.

We observe that the convergence always becomes slower when problem size increase, so when it comes to high-dimensional problems, a larger number of iterations occurs. Researchers Hatanaka et al [6] developed a PSO model, where velocity values are updated, by considering the rotation of the coordinate system. This model is aimed at problems of high dimensionality and it showed good results when applied to all functions of De Jong, (larger dimensions).

Numerical Examples

In order to test the standard PSO algorithm and two variants with incorporated and inertial weight factor χ, we have used some unrestricted functions which are commonly referred as *De Jong functions*'. The minimum of these functions is located in the search space. They were originally proposed by de Jong to measure the performance of genetic algorithms. However they have also been used to test the performance in PSO algorithms. Some of the other functions are unimodal and multimodal i.e Ring (*lbest*) and star (*gbest*) topologies. In the ring topology each particle is related to its two neighbors. In the star topology all particles are interconnected. A population of 20 particles was considered. Table 1 functions are tested with the standard PSO algorithm and two variations: inertia weight and constriction factor. The first three functions are unimodal and have the optimal solution $x^* = 0.0^d$ and the minimum value $f_i(x^*) = 0.0$. The following are multimodal functions, the function f_4 has the minimum value 0.0, the optimal solution is $\pm n\frac{\pi}{2}^d$, the function f_5 has optimal solution $x^* = 0.0^d$ and the minimum value of the function: $f(x^*) = 0.0$ and f_6 has the optimal solution $x^* = 420.968^d$ and the minimum value of the function : $f(x^*) = -12.569, 4866$.

Tables (2) and (3) show the results after 20 executions of the standard algorithm. Not only the inertial weight has been modified in this algorithm but also the constriction factor for each functions by using neighborhood models *lbest* and *gbest*. The algorithm stops when two successive values of the best assessments of the swarm get close to each other. (A ϵ value is prefixed and so it is a maximum number of iterations). NPE

Function	Dim.	Search space.	Name
$f_1(x) = \sum_{i=1}^{d} x_i^2$	30	[-100,100]	Sphere/Parabola
$f_2(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	Rosenbrock Generaliz.
$f_3(x) = \sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	30	[-100,100]	Schwefel 1.2
$f_4(x) = \sum_{i=1}^{d} cos(x_i)^2$	30	$(-\infty, \infty)$	
$f_5(x) = \sum_{i=1}^{d}(x_i^2 - 10cos(2\pi x_i) + 10)$	30	[-5.2, 5.2]	Rastrigin Generaliz.
$f_6(x) = -\sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_d \rvert})$	30	[-500,500]	Schwefel Generaliz. 2.6

Table 1: Functions tested with PSO algorithm

(average number of assessments) shows the average number of evaluations for the function when applying PSO and its variants.

Name	PSO Original		Peso Inertial		Factor Constriction	
	Best Solut.	NPE	Best Solut.	NPE	Best Solut.	NPE
f_1	**l**: 3,70889e-07	190.480	7,68359e-07	28.800	2,69078e-08	19.100
	g: 5,9803e-04	2×10^5	2,45656e-20	151.500	3,63055e-20	30.820
f_2	**l**: 5,60357e-06	2×10^5	2,2112e-05	831.580	4,5089e-06	1.257.520
	g: 4,51068e-02	4×10^6	9,24376e-12	4×10^6	2,59676e-09	5×10^5
f_3	**l**: 5,76231e-06	198.280	5,57901e-15	1.425.900	2,83076e-16	399.140
	g: 4,91261e-06	2×10^5	1,81786e-12	2.596.780	1,44177e-13	125.640

Table 2: Results obtained by applying the PSO algorithm to unimodal functions, l indicates the model *lbest*; g refers to model *gbest*.

After PSO Algorithm and its variants are executed, results are collected; best results are found in approximately equal number of cases regardless the model is used (*lbest* or *gbest*), so we can not assure which topology is the optimal. Moreover, we notice that when using the constriction factor, the convergence accelerates and the results are better when compared to the exact solution. In some cases, we notice that the region in which the swarm of particles starts can influence the results. Regarding the number of particles of the swarm, when increased to more than 20, results did not improve.

Problems with Constraints

An optimization problem (minimization) with restrictions is defined as follows:

Minimize $\mathbf{f}(\mathbf{x})$

Constrains: $\mathbf{g_j}(\mathbf{x}) \leq \mathbf{0} \quad (j = 1, \dots, p)$

$\mathbf{h_j}(\mathbf{x}) = \mathbf{0} \quad (j = p+1, \dots, m)$

$x \in \Re^n$

Name	**PSO Original**		**Weight Inertial**		**Factor Constriction**	
	Best solut.	NPE	Best solut.	NPE	Best solut.	NPE
f_4	**l**: 2.18719e-05	2×10^5	4.36373e-19	114.560	6,7306e-19	4.174
	g: 3,9543e-12	180.820	1.3000e-12	15.120	1.7063e-15	14.160
f_5	**l**: 1,70688e-14	216.240	1,81227e-14	339.020	1,07181e-11	9.480
	g: 3,00262e-04	2×10^5	6,82288e-12	12.180	2,81599e-12	11.040
f_6	**l**: -12.568,2	2×10^5	-12.569,5	2×10^5	-12.569,5	2×10^5
	g: -12.569,5	6×10^5	-12.569,5	6×10^5	12.352,3	2×10^5

Table 3: Results obtained by applying the PSO algorithm to unimodal functions, l indicates the model *lbest*; g refers to model *gbest*.

The problem of nonlinear constrained optimization arises frequently in engineering.In general it does not have a deterministic solution. In the past, nonlinear optimization methods were developed and now it is a challenge to work with differentiable functions.Before gradient methods were used succesfully for solving some problems. Evolutionary methods provide a new possibility for solving such problems. The PSO technique has been used successfully in optimizing real functions without restrictions, but it has been little used for problems with restrictions. This has happened mainly because there are no mechanism to incorporate restrictions on the *fitness* function. Evolutionary Computation has tried to solve the constrained optimization problem, either by bypassing nonfeasible solutions sequences, or by using a penalty function for nonfeasible sequences. Some researchers suggest to use two subfunctions of *fitness*. One helps to evaluate feasible elements and the other one evaluates the unfeasible onea . In this regard, there are many criteria. Moreover, some special self adaptive functions have been designed to implement the penalty technique.

Hu and Eberhart [8] presented a PSO algorithm. This algorithm bypasses nonfeasible sequences. it also creates a random initial population,in which nonfeasible sequences are bypassed until the entire population has only feasible particles. By upgrading the positions of the particles nonfeasible sequences are bypassed automatically. The cost of the technique that creates the initial populations is high, especially when it comes to problems with nonlinear constraints because then it must create an entire population of feasible individuals. In his work, Cagnina et al [3] proposed the following strategies for implementing the PSO into problems with restrictions: **a)** If two particles are feasible, select the one with the best *fitness*. **b)** When one particles is feasible and the other is not, the feasible one is chosen. **c)** If two particles are nonfeasible, the one with the lowest degree of nonfeasibility is selected. These strategies are applied when the particles *gbest* and *lbest* are selected . The same authors also proposed an update in (1). This update considers three elements:

1. $p_d^{(i)}$ which is the best position reached by the particle i in its history,
2. $g_d^{(i)}$ which is the best position reached by the particles in its neigborhood and,
3. t_d which is the best position achieved by any particle in the whole swarm.

$$v_d^{(i)} = w(v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) + c_3\epsilon_3(t_d - x_d^{(i)})) \tag{6}$$

where c_1 is the personal learning factor and c_2 and c_3 are the social learning factors. According to Michalewicz et al [10] y [11] constrained optimization methods are classified as:

1. Methods based on preserving feasibility of solutions.
2. Methods based on penalty functions
3. Methods that make a clear distinction between feasible solutions and infeasible sequences.
4. Methods based on decoders
5. Hybrid methods

Part of our work is to analyze penalty methods under E.A. perspective (Evolutionary Algorithms). The penalty methods use functions (penalty functions) that degrade the quality of the nonfeasible solution. In this way the constrained problem becomes a problem without constraints by using a modified evaluation function:

$$eval(x) = \begin{cases} \text{f(x)} & x \in \mathcal{F} \\ \text{f(x) + \textit{penalty}(x)} & eoc \end{cases}$$

where $\mathcal{F}$ is the set created by the intersection of all sets that are the restrictions of the problem (Feasible region). The penalty is zero if no violation occurs and it is positive otherwise. The penalty function is based on the distance beteween a nonfeasible sequence and the feasible region $\mathcal{F}$, It also works for repairing solutions outside of the feasible region $\mathcal{F}$.

There are many penalty methods. The main difference between the methods is the way the penalty function is designed and applied to the nonfeasible sequences. Some methods associate a penalty function f_j, $(j = 1, \ldots, m)$ with a constraint, which measures the violation of the restriction j as follows:

$$f_j(x) = \begin{cases} max\{0, g_j(x)\}, & si \;\; 1 \leq j \leq p \\ |h_j(x)| & si \;\; p+1 \leq j \leq m \end{cases}$$

According to Michalewicz al[10], penalty methods are classified as **a)** Static, **b)** Dynamic, **c)** Annealing **d)** Adaptive **e)** death. There are other hybrid methods that combine different techniques. Static methods use l violation levels of constraints; to do that it creates a penalty coefficient $R_{ij}, (i = 1, 2, \ldots, l, j = 1, 2, \ldots, m)$, for each constraint. The greater the violation of the restriction, the greater its value. It begins with a population that may have nonfeasible elements. *fitness* function is as follows:

$$eval(x) = f(x) + \sum_{j=1}^{m} R_{ij} f_j^2(x)$$

The main problem of this methods is the number of parameters involved.

Dynamic penalties method

Individuals are evaluated in each iteration (generation) k, by using the following formula:

$$eval(x) = f(x) + (C.k)^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(x)$$

C, α and β are constant values. The penalty changes over time. A reasonable selection of these parameters is: $C = 0,5$, $\alpha = \beta = 2$. $(Ck)^{\;alpha}$ is larger when k increases. $Functions f_j$ represent restrictions violation. They are calculated as $f_j = max\ o, g_j(x)$

This method obtains the following results when applied to the following set of test functions:

Function	$f(x^*)$	Best solution	Media	Est Deviation.
g01	-15.0	-15.0	-13.86241	1.29431593
g02				
g03				
g04	-30.665,539	-30.687,8	30.992.42	657.818933
g05				
g06	-6961.63	-6962.63	-6961.6725	12.93028533
g07	24,3062091	24,4253	26,18224	1,61696066
g08	-0,095825	-0.095825	-0,09576949	0,00010912
g09	680,6300573	680,649	680.86865	0,22674309
g10				

Restrictions such as 'equal to' are difficult to work with;

$|h_j(x)| \leq \epsilon,$

That is the reason why we decide to relax these constraints by setting a prefixed value to ϵ=10^{-6}

A proper adjustment of parameters plays an important role in PSO, for example the adjustment of maximum speed to the the range of values that the inertial weight can take. Upgrading the position and speed of particles (original equations 1 and 2) is another parameter that plays a major role in PSO. The latter are related to social influence exerted by each particle in the swarm (topology of the particles) and the overall behavior of the swarm. The other parameters apply to the initial generation of particles.

$$v_d^{(i)} = v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{7}$$

$$x_d^{(i)} = x_d^{(i)} + v_d^{(i)} \tag{8}$$

The data included in the table are the satisfactory results. We have detected cases that need to be improved. In these cases we re working on the adjustment to the parameters of the particles and to the penalty function. Maurice Clerc(*Stagnation Analysis in PSO*, 2006) proposes amendments to the standard model PSO-2007 Kennedy. PSO has developed several models to improve the classical model by following that way. They are no differences regarding the values of c_1 and c_2. In this regard we have tried some configurations, but the improvements have not been significant. In connection with the neighborhoods, Kennedy and Eberhart (textit Bare Bones Particle Swarm, 2003) have proposed the use of Gaussians neighborhoods. This is an adjustment to the particle position by using an estimation of the average distribution between the best recorded position of the particle ($p^{(i)}$) and the best position within its neighborhood ($g^{(i)}$). The standard deviation is the difference between these two values (see equation ref gauss).This will gain new positions of the particles by using $N(mu, sigma)$. Some authors compare this method to the random variation of the mutations occuring in Evolutionary Computing because the particle position suffers a random change caused by a different rule.

$$N(\frac{|p^{(i)} - g^{(i)}|}{2}, |p^{(i)} - g^{(i)}|) \tag{9}$$

Moreover Kennedy and Mendes in (*Neighborhood Topologies in Fully Informed and Best Of Neighborhood Particle Swarms*, 2006) have used the Von Neumann neighborhood to deal succesfully with other types of problems. This will be object of our study in further work.

Test Functions

Below is a description of the functions to be tested using the penalty methods:

g01: Minimizing $f(x) = \sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$

constraints:

$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$

$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$

$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$

$g_4(x) = -8x_1 + x_{10} \leq 0$

$g_5(x) = -8x_2 + x_{11} \leq 0$

$g6(\ x) = -8x_3 + x_{12} \leq 0$

$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0$

$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0$

$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0$

$0 \leq x_i \leq 1 (i = 1, \ldots, 9)$, $0 \leq x_i \leq 100$ (i = 10; 11; 12) y $0 \leq x_{13} \leq 1$

Global optimum is : $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, $f(x^*) = -15$.

g02: Maximizing: $f(x) = \left| \frac{\sum_{i=1}^{n} cos^4 x_i - 2\prod_{i=1}^{n} cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$

constraints:

$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$

$g_2(x) = \sum_{i=1}^{n} x_i - 7.5 \leq 0$

and $n = 20$ y $0 \leq x_i \leq 10$, $(i = 1 \ldots n)$.

Global optimum is unknown, the best output value for $f(x)$ is $f(x^{\ ast}) =$ 0.803619.

g03: Maximizing: $f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$

constraints:

$h(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$

$n = 10$ y $0 \leq x_i \leq 1, (i = 1, \ldots, n)$.

Local optimum is $x_i^* = 1/\sqrt{n}$, (i=1,...,n), $f(x^*) = 1$.

g04: Minimizing: $f(x) = 5,3578547x_3^2 + 0,8356891x_1x_5 + 37,293239x_1 - 40792,141$

constraints:

$g_1(x) = 85,334407 + 0,0056858x_2x_5 + 0,0006262x_1x_4 - 0,0022053x_3x_5 - 92 \leq 0$

$g_2(x) = -85,334407 - 0,0056858x_2x_5 - 0,0006262x_1x_4 + 0,0022053x_3x_5 \leq 0$

$g_3(x) = 80,51249 + 0,0071317x_2x_5 + 0,0029955x_1x_2 + 0,0021813x_3^2 - 110 \leq 0$

$g_4(x) = -80,51249 - 0,0071317x_2x_5 - 0,0029955x_1x_2 - 0,0021813x_3^2 + 90 \leq 0$

$g_5(x) = 9,300961 + 0,0047026x_3x_5 + 0,0012547x_1x_3 + 0,0019085x_3x_4 - 25 \leq 0$

$g_6(x) = -9,300961 - 0,0047026x_3x_5 - 0,0012547x_1x_3 - 0,0019085x_3x_4 + 20 \leq 0$

$78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45, 27 \leq x_i \leq 45 (i = 3; 4; 5)$.

Optimal solution is: $x^* =$ (78, 33, 29.995256025682, 45, 36.775812905788), $f(x^*) =$ -30665.539

g05: Minimizing: $f(x) = 3x_1 + 0,000001x_1^3 + 2x_2 + (0,000002/3)x_2^3$

constraints:

$g_1(x) = -x_4 + x_3 - 0,55 \leq 0$

$g_2(x) = -x_3 + x_4 - 0,55 \leq 0$

$h_3(x) = 1000sin(-x_3 - 0,25) + 1000sin(-x_4 - 0,25) + 894,8 - x_1 = 0$

$h_4(x) = 1000sin(x_3 - 0,25) + 1000sin(x_3 - x_4 - 0,25) + 894,8 - x_2 = 0$

$h_5(x) = 1000sin(x_4 - 0,25) + 1000sin(x_4 - x_3 - 0,25) + 1294,8 = 0$

and $0 \leq x_1 \leq 1200, 0 \leq x_2 \leq 1200, -0,55 \leq x_3 \leq 0,55, -0,55 \leq x_4 \leq 0,55$

Optimal solution: $x^* =$ (679.9453, 1026.067, 0.1188764, -0.3962336), and $f(x^*) =$ 5126.4981.

g06: minimizing: $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

constraints:

$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$

$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82,81 \leq 0$

and $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$

optimal solution: $x^* =$ (14.095, 0.84296), and $f(x^*) =$ -6961.81388.

g07: minimizing: $f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2$
$+2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$

constraints:

$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$

$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$

$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$

$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 \leq 120$

$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$

$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$

$g_7(x) = 0,5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 \leq 30$

$g8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$

and $-10 \leq x_i \leq 10 (i = 1, \ldots, 10)$.

Optimal Solution: $x^* =$ (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927), and $f(x^*) =$ 24.3062091.

g08: Maximizing: $f(x) = \frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1+x_2)}$

constraints:

$g_1(x) = x_1^2 - x_2 + 1 \leq 0$

$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

and $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$.

Optimal Solution: $x^* =$ (1.2279713, 4.2453733), and $f(x^*)$ = 0.095825.

g09: minimizing $f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2$
$+x_7^4 - 4x_6x_7 + 10x_6 - 8x_7$

constraints:

$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$

$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$

$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$

$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$

and $-10 \leq x_i \leq 10 (i = 1, \ldots, 7)$.

Optimal Solution: $x^* =$ (2.330499, 1.951372,-0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227) and $f(x^*) =$ 680.6300573.

g10: minimizing $f(x) = x_1 + x_2 + x_3$

constraints:

$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$

$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$

$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$

$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$

$g_5(x) = -x_2x_7 + 1250x_5 + x_1x_4 - 1250x_4 \leq 0$

$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$

and $100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000, (i = 2, 3), 10 \leq x_i \leq 1000, (i = 4, \ldots, 8)$.

Optimal solution: $x^* =$ (579.3167,1359.943, 5110.071,182.0174, 295.5985, 217.9799,286.4162, 395.5979) and $f(x^*) =$ 7049.3307.

Conclusions and Future Remarks

This paper has reviewed some natural computation strategies as a survey concerning optimization strategies. The so-called collaborative model PSO has been exposed in order to understand the ability to extract some biological concepts and apply them in computational models as described along the paper. Such biological inspired models have proof to be a powerful tool in order to solve non common problems in a collaborative/competitive way.

Particle Swarm Optimization is often failed in searching the global optimal solution in the case of the objective function has a large number of dimensions. The reason of this phenomenon is not only existence of the local optimal solutions, the velocities of the particles sometimes lapsed into the degeneracy, so that the successive range is restricted in the sub-plain of the whole search hyper-plain [17]. The sub-plane that is defined by finite number of particle velocities is a partial space in the whole search space. The issue of local optima in PSO has been studied and proposed several modifications on the basic particle driven equation [18; 19]. There used a kind of adaptation technique or randomized method (e.g. mutation in evolutionary computations) to keep particles velocities or to accelerate them. Although such improvements work well and have ability to avoid fall in the local optima, the problem of early convergence by the degeneracy of some dimensions is still remaining, even if there are no local optima. Hence the PSO algorithm does not always work well for the high-dimensional function.

Usually, neural network parameters are in a high-dimensional space and then PSO algorithms are not very efficient ones dealing with such individuals. Best solution could be the integration of PSO and GA in a new model $GPSO$ taking advantages of both models, or at least to improve the impact of the high dimensional individuals in the PSO algorithm, see figures 3 and 4.

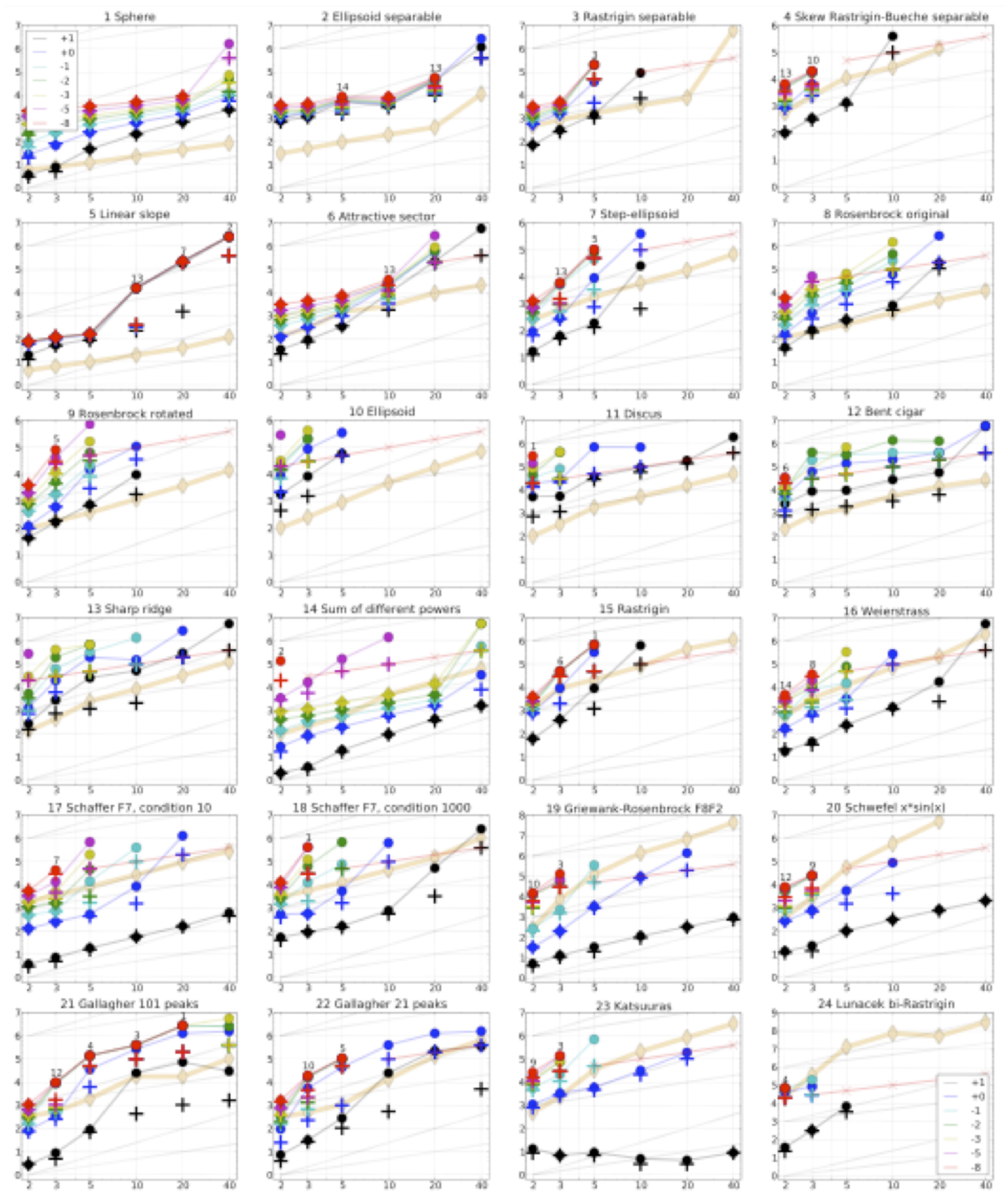


Figure 3: PSO Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of f-evaluations from successful trials (+), for $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. For each function and dimension, $\mathrm{ERT}(\Delta f)$ equals to $\#\mathrm{FEs}(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{\mathrm{opt}} + \Delta f$ was surpassed. The $\#\mathrm{FEs}(\Delta f)$ are the total number (sum) of f-evaluations while $f_{\mathrm{opt}} + \Delta f$ was not surpassed in the trial, from all (successful and unsuccessful) trials, and f_{opt} is the optimal function value. Crosses ($\times$) indicate the total number of f-evaluations, $\#\mathrm{FEs}(-\infty)$, divided by the number of trials. Numbers above ERT-symbols indicate the number of successful trials. Y-axis annotations are decimal logarithms. The thick light line with diamonds shows the single best results from BBOB-2009 for $\Delta f = 10^{-8}$. Additional grid lines show linear and quadratic scaling.

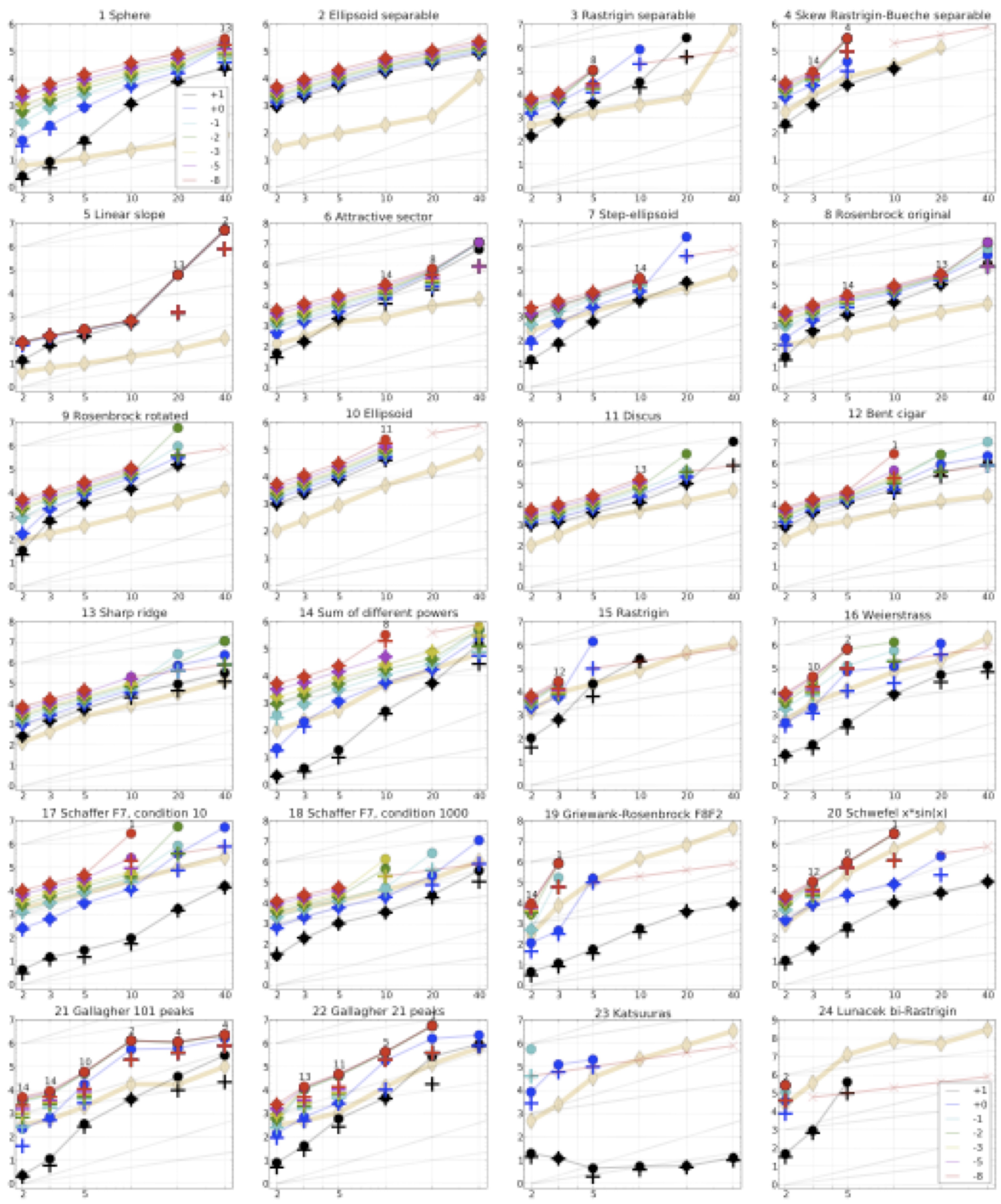


Figure 4: DE-PSO Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of f-evaluations from successful trials (+), for $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. For each function and dimension, $\mathrm{ERT}(\Delta f)$ equals to $\#\mathrm{FEs}(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{\mathrm{opt}} + \Delta f$ was surpassed. The $\#\mathrm{FEs}(\Delta f)$ are the total number (sum) of f-evaluations while $f_{\mathrm{opt}} + \Delta f$ was not surpassed in the trial, from all (successful and unsuccessful) trials, and f_{opt} is the optimal function value. Crosses ($\times$) indicate the total number of f-evaluations, $\#\mathrm{FEs}(-\infty)$, divided by the number of trials. Numbers above ERT-symbols indicate the number of successful trials. Y-axis annotations are decimal logarithms. The thick light line with diamonds shows the single best results from BBOB-2009 for $\Delta f = 10^{-8}$. Additional grid lines show linear and quadratic scaling.

Bibliography

[1] Beni G. E. Ahin, Spears W.M., *Swarm Robotics WS 2004, From Swarm Intelligence to Swarm Robotics*. LNCS 3342, pp. 1-9, 2005. Springer-Verlag Berlin Heidelberg, 2005.

[2] Bratton, D., Kennedy, J.,*Defining a Standard for Particle Swarm Optimization*.Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007).

[3] Cagnina, L.C., Esquivel,S. C., Coello, C.,*Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer*, 2008.

[4] Clerc, M., Kennedy, J.,*The particle Swarm: Explosion, Stability, and Convergence in a Multidimensional Complex Space*,IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 5873, Feb. 2002.

[5] HernÃąndez, A., Muñoz, A., Villa, E., Botello, S., *COPSO: Constrained OptimizaciÃşn Via PSO Algorithm*,ComunicaciÃşn TÃľcnica No I0704/22022007.(CC/CIMAT), 2007.

[6] Hatanaka, T., Korenaga, T., Kondo, N., Uosaki, K.*Search Performance Improvement for PSO in High Dimensional Space*,2007.

[7] Hvass, P.,M.E.,*Tuning & Simplifying Heuristical Optimization*,Thesis PhD,University of Southampton, 2010.

[8] Hu, X., Eberhart, R., *Solving Constrained Nonlinear Optimization Problems wirh Particle Swarm Optimization*. Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informaticsics, SCI2002 , Vol. 5, IIIS, July 2002.

[9] Kennedy, J., Eberhart, R., *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.

[10] Michalewicz, Z. , Fogel, D., *How to solve it. Modern Heuristics*. Springer,1998.

[11] Michalewicz, Z. , Shoenauer, M., *Evolutionary Algorithms for Constrained Parameter Optimization Problems*. Evolutionary Computation, 1996, Vol. 4, pp. 132.

[12] Parsopoulos, K.E., Vrahatis, M. N.,Particle Swarm Optimization Method for Constrained Optimization Problems,2002.

[13] Shi, Y., Eberhart, R.,*A Modified Particle Swarm Optimizer*, Proc. IEEE World Congr. Comput. Intell., 1998, pp. 69 73.

[14] Tan, C. , Tapabrata, R. , Pawan, D.,*Supplementary Material on Parameter Estimation using Swarm Algorithm*, Preprint Elsevier Science, 2004.

[15] Weise, T.,*Global Optimization Algorithms-Theory and Application*, e-book, 2009.

[16] Zhan, Z., Zhang, J., Li, Y., Chung, H. S. -H., *Adaptative Particle Swarm Optimization*. IEEE Transactions On Systems, Man, and Cybernetics,Part B: Cybernetics,Digital Object Identifier: 10.1109/TSMCB.2009.2015956, 2009.

[17] Rapaic, M. R., Kanovic, Z. and Jelicic, Z. D., "A Theoretical and Empirical Analysis of Convergence Related Particle Swarm Optimization", *WSEAS Transactions on Systems and Control, 4*, 2009.

[18] Parsopoulos, K., Plagianakos, V. P. and Magoulas, G. D., "Stretching technique for obtaining global minimizers through particle swarm optimization" in *Proceedings of the Particle Swarm Optimization Workshop* (Indianapolis), pp. 22-29, 2001.

[19] Hendtlass, T., "A particle swarm algorithm for high dimensional, multi-optima problem spaces", in *Proceedings of Swarm Intelligence Symposium 2005* (Pasadena), pp. 149-154, 2005.

Authors' Information

Nuria Gómez Blas - *Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing sn, 28031 Madrid, Spain; e-mail: nuria.gomez.blas@upm.es*
Major Fields of Scientific Research: Bio-inspired Algorithms, Natural Computing

Luis Fernando de Mingo López - *Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing sn, 28031 Madrid, Spain; e-mail: fernando.demingo@upm.es*
Major Fields of Scientific Research: Artificial Intelligence, Social Intelligence

Juan Castellanos Peñuela - *Departamento de Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, Universidad Politécnica de Madrid, Campus de Montegancedo sn, 28660 Madrid,, Spain; e-mail: jcastellanos@fi.upm.es*
Major Fields of Scientific Research: Formal Languages, Neural Networks

ISOMORPHISM OF PREDICATE FORMULAS IN ARTIFICIAL INTELLIGENCE PROBLEMS

Tatiana Kosovskaya

Abstract: *The paper discusses various aspects of application of the notion of isomorphism of elementary conjunc- tions of predicate formulas in Artificial Intelligence (AI) problems that can be formalized by means of predicate calculus. The notion of isomorphism of various objects is widespread in mathematics. Moreover, isomorphic objects have a large number of identical properties. The definition of isomorphic elementary conjunctions of predicate formulas is given in the paper. The main property of such formulas is that they define the same relation between their arguments. The main difference between the notion of isomorphism and the notion of equivalence is that the equivalent formulas must have the same arguments, and the arguments of isomorphic formulas may be significantly different. In the framework of the logic-objective approach to solving AI problems, the following problems, for solving which the notion of isomorphism is used, are considered: the problem of object classification; creating a level description of classes to decrease the computational complexity of the analysis problem of a complex object; creating a level description of the database to decrease computational complexity while multiple solution of the problem Conjunctive Boolean Query; definition of a metric in the space of elementary conjunctions of predicate formulas.*

Keywords: *Logic-objective recognition, isomorphism of predicate formulas, NP-completeness, GI-completeness.*

ITHEA Keywords: *G.2. Mathematics of Computing, Discrete mathematics.*

Introduction

When solving an AI problem, much attention is paid to the methods of this problem formalizing. Among such methods, logical methods are widely used. Usually an object is simulated by a string of binary attributes. With this simulation, the structure of an object is lost, but the solving algorithms are usually polynomial or even linear.

An example of economic problem simulation using a binary string is presented in Russel, etc., [2008]. At the same time, the length of such a string is exponential under the length of the initial data presented as a set of properties and relations between elements of an object under consideration.

In the frameworks of a logic-objective approach to AI problems Kosovskaya, [2018], an object under consideration is represented as a set of its elements, and a description of an object is a set of literals (atomic formulas or their negations) with predicates that define properties of an object elements or relations between them. Such a way formulated problems are usually NP-complete or NP-hard (if it is required not only to recognize the presence of a part with specified properties in a complex object, but also to select such a part) Kosovskaya, [2007].

The notion of isomorphism of elementary conjunctions of predicate formulas is under consideration in this paper. Isomorphic formulas define the same relation between their arguments and, therefore, the extraction of sub-formulas isomorphic to each other from a set of more complex formulas allows us to find out the structure of objects. Extracting of such a structure in the input data makes it possible to decrease computational complexity of many AI problems.

The following problems, for solving which the notion of isomorphism is used, are considered: the problem of object classification; creating a level description of classes to decrease the computational complexity of solving the analysis problem of a complex object; creating a level description of the database to decrease computational complexity while multiple solution of the problem Conjunctive Boolean Query; definition of a metric in the space of elementary conjunctions of predicate formulas.

Logic-objective approach to AI problems solving

A detailed description of the logic-objective approach to AI problems is presented in Kosovskaya, [2018]. Here only general setting of the problems and the main methods for their solving will be formulated.

Let an investigated object be represented as a set of its elements $\omega = \{\omega_1, \dots, \omega_t\}$. A set of predicates $p_1, \dots, p_n$, which characterize properties of elements of ω and relations between them, is defined on ω. The logical description $S(\omega)$ of an object ω is a set of all literals true on ω. The set of all objects is divided into classes $\Omega = \cup_{k=1}^{K} \Omega_k$. The logical description of the class Ω_k is the formula $A_k(\overline{x}_k)$ given as a disjunction of elementary conjunctions, such that if $A_k(\overline{\omega})$ is true, then $\omega \in \Omega_k$.[1]

The following problems may be formulated in the framework of logic-objective approach.

Identification problem. *To check whether the object ω or its part satisfies the description of the class $A_k(\overline{x}_k)$ and to extract this part of the object.*[2]

$$S(\omega) \Rightarrow \exists \overline{x}_{k\neq}\, A_k(\overline{x}_k) \tag{1}$$

Classification problem. *To find all such numbers k that the formula $A_k(\overline{\omega})$ is true.*

$$S(\omega) \Rightarrow \bigvee_{k=1}^{M} A_k(\overline{\omega}) \tag{2}$$

Analysis problem. *To find and classufy all parts τ of an object ω such that $A_k(\overline{\tau})$ is true for some permutation $\overline{\tau}$ of elements of τ.*

$$S(\omega) \Rightarrow \bigvee_{k=1}^{M} \exists \overline{x}_{k\neq}\, A_k(\overline{x}_k) \tag{3}$$

The problems (1) and (3) are NP-complete ones Kosovskaya, [2007].

[1] Hereinafter, $\overline{x}$ denotes the list of elements of a finite set x corresponding to a certain permutation of its elements. The fact that the elements of the list $\overline{x}$ are elements of the set y will be written in the form $x \subseteq y$.

[2] To denote that values for the list of variables $\overline{x}$, satisfying the formula $A(\overline{x})$ are different, instead of $\exists x_1 ... \exists x_m (\&_{i=1}^{m} \&_{j=i+1}^{m} (x_i \neq x_j) \; \& \; A(x_1, ..., x_m))$ the notation $\exists \overline{x}_{\neq} A(\overline{x})$ will be used.

Note that to prove (1) or (3), it is sufficient to be able to prove the logical sequent

$$S(\omega) \Rightarrow \exists \overline{x}_{\neq}\, A(\overline{x}), \tag{4}$$

where $A(\overline{x})$ is an elementary conjunction of atomic formulas or their negations. Estimates of the number of steps of the algorithms solving problem (4), as well as problems (1) and (3), are proved in Kosovskaya, [2007]. These bounds have an exponential on the length of formula $A(\overline{x})$ form.

So, for example, for an algorithm based on exhaustive search method of complete enumeration of all possible substitutions of constants from ω instead of variables from $A(\overline{x})$, the number of steps is $O(t^m \cdot \sum_{i=1}^{n} a_i \cdot s_i)$, where t is the number of constants in ω, n are the numbers of variables in $\overline{x}$, s_i, a_i are the numbers of occurrences of atomic formulas with the predicate p_i in $S(\omega)$ and in $A(\overline{x})$, respectively. Note, that this estimation coincide with the one for simulation of an economic problem by a binary string, given in Russel, etc., [2008].

For algorithms based on the construction of the derivation in the predicate calculus, the estimates are $O(s_1^{a_1} \cdot ... \cdot s_n^{a_n}) = O(s^a)$, where $s = \max_i(s_i)$, $a = \sum_{i=1}^{n} a_i$.

The problem (2) may be reduced to a sequential check for $k = 1, \ldots, M$ of an elementary conjunction $A_k(\overline{\omega})$ and a conjunction of literals from $S(\omega)$ isomorphism.

Isomorphism of predicate formulas

Definition. *Two elementary conjunctions of atomic predicate formulas P and Q are called isomorphic if there exists such an elementary conjunction R and substitutions of arguments $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$ of formulas P and Q, respectively, instead of all occurrences of the variables $x_1, \ldots, x_m$ of the formula R, that the results of these substitutions in R coincide with the formulas P and Q, respectively, up to the order of literals.*

The recieved substitutions $(a_1 \to x_{i_1}, \ldots, a_m \to x_{i_m})$ and $(b_1 \to x_{j_1}, \ldots, b_m \to x_{j_m})$ are called unifiers of the formulas P and Q with formula R.

Note that the arguments of elementary conjunctions P and Q may be either object variables or object constants. In addition, the notion of isomorphism of elementary conjunctions of atomic predicate formulas differs from the notion of these formulas equivalence, since they may have significantly different arguments. In fact, for isomorphic formulas there are such permutations of their arguments, that they define the same relation between their arguments.

An algorithm for checking the isomorphism of formulas, which has an exponential computational complexity, as well as an approximate polynomial algorithm for solving this problem is in Petrov, [2016]. If the approximate algorithm gives the answer No, then the formulas are not isomorphic. An example of formulas that are not isomorphic but the algorithm gives the answer Yes is given. If Yes, the computational experiment showed that in 99.95% of cases the formulas are really isomorphic.

Consider two problems.

Predicate formula isomorphism (PFI).

Instance. Two elementary conjunctions of atomic predicate formulas P and Q.

Question. Are P and Q isomorphic?

Graph isomorphism (GI) Garey, etc., [1979].

Instance. Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.

Question. Are $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ isomorphic?

It is not now known if **GI** belongs to the class **P** nor it is an NP-complete one. A special term GI-complete problem is introduced for problems that are polynomially equivalent to it. Bellow a sketch of the proof that **PFI** and **GI** are polynomially equivalent is presented.

Theorem 1. **GI** *is polynomially reducible to* **PFI***.*

Proof. **GI** is a restriction of **PFI**. The restriction consists in the condition that there is the only one binary predicate p and elementary conjunctions P and Q do not contain negations. In such a case the set of arguments of every elementary conjunction is the set of nodes, and the set of argument pairs in literals is the set of edges. $\square$

To prove the polynomial equivalence of the problems it would be shown that according to every pair of elementary conjunctions P and Q it s possible to construct two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ which are isomorphic if and only if P and Q are isomorphic.

Introduce the notion of **graph associated with an elementary conjunction** of predicate formulas every literal of which does not contain the same arguments.

For each predicate symbol p_i, we associate two nodes w_{2i-1} and w_{2i}. The first one corresponds to the occurrences of literals with p_i without negations, and the other to the occurrences of literals with p_i with negation. In addition, each of these nodes w_j has $j + 2$ adjacent "hanging" nodes (nodes with the degree 1) to indicate the index of a predicate symbol.

For each occurrence of a literal corresponding to the node w_i $(1 \leq i \leq 2n)$, we associate a sub-graph obtained by successively connecting of k_i (the number of predicate arguments) nodes with edges (the first of them mark with the help of a hanging node), as well as with the node w_i. The jth node $j = 1, \ldots, k_i$ is connected with the node corresponding to the jth argument of the literal.

For example, the sub-graph associated with the literal $\neg p_i(a, b, c)$ is presented on the Figure 1.

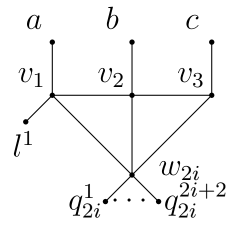


Figure 1. Sub-graph associated with the literal $\neg p_i(a, b, c)$.

All such sub-graphs, associated with different literals with the same predicate, have a common node w_{2i-1} (or w_{2i} if its occurrence is with negation) with $2i + 1$ (or $2i + 2$) adjacent "hanging" nodes. For example, elementary conjunction $p(a)$ & $q(x, y)$ & $q(x, a)$ & $\neg q(a, y)$ is associated with the graph presented on Figure 2. Here the nodes w_1, w_3 and w_4 correspond to literals with p, q and $\neg q$, respectively.

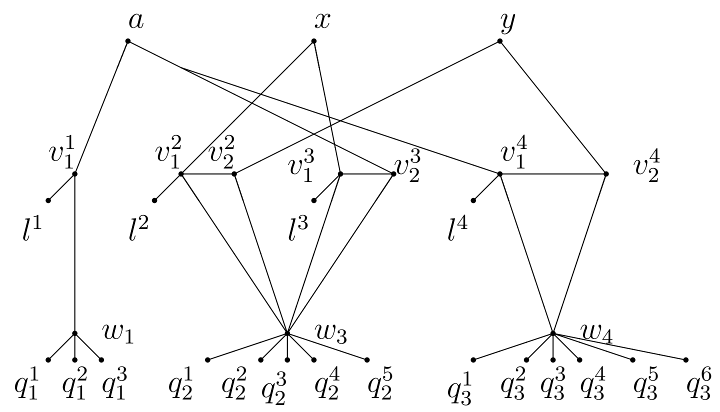


Figure. 2. Graph associated with the elementary conjunction $p(a)\ \&\ q(x,y)\ \&\ q(x,a)\ \&\ \neg q(a,y)$.

Theorem 2. PFI *is polynomially reducible to* **GI**.

Scheme of proof. According to input data P and Q for the problem of the **PFI** we construct input data G_P and G_Q in the form of graphs associated with P and Q. Graphs G_P and G_Q may be written down in number of steps bounded by a polynomial under the length of P and Q.

Elementary conjunctions P and Q are isomorphic if and only if graphs G_P and G_Q are isomorphic.

For an arbitrary isomorphism of graphs G_P and G_Q, sub-graphs with the same node w_j may not necessarily be identicall. This corresponds to a permutation of literals in elementary conjunctions. $\square$

Polynomial equivalence of the problems of **PFI** and **GI** directly follows from thees theorems.

Theorem 3. PFI *is polynomially equivalent to* **GI**.

Classification problem

The classification problem (2) formulated in the first section is the one which is the most spread among recognition problems in the frameworks of Artificial Intelligence. While its solving it is not needed to extract the object from a more complicated one. There is no a so called "vicious circle": extract for recognition and recognize to extract.

The solution of this problem is a multiple (for $k = 1, \ldots, M$) logical sequence checking

$$S(\omega) \Rightarrow \exists \Pi(\omega) A_k(\Pi(\omega)),$$

where $\Pi(\omega)$ is a permutation of the elements of the set ω.

That is, if the sign $\&$ is inserted between the literals in the description of the object $S(\omega)$, then the resulting formula $S_f(\omega)$ is isomorphic to one of the disjunctive terms of the class description $A_k(\overline{x}_k)$.

So, the classification problem, in contradistinction to identification problem and analysis problem, is proved to be GI-complete.

Level description of classes

To decrease the number of steps while running an algorithm that solves the described problems, a level description of classes of recognizable objects is proposed in Kosovskaya, [2008], which is essentially a hierarchical description of classes. It is based on the extraction from the class description of sub-formulas that are isomorphic to each other and which define the generalized characteristics of objects of the same class Kosovskaya, [2014].

In particular, this can be done by extracting the formulas $P_i^1(\overline{y})$ ($i = 1, \dots, n_1$), isomorphic to the "often occurred" sub-formulas of the formulas $A_k(\overline{x}_k)$ with "small complexity". In such a case, a system of equivalences of the form $p_i^1(y_i^1) \leftrightarrow P_i^1(\overline{y}_i)$ is written, where p_i^1 are new predicates, which will be called the first-level predicates, and the variables y_i^1 are new variables for the lists of the initial variables, which will be called the first-level variables.

Denote the formulas obtained from $A_k(\overline{x}_k)$ by replacing all occurrences of subformulas isomorphic to $P_i^1(\overline{y}_i^1)$ with the atomic formulas $p_i^1(x_{i\ k}^1)$ by $A_k^1(\overline{x}_k^1)$. Here $\overline{x}_k^1$ is the list of all variables of the formula $A_k^1(\overline{x}^1)$, consisting of some (perhaps all) initial variables of the formula $A_k(\overline{x}_k)$, and of first-level variables that appear in the formula $A_k^1(\overline{x}_k^1)$. Such formulas $A_k^1(\overline{x}_k^1)$ can be considered as class descriptions in terms of predicates of the initial (zero) and the first levels.

The first-level description $S^1(\omega)$ is a union of $S(\omega)$ and the set of all atomic formulas of the form $p_i^1(\omega_{ij}^1)$ for which the defining sub-formula $P_i^1(\overline{\tau}_{ij}^1)$ is true with $\tau_{ij}^1 \subset \omega$, and the first-level object ω_{ij}^1 is a list of initial objects $\overline{\tau}_{ij}^1$.

The procedure for extracting sub-formulas that are isomorphic to "often" occurring subformulas with "small complexity" can be repeated with formulas $A_k^1(\overline{x}_k^1)$.

As a result of constructing predicates of various levels and a level class description, the original class description system can be written using the equivalent level class description system

$$\left\{\begin{array}{rcl} & A_k^L(\overline{x}_k^L) & \\ p_1^1(x_1^1) & \Leftrightarrow & P_1^1(\overline{y}_1^1) \\ & \vdots & \\ p_{n_1}^1(x_{n_1}^1) & \Leftrightarrow & P_{n_1}^1(\overline{y}_{n_1}^1) \\ & \vdots & \\ p_i^l(x_i^l) & \Leftrightarrow & P_i^l(\overline{y}_i^l) \\ & \vdots & \\ p_{n_L}^L(x_{n_L}^L) & \Leftrightarrow & P_{n_L}^L(\overline{y}_{n_L}^L) \end{array}\right. .$$

The solution of the considered recognition problems can be reduced to sequential implementation of the next operation for $l = 1, \dots, L$.

- For each $j = 1, \dots, n_l$ check the sequence from $S^{l-1}(\omega)$ the formula $\exists \overline{x}_{j\neq}^l\ P_j^l(\overline{x}_j^l)$ and find all such l-th level objects ω_j^l, whose existence is stated on the right-hand side of the logical sequence, and therefore the atomic formulas of $p_j^l(\omega_j^l)$ are true. In this case, a description of the l-th level $S^l(\omega)$ object will be obtained.

It should be noted that for each l a logical sequence of the form (4) with a shorter recording length of the right-hand side is checked. So, it is possible to clarify the concept of "small complexity".

For the exhaustive algorithm, this means a "small" number of object variables in the formulas. For algorithms based on the derivation in the predicate calculus, "small complexity" means "small" number of atomic formulas.

Note, that in spite of GI-completeness of the classification problem, the problems of idendification and of analysis remain to be NP-complete. This corresponds the facts that the problem **Isomorphism of a sub-graph** Garey, etc., [1979] is NP-complete.

Conjunctive Boolean Query problem

The following problem is presented in Garey, etc., [1979].

Conjunctive Boolean Query (CBQ)

Instance: Finite domain D, a collection $R = \{R_1, R_2, \ldots, R_n\}$ of predicates, where each R_i defines a d_i-ary relation between entries from D, a set $S(D)$ of all atomic formulas with predicates from R which are true on D, and a conjunctive Boolean query Q over R and D, where such a query Q is of the form $A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_r$ with each A_i of the form $R_j(u_1, u_2, \ldots, u_{d_j})$ where each $u \in \{y_1, y_2, \ldots, y_l\} \cup D$.

Question: Is $\exists y_1, y_2, \ldots, y_l Q$, when interpreted as a statement about R and D, true?

That is whether

$$S(D) \Rightarrow \exists y_1, y_2, \ldots, y_l(A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_r)?$$

Such setting of the problem **CBQ** is very similar to the earlier investigated in Kosovskaya, [2014] problem Satisfiability in a Finite Interpretation appeared while recognition of an object in the frameworks of logic-objective approach to the pattern recognition and the question of which is the formula (4).

Essential difference in implementation of these problems consists in the following:

— data base may be not changeable at all or have very small changes, but queries may differ every time ($S(D)$ is fixed, but the query Q often may be changed);

— while pattern recognition the set of goal formulas (description of classes) may be not changeable at all or has changes very rarely, but the recognized objects may be different every time (the set of all possible formulas $A(\overline{y})$ is fixed, but the object ω and its description $S(\omega)$ often may be changed).

While multiple solution of the CBQ problem, the researcher does not have a set of elementary conjunctions. Analogues of frequently occurred sub-formulas have to be extracted from the set of literals, and the algorithm for constructing a multi-level class description cannot be applied. When creating a database, we cannot say with certainty what queries the user may have. However, the database itself with all requests remains almost unchanged. Therefore, regularities should be sought in the base itself (the set of constant literals $S(D)$).

An algorithm for finding such regularities (sets of conjunctions of atomic formulas that are isomorphic to each other) is given in Kosovskaya, [2018a]. This algorithm has an exponential upper estimate of computational complexity of the database record length. However, this algorithm may be applied to a database formulas only once.

Metrics in the space of elementary conjunctions of predicate formulas

When solving many AI problems, the question arises of how similarly two objects under study are alike. When simulation an object using binary (or finite-valued) strings, there naturally arises a feature space of a given dimension n. The distance between the descriptions of objects $(x_1, \dots, x_n)$ and $(y_1, \dots, y_n)$ in this space can naturally be calculated using the formula $\sqrt[k]{\sum_{i=1}^{n} |x_i - y_i|^k}$. Or, if the "weights" w_i of the features are known, then by the formula $\sqrt[k]{\sum_{i=1}^{n} w_i |x_i - y_i|^k}$.

In the logic-objective approach, the descriptions of objects can have a different number of literals, and the same predicate symbol can enter the description of an object a different number of times and with different arguments.

A metric for elementary conjunctions of predicate formulas is proposed in Kosovskaya, [2012].

Let $S(\omega_1)$ and $S(\omega_2)$ be descriptions of ω_1 and ω_2, respectively. Elementary conjunctions $S_f(\omega_1)$ and $S_f(\omega_2)$ are obtained from $S(\omega_1)$ and $S(\omega_2)$ by means of writing the sign $\&$ between literals. $S_f(\omega_i)$ ($i = 1, 2$) contains a_i literals.

Let us find a maximal, over the number of arguments, elementary conjunction of C, isomorphic to the subformulas of the formulas $S_f(\omega_1)$ and $S_f(\omega_2)$. Let C contains a' literals. Denote by $\Delta a_i = a_i - a'$ the number of literals from $S_f(\omega_i)$ not included in C. Then the formula

$$\rho(\omega_1, \omega_2) = \Delta a_1 + \Delta a_2 = a_1 + a_2 - 2a'$$

defines the distance between the descriptions of $S(\omega_1)$ and $S(\omega_2)$, satisfying all properties of the distance, namely:

— non-negativity,

— symmetry,

— equality to zero if and only if the descriptions are isimorphic,

— triangle inequality, i.e. for every three objects ω_1, ω_2 and ω_3 the inequality $\rho(\omega_1, \omega_2) + \rho(\omega_2, \omega_3) \geq \rho(\omega_1, \omega_3)$ is true.

The disadvantage of such a way defined distance is that it does not reflect (illustrate) the degree of similarity of objects. For example, for objects with descriptions containing a large number of literals, if they coincide by 90%, there can be a large distance. For objects with a small number of literals, if they coincide by 10%, there may be a small distance. You can overcome of this disadvantage by normalizing the distance.

$$d(\omega_1, \omega_2) = \frac{\rho(\omega_1, \omega_2)}{a_1 + a_2}.$$

Unfortunately, this formula does not define a metric, since the triangle inequality is not satisfied. Therefore, the function d can be called the degree of similarity of the descriptions of the objects ω_1 and ω_2.

Conclusion

The paper considers various aspects of the application of the notion of isomorphism of elementary conjunctions of predicate formulas in AI problems. The term of isomorphism of various objects is widespread in mathematics. However, in the most of the author's works cited here, instead of the term "isomorphism", the term "coincidence up to the names of variables" was used.

It was the understanding of the fact that the relation under consideration is an isomorphism relation that led to the appearance of this article. Moreover, a connection between problems using this relation and the graph isomorphism problem was realized. For the last one Laslo Babai in 2017 Babai, [2017] proposed a quasi-polynomial algorithm for solving it. His estimate $2^{O((\log\ n)^3)}$ is still being verified.

Bibliography

Babai L. "Graph Isomorphism in Quasipolynomial Time" (Version 2.1 Unfinished revision of Version 2 posted on arXiv May 23, 2017). http://people.cs.uchicago.edu/∼laci/17groups/version2.1.pdf

Garey M.R., Johnson D.S., "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, New York, 1979.

Kosovskaya T. "Discrete Artificial Intelligence Problems and Number of Steps of their Solution" // International Journal on Information Theories and Applications, Vol. 18, Number 1, 2011. P. 93 ï£¡ 99.

Kossovskaya T. M. "Level descriptions of classes for decreasing step number of pattern recognition problem solving described by predicate calculus formulas" // Vestn. St. Petersburg Univ.: Seria 10, Issue. 1. P. 64 – 72 (2008). (In Russian)

Kosovskaya T. "Distance between objects described by predicate formulas" // International Book Series. Information Science and Computing. Book 25. Mathematics of Distances and Applications (Michel Deza, Michel Petitjean, Krasimir Markov (eds)), ITHEA ï£¡ Publisher, Sofia, Bulgaria, 2012. P. 153 ï£¡ 159.

Kosovskaya T. "Construction of Class Level Description for Efficient Recognition of a Complex Object" // International Journal "Information Content and Processing", Vol. 1,No 1. 2014. P. 92 – 99.

Kosovskaya T.M. "Predicate Calculus as a Tool for AI Problems Solution: Algorithms and Their Complexity", Chapter 3 in: Intelligent System. Open access peer-reviewed Edited volume. Edited by Chatchawal Wongchoosuk Kasetsart University (2018) 1 – 20. https://www.intechopen.com/books/intelligent-system/predicate-calculus-as-a-tool-for-ai-problems-solution-algorithms-and-their-complexity

Kosovskaya T.M. "Construction of a level data base for decreasing of computational complexity of the Conjunctive Boolean Query problem" // 11th Russian Multiconference on the Control Problems. Proc. of the conference "Information Technology in Management". St. Petersburg, 2018. P. 33 –38. (In Russian)

Petrov D.A. "Algorithms of extraction of a maximal common up to the names of variables predicate formulas and their implementation" // Proc. of the 9-th Conference "Information Technology in Management". St.Petersburg, 2016. P. 97 –102. (In Russian)

Russell S. and Norvig P. "Artificial Intelligence: A Modern Approach", Third edition, Prentice Hall Press Upper Saddle River, NJ, 2009.

Authors' Information

Tatiana Kosovskaya - *Dr., Professor of St.Petersburg State University, University emb., 7 – 9, St.Petersburg, 199034, Russia; e-mail: kosovtm@gmail.com*
Major Fields of Scientific Research: Logical approach to artificial intelligence problems, Theory of Computational Complexity of Algorithms.

DETECTING COMMUNITIES FROM NETWORKS: COMPARISON OF ALGORITHMS ON REAL AND SYNTHETIC NETWORKS

Karen Mkhitaryan, Josiane Mothe, Mariam Haroutunian

Abstract: Communities in real world complex networks correspond to hidden structures that are composed of nodes tightly connected among themselves and weakly connected with other nodes in the network. There are various applications of automatic community detection in computer science, medicine, machine learning, sociology, etc. In this paper, we first present the existing community detection algorithms and evaluation measures used in order to consider the algorithms effectiveness. We then report a deep comparison of the algorithms using both large scale real world complex networks and artificial networks generated from stochastic block model. We found that Louvain algorithm is consistently the best across both the measures and the networks (8 real world and many varied synthetic networks) we tested. Fast Greedy and Leading Eigenvector algorithms are also good alternatives. Moreover, compared to related work, our paper considers both more algorithms and more networks.

Keywords: Complex Networks, Community Detection, Stochastic Block Model, Evaluation Measures.

Introduction

Network science gets more and more attention with the increase of computing power enabling to challenge more complex problems and with the rapid increase in the amount and types of *related* data [Fortunato, 2010]. Network science aims at studying complex relational data and networks both theoretically and on the application point of view with real world networks.

Real world networks are represented as undirected, directed or weighted graphs, composed of nodes and edges where edges connect the nodes. Real networks and synthetic random graphs differ in their structure. In random graphs the distribution of the node degree is homogeneous which means the probability of having an edge between any two nodes is the same. On the other hand, in real world complex networks the structure is usually heterogeneous which results in groups of nodes that are tightly connected to each other but weakly connected with nodes from other groups.

This property of real world complex networks is known as *community structure.* Communities (or clusters) are defined as groups of nodes that have high inter connectivity (strong links inside the groups) and weak inter-connectivity (weak link between groups) [Fortunato, 2010]. For example in Digital Bibliography and Library Project (DBLP) collaboration network which will be used later, nodes represent the authors and edges represent the connections between authors i.e. published a paper together. Communities in DBLP network can be interpreted as groups of authors with the same research interest. Synthetic networks can also mimic the complexity of real world ones.

The aim of community detection is to identify the groups with high intra-connectivity by analyzing the information that the graph topology encodes.

There are a lot of applications of community extraction in networks in various domains such as in biology (e.g. protein to protein interactions [Mahmoud et al., 2014]), sociology (e.g. social network analysis), [Girvan and Newman, 2001, Bedi and Sharma, 2016, He et al., 2018]), information retrieval (e.g. recommendation systems) [Parimi and Caragea, 2014]), etc.

There is no globally accepted definition of *community* but various definitions are used in the literature [Fortunato, 2010, Hu et al., 2008, Radicchi et al., 2004]. In [Fortunato, 2010], Fortunato defined community using local and global definitions as well as based on vertex similarity. Local definitions consider a community as a separate autonomous entity while in global definitions communities are important parts of a whole graph and considering them separately affects the functionality of the system. In definitions that are based on vertex similarity, a similarity measure is defined for each pair of nodes (e.g., Manhattan distance, cosine similarity, etc.) and they are joined based on the chosen similarity measure [Fortunato, 2010]. Radicchi et al. defined community in a strong and weak sense using degree of the nodes [Radicchi et al., 2004]. There are also community scoring functions used to quantitatively assess how community-like are the groups of nodes (e.g., conductance, modularity, global clustering coefficient) [Leskovec and Yang, 2015, Newman and Girvan, 2004] that we will detail later in section 3.

Modern real world networks like Facebook and DBLP may contain millions or billions of nodes and edges which obscure the process of community detection due to computational issues. However there are well designed approximation algorithms with low complexity fitting large scale networks [Fortunato, 2010, Lancichinetti and Fortunato, 2011], which we evaluate and compare later in this paper (see section 5). Since the number of easily and openly accessible large real world networks is limited, benchmark models such as Lancichinetti Fortunato Radicchi (LFR) benchmark [Lancichinetti et al., 2008] and the Stochastic Block Model (SBM) [Abbe, 2018] which generate networks with community structure are used to overcome this issue. Although the use of artificial networks is beneficiary to generate random networks with different parameters and properties (e.g., degree distribution, number of communities, number of vertices inside communities etc.), they are far from being real world networks as the structure of real networks can be unpredicted resulting in different topologies.

In the literature of the domain, algorithms have been mostly evaluated and compared either on artificial networks [Lancichinetti and Fortunato, 2011, Lancichinetti et al., 2008, Orman and Labatut, 2009, Yang et al., 2017] or small real world networks [de Sousa and Liang, 2014, Moradi et al., 2012]. These settings are not enough to make strong conclusions. Since there are many reasons why the results may differ from network to network, tests and comparisons should be made on many different networks of various types and properties to make acceptable conclusions. Therefore in our research we included both artificial networks and real networks (see section 4). We generated artificial networks using stochastic block model [Abbe, 2018]. As for real networks, we used SNAP datasets where number of nodes range from 1589 to 334,863 [Leskovec and Krevl, 2014].

The evaluation of a community detection algorithms is a crucial research problem. It implies to consider one or several networks, to apply the algorithm and to analyze how "good" or "bad" the detected communities are. This can be done by comparing the estimated community structure with a reference structure or ground truth using external measures or by assessing the quality of detected communities internally. In this paper, we present the main measures from the literature (See Section 3). We also use them in order to compare popular community detection algorithms (See Section 5).

The aim of this paper is to provide the reader with an overview of the methods and algorithms that have been developed for community detection and to compare them using different measures on different types of networks, both real world and synthetic. We did not study specific cases of community detection and some specific algorithm behavior. A few studies could be mentioned on this topic. For example, Abbe studied the limits for community detection in the SBM, both with respect to information-theoretic and computational thresholds [Abbe, 2018]. Overlapping community detection is also an important topic. In [Xie et al., 2013], Xie et al. considered fourteen algorithms for their ability to detect overlapping nodes; which is a different problem although related to. The authors first analyzed community detection ability using NMI and Omega index. Omega index measures the number of node pairs are together in no clusters, the number of pairs that are together in one cluster only, in two clusters, and so on.

The rest of the paper is organized as follows: In Section 2 we describe the community detection algorithms from the literature. The two next sections present the evaluation framework with regards to measures (Section 3) and benchmark networks (Section 4) including artificial networks generated automatically and real networks. Section 5 presents the evaluation of the algorithms using various measures and considering the different networks. Section 6 presents the related work. Section 7 draws some conclusions.

Community Detection Algorithms

Various community detection algorithms have been developed which differ in terms of complexity and network types they target (e.g., undirected, directed, weighted, etc.). In this paper, we chose to present them briefly and to provide the link to the papers where details of the algorithms and implementations can be found for the interested reader. Heuristics or approximation algorithms are used to optimize some given objective function (e.g. modularity) to detect communities. Despite these barriers, various algorithms exist in the literature, including those that were initially developed for cluster analysis. These algorithms fall mainly into the following main categories [Fortunato, 2010]: modularity-based algorithms, spectral algorithms, algorithms based on random walks, label propagation and information-theoretical measures that we detail in the next sub-sections.

Algorithms based on modularity optimization. Modularity is a community scoring function which characterizes how community-like are the groups of nodes in the network. Modularity is defined as :

$$\text{Modularity} = \frac{1}{2M}\sum_{x,y}(A_{x,y} - \frac{k_x k_y}{2M})\delta(c_x, c_y) \tag{1}$$

where x and y are nodes, M is the number of edges in the network, k_x and k_y the degrees of x and y respectively and $\delta(c_x, c_y)$ is 1 when x and y fall in the same community and 0 otherwise.

Algorithms based on modularity optimization such as Newman's greedy algorithm [Newman and Girvan, 2004] and its updated version by Clauset et. al, namely *Fast Greedy* [Clauset et al., 2004] join vertices which result in highest increase in modularity. After an iterative process when modularity cannot be maximized anymore, the network is partitioned into communities. Another popular modularity optimization method is *Louvain's* algorithm which initially finds small communities by optimizing modularity locally and then aggregates nodes belonging to the same community and creates a network where nodes represent the communities. This process is iterated until maximum modularity is reached and a hierarchy of communities is produced [Blondel et al., 2008].

Infomap. *Infomap* is a flow based information theoretic method used to reveal community structure in the networks. At the beginning every node is assigned to its own community. Then nodes are moved to neighboring communities that results in the largest decrease of the map equation which is the theoretical limit to describe the path of a random walker on the network for a given partition.

After an iterative process, when map equation is minimized over all possible network partitions and no move results in decrease of it, network splits into communities [Rosvall and Bergstrom, 2007].

Random Walks. In general, communities in networks have more intra connectivity (inside communities) than inter connectivity (between communities). Thus it is expected to have more edges inside those groups than between them. When implementing a short random walk, the probability that both the starting and ending points will be in the same group rather than in different groups is thus higher. Algorithms based on random walks like *Walktrap* [Pons and Latapy, 2005] use this idea to detect communities in networks.

Algorithm based on eigenvectors of modularity matrix. This algorithm proposed by Newman, namely *Leading Eigenvector* [Newman, 2006] uses eigenspectrum of modularity matrix. This algorithm initially creates the modularity matrix $M = A - P$, where A is the adjacency matrix of the network and P is the probability matrix, where $P[i, j]$ is the probability that there is an edge between vertices i and j in a random network which has the same degree distribution as the original network. Then the method takes the eigenvector of the modularity matrix for the largest positive eigenvalue and partitions the network into communities considering the sign of the corresponding element in the eigenvector.

Label Propagation. Unlike other community detection algorithms, *label propagation* does not optimize any given objective function and it does not require to have a priori information about the network structure. Initially every node has its own label (corresponding to its community) and during an iterative process nodes gain the label which is frequent in their neighborhood. When every node has the label that the maximum number of its neighbors have, the algorithm stops, resulting in densely connected groups.

Spinglass. *Spinglass* is an approach from statistical mechanics introduced by Reichardt and Bornholdt [Reichardt and Bornholdt, 200 which is based on the Potts model (model of interacting spins). In the network each vertex can be in one spin state. The number of total spin states equals the number of communities in the network. The edges of the network identify the pairs of vertices that state in the same spin state and vice versa. At the end, after running the model for a number of steps, spin states represent the community assignments.

Girvan-Newman Algorithm. This algorithm proposed by Girvan and Newman [Girvan and Newman, 2001] uses the concept of betweenness centrality and generalizes it to define *Edge betweenness* centrality which separates tightly connected vertices. Betweenness centrality of a vertex v is defined as:

$$B(v) = \sum_{p \neq v \neq q} \frac{S_{p,q}(v)}{S_{p,q}} \tag{2}$$

where $S_{p,g}$ is the total number of shortest paths from vertex p to vertex q and $S_{p,g}(v)$ is the number of shortest paths from vertex p to vertex q that pass through node v. The first step of the algorithm is to calculate betweenness of all edges, then remove the edge with highest betweenness score and recalculate betweenness of all edges again. After an iterative process, when no edges remain in the network, result can be shown as a dendrogram. According to dendrogram and modularity score, the proper partition is chosen.

Table 1: Algorithms - the underlying theory used, computing complexity where N is the number of nodes and E is the number of edges in the network and main reference.

Algorithm	Based on	Complexity	Reference
Fast Greedy	Modularity	$O(NElog2E)$	[Clauset et al., 2004]
Louvain	Modularity	$O(E)$	[Blondel et al., 2008]
Infomap	Information theory	$O(N(N + E))$	[Rosvall, Bergstrom, 2007]
Walktrap	Random walk	$O(N^2E)$	[Pons, Latapy, 2005]
Leading Eigenvector	Eigen vectors	$O(N(N + E))$	[Newman, 2006]
Label Propagation	Neighborhood	$O(N + E)$	[Raghavan et al., 2007]
Spinglass	Spin states	$O(N^{3:2})$	[Reichardt, Bornholdt, 2006]
Edge betweenness	Centrality	$O(NE^2)$	[Girvan and Newman, 2001]

Table 1 reports the main algorithms to detect communities along with the underlying theory and their complexity in terms of calculation depending on the number of nodes *N* and edges *E.* We also provide in this table the main reference where the reader can find the details of the eight algorithms.

Measures to Compare Algorithms

Detecting communities in large networks can sometimes be much complicated due to computational complexity of algorithms specifically for networks with large number of nodes and edges. Indeed, exact detection can be a NP-hard problem (see Table 1). Distinguishing between algorithms and deciding which algorithm works the best on a particular network is not obvious.

Algorithms can be compared in terms of efficiency and effectiveness. In the case of community detection algorithms, in addition to algorithm complexity (see Table 1), efficiency refers to the time taken to partition the network while effectiveness is a qualitative measure describing how "good" the derived communities are.

Measures used to assess the quality of detected communities are in turn divided into two main categories [Halkidi et al., 2002, Chakraborty et al., 2017] as described below.

Internal measures

Internal measures are used to quantitatively assess how community-like is the given set of nodes in the network. As the global definition of community is based on the idea that it has high connectivity within a group and weak connectivity with other groups, scoring functions are also based on this intuition. In the following, we will focus on modularity, conductance, and global clustering coefficient measures. Conductance give optimal results when identifying ground truth communities [Leskovec and Yang, 2015] and modularity is the most widespread evaluation criteria used in the literature.

Modularity. Modularity is a well accepted measure to evaluate a partition of a network into communities that is to say to assess the quality of the network division into communities. The basic idea is that random networks do not contain community structures thus its structure can be compared to a give network to have an idea of the network community-like structure. Modularity is then defined as "the fraction of the edges that fall within the given groups minus the expected fraction if edges were distributed at random" [da Costa Alves, 2010]

$$\text{Modularity} = \frac{1}{2M}\sum_{xy}(A_{xy} - \frac{k_x k_y}{2M})\delta(c_x c_y). \tag{3}$$

where M is the number of edges in the network, kx and ky the degrees of nodes. x and y respectively and $\delta(c_x c_y)$ is 1 when x and y fall in the same community and 0 otherwise.

Partitions of the network with strong community structure where vertices are densely connected inside communities and sparsely connected between communities have higher modularity score [Newman and Girvan, 2004].

Experiments showed that modularity suffers from resolution limit merging small groups in case of low resolution and splitting large groups in case of high resolution i.e. missing important structures in the network [Lancichinetti and Fortunato, 2011] and often it is not possible to eliminate both biases simultaneously.

Conductance. Conductance measures how strong or dense the community structure of the graph is. Conductance is the fraction of the total edges that goes outside the community and is defined as:

$$Conductance = \frac{O_c}{2I_c + O_c} \tag{4}$$

where O_C is the number of edges pointing outside from the community c and I_c is the number of edges within c.

Using conductance as a community goodness metric Leskovec et al. showed that the best possible communities get less community-like structure when network size grows [Leskovec et al., 2008]. In their other study while experimenting on large networks from 0.33 million up to 117.7 million nodes, conductance and triangle participation ratio gave best results in identifying ground truth communities [Leskovec and Yang, 2015].

Global clustering coefficient. A clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together [Deng et al., 2016]. The global clustering coefficient is designed to give an overall indication of the clustering in the network. Global clustering coefficient is defined as the number of triangles in the network divided by the number of connected triplets (subgraphs having three vertices and two edges) as follows [Wasserman and Faust, 1994]:

$$T = \frac{3 \times \text{number of triangles}}{\text{number of connected triplets of vertices}} \tag{5}$$

In real world networks where the degree distribution is highly heterogeneous, global clustering coefficient shows how much the vertices tend to cluster together to form dense groups [Wasserman and Faust, 1994].

External measures

Unlike internal measures which are used to assess the quality of detected communities, external measures are used to evaluate and compare a given partition X with another or with ground truth if available, denoted Y in what follows.

Normalized Mutual Information Mutual Information (MI) is an information-theoretic measure that quantifies the mutual dependence between two random variables. It is applied in community detection to compare two partitions. MI measures how much information can be obtained about one partition (random variable) from another one.

Normalized Mutual Information (NMI) is the normalized variant of MI [Collignon et al., 1995]. NMI is defined as :

$$\text{NMI}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)} \tag{6}$$

where $H(X)$ and $H(Y)$ are the entropies of random variables corresponding to two partitions X and Y respectively. Thus NMI(X, Y) measures the similarity of the two partitions [Cover and Thomas, 2006].

Variation of information Variation of information is also an information-theoretic measure of the distance between two random variables similar to MI and NMI. However unlike these measures, it is a true metric measure defined as:

$$\mathrm{VI}(X,Y) = H(X|Y) + H(Y|X) \quad (7)$$

Considering X and Y as two different partitions of the network, VI(X, Y) measures the variation of information between the two partitions [Meila, 2007].

Adjusted Rand Index Adjusted Rand Index (ARI) is another similarity measure of two different partitions. Given a set of n elements, $S = (d_1, d_2, ..., d_n)$, and two partitions of S, X and Y respectively, ARI is defined as:

$$\mathrm{ARI} = \frac{SS + DD}{SS + SD + DS + DD} \quad (8)$$

where SS is the number of pairs of elements in S that are in the same subset in X and in the same subset in Y, DD is the number of pairs of elements in S that are in different subsets in X and in different subsets in Y, SD is the number of pairs of elements in S that are in the same subset in X and in different subsets in Y, DS is the number of pairs of elements in S that are in different subsets in X and in the same subset in Y [Rand, 1971, Hubert and Arabie, 1985].

In the evaluation part of this paper we use conductance, global clustering coefficient and modularity to assess the quality of detected communities internally and NMI, VI and ARI to compare partitions externally. We will also measure efficiency considering the processing time of the algorithms in various configurations.

Networks Used for the Evaluation of the Algorithm

In our experiments we use eight real world networks (See Table 2) provided from SNAP Stanford database (http://snap.stanford.edu/data/index.html). However even if large networks are available, the number of networks with pre-known community structure or ground truth is limited. This limitation is over-passed by generating networks similar to real networks using the LFR benchmark and the SBM. In this paper, we consider both real networks and simulated networks in order to evaluate the algorithms on the same basis.

While real-world networks correspond to real cases from observed links or interactions, generally there is no ground truth associated to them with regard to underlying communities. Synthetic networks offer this possibility and can be built for use as benchmarks [Barrett et al., 2009].

Table 2: Eight real world networks statistics - SNAP Stanford database.

Network	Number of Nodes	Number of Edges	Average Degree	Average Transitivity	Diameter
Net Science	1,589	2,742	3.45	0.878	17
Facebook	4,039	88,234	43.69	0.617	8
Power Grid	4,941	6,594	2.67	0.106	46
Astrophysics	16,706	121,251	14.52	0.726	14
Enron Email	36,692	183,831	10.02	0.715	13
Condensed Matter	40,421	175,692	8.69	0.718	18
DBLP	317,080	1,049,866	6.62	0.732	23
Amazon	334,863	925,872	5.53	0.429	47

Real world networks

One interesting real network collection is the SNAP Stanford database which contains eight real world networks where the number of vertices, edges, Average degree[1], global clustering coefficient and diameter[2] are reported in

[1]The *degree* of a node is the number of edges connected to it. The average degree is computed as $\frac{2E}{N}$)

[2]The *diameter* of a network is the longest of all the calculated shortest paths in a network. In other words, it is the shortest distance between the two most distant nodes in the network.

Table 2. *Facebook* dataset consists of Facebook profiles and friendships circles, *Power Grid* represents the topology of the western states power grid of the United States, *Enron Email* represents an email communication network, *Amazon* data set represents co-purchasing network of products. We also use the four co-authorship networks as follows: *Net Science, Astrophysics, Condensed Matter* and *DBLP,* representing the networks of co-authorship between scientists in science of networks, astrophysics e-print archive, condensed matter e-print archive and in computer science respectively.

From Table 2 we can see that the networks are quite varied. The number of nodes varies from a few thousands to a few hundred thousands while the number of edges varies from a few thousands to about one million. The average degree is also varied (from 2.66 for Power Grid to 43.69 for Facebook). Finally, the diameter of the networks varies from 8 (Facebook) to 47 (Amazon).

Synthetic networks

One of the earlier synthetic based networks is the Girvan-Newman (GN) benchmark [Girvan and Newman, 2002]. It makes possible to define equal size communities with a given degree and a given ratio between internal and outgoing connections. However, the resulting networks are quite homogeneous (equal size communities, same expected degree), the number of nodes is very limited and communities do not overlap. As a result, most of the detection algorithms works well on GN and it is not possible to GN to reveal the specific limitations of a given algorithm. GN is thus considered as providing non realistic networks [Yang et al., 2017]. LFR (Lancichinetti, Fortunato, Radicchi) benchmark [Lancichinetti et al., 2008] and SBM [Karrerand Newman, 2011] overtake these problems and for this reason are preferred to generate test bed and benchmarks to evaluate community detection methods.

LFR benchmark (https://sites.google.com/site/santofortunato/inthepress2) generalizes GN by using power-law distributions of degree and community size in the graph generation. LFR generates networks based on a set of parameters the user specifies. The resulting networks are with pre-known community structure where network size, node degree range and community size distributions are heterogeneous and power-law. Mixing parameter μ is used to control the fraction of nodes a node shares inside and outside the community [Lancichinetti et al., 2008]. More specifically, for a given node, μ is defined as the fraction of the number of edges connecting it to other nodes belonging to different communities by the total degree of that node. In LFR, the mixing parameter μ is the same for any of the nodes. μ has been shown to be the most influential parameter and thus it makes sense to make it varying when comparing various algorithms [Orman and Labatut, 2009]; moreover, to keep the notion of communities, one should consider $\mu < 0.5$.

SBM (http://igraph.org/r/doc/sample_sbm) is a generative model for random graphs, generating networks with community structure where it is possible to choose the number of nodes in the network, the size of each community, and the community structure considering a CxC Bernoulli matrix, where C is the number of communities and Cij is the probability of an edge between nodes from community i and community j [Abbe, 2018]. The LFR benchmark is a special version of the stochastic block model [Fortunato and Hric, 2016]. SBM is considered as the simplest model of a graph with communities [Abbe, 2018].

LFR and SBM can thus generate an unlimited number of networks with predefined community structure simulating real world networks. In Section 5, when evaluating and comparing the algorithms on synthetic networks we considered SBM. Indeed, SBM is generally preferred as a test bed for algorithms [Goldenberg et al., 2010, McDaid and Hurley, 2010, Abbe and Sandon, 2015, Zhang et al., 2016, Caltagirone et al., 2017, Gulikers et al., 2017]. Synthetic networks and real networks complement well each other. Synthetic networks correspond to insightful models for which ground truth is known by construction, if not fully realistic. For example, *Orman et al.* report that the "distribution of links is not always appropriate" and "the small communities are too dense and clique-like" [Orman and Labatut, 2009].

Comparison of the Algorithms

In this section we evaluate and compare community detection algorithms surveyed in Section 2 on both large real world networks and artificially generated ones using SBM. The same "igraph" package was used to assess the communities by internal and external measures. Calculation of conductance was implemented by us.

Results on real networks

We applied the eight state of the art community detection algorithms presented in Section 2 on the eight real world networks as presented in Section 4. Modularity, global clustering coefficient and conductance were used to assess the quality of the detected communities.

Modularity and number of detected communities

Table 3 reports the modularity scores of the partition while Table 4 reports the number of communities detected across real networks and algorithms. The reason why some columns in Table 3 and 4 have NA values is that the Spinglass algorithm cannot work with unconnected graphs and Edge betweenness algorithm was computationally extensive for few networks due to its high complexity (See Table 1).

Table 3: **Modularity** of the network when partitioned by each algorithm on real networks. For each network the value in bold highlights the best algorithm. The larger the value, the more the community structure is revealed.

Algorithm	Net Science	Facebook	Power Grid	Astro physics	Enron Email	Condensed Matter	DBLP	Amazon
Fast Greedy	0.955	0.777	**0.933**	0.633	N/A	0.632	0.735	0.880
Louvain	**0.960**	0.835	0.926	**0.727**	**0.605**	**0.722**	**0.820**	**0.926**
Infomap	0.929	0.810	0.817	0.658	0.130	0.631	0.722	0.825
Walktrap	0.957	0.812	0.831	0.636	0.512	0.599	0.672	0.849
Leading Eigenvector	0.951	0.799	0.825	0.595	0.439	0.359	0.024	0
Label Propagation	0.907	0.821	0.806	0.550	0.340	0.425	0.677	0.784
Spinglass	N/A	**0.835**	0.916	N/A	N/A	N/A	0.799	0.863
Edge betweenness	0.958	N/A	0.933	N/A	N/A	N/A	N/A	N/A

Table 4: **Number of communities** detected by each algorithm on real networks.

Algorithm	Net Science	Facebook	Power Grid	Astro physics	Enron Email	Condensed Matter	DBLP	Amazon
Fast Greedy	403	13	41	1,168	N/A	2,253	3,077	1,480
Louvain	406	17	40	1,080	1,319	1,876	275	249
Infomap	442	93	492	1,860	10,291	3,816	17,023	17,286
Walktrap	416	77	364	2,673	3,066	5,252	30,425	14,905
Leading Eigenvector	404	18	35	1,067	1,068	1,801	2	1
Label Propagation	456	53	481	1,506	1,989	3,381	21,040	22,532
Spinglass	N/A	23	25	N/A	N/A	N/A	25	25
Edge betweenness	405	N/A	45	N/A	N/A	N/A	N/A	N/A

Modularity reported in Table 3 varies from 0.024 (Leading Eigenvector on DBLP) to 0.960 (Louvain on Net Science). The various algorithms obtain similar and high modularity (from 0.907 to 0.960) on Net Science network which is the smallest network in terms of number of nodes and edges and one of the networks that have small average degree. Results are also high and similar (0.806 to 0.933) for the algorithms on Power Grid network which is quite similar to Net Science in terms of number of nodes, edges and average degree.

Larger networks such as DBLP and Amazon, which are also not very dense, obtain higher modularity for most of the algorithms. The lowest modularity values occurs on Enron Email, Astrophysics and Condensed Matter networks for which average degree is between 8 and 14, there are a few ten thousand nodes and a few hundred thousand edges.

From the Table 3 we can see that partitions obtained by Louvain have consistently high modularity scores across the networks, indicating that the network partitions are more community-like. Fast Greedy, Infomap and Walktrap algorithms also have high modularity scores.

The algorithms also differ in terms of the number of communities being detected. Table 4 reports this number across collections and algorithms. Fast greedy, Louvain and Leading eigenvector methods detect small number of communities while Infomap, Walktrap and Label propagation detect more. On the smallest network (Net Science) all the algorithms detect about the same number of communities but not on Power Grid. While the two networks are similar in terms of number of edges, nodes and degree, they differ in terms of diameter (17 for Net Science and 46 for Power Grid). On Power Grid network that has a longer shortest distance between the two most distant nodes than Net Science, Infomap, Walktrap and Label propagation algorithms detect a few hundred communities while the other algorithms detect rather a few tens. This result is not surprising considering the propagation methods used in these three algorithms. On Amazon and DBLP networks which also have a large diameter, the number of communities detected by the algorithms also varies a lot with Infomap, Walktrap and Label propagation, consistently detecting much more communities than the other algorithms. Although the variation is much higher when the diameter is larger, these three algorithms tend to detect more communities than the other algorithms whatever the networks are.

Conductance and global clustering coefficient

In this sub-section, we analyze the results obtained for the two other internal measures. The results are presented in two tables which present the percentage of detected communities for which the global clustering coefficient (T) (Table 5) and conductance (C) (Table 6) satisfy the inequality given in the left part of the tables. Values in bold correspond to the highest percentage of communities in particular intervals. We consider four intervals for the values of C and T in order to make categories related to the quality of the detected communities.

The higher the global clustering coefficient T and the lower the conductance C, the better the community structure. So when $C \in$ (0; 0.01) and/or $T \in$ (0.75; 1), the community structure is good while for $C \in$ (0.1; 1) and/or $T \in$ (0; 0.25), the community structure is weak. Tables 5 and 6 are ordered from the best community structure to the weakest according to the measure used. In these tables, we did not report Spinglass nor Edge betweenness because they had too many NA values.

For example, we can read in Table 5 that on Net Science network, Fast Greedy algorithm gets 69.9% of the communities it detects having a global clustering coefficient T higher than 0.75,18.4% of the detected communities with clustering coefficient in between 0.5 and 0.75, 5.2% in between 0.25 and 0.5 and 6.3% lower than 0.25. Fast Greedy is a good algorithm when considering global clustering coefficient and Net Science network. Reversely, on Amazon, Leading Eigenvector gets 100% of the communities it detects with global clustering coefficient lower than 0.25; this algorithm fails detecting the communities on Amazon network.

The algorithms all provide good structure communities with low conductance and high global clustering coefficient on Net Science network (the smallest network). Reversely, most of them fail on Power Grid (high conductance, low global clustering coefficient). Power Grid is the only network that has a very low average transitivity; although we cannot make a clear correlation with this feature, it might be a reason.

In Table 6, we also can see that Fast Greedy, Louvain, and Leading EigenVector algorithms get low conductance for most of the communities they detected and this across most of the real networks we used. These algorithms also have the highest global clustering coefficient (see Table 6). These algorithms are thus the ones that are considered the best with regard to conductance and global clustering coefficient.

When considering global clustering coefficient only, the next best algorithms are Walktrap and Label propagation (but they are in the worth when considering conductance). On the other hand, Infomap, Walktrap and label propagation have the worth (high) conductance and Infomap is also in the worth category when considering global clustering coefficient. When $T \in$ (0.75; 1) we have a good community structure while for $T \in$ (0;0.25) community structure is weak.

As we can see in Table 5 and Table 6 based on global clustering coefficient and conductance, Louvain, Fast greedy, Leading eigenvector and Label propagation algorithms were able to detect mostly good communities (i.e. highest global clustering coefficient and lowest conductance).

Table 5: Percentages of communities for which the **conductance** score is in the respective interval on real networks. The top part of the table indicates good community structure according to conductance, while the conductance is lower as we go on the bottom of the table. Low conductance (top part of the table) shows a good community structure while high conductance reflects a weak community structure (bottom part of the table).

C	Algorithm	Net Science	Facebook	Power Grid	Astro physics	Enron Email	Condensed Matter	DBLP	Amazon
$C \in [0; 0.01)$	Fast Greedy	**97.4**	23	7.3	**72.6**	NA	**67.6**	0.2	7.3
	Louvain	**96.4**	23.5	10	**88**	**80.8**	**92.4**	1.8	24.8
	Infomap	**83.4**	0	0	30.4	8.5	32	0.06	0.5
	Walktrap	**91.6**	0	0.2	17.2	33.4	20.4	0.02	0.5
	Leading Eig.	**97.1**	16.6	2.8	**90.4**	**99.6**	**99.5**	**50**	**100**
	Label Prop.	**77.2**	0	0	32.6	**52.6**	27.4	0.02	0.2
$C \in [0.01; 0.02)$	Fast Greedy	0	0	7.3	0.3	NA	0.2	1.3	10.9
	Louvain	0	0	10	0.4	0.3	0	4.7	11.6
	Infomap	0	0	0	0.3	0.01	0.2	0.4	1.8
	Walktrap	0.3	0	0	0.1	0.1	0.04	0.09	1.8
	Leading Eig.	0	5.5	2.8	0.2	0	0	0	0
	Label Prop.	0.3	0	0	0.5	0.1	0.1	0.1	0.9
$C \in [0.02; 0.1)$	Fast Greedy	2.5	**61.5**	**82.9**	6.2	NA	5.4	33.4	59.1
	Louvain	3.5	**58.8**	**80**	3.5	9.7	2.2	**50.9**	**59.4**
	Infomap	6.6	17.7	12	4.4	0.03	5.2	14	25.4
	Walktrap	6.5	18.1	16.2	1.6	4.5	3	5.2	28.7
	Leading Eig.	2.1	**38.8**	**57.1**	0.9	0	0.1	0	0
	Label Prop.	4.8	28.8	8.1	3.7	7.5	3.9	7.6	15.9
$C \in [0.1; 1]$	Fast Greedy	0	15.3	2.4	20.6	NA	26.6	**64.9**	22.5
	Louvain	0	17.6	0	7.8	9	5.3	42.5	4
	Infomap	9.8	**82.2**	**87.9**	**64.7**	**91.4**	**62.4**	**85.4**	**72**
	Walktrap	1.3	**81.8**	**83.5**	**81**	**61.8**	**76.4**	**94.5**	**68.9**
	Leading Eig.	0.7	**38.8**	37.1	8.3	0.3	0.3	**50**	0
	Label Prop.	17.5	**71.1**	**91.8**	**63**	39.6	**68.5**	**92.1**	**82.8**

Table 6: Percentage of communities for which the global clustering coefficient is in the respective interval on real networks. High global clustering coefficient shows a good community structure (top part of the table) while low global clustering coefficient reflects a weak community structure (bottom part of the figure).

T	Algorithm	Net Science	Facebook	Power Grid	Astro physics	Enron Email	Condensed Matter	DBLP	Amazon
$T \in [0.75; 1]$	Fast Greedy	**69.9**	38.4	0	**63.7**	NA	**57.2**	**36.9**	18.4
	Louvain	**69.3**	29.4	0	**67**	**57**	**61.8**	19.2	0.4
	Infomap	65	**40.6**	0.4	**39.6**	23	29.4	17	9.8
	Walktrap	**67.2**	**40.5**	0.5	**58**	**59.9**	**46.6**	**34.8**	17.9
	Leading Eig.	**69.5**	23.5	0	**68.7**	**61.2**	**68.5**	0	0
	Label Prop.	**69.3**	**43.1**	3	**68.2**	**66.7**	**55.9**	**43.4**	20.4
$T \in [0.5; 0.75)$	Fast Greedy	18.4	15.3	0	19.4	NA	23	31.7	36.7
	Louvain	19.3	**35.2**	0	20.1	15.3	17.3	29.8	12.8
	Infomap	**23.1**	27.9	2	33.9	2.8	**35**	30.5	27
	Walktrap	21.5	31.8	1.6	20.4	14.2	25.2	26.1	30.8
	Label Prop.	20.2	39.6	3.8	19.6	14.6	24.6	26.4	31.8
	Leading Eig.	18.9	23.5	2.8	18.1	7.9	15.3	**50**	0
$T \in [0.25; 0.5)$	Fast Greedy	5.2	**46.1**	9.7	7.5	NA	9.3	20.9	**36.8**
	Louvain	5.1	**35.2**	5	4.6	5.2	10.1	**48**	**68.2**
	Infomap	6.6	22	10	20.1	1	29.1	**39**	**45.3**
	Walktrap	5.3	21.7	6.7	8.9	6.2	15.3	20.3	**33**
	Leading Eig.	5.1	**52.9**	8.5	4.5	1.1	4.4	**50**	0
	Label Prop.	4.9	17.2	10.1	6.9	4.5	13.4	20.9	**35.7**
$T \in [0; 0.25)$	Fast Greedy	6.3	0	**90.2**	9.2	NA	10.3	10.2	7.9
	Louvain	6.2	0	**95**	8.1	22.2	10.6	2.9	18.4
	Infomap	5.1	9.3	**87.4**	6.2	**73**	6.3	13.4	17.7
	Walktrap	5.9	5.7	**90.9**	12.4	19.5	12.8	18.5	18.1
	Leading Eig.	6.3	0	**88.5**	8.6	29.6	11.6	0	**100**
	Label Prop.	5.4	0	**83**	5	14	5.9	9.1	11.9

External Measures

Finally, the external measures NMI, VI and ARI were used to compare partitions obtained by different algorithms. We calculated the values for the partitions obtained by each pair of algorithms. In Table 7, we report the highest values only.

Pairs of algorithms which partitions resulted in the highest NMI, VI and ARI scores for each network are reported in Table 7. When considering NMI (first row in the Table), we can see that on 4 networks, namely Astrophysics networks, Enron email, Condensed matter, and DBLP, the partitions obtained by Infomap and Walktrap algorithms are the most similar. Infomap and Label propagation obtain similar results on two other networks (Power Grid and Amazon networks). On 5 over 8 networks, the same couples of algorithms give the highest NMI and VI, this is on Net science, Facebook, Astrophysics, Condensed matter and Amazon networks. Similar results indicate the analogy of NMI and VI.

Table 7: Pairs of algorithms for which the score is the highest for a given measure and a network - Using real networks and the three measures: normalized mutual information, variation of information and adjusted rand index.

External Measure	Net Science	Facebook	Power Grid	Astrophysics	Enron Email	Condensed Matter	DBLP	Amazon
NMI	0.992 Louvain & Fast Greedy	0.937 Louvain & Spinglass	0.903 Infomap & Label prop.	0.848 Infomap & Walktrap	0.743 Infomap & Walktrap	0.835 Infomap & Walktrap	0.855 Infomap & Walktrap	0.942 Louvain & Label prop.
VI	0.083 Louvain & Fast Greedy	0.344 Louvain & Spinglass	0.997 Louvain & Fast Greedy	1.965 Infomap & Walktrap	2.298 Louvain & Label prop.	2.361 Infomap& Walktrap	1.789 Infomap& Label prop.	1.102 Infomap& Label prop.
ARI	0.898 Louvain & Edge betweenness	0.914 Louvain & Spinglass	0.680 Louvain & Fast Greedy	0.151 Walktrap & Fast Greedy	0.238 Louvain & Walktrap	0.216 Walktrap & Fast Greedy	0.238 Fast Greedy& Label prop.	0.587 Infomap & Label prop.

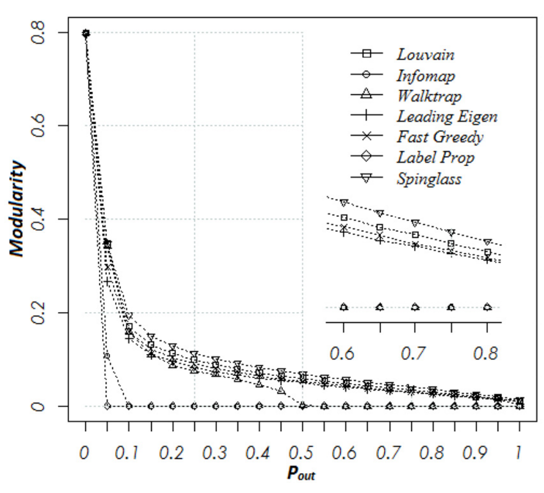
Results on artificial networks

In this section, we used SBM to generate random networks with community structure, where the number of communities, community sizes and probabilities of edges between communities and inside communities are known *a priori.*

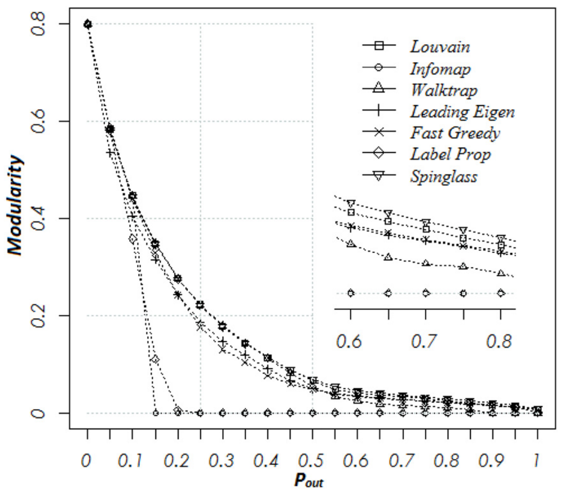
Using such networks help in analyzing the impact of the network properties on the results of algorithms or vice versa. In our experiments we generated more than 400 SBMs, where the number of nodes is 200 and they are divided into 5 equally sized communities. We did our experiments for P_{in} = 0.25 (weak connectivity inside communities) and P_{in} = 0.75 (strong connectivity inside communities).

Modularity

Fig. 1 (a) displays the modularity scores of the partitions based on the probability of edges between communities (P_{out}) for P_{in} = 0.25 (weak connectivity inside communities) while Fig. 1 (b) is for P_{in} = 0.75 (strong connectivity inside communities).



(a) For $P_{in} = 0{:}25$ (low connectivity inside communities).



(b) For $P_{in} = 0{:}75$ (strong connectivity inside communities).

Figure 1: Probability of edges between communities (P_{out}) versus modularity for a fixed P_{in}. A zoom on curves when $P_{out} \in [0{:}6;\ 0{:}8]$ is provided in the right bottom part.

In both Figures (Fig.1 (a), Fig. 1 (b)) for each P_{out} = 0.05n, where $n \in$ {0,1,20}, we generated 100 SBMs and averaged the values of modularity of the partitions. As the pattern of the majority of the algorithms are close to each other, a zoom is provided on the denser interval $P_{out} \in$ [0.6,0.8] to distinguish the curves for the different algorithms easily.

Results show that when the probability of edges between communities (P_{out}) increases, the partitions become less community-like resulting in smaller modularity values. However for any P_{out} value, Louvain and Spinglass methods still detect community structure with highest modularity compared with other methods. With an increase of P_{out}, Infomap, Walktrap and Label Propagation algorithms fail to detect communities (i.e. modularity of the partition becomes zero at $P_{out} \geq 0.1$ for Infomap, $P_{out} = 0.5$ for Walktrap and $P_{out} = 0.05$ for Label Propagation).

Processing time

We also evaluated algorithms based on the processing time and the number of nodes in the network; results are displayed in Fig. 2. From Fig. 2 that reports the relationship between P_{out} and processing times of the algorithms, we can see that when Pout increases, the processing time of the majority of the algorithms increases as the network structure becomes more dense. Moreover, in the same Fig. 2, we see that Infomap is the most time consuming algorithm whatever the probability of edges between communities (P_{out}) is. On the contrary, processing time for Label propagation is constant with regard to the P_{out} value. The processing time for Spinglass algorithm is not displayed in the figure and is much longer than for other methods.

Finally Fig. 3 gives the relationship between the number of nodes in the network and the processing time of the algorithms. We can see that when the number of vertices in the network increases, processing time for Spinglass, Fast Greedy and Walktrap increases the most, while the other methods detect community structure in a relatively constant time.

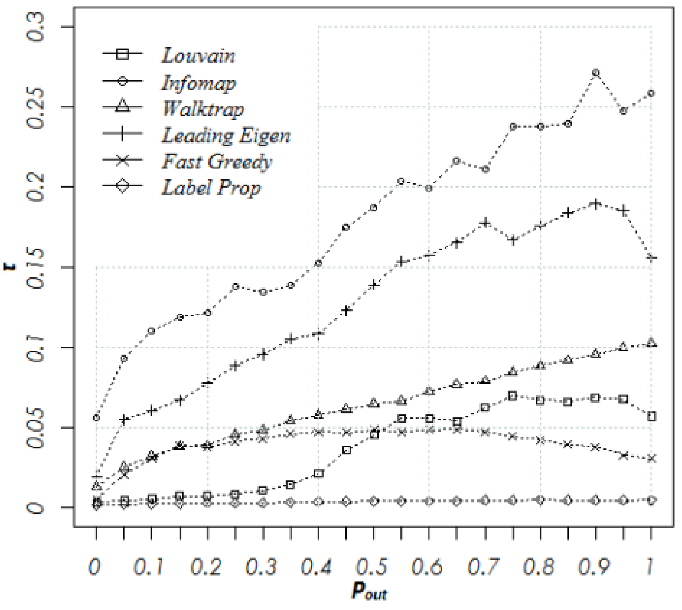


Figure 2: Probability of edges between communities (P_{out}) versus processing time in seconds (t) for P_{in} = 1 (Strong community structure).

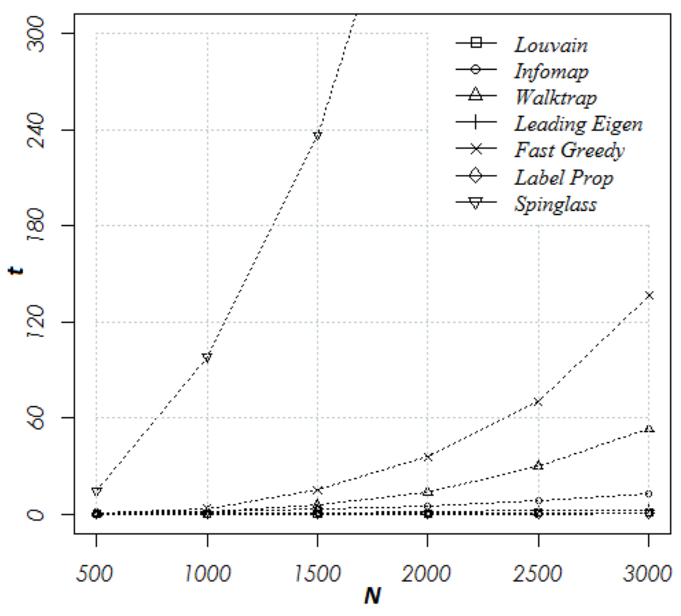


Figure 3: Number of vertices (N) versus time in seconds (t)

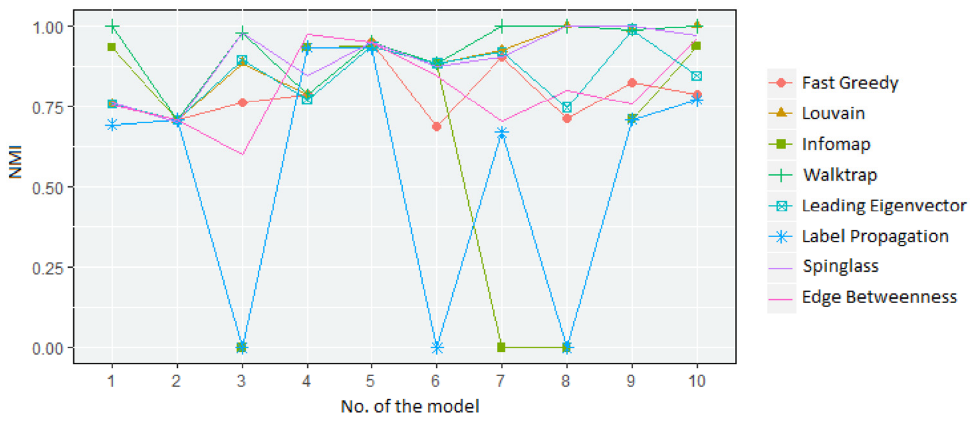
Comparison with ground truth

To complete the comparison of the algorithms, we evaluated and compared the community structures the algorithms detected with pre-known ground truth using stochastic block model.

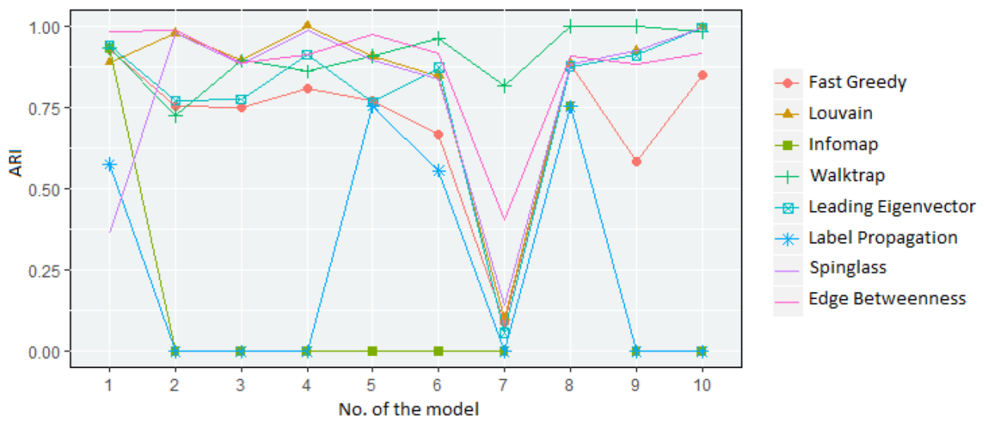
From the definition of SBM we already know that networks are generated based on the CxC Bernoulli matrix, where C_{ij} is the probability of an edge between nodes from community i and community j and relative sizes of communities. Thus specifying higher probabilities for the edges inside communities ($C_{ij}\in$(0.5,1)) and lower probabilities for the edges between communities (($C_{ij}\in$(0,0.25)), where $i = j$) and taking random numbers for relative community sizes which sum up to 200, we can generate an unlimited number of ground truth community structures.

The results obtained on a sample of 10 random networks are presented in Fig.4. We report NMI, ARI, and VI scores.

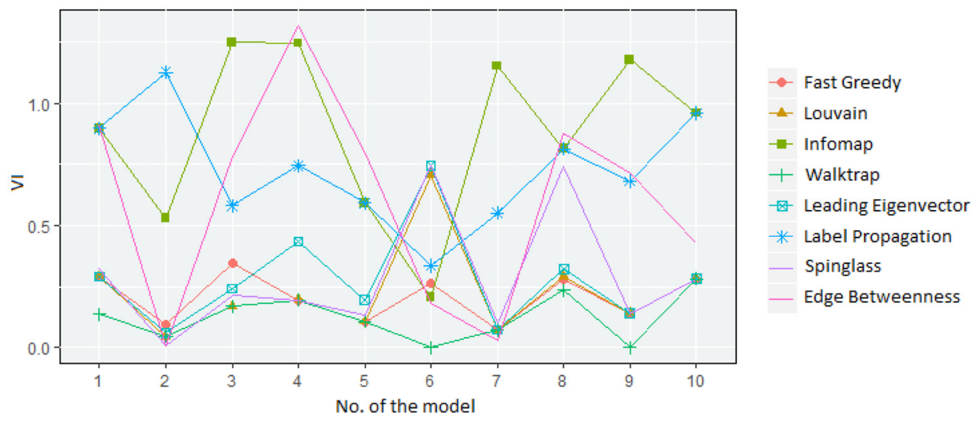
On Fig. 4 (a), we can see that the results regarding normalized mutual information are unstable for Label propagation and Infomap algorithms; while they can achieve good results when applied to some network structures, they can fail on others. Moreover, they never clearly overtake the other algorithms.



(a) Normalized mutual information



(b) Adjusted rand index



(c) Variation of information

Figure 4: NMI, ARI, and VI scores on 10 sample networks using the 8 algorithms when comparing detected community structures and ground truth.

Other algorithms are more stable. We can mention the high performance of Louvain, Walktrap and Spinglass algorithms and slightly lower performance for Fast Greedy. With regard to adjusted rand index displayed on Fig. 4 (b), the results are similar with some instability for Label propagation and Infomap algorithms, good performance for the other algorithms, but slightly better for Louvain, Walktrap and Spinglass, while Fast Greedy is slightly lower. Finally the results of variation of information are displayed on Fig. 4 (c). We can again see the instability among some algorithms (Edge betweenness, Infomap and Label propagation) while in the majority of the cases Walktrap, Spinglass and Louvain algorithms outperform the others. The similarity in the results obtained by NMI and VI comes from their analogy as they both have mutual information component in common.

We also report the numerical values obtained when averaged over the 100 random networks in Table 8. From Table 8 we entail that Walktrap algorithm overtakes all other algorithms considering the three measures when identifying ground truth community structure, although Louvain and Spinglass are very close, followed by Leading Eigenvector.

Table 8: Normalized mutual information, adjusted rand index and variation of information averaged over 100 trials (random networks) when comparing the community structures detected by the eight algorithms and the ground truth. Best results are in bold.

Agorithm Measure	Fast Greedy	**Louvain**	InfoMap	**Walktrap**	Leading Eig.	Label Prop	**Spinglass**	Edge between
NMI	0.791	0.879	0.471	**0.939**	0.839	0.444	0.883	0.807
ARI	0.794	0.883	0.449	**0.937**	0.849	0.404	0.889	0.891
VI	0.356	0.216	0.641	**0.114**	0.291	0.657	0.220	0.527

Summary of the results

On real world networks, Louvain algorithm achieves the detection of communities which are densely connected inside communities and sparsely connected between communities (almost always the highest modularity score), although it leads to detecting much less communities on some networks compared to other algorithms, specifically on the large networks and on networks with large diameter (Power Grid, DBLP, and Amazon). Louvain algorithm remains both effective and efficient also when the probability of edges between communities increases (results on artificial networks). Our experiments vote in for Louvain algorithm. Fast Greedy, Leading Eigenvector and Label propagation are also good algorithms. Spinglass and Edge betweenness suffer from their complexity. Infomap and Walktrap are weak on too many measures, while Walktrap was the most effective one when detecting ground truth communities on synthetic networks.

Table 9: Overview of the results. ++ indicates that the algorithm has good properties according to the measure while -that it has low performance.

Algorithm	Complexity	Modularity	Number of communities	Conductance	Clusteringco efficient	High P_{out} Modularity	High P_{out} Time	Comparisong round truth
Fast Greedy	+	++	Small	++	+			+
Louvain	++	++	Small	++	++	++		++
Infomap	+	++	Large	--	-	--	--	--
Walktrap	-	++	Large	-	+	--		++
Leading Eigenvector	+	+	Small	++	++			+
Label propagation	++	+	Large	+	++	--	++	--
Spinglass	--	NA	Small	NA	NA	++		++
Edge betweenness	-	NA	NA	NA	NA			+

Related Work

Several papers consists in community detection survey but many do not compare them [Porter et al., 2009, Coscia et al., 2011, Fortunato and Hric, 2016, Bedi and Sharma, 2016] while a few others made comparison but either on fewer measures than our survey, or on either real or synthetic networks [Danon et al., 2005, Fortunato, 2010, Leskovec et al., 2010, Orman et al., 2011, de Sousa and Liang, 2014, Yang et al., 2017].

In these studies, the authors did not aim at deeply comparing the algorithms. For this reason, we do not mention them in this section nor in the Table 10 where an overview of survey papers is reported.

Table 10: Overview of the literature. For each of the reviewed papers (first column), we report the number of algorithms that are reviewed in that paper (2nd column), the number of real networks used (3rd column), the type or number of synthetic networks used (4th column), the measures used to compare algorithms (5th column), and the outcome in terms of the best algorithms mentioned by the paper (6th column).

Survey	**Algorithms**	**Real networks**	**Synthetic net.**	**Measures**	**Best**
[Danon et al., 2005]	16 algorithms	None	GN	Modularity	
[Fortunato et al., 2010]	12 algorithms	None	LFR various μ	NMI, Precision	
[Leskovec et al., 2010]	2 algorithms	Yes	None	Conductance & Modularity	
[Orman et al., 2011]	5 algorithms	None	LFR,10k & 100k nodes	NMI, Topology	Infomap & Walktrap
Moradi et al., 2014 [de Sousa and Liang, 2014]	8 igraph algo.	3 real	Yes (4 own)	Modularity & time	Walktrap, Spinglass
[Yang et al., 2017]	8 igraph algo.	None	LFR, 200 to 32k nodes various μ	NMI, $\overline{C}/C$, Time	Louvain
[Chakrabortyet al., 2017]	6 algorithms	6 real	LFR	VI, NMI, ARI, F, Purity	
Our contribution	8 algorithms	8 real	SBM various P_{in} and P_{out}	Modularity Conductance & NMI	Louvain, Leading Eig. Fast Greedy

One of the earlier studies worth mentioning is Danon et al. [Danon et al., 2005]. The authors compare 16 algorithms mainly on modularity and fraction of nodes correctly identified on GN network. The authors conclude that most of the methods are very good at detecting the structure, but the synthetic network used to evaluate the algorithm was quite homogeneous and small (state of the art test bed set at that time).

In [Leskovec et al., 2010], Leskovec et al. first compare two graph partitioning algorithms namely the Local Spectral Partitioning algorithm [Andersen et al., 2006] and the flow-based Metis+MQI algorithm[Karypis and Kumar, 1998] considering conductance evaluation measure using several real networks including DBLP, Enron email and Astrophysics networks. The authors also briefly report the comparison of other algorithms still on conductance plus community score.

The survey by Fortunato [Fortunato, 2010] is probably still the largest reported study (100 pages long). This survey also suggests a classification of algorithms based on the underlying type of method they use; classes are in agreement with the ones previously mentioned.

Orman et al. uses LFR benchmark and generated 3 different network structures [Orman et al., 2011]. The authors first analyzed the properties of the synthetic network generated as compared to real networks. They found out that while it is possible to generate networks that are realistic in terms of size, the distribution of links is not always appropriate. For example, the small communities seem to be too dense and clique-like. They evaluated 4 of the 8 algorithms we also evaluated (namely Louvain, Fast Greedy, Infomap, and Walktrap) and added MarkovCluster. When evaluated on networks either composed of 10,000 and 100,000 nodes and using NMI, the authors found out that Infomap got the best results while Fast Greedy got the lowest. When considering the topology of the network as extracted by the algorithm and when compared to the ground truth, Waltrap seemed to reflect it better than the other algorithms.

Moradi et al., 2014 [de Sousa and Liang, 2014] evaluated the 8 algorithms from igraph package on 3 small real networks (ZacharyaAZs Karate Club, Football

Network, and NetScience) and 4 small synthetic networks they built (up to 100 nodes). They found that Infomap and Walktrap are the best when considering NMI and the topology of the extracted structure when compare to the ground truth structure. However, the size of the networks used are not fully realistic when compare to current real networks.

Yang et al. is one of the most recent survey [Yang et al., 2017]. In their paper, they compared the same 8 algorithms that we used in this paper and consider LFR synthetic networks making the networks varying from 200 to 32,000 nodes and also making the structure of the network varying through the μ parameter. To evaluate the algorithms, [Yang et al., 2017] considered NMI, $\overline{C}/C$ where $\overline{C}$ is the average number of detected communities the algorithm extracted (the experiment was repeated using 100 different networks) and C is the average number of communities in the same networks as given by the LFR benchmark, and processing time. The authors report that the number of nodes has little impact on the NMI apart for leading eigenvector algorithm. They also found that Label propagation and Louvain algorithms are much faster than the others. Finally, they concluded that Louvain algorithm is the best when considering the various aspects. While our study is more complete in terms of the measures we used as well as the type of networks we used since we used both real and synthetic networks, we could make the same conclusion on that algorithm.

Conclusion

In this paper we surveyed state of the art traditional community detection algorithms, internal evaluation measures to assess the quality of community structure and external evaluation measures to compare partitions. We applied the algorithms on eight real world networks and on artificially generated stochastic block model.

Thereafter we used internal evaluation measures such as conductance, global clustering coefficient and modularity to quantitatively assess the detected community structure and external evaluation measures such as normalized mutual information, variation of information, and adjusted rand index to compare network partitions of each algorithm. We also compared the algorithms based on performance (processing time). Overall, Louvain algorithm appears to be the most robust algorithm across the real and synthetic networks and across measures. Fast Greedy and Leading Eigenvector algorithms are also interesting alternatives.

Acknowledgment

The work benefited from the support of the European program Erasmus+ Convention KA107 No036342.

Bibliography

[Abbe, 2018] Abbe, E. (2018). Community detection and stochastic block models. Foundations and Trends in Communications and Information Theory, 14(1-2):1-162.

[Abbe and Sandon, 2015] Abbe, E. and Sandon, C. (2015). Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on, pages 670-688. IEEE.

[Andersen et al., 2006] Andersen, R., Chung, F., and Lang, K. (2006). Local graph partitioning using pagerank vectors. In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pages 475-486.

[Barrettetal.,2009] Barrett, C. L., Beckman, R. J., Khan, M., Kumar, V. A., Marathe, M. V., Stretz, P. E., Dutta, T., and Lewis, B. (2009). Generation and analysis of large synthetic social contact networks. In Simulation Conference (WSC), Proceedings of the 2009 Winter, pages 1003-1014. IEEE.

[Bedi and Sharma, 2016] Bedi, P. and Sharma, C. (2016). Community detection in social networks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 6(3):115-135.

[Blondel et al., 2008] Blondel, V., Guillaume, J., and Lambiotte, R. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008.

[Caltagirone et al., 2017] Caltagirone, F., Lelarge, M., and Miolane, L. (2017). Recovering asymmetric communities in the stochastic block model. IEEE Transactions on Network Science and Engineering.

[Chakrabortyetal., 2017] Chakraborty, T., Dalmia, A., Mukherjee, A., and Ganguly, N. (2017). Metrics for community analysis: A survey. ACM Comput. Surv., 50:54:1-54:37.

[Clauset et al., 2004] Clauset, A., Newman, M., and Moore, C. (2004). Finding community structure in very large networks. Phys. Rev. E 70,70(066111).

[Collignon et al., 1995] Collignon, A., Maes, F., Delaere, D., Vandermeulen, D., Suetens, P., and Marchal, G. (1995). Automated multi-modality image registration based on information theory. volume 3, pages 263-274.

[Coscia etal., 2011] Coscia, M., Giannotti, F., and Pedreschi, D. (2011). A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining: The ASA Data Science Journal, 4(5):512-546.

[Cover and Thomas, 2006] Cover, T. and Thomas, J. (2006). In Elements of Information Theory, 2nd Edition. Wiley.

[da Costa Alves, 2010] da Costa Alves, C. S. (2010). Social network analysis for business process discovery. The Technical University of Lisbon.

[Danon et al., 2005] Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. (2005). Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment, 2005(09):P09008.

[de Sousa and Liang, 2014] de Sousa, F. B. and Liang, Z. (2014). Evaluating and comparing the igraph community detection algorithms. In Intelligent Systems (BRACIS), 2014 Brazilian Conference on, pages 408-413. IEEE.

[Deng et al., 2016] Deng, S.-P., Zhu, L., and Huang, D.-S. (2016). Predicting hub genes associated with cervical cancer through gene co-expression networks. IEEEACM Transactions on Computational Biology and Bioinformatics (TCBB), 13:27-35.

[Fortunato, 2010] Fortunato, S. (2010). Community detection in graphs. Physics Reports, 486:75-174.

[Fortunato and Hric, 2016] Fortunato, S. and Hric, D. (2016). Community detection in networks: A user guide. Physics Reports, 659:1-44.

[Girvan and Newman, 2002] Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. Proceedings of the national academy of sciences, 99(12):7821-7826.

[Girvan and Newman, 2001] Girvan, M. and Newman, M. E. J. (2001). Community structure in social and biological networks. PNAS, 99(12):7821-7826.

[Goldenberg et al., 2010] Goldenberg, A., Zheng, A. X., Fienberg, S. E., Airoldi, E. M., et al. (2010). A survey of statistical network models. Foundations and Trends® in Machine Learning, 2(2):129-233.

[Gulikersetal., 2017] Gulikers, L., Lelarge, M., and Massoulié, L. (2017). A spectral method for community detection in moderately sparse degree-corrected stochastic block models. Advances in Applied Probability, 49(3):686-721.

[Halkidi etal., 2002] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2002). Clustering validity checking methods: part ii. ACM SIGMOD Record, 31(3):19-27.

[He et al., 2018] He, K., Li, Y., Soundarajan, S., and Hopcroft, J. E. (2018). Hidden community detection in social networks. Information SciencesaATInformatics and Computer Science, Intelligent Systems, Applications: An International Journal, 425(C).

[Hu et al., 2008] Hu, Y., Chen, H., Zhang, P., Li, M., Di, Z., and Fan, Y. (2008). Comparative definition of community and corresponding identifying algorithm. Phys. Rev. E, 78(026121).

[Hubert and Arabie, 1985] Hubert, L. and Arabie, P. (1985). Comparing partitions. Journal of Classification, 2(1):193-218.

[Karrer and Newman, 2011] Karrer, B. and Newman, M. E. (2011). Stochastic blockmodels and community structure in networks. Physical review E, 83(1):016107.

[Karypis and Kumar, 1998] Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing, 20(1):359-392.

[Lancichinetti and Fortunato, 2011] Lancichinetti, A. and Fortunato, S. (2011). Limits of modularity maximization in community detection. Physical Review E, 84(066122).

[Lancichinetti et al., 2008] Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. Physical review E, 78(4):046110.

[Leskovec and Krevl, 2014] Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

[Leskovec et al., 2008] Leskovec, J., Lang, K., Dasgupta, A., and Mahoney, M. (2008). Statistical properties of community structure in large social and information networks. Proceedings of the 17th international conference on World Wide Web, 3733(5):695-704.

[Leskovec et al., 2010] Leskovec, J., Lang, K. J., and Mahoney, M. (2010). Empirical comparison of algorithms for network community detection. In Proceedings of the 19th international conference on World wide web, pages 631-640. ACM.

[Leskovec and Yang, 2015] Leskovec, J. and Yang, J. (2015). Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems, 42(1):181-213.

[Mahmoud et al., 2014] Mahmoud, H., Masulli, F., Rovetta, S., and Russo, G. (2014). Community detection in protein-protein interaction networks using spectral and graph approaches. Computational Intelligence Methods for Bioinformatics and Biostatistics (CIBB), 8452.

[McDaid and Hurley, 2010] McDaid, A. and Hurley, N. (2010). Detecting highly overlapping communities with model-based overlapping seed expansion. In Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, pages 112-119. IEEE.

[Meila, 2007] Meila, M. (2007). Comparing clusteringsaAfan information based distance. Journal of Multivariate Analysis, 98(5):873-895.

[Moradi et al., 2012] Moradi, F., Olovsson, f., and fsigas, P. (2012). An evaluation of community detection algorithms on large-scale email traffic. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 7276:283-294.

[Newman, 2006] Newman, M. (2006). Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E, 74(036104).

[Newman and Girvan, 2004] Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. Phys. Rev. E, 69(026113).

[Orman and Labatut, 2009] Orman, G. and Labatut, V. (2009). A comparison of community detection algorithms on artificial networks. Lecture Notes in Computer Science (LNCS), pages 242-256.

[Orman et al., 2011] Orman, G., Labatut, V., and Cherifi, H. (2011). Qualitative comparison of community detection algorithms. Communications in Computer and Information Science, Springer, Berlin, Heidelberg, 167:265-279.

[Parimi and Caragea, 2014] Parimi, R. and Caragea, D. (2014). Community detection on large graph datasets for recommender systems. 2014 IEEE International Conference on Data Mining Workshop, pages 589-596.

[Pons and Latapy, 2005] Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks. Computer and Information Sciences - ISCIS 2005, pages 284-293.

[Porter et al., 2009] Porter, M. A., Onnela, J.-P., and Mucha, P. J. (2009). Communities in networks. Notices of the AMS, 56(9):1082-1097.

[Radicchi et al., 2004] Radicchi, F., C.Castellano, Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. PNAS, 101(9):2658aAS2663.

[Raghavan et al., 2007] Raghavan, U., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 76(036106).

[Rand, 1971] Rand, W. (1971). Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association, 66(336):846-850.

[Reichardt and Bornholdt, 2006] Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. Phys. Rev. E, 74(016110).

[Rosvall and Bergstrom, 2007] Rosvall, M. and Bergstrom, C. (2007). Maps of random walks on complex networks reveal community structure. PNAS, 105(4):1118-1123.

[Wasserman and Faust, 1994] Wasserman, S. and Faust, K. (1994). Social Network Analysis: Methods and Applications. Cambridge: Cambridge University Press.

[Xie et al., 2013] Xie, J., Kelley, S., and Szymanski, B. K. (2013). Overlapping community detection in networks: The state-of-the-art and comparative study. Acm computing surveys (csur), 45(4):43.

[Yang et al., 2017] Yang, Z., Algesheimer, R., and Tessone, C. J. (2017). A comparative analysis of community detection algorithms on artificial networks. Scientific Reports, 6(30750):Jun.

[Zhang et al., 2016] Zhang, A. Y., Zhou, H. H., et al. (2016). Minimax rates of community detection in stochastic block models. The Annals of Statistics, 44(5):2252-2280.

Authors' Information

Karen Mkhitaryan *- PhD student, Institute for Informatics and Automation Problem,*

0014, Yerevan, Armenia;

e-mail: *karenmkhitaryan@gmail.com*

Major Fields of Scientific Research: *Network Science, Community Detection*

Josiane Mothe *- Professor, Institutde Recherche en Informatique de Toulouse, URM5505 CNRS, Universite de Toulouse, Toulouse;*

e-mail: *josiane.Mothe@irit.fr*

Major Fields of Scientific Research: *Information Systems, Information Retrieval, Data analytics, Big Data*

Mariam Haroutunian *- Professor, Institute for Informatics and Automation Problems, 0014, Yerevan, Armenia;*

e-mail: *armar@ipia.sci.am*

Major Fields of Scientific Research: *Information Theory, Information Security, Probability Theory and Statistics*

MINIMAL AND MAXIMAL MODELS IN REINFORCEMENT LEARNING[1]

Dimiter Dobrev

***Abstract**: Each test gives us one property which we will denote as test result. The extension of that property we will denote as the test property. This raises the question about the nature of that property. Can it be a property of the state of the world? The answer is both yes and no. For a random model of the world the answer is negative, but if we look at the maximal model of the world the answer would flip to positive. There can be various models of the world. The minimal model knows about the past and the future the indispensable minimum. Conversely, in the maximal model the world knows everything about the past and the future. If you threw a dice the maximal model would know which side will fall up and would even know what you will do. For example, it would know whether you will throw the dice at all.*

***Keywords**: Artificial Intelligence, Reinforcement Learning, Partial Observability, Event-Driven Model, Double-State Model, Test Property, Test State.*

***ITHEA Keywords**: I.2.6 Learning*

Introduction

We try to understand the world (the environment) and for this purpose will use tests. One example of a test is the following:

---

[1] The author is grateful to the Bulgarian National Scientific Fund for the provided financial support, under project No. DN 19/1 (2017).

If I press the door handle $\Rightarrow$ the door will open.

Each test contains a condition (prerequisite). The condition in this case is that I must press the door handle. When the condition is met, the test is done and we get the test result, which can be true or false. In our example the door will open or will not open.

Each test gives us one property (*the test result*). The property in our example is “the door is locked”. We cannot know the value of that property at each and every point in time. We only know that value at the time points when the test is done.

We will assume that the property is meaningful at any time point and will thus try to extend the characteristic function of that property beyond the subset of time points at which the test is done. The property may not necessarily be total or defined at each and every time point. Say, if somebody stole the door, the question of whether it is locked becomes meaningless. In other words, while we will do our best to extend the characteristic function of the property, we may not always end up with a total function.

What is the idea behind extending the test result to a test property? If I gave you a slice from a cucumber, would you be able to reconstruct the whole cucumber out of that slice? We certainly mean a mental reconstruction of the cucumber, not a physical one. That reconstruction however is not unique as it may generate various objects. E.g. from a cucumber slice we may conceive a rhinoceros if we imagine that the slice comes from its tusks. Certainly, we will try to find extensions which are as simple, natural and credible as possible.

Bear in mind that the cucumber slice is real, while the cucumber as such is imaginary or conjured up. If I let you see the full cucumber it would be easier for you to imagine it; however, you are more likely to imagine a healthy cucumber, whilst the actual one may turn out to be rotten inside. That is, you never get the full information. You always receive a fraction (a slice of information) from which you have to conjure up the whole thing.

Now let us discuss the question of what is this property (the test result and its extension). At different moments its value can be true or false. However, it is not a property of time because it depends on the developments playing out around the time point rather than on the time point proper. Maybe this is a property of the state of the world (the subset of states in which the property is TRUE).

Does this property depend on the past and on the future? Typically, the test takes place within a certain period of time rather that at a single moment. The test result depends therefore on the temporal context (the near past and the near future within the test period). If we take the test property it depends on a wider temporal context and may tell us something about the distant past and the distant future of the world.

Let us have the property “This letter brings good news”. The test for this property is “Open the envelope and check what the letter says”. The property tells us something about the future. To put it more precisely, it tells us what we will read in the letter after we open the envelope. Is it a property of the world? Does the world know what the letter says before we open it? We tend to think that yes, the world knows, but it can afford not to know, as well. For example, the majority of computer games do not bother to calculate the whole world, but take care only of the fraction of the world the gamer sees now. If the world were such a game, it would decide what the content of the letter is only when you open it. In a similar example, let us imagine life as a TV serial. In series 1354 you receive a letter, but open it 10 series later. When will the scriptwriter decide what is in the letter? When he or she writes the script of series 1354 or the script of series 1354+10? So we see that the world may or may not know in advance how the future will unfold.

Similar is the situation with the past. The property “I am back from a vacation” provides some clues about the past. The test to verify this property is: “I check whether I am on a vacation or on a business trip and then come back”. We assume that if you are back from a vacation and are still at home (have not gone elsewhere), then you are still back from a vacation. That is, we extended the property to time points at which it is not tested.

Let us assume that the future of the world does not depend howsoever on whether you are back from a vacation or from a business trip. Then why should the world remember that fact at all? The question which keeps historians awake at night is whether the world remembers the past. Did a historical event leave any documents or other traces of its occurrence? The answer is that the past can be remembered but it is perfectly possible some facts to be totally discarded.

This article will discuss two models of the world – minimal and maximal. In the minimal model the world remembers the indispensable minimum from the past and knows the indispensable minimum about the future. Conversely, in the maximal model the world remembers everything from the past and knows everything about the future. In the maximal model the world knows exactly what is going to happen and even what you (the agent) would do or not do.

What are we looking for?

What is given and what are we trying to find out? In this article we will try to find an explanation of the world. With Reinforcement Learning [1] we have an agent who lives in a certain world. The world is an oriented graph similar to Figure 1. The agent follows the arrows, moves from one state to another and collects certain rewards. It is vital for the agent to explore the world and understand it well, otherwise he would not get to the rewards. Many articles assume that the world is a given entity and what we look for is a policy which would be successful in that world (e.g. [3]). In this article we will assume that the world is unknown.

In Figure 1, the possible moves of the agent are represented by arrows and the possible observations are denoted with various numbers and colours. Each arrow should be tagged with a label indicating the action corresponding to that arrow. Although we have not shown these labels, the figure clearly demonstrates that sometimes there is a single arrow (possibility) and at other times there are more possibilities (at states 2 and 3). We will assume that not all actions are possible and that the transitions are nondeterministic. In other words, for a particular state/action there may not be even a single arrow with right label departing from that state, or there may well be more than one arrow with the right label.

By allowing for nondeterministic transitions we have in fact embraced randomness. In [5] we demonstrated that there are two types of randomness, predictable and unpredictable. Here we will work with unpredictable randomness (with a probability in the interval [0, 1]). This kind of randomness is also used in Nondeterministic Finite Automaton (NFA). With NFA something may or may not occur, and we do not know the probability of its occurrence. The Partially Observable Markov Decision Process (POMDP) uses predictable randomness (something occurs with a precisely determined probability). In [5] we demonstrated the equivalence of the four types of models (the deterministic one, the models with the two types of randomness, and the model which combines the two types of randomness). So we can use whichever model suits us best and have thus opted for the model with unpredictable randomness used here.

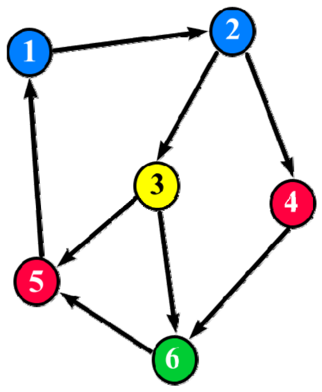


Figure 1

Full Observability means that we can tell which is the state just by the observation. The opposite is referred to as Partial Observability. Going back to Figure 1, when we are able see the number of the state, we enjoy Full Observability. Seeing the colour only does not tell us the full story. If all we see is a red circle we cannot not know whether it is state 4 or 5.

If we had the world’s model, we would be able to foretell the future. Thus, if we find ourselves in state 4 we will predict that the next state will be 6. In a red state we will know that the next state will be either green or blue. Likewise, we can back-tell the past. Suffice it to turn the arrows in opposite direction and the past will become the future. The only problem is that turning the arrows may force a deterministic graph become nondeterministic, but we have chosen to use nondeterministic graph models anyway.

What is given? The given body of facts is the history until the present time point, that is the sequence of actions (outputs) and observations (inputs or views).

$$a_1, v_2, a_3, v_4, \dots , a_{t-1}, v_t$$

The numbering notation here identifies the number of the moment and not the number of the step. There are two moments in each step. At the first moment we produce information (this is our action) and in the second moment we enter what we see. That is, the step number will be the moment number divided by two.

Why do we choose to divide time in moments and not in steps? Because of the event-driven models, where the states will change at certain moments. At each step the state may change twice because the step has two moments.

There will be two types of moments: input and output moments. They will also be referred to as even and odd moments.

Note that the inputs and outputs will be vectors of scalars. We will assume that these scalars are finite. We could have stayed with Boolean vectors, but have not done so in order to avoid redundant coding (cf. [4]).

The history will also include all the incorrect (bad) moves we have tried before we play our next move.

$$bad_1, a_1, v_2, bad_3, a_3, v_4, \dots , bad_{t-1}, a_{t-1}, v_t$$

Here we can imagine the ***bad*** element as a list or as a set of incorrect moves (because the order in which we have tried the incorrect moves is not essential). We will assume that the incorrect moves were tried at the same moment when the correct move was played (that is why the set of moves ***bad*** and the next correct move ***a*** have the same index).

Definition: For our purposes, *life* is a history which cannot be continued.

A history cannot be continued when it is either infinite or terminates in a state which does not have any arrows coming out of it. The term we used for these states in [6] was “sudden death”.

The Double-State Model

Figure 1 depicts the standard model, which is based on *steps*. We will use a *double-state* model, which is based on *moments*.

Figure 2 explains the difference between the standard model and the double-state model. The latter is obtained from the first by replacing each state with two states connected with an arrow. All arrows which previously entered the one state now enter the first one of the two states. The arrows which previously departed from the one state now depart from the second one of the two states. The former label of the state now goes to the arrow which connects the new states. In the double-state model, only arrows have labels and states have none. We have used smaller circles for odd-number states in order to stress that there are two types of states.

Note: Here we will deem that a state relates to a *time period* while an arrow relates to a *time point* which is the onset of the text time period. In event-driven models however, we will deem that states relate to longer time periods and arrows relate to very brief periods (lasting one moment or bit more).

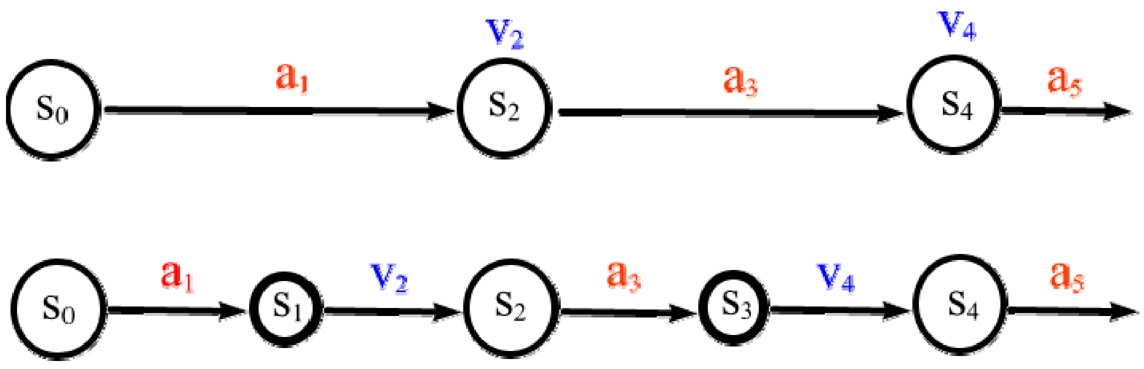

$$\begin{tikzcd} S_0 \arrow[r, "a_1"] & S_2 \arrow[r, "a_3"] & S_4 \arrow[r, "a_5"] & {} \end{tikzcd}$$
V_2
V_4
$$\begin{tikzcd} S_0 \arrow[r, "a_1"] & S_1 \arrow[r, "V_2"] & S_2 \arrow[r, "a_3"] & S_3 \arrow[r, "V_4"] & S_4 \arrow[r, "a_5"] & {} \end{tikzcd}$$

Figure 2

Although it appears that the double-state model includes twice as many states, this is not actually the case, because when two states are equivalent from a future perspective they can be merged in a single state. The standard model allows the merging of only those states which are equivalent from both present and future perspective. I.e. in the standard model each state must remember the present, but need not do so in the double-state model. This is one of the reasons why we introduce that model. In the present article we attach paramount importance to the information we can derive from the state – what can the state tell about the past and the future. The present is part of the past because it has already occurred.

As the name suggests, the double-state model deals with two types of states: post-input and post-

output. We will call them evens and odds, meaning even-number and odd-number states. Evens are more important because they are the states in which we think. In odd states we do not think, but just wait to see what information the world will give us. (It may be assumed that the world does the thinking in the odd states. So we take turns – at one time point we put the thinking hat on and at the next time point we pass the thinking hat over to the world.)

Let use as an example the world in which we play chess against an imaginary opponent. Our actions (outputs) will be vectors in the form of (x_1, y_1, x_2, y_2). These vectors describe our move. We will be able to see the opponent’s move and the reward. That is, the input will be a (x_1, y_1, x_2, y_2, R) vector. The states in the double-state model will be the positions on the chessboard. Even-number states will be those in which it is the white’s (ours) turn to make a move. If our move terminates the game (e.g. checkmate) the opponent must play some idle move and return to us only the reward of the game. The idle move is a vector in the form of (0, 0, 0, 0, R) and must lead to the initial position so that the game can be restarted.

Things with the standard model will be more complicated. The states will only be the positions when it is white’s turn to move, but the model must also remember the opponent’s move which produced that position. Hence there will be more states because “white ahead” positions will be twice less, but if there are averagely 100 ways to produce a position (from 100 different positions with 100 different moves of the opponent) the standard model will end up with 50x more states.

As mentioned already, the standard model needs to remember the present while the double-state model does not. How about the future? When we play chess against a deterministic opponent, then both models produce deterministic graphs. Let us assume that the opponent is nondeterministic and enjoys the liberty to choose among several possible moves at each position. In that case, the standard model would produce a nondeterministic graph (a single move from us triggers several different responses/counter-moves and their respective positions). The double-state model will remain deterministic, because our move creates a single, determinate position. Two or more arrows, which correspond to the possible moves of the opponent, will depart from that position (if the opponent were deterministic, there would be just one arrow).

Does the double-state model always produce a deterministic graph? Not always, but we can always transform the graph to a deterministic one. Now we will go back to our chess game example, but will slightly change the rules: we will not be able to see the opponent's move, but only the piece moved by the opponent. So we will not see (x_2, y_2). Now that we know the piece, but not the place to which it was moved, we will have several possible positions. While the obvious graph representing this kind of game is nondeterministic, we can perfectly apply a deterministic graph. If the state of the world is not a specific position on the chessboard, but a set of possible positions, then each move will trigger a single arrow which points to a set of possible positions. That is, in the first (nondeterministic) scenario the world knows more about the future than in the second scenario. In the deterministic scenario, the world does not know the exact position. What the world knows is the set of possible positions. When will the world get to know the exact position, however? Not until a later time point, when the exact position will transpire from the input (observations). Same as the letter in our TV serial. In the first case the world knows what is in the letter, while in the second case it will not decide what the letter says until you open the letter.

The Minimal Model

In our concept, a double-state model of the world will be minimal when the world’s knowledge about the past and the future is limited to the indispensable minimum. In the minimal model, if two states are equivalent vis-à-vis the future, they coincide. In other words, the model does not remember anything about the past unless it is necessary in order to determine the future.

Furthermore, the minimal model is a deterministic graph. This means that the branches are pushed forward (to the future) as much as possible. Hence, nothing of the future will be known in advance (the thing will become known only when it has left its imprint on the observation, not earlier).

A minimal model is not tantamount to a least-state model. Minimalism does reduce the number of states with regards to the past, but tends to increase them with regards to the future (because we have replaced the specific possibilities with sets of specific possibilities).

The determinization procedure can always be applied so we can always obtain a deterministic graph. A deterministic graph does not necessarily mean a deterministic agent or world. For the agent to be deterministic there must not be any branches from the even states. (Here by saying that an agent is deterministic we meant that the agent is forced to play a deterministic game, because the agent has only one possible move. Generally speaking, however, a deterministic agent is understood as an agent who plays deterministically without being forced to do so.) For the world to be deterministic there must not be branches from the odd states. A deterministic graph means that the states know only the indispensable minimum about the future. (If two states are different, their past has been different, too. Therefore, the differentiating factor is their different past, not their different knowledge about the future).

The Total Model

An incorrect move is one which has not any arrow for it, i.e. this move simply cannot be made. However, we would like to enable the agent try incorrect moves without further implications. That is, the agent will only receive information that the move is not correct, but will remain in the same state.

For this purpose, to each even state (at which incorrect moves exist) we will add an odd state (see Figure 3). All incorrect moves from the even state will point to the new odd state. From the new odd state we will go back to the even state with an arrow labeled “bad”. That label will be a special new vector, which we have added for our purpose and which will be received as an input only when we try an incorrect move.

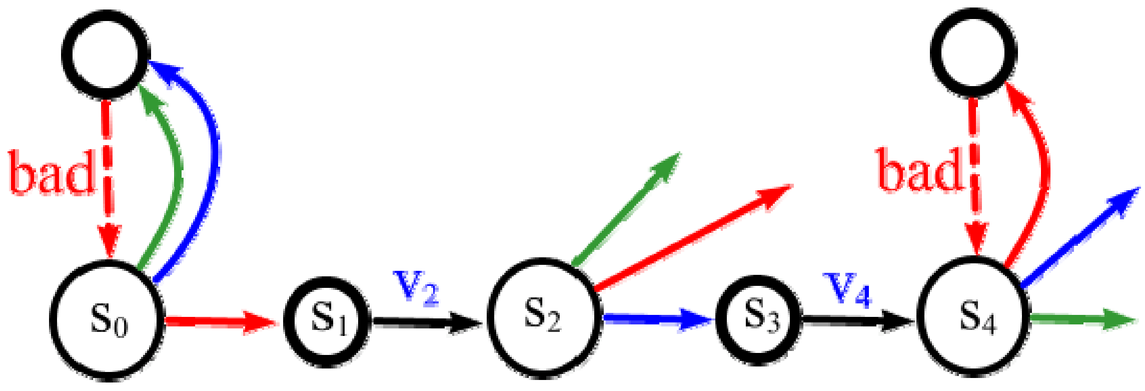


Figure 3

In Figure 3, the possible moves are represented by red, blue and green arrows. There are two incorrect moves at state s_0 and only one incorrect move at s_4. State s_2 has not an additional odd state due to the absence of incorrect moves at s_2.

This gives us a total model, where all moves can be tried, but only the correct ones change the state of the world, while the incorrect ones only return information which confirms that they are not correct. Thus we obtain a whole new total model, which describes the same world as the previous model, but has an added value in that it lets us try the incorrect moves.

The Maximal Model

Now that we saw the minimal model well and alive, we might think there is a maximal model, too. That should be the model where the state knows everything about the past, everything about which moves are correct and everything about the future.

Knowing everything about the past is easy as long as we do not stumble into branches when we walk back (in the direction opposite to that of the arrows). In other words, if the model is tree-shaped then the state will know everything about the past (we will able to reconstruct the entire history by going back from that state).

We can easily add also the information about which moves are incorrect, because this information is finite.

If we were to know everything about the future, we must dispense with any nondeterminism. We want to know which side will fall up before we throw the dice. We will thus construct a model where all nondeterminism is precipitated in the initial state. Once we select the initial state (in a nondeterministic way), the way ahead will only be deterministic.

Let us take the tree of all reachable states. (This is a true tree, because equivalent states are *not* merged into one.) We will make determinization on that tree, although this is not strictly necessary. From that tree we will obtain all policies of the world. These are sub-trees, which have no branches at observation points, but keep their branches at action points. These trees are many (cardinality of the continuum). We will take all those trees and make a model where the initial states will be the roots of all these trees.

“Policy” is maybe not the most appropriate word to use here, because a policy implies a certain objective. Here we assume that only the agent has an objective, and the world has none. If we claim that the entire world tries to help us or disrupt us, that would be far too egocentric. Nevertheless, we will imagine that in the world there are agents who are up to something (have their objectives). So, the world has not an objective as such, but we will still use “policies” to denote the various behaviours of the world.

We have thus made a model which consists of all policies of the world. Yet before life begins, the world randomly chooses one of its policies and follows it to the end of the agent’s life. The idea is to preconceive how to play the game before the games begins. We may decide that if the opponent plays a rook, we will respond with a bishop and so forth. This preconception is a policy and is depicted with an infinite tree. These trees are uncountable. We may decide to play using a certain deterministic program, but in this way we can preconceive only a computable policy. The number of computable policies is smaller (they are countable).

The so-obtained model is equivalent to the one we began with, because any history which is possible in one of these models is also possible in the other model. With the new model however the world behaves in a deterministic pattern except for the initial time point when the initial state is selected.

In that world, any randomly selected state knows almost everything about the future. The only thing it does not know is what action will the agent choose. We would like to construct a model which makes sure that the state of the world knows even this missing piece.

There is one hurdle, however. We tend to assume that the world is given and the agent is random. Therefore, the world cannot know what the agent will do because it does not have any idea which agent will come by. Now let us assume that the agent is fixed and the world may know something about the agent. The world may know, for example, that in a certain situation the agent will not play a certain move, although the move is correct and doable. In the model graph, the move not to be played by the agent will be shown with a missing arrow.

We will further imply that before life begins both the world and the agent have figured out how to play. They have selected their policies and will stick to them right to the end of the agent's life. This can be described by a tuple of two infinite trees or by one life (an infinite path in the tree). This is because the result from the application of two fixed policies is a fixed path.

Thus we arrived at the maximal model of the world. It consists of all possible lives (paths in the tree of reachable states). The only missing piece is information about the incorrect moves. To bridge this gap, we will add loops similar to those in the total model. But, this will not be a total model, because loops will be added only for incorrect moves, and not for all missing arrows. This gives the model depicted on Figure 4. (No loop at s_2 because there are no incorrect moves from that state.)

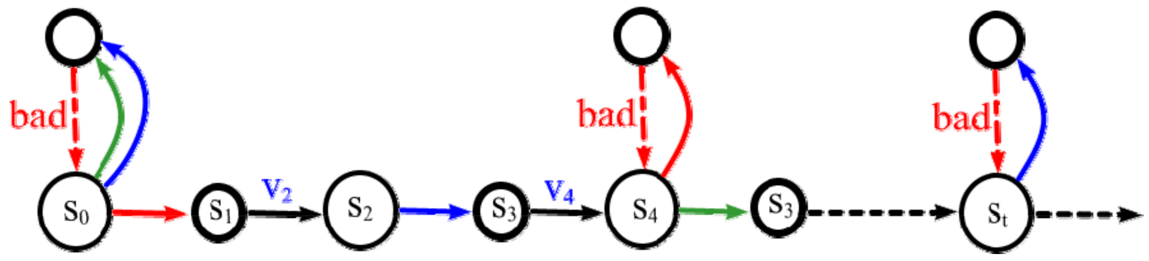


Figure 4

Figure 4 depicts only one life rather than all possible lives which form the maximal model. We care only about one life and this is the life we live in. This lets us assume that the maximal model has only one life, namely the life we are interested in. So we will lose the equivalence with the initial model, but the so-obtained model is what we need, because all other possible lives are not relevant.

Any state in our maximal model can be used to reconstruct the full history or even the full life. The forward and backward paths are branchless. Our added loops will not count as branches, because the *bad* symbol does not occur at observations after correct moves. Here the state knows which moves are incorrect, but does not know which of those moves have been tried by the agent. This information is not important to the world, because it has not any implications for the past or for the future. The information is certainly essential to the agent, because he may not know which moves are incorrect and will obviously benefit from knowing which moves he has already tried.

Conclusion

We try to understand the world, i.e. to find its model. The problem however is that there is not a single model but a raft of models. There may be unreachable and equivalent states, but this is not much of a problem. There may be parts of the world which we have not visited and will never visit. Let us take the following example: Is there life on planet Mars? We have never been there and will never go there, so the matter is irrelevant (will not howsoever affect our life).

The most serious problem is that there are models in which the world knows more and models in which the world knows less. Which model are we looking for? The answer is that we need the maximal model so that we can obtain maximum insight of the past and maximum foresight of the future. We will even venture to predict our own behaviour, because we are part of the world and trying to understand the world means that we should also try to predict our own behaviour in that world.

In order to describe the maximal model, we will use the so called extended model. That model will present the state of the world as a vector with a huge number of coordinates (variables). They will be thousands or even countless. In theory they are countless, but for practical purposes we will select only the most interesting ones. Probably this is the model referred to by Sutton in [2] (state representation which contains many state variables).

The first coordinates (variables) to be applied in the vector which describes the extended state will represent what we see at the moment. With the double-state model, an observation is not a function of the state of the world, because if the state is even-numbered there may be many arrows with different observations pointing to it. Accordingly, an odd state may have many arrows departing from it. But, here we discuss the maximal model, which always has one incoming and one outgoing arrow. In the maximal model, therefore, it is the state of the world which determines what we see at the moment.

To what we see at the moment we will add the test properties of various tests. These are not the real-world cucumber slices. These are imaginary cucumbers which we have conjured up from the real slices. Thus, the extended model will be an imaginary rather than a real thing.

Question: If we have Full Observability, do we need to add test properties to the state vector in the extended model? Answer: Yes, we need to do so, because in Full Observability case we know the state of the world, but this state comes from another model, not from the maximal one. Having Full Observability with a maximal model would make the world so simple that it will be not interesting at all.

How can we design a test property? Consider the property "Is the door locked". We test this property and sometimes get "locked", sometimes "unlocked". It would be fairly difficult to create a model which tells us when exactly the door is locked or unlocked. We would be much more successful if we imagine that the doors are more than one (especially if the doors are actually more than one). In the new model we should have an idea about which door we are standing in front of now. Some doors in that new model can be always locked, others would be always unlocked, and a third group of doors would change their state according to certain rules. In this case, we will not add to the extended model just one variable which reflects the test property. We will add many variables – one for each door (which reflects the state of the door) and one variable which indicates the exact door we are standing in front of now. In [5] this presentation was termed *test state*.

Which door are we standing in front of right now? This is determined by event-driven models which will be discussed in the next article.

Bibliography

[1] Richard Sutton, Andrew Barto (1998). Reinforcement Learning: An Introduction. *MIT Press, Cambridge, MA (1998).*

[2] Richard Sutton (2008). Fourteen Declarative Principles of Experience-Oriented Intelligence. www.incompleteideas.net/RLAIcourse2009/principles2.pdf

[3] Johan Åström. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications. 10: 174–205.*

[4] Dimiter Dobrev (2013). Giving the AI definition a form suitable for engineers. *April, 2013.* arXiv:1312.5713.

[5] Dimiter Dobrev (2017). How does the AI understand what's going on. *International Journal "Information Theories and Applications", Vol. 24, Number 4, 2017, pp.345-369.*

[6] Dimiter Dobrev (2018). The IQ of Artificial Intelligence. *Jun 2018. arXiv:1806.04915.*

Authors' Information

***Dimiter Dobrev** – Institute of Mathematics and Informatics, Bulgarian Academy of Sciences. Acad. Georgi Bonchev Str., Block 8, 1113 Sofia, Bulgaria; e-mail: d@dobrev.com*

Major Fields of Scientific Research: Artificial Intelligence, Logic Programing

DEVELOPING ENTERPRISE FINANCIAL SUSTAINABILITY AND BANKRUPTCY FORECASTING MODEL IN SPECIFIC ECONOMIC-MATHEMATICAL ENVIRONMENT USING SIMULATION METHOD

Tea Munjishvili

Abstract: *The possibility of developing a specific economic-mathematical model of financial stability and prediction for simulation in the simulated mode is justified by optimal choice of multivariate model. The software package developed by us allows simulations mode based on a general model to develop a number of concrete models of the enterprise and optimal options. Using simulator's techniques and technology is described below.*

Keywords: *Simulator, Financial Sustainability, Model development*

ITHEA Keywords: *I.6.7 Simulation Support Systems*

Introduction

There are many publications dedicated to financial sustainability and bankruptcy forecasting of enterprises. In the advanced countries of the world are developed and used different economic- mathematical models, such as: Altman models, Olsen, Lisa, Dzivjevski and others. The long-term statistical data of bankrupt and financially sustainable enterprises of this country and the sector are based on the determination of the values of the coefficients in any model. Countries, which are moving on a market economy, like Georgia, many statistical data of enterprises bankruptcy does not exist, so the rates statistical methods is not possible, and using models without modifying them in developing and post-Soviet countries invariably serious reduced to problems linked.

Simulation modeling is the most rational way to solve the problem. The purpose of simulation of financial stability and bankruptcy prediction of any simulator and the enterprise is:

- On the basis of a general model that can be considered as a standard model, a particular economic mathematical model of the financial sustainability assessment of the enterprise can be used in the actual activity of the problematic area;
- In the process of teaching: first – getting knowledge using simultaneous modeling, second - simulation of typical situations and decision-making skills, and third - identification and evaluation of received knowledge;
- In the course of scientific research, getting the factors involved in financial sustainability of a particular enterprise or enterprise group.

Setting the task

Financial analyst has three main tasks:

- Assess the financial position of an enterprise based on actual data;
- Prognosis of financial sustainability according to actual data;
- In case of unwanted forecast results, elaborate the plan's plan during which future financial sustainability will be ensured;
- Establish organizational and technical measures based on the results of the forecast and control its implementation.

Specialist in the preparation of the financial analysis has the same tasks, the only difference is that he should acquire financial mathematical methods and models used in the analysis, using a mathematical model capable of modeling - simulation, identify the model of which the characteristics of what can be achieved to the desired values financially stable situation.

The aim of the research was to use the mathematical models used in technology and other areas of the economy (securities market and other) used to assess the financial sustainability of the enterprise without any statistical data:

1. Based on modeling, hypotheses about the structure and content of an integrated indicator of the financial condition of the enterprise. Ie Let's justify the possibility of assessing the financial position of an enterprise with selected economic indicators;
2. The probability of reliability / insecurity of the combination of each indicator and indices shall be calculated by considering the possible financial conditions of an enterprise;
3. Compute the share of each of the indicators and indicators in the financial position of the enterprise;
4. Implementation of the impact of the share of each indicator and the combination of indicators on the expected financial condition of the enterprise;
5. Based on modeling, the financial development strategies of the enterprise should be shown;
6. Calculate the risk of financially unsustainable enterprise of the enterprise and develop ways to prevent it;
7. Based on modeling, the optimum option for the development of the enterprise was chosen and this option should be used to monitor the financial stability of the enterprise;
8. Based on the actual data, the financial stability of the enterprise will be assessed and the bankruptcy assessment forecasts of the enterprise will be implemented.

Our goal is to provide the basis for the financial mathematical model of financial sustainability of the enterprise to achieve the tasks that are listed on the bankruptcy of enterprises which are the basis of the existing models.

Our method requires that achievement of the goals listed here must be achieved only in terms of logical (economic content) connections between economic indicators simulation mode.

We analyzed the mathematical models and algorithms used in the assessment of the financial sustainability of the enterprise - the 14 models: Altman (4 model), Altman-Sabato, Fulmer, Sprengite, Dzivjevsk, Olson, Conan-Goldend, Taffler, Lego, Lisa, Chester models.

Besides the selected indicators based on the economic essence in any model, it is important to determine the values of the **b_i**-coefficients of this model, which is carried out by the processing of statistical data.

In developing and post-Soviet countries (such as Georgia) there is no long-term statistics on bankruptcy of enterprises, so the next issue on the agenda - designed for such method, which would allow the financial sustainability of the economic mathematical model based on the determined risk of bankruptcy and is not used for the statistical data.

Problem solving method

We have two ways of realizing this task:

First - In economic mathematical model, logical connections between selected economic indicators should be considered. Assess the financial sustainability of the enterprise by comparing logical connections, factual values of the normative values of experts evaluated by the expert (experts);

Second - To develop a specific economic mathematical model of financial sustainability assessment and forecasting for a specific enterprise. Found. The models of P_i and P_{i+1} are different from the values of the coefficients.

Today we have a program that is the second approach to developing economic mathematical model of financial sustainability and forecasting of enterprises.

The second, the essence of the approach chosen by the economic mathematical model of financial sustainability and forecasting of enterprises is the following:

The $\mathrm{m_i} \in \mathrm{M}$ model is chosen from selected models range and by $S_j$enterprise’s actual data model’s arguments (values) average will be counted.

The first step of solving problem

The first step is realizing with a simulation model. Simulation object is a selected model.

The task is set, when in equation $Z = f_i \emptyset k_t$ model’s arguments (values) $\mathrm{f_i} = \mathrm{Const}$, should be found such values of k_i, when: first – Z shows enterprise financial sustainability, or second – Z’s different values are counted for different values when Z is changing with Δ step.

So many of the Z's options are different, which are different from the values of the modes in the model. Assessment of the results of any variant is automatically carried out by the rules adopted in the selected model, or in general

$Z = \sum_{i=1}^{n} b_i\, x_i \rightarrow max;\;\; Z = \sum_{i=1}^{n} b_i\, x_i \rightarrow min$

$b_{i,min} \leq b_i \leq b_{i,max}$

$b_{i,min} = 0{,}001;$

x_i*- i argument’s actual data’s average*

Second: The optimum option will be chosen from the variants. The selection criterion is the minimum margin of bankruptcy forecasts. The economic mathematical model of the financial sustainability assessment of the obtained S_j enterprise is different from the sampling $m_i \in M$ model with the values of the coefficients;

Third: Correction of the model adopted according to the actual data of S_j.

The second step of solving problem

First, in the equation $Z = f_i \emptyset k_t$ by using simulation model were found values of input coefficients and many variations of Z, from which was chosen one and tested on actual data. Let’s set the task. We have b_i=const, we have to find all x_i arguments’ (variables) values. We have to find for t_i, i=1,n year such values, when Z show us stability of the enterprise's financial stability, or get Z’s different values for changing model’s variables by Δ step. Generally,

$Z = \sum_{i=1}^{n} b_i\, x_i \rightarrow max;\;\; Z = \sum_{i=1}^{n} b_i\, x_i \rightarrow min$

$x_{i,min} \leq x_i \leq x_{i,max}$

$x_{i,min}$ - minimal value of actual values;

$x_{i,max}$ - maximal value of actual values;

b_i – values, found with using coefficients' simulation model

For any $m_i \in M$ model plan's many variances is calculated for t_i ,$i = 1,n$ year. Then, the optimal one is chosen from calculated majority of variances, where criteria of optimization is equal to minimum margin of bankruptcy forecasts.

This model is taken to guide the financial stability of the given enterprise and as a model of the bankruptcy forecasts.

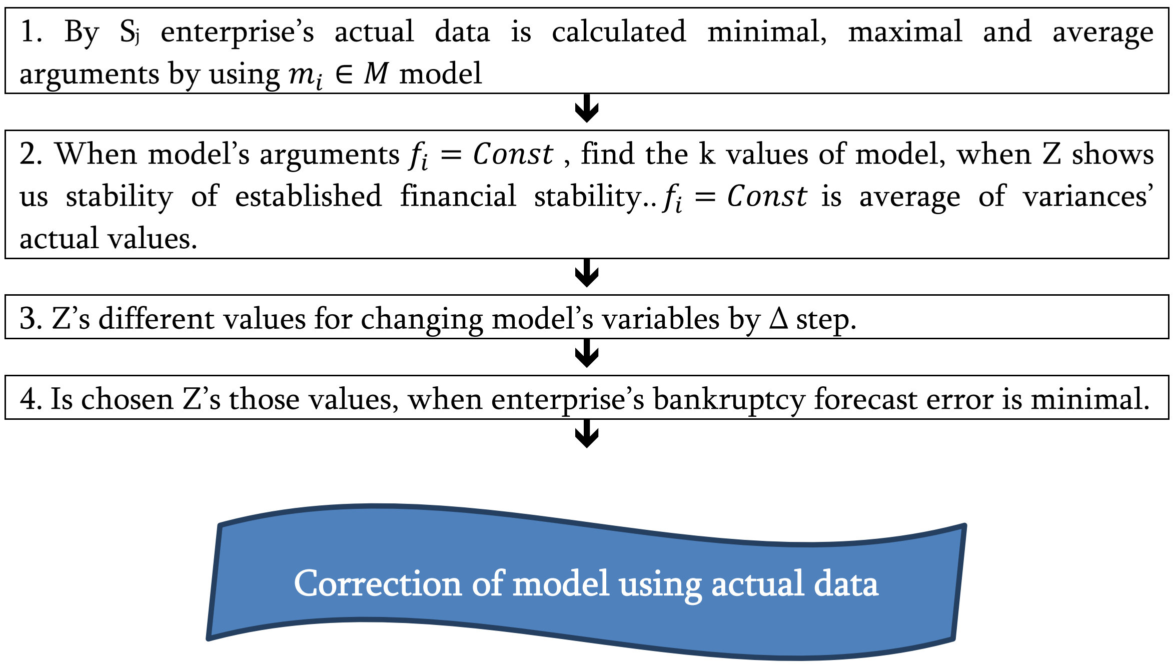
Both tasks are the task of optimization. For the 14 reviewed models only two of them (Fulmer, Olson) are not linear optimization, and the rest twelve - linear optimization.

The first task - based on the sample model S_j - the development of economic mathematical model of financial sustainability of the enterprise is simulated by the method /Figure 1/

Developing enterprise financial sustainability and forecasting special model

Development of the economic-mathematical model of the financial sustainability assessment of the enterprise (simplified model of the concrete enterprise and selection of the coefficients of this model) is implemented by the software package FINSIM1_PRO2019[89]. Program's interface is multilingual. One can work with it in Georgian, English and other languages [1,2].

For S_i enterprise, specifically JSC "Telasi", compilaiton of financial stability's assessment and bankruptcy forecast of economic-mathematical model is carrying out by a user, who has got appropriate permissions. The user is allowed to edit data, work with the system in training mode and, what's more important, create specific model for certain enterprise and save calculated results in database. Work with FINSIM1_PRO2019 is carried out in the following sequence:



1. By S_j enterprise’s actual data is calculated minimal, maximal and average arguments by using $m_i \in M$ model

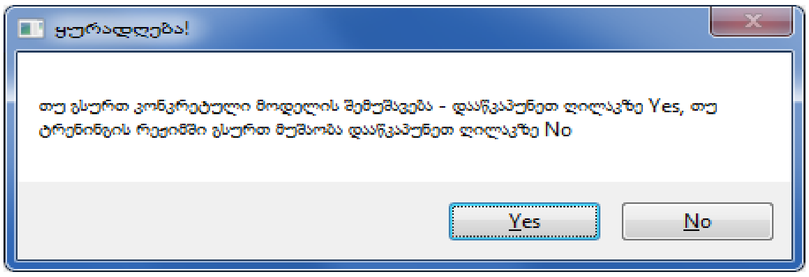
2. When model’s arguments $f_i = Const$, find the k values of model, when Z shows us stability of established financial stability.. $f_i = Const$ is average of variances’ actual values.

3. Z’s different values for changing model’s variables by Δ step.

4. Is chosen Z’s those values, when enterprise’s bankruptcy forecast error is minimal.

Figure. 1. The stages of developing an economic mathematical model of the financial sustainability assessment and forecasting of the enterprise

1. After choosing interface language and writing ID number and password the next window will pop out /Figure 2/



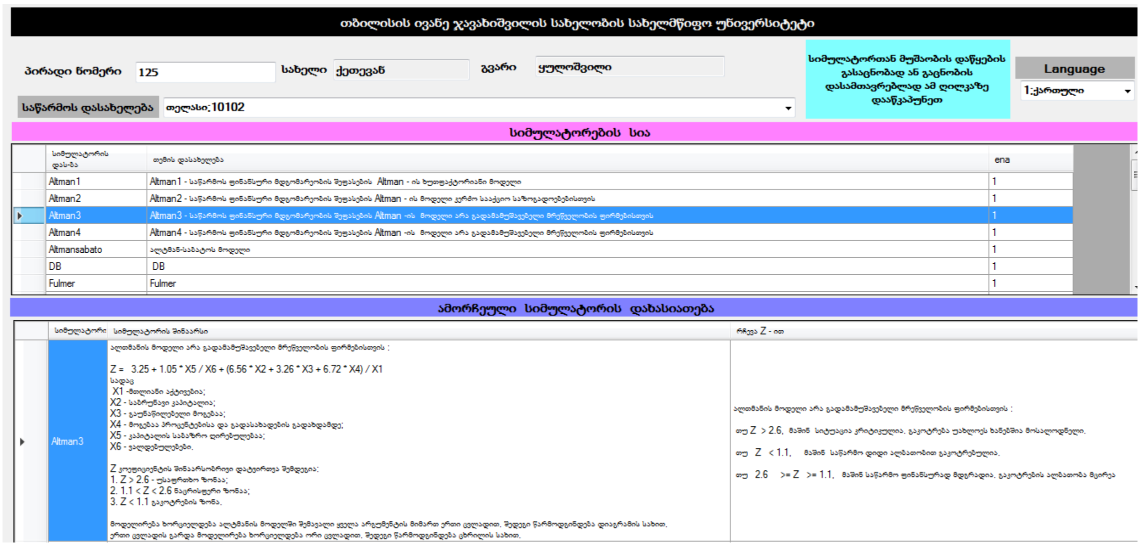
ყურადღება!

თუ გსურთ კონკრეტული მოდელის შემუშავება - დააწკაპუნეთ ღილაკზე Yes, თუ ტრენინგის რეჟიმში გსურთ მუშაობა დააწკაპუნეთ ღილაკზე No

Figure 2. Model development message

2. After pressing “Yes” button, enterprise’s name will automatically appear and list of ethernon models of financial sustainability assessment and forecasting of enterprise will roll out, based on which enterprise’s concrete economic-mathematical model must be developed. If we change enterprise, which was

chosen by system, after it list of ethernon models of financial sustainability assessment and forecasting of enterprise will roll out. In both cases we can choose preferred model from the list of models, particularly Altman's model for non-manufacturing enterprises with a conditional name Altman3 /Pic. 3/. In the same window, click on "Click on this button to get started working with the simulator" to start watching the video about how to get started with the simulator.


თბილისის ივანე ჯავახიშვილის სახელობის სახელმწიფო უნივერსიტეტი

პირადი ნომერი	125	სახელი	ქეთევან	გვარი	ყულოშვილი
საწარმოს დასახელება	თელასი;10102				

სიმულატორთან მუშაობის დაწყების გასაცნობად ან გაცნობის დასამთავრებლად ამ ღილკაზე დააწკაპუნეთ

Language: 1;ქართული

სიმულატორების სია

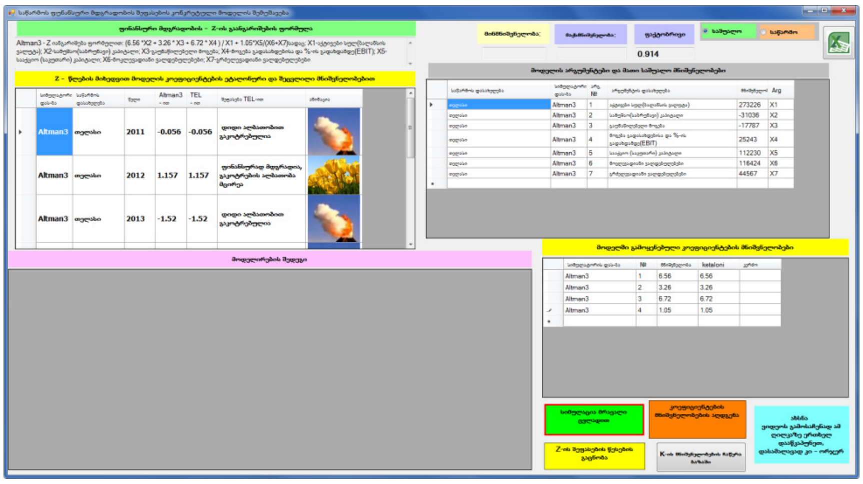
სიმულატორის დას-ბა	თემის დასახელება	ena
Altman1	Altman1 - საწარმოს ფინანსური მდგომარეობის შეფასების Altman - ის ხუთფაქტორიანი მოდელი	1
Altman2	Altman2 - საწარმოს ფინანსური მდგომარეობის შეფასების Altman - ის მოდელი კერძო სააქციო საზოგადოებებისთვის	1
Altman3	Altman3 - საწარმოს ფინანსური მდგომარეობის შეფასების Altman -ის მოდელი არა გადამამუშავებელი მრეწველობის ფირმებისთვის	1
Altman4	Altman4 - საწარმოს ფინანსური მდგომარეობის შეფასების Altman -ის მოდელი არა გადამამუშავებელი მრეწველობის ფირმებისთვის	1
Altmansabato	ალტმან-საბატოს მოდელი	1
DB	DB	1
Fulmer	Fulmer	1

ამორჩეული სიმულატორის დახასიათება

სიმულატორი	სიმულატორის შინაარსი	რჩევა Z - ით
Altman3	ალთმანის მოდელი არა გადამამუშავებელი მრეწველობის ფირმებისთვის : Z = 3.25 + 1.05 * X5 / X6 + (6.56 * X2 + 3.26 * X3 + 6.72 * X4) / X1 სადაც X1 -მთლიანი აქტივებია; X2 - საბრუნავი კაპიტალია; X3 - გაუნაწილებელი მოგებაა; X4 - მოგებაა პროცენტებისა და გადასახადების გადახდამდე; X5 - კაპიტალის საბაზრო ღირებულებაა; X6 - ვალდებულებები. Z კოეფიციენტის შინაარსობრივი დატვირთვა შემდეგია: 1. Z > 2.6 - უსაფრთხო ზონაა; 2. 1.1 < Z < 2.6 ნაცრისფერი ზონაა; 3. Z < 1.1 გაკოტრების ზონა. მოდელირება ხორციელდება ალტმანის მოდელში შემავალი ყველა არგუმენტის მიმართ ერთი ცვლადით. შედეგი წარმოდგინდება დიაგრამის სახით. ერთი ცვლადის გარდა მოდელირება ხორციელდება ორი ცვლადით. შედეგი წარმოდგინდება ცხრილის სახით.	ალთმანის მოდელი არა გადამამუშავებელი მრეწველობის ფირმებისთვის : თუ Z > 2.6, მაშინ სიტუაცია კრიტიკულია, გაკოტრება უახლოეს ხანებში მოსალოდნელი. თუ Z < 1.1, მაშინ საწარმო დიდი ალბათობით გაკოტრებულია. თუ 2.6 >= Z >= 1.1, მაშინ საწარმო ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა

Pic. 3. Window after user's identification and choosing enterprise and model

First step of specific model developing: In pic.3 in the table "Description of selected simulator" after marking row Altman3 a window /Pic. 4/ will appear. The values of the coefficients used in the table "model" column "K etalon" are recorded in the sampled model by us - the values of the coefficients used in the Altman model, namely: b1 = 6,56; b2 = 3,26; b3 = 6,72; b4 = 1,05. The same values are repeated in the column "value". In the table "The model arguments and their average values" the names of the arguments in the Altman's model are written and their average values are calculated, in the field "Z financial sustainability's calculation formula" Z's calculating formula with selected model is shown, in field "actual" Z's average actual value, which is calculated from Z's average values, is shown, and in the table "Z-year-by-year model coefficients of phonetic and altered values" is calculated according to the values used in the Altman sampling model of the Z values - column "Altman3" and coefficients, changed by us, - (column "value", table "coefficients, used in the model") Z's values from the column "TEL". According to Z's value, enterprise sustainability's rates are visualized by text and pictures. On the first step of modeling our coefficients aren't changed, so Altman3 and TEL values are equal.


საწარმოს ფინანსური მდგრადობის შეფასების კონკრეტული მოდელის შემუშავება

ფინანსური მდგრადობის - Z-ის გაანგარიშების ფორმულა

Altman3 - Z იანგარიშება ფორმულით: (6.56 *X2 + 3.26 * X3 + 6.72 * X4) / X1 + 1.05*X5/(X6+X7)სადაც: X1-აქტივები სულ(ბალანსის ვალუტა); X2-სამუშაო(საბრუნავი) კაპიტალი; X3-გაუნაწილებელი მოგება; X4-მოგება გადასახდებისა და %-ის გადახდამდე(EBIT); X5-სააქციო (საკუთარი) კაპიტალი; X6-მოკლევადიანი ვალდებულებები; X7-გრძელვადიანი ვალდებულებები

Z - წლების მიხედვით მოდელის კოეფიციენტების ეტალონური და შეცვლილი მნიშვნელობებით

სიმულატორის დას-ბა	საწარმოს დასახელება	წელი	Altman3 -ით	TEL -ით	შეფასება TEL-ით	ამინდი
Altman3	თელასი	2011	-0.056	-0.056	დიდი ალბათობით გაკოტრებულია	
Altman3	თელასი	2012	1.157	1.157	ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა	
Altman3	თელასი	2013	-1.52	-1.52	დიდი ალბათობით გაკოტრებულია	

მოდელირების შედეგი

მინიმნიშვნელობა: | მაქსიმნიშვნელობა: | ფაქტორივი 0.914 | ● საშუალო ○ საწარმო

მოდელის არგუმენტები და მათი საშუალო მნიშვნელობები

საწარმოს დასახელება	სიმულატორის დას-ბა	არგ. №	არგუმენტის დასახელება	მნიშვნელობა	Arg
თელასი	Altman3	1	აქტივები სულ(ბალანსის ვალუტა)	273226	X1
თელასი	Altman3	2	სამუშაო(საბრუნავი) კაპიტალი	-31036	X2
თელასი	Altman3	3	გაუნაწილებელი მოგება	-17787	X3
თელასი	Altman3	4	მოგება გადასახდებისა და %-ის გადახდამდე(EBIT)	25243	X4
თელასი	Altman3	5	სააქციო (საკუთარი) კაპიტალი	112230	X5
თელასი	Altman3	6	მოკლევადიანი ვალდებულებები	116424	X6
თელასი	Altman3	7	გრძელვადიანი ვალდებულებები	44567	X7

მოდელში გამოყენებული კოეფიციენტების მნიშვნელობები

სიმულატორის დას-ბა	№	მნიშვნელობა	ketaloni	[illegible]
Altman3	1	6.56	6.56	
Altman3	2	3.26	3.26	
Altman3	3	6.72	6.72	
Altman3	4	1.05	1.05	

სიმულაცია მრავალი ცვლადით
კოეფიციენტების მნიშვნელობების აღდგენა
ახსნა ვიდეოს გამოსაჩენად ამ ღილაკზე ერთხელ დააწკაპუნეთ, დასამალად კი - ორჯერ
Z-ის შეფასების წესების ჯგუფი
K-ის მნიშვნელობების ნაწილი

Pic.4. Model developing first step's window

Second step of specific model developing: The determination of the values of the b_i-coefficients in the model is performed on the second and third steps of the development of a specific model. In the second step, the $b_i \in B$ will be selected from the multiplicity of the coefficients, during which the value of Z is accepted within the permissible limits.

Any sample model consists of N number coefficient, so before the start of the second step of the modeling it is necessary to select the coefficient that significantly affects on Z's importance. For this purpose it is necessary to examine the value of Z in the value of the output of any coefficient of the $\forall b_i$ component in the model. The argument of the $\forall b_i$ significance Δ is the change in b.

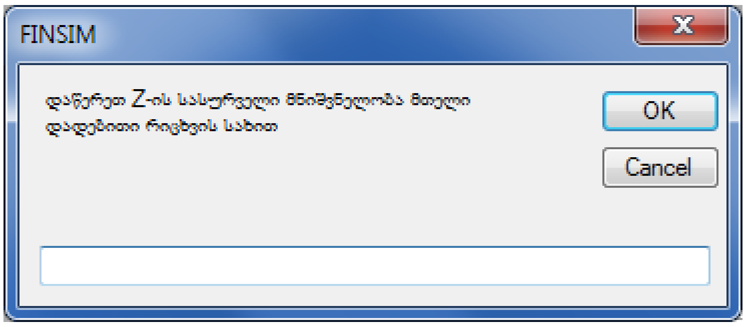
In order to determine any $\forall b_i$ portion of the model in the value of Z, press the "weight calculator" button. The message "If the value of the coefficient is increased, write down the number of cycles in the whole number, and if the value of the coefficient is reduced - write the decimal or integer minus sign" will appear.

The number of cycles is calculated when writing a negative number N=b_i/n, where n<0.

The calculation revealed that the values of the B3 coefficient 22.72-38.22 in the values of Z are derived from the values of Z, whereas the production is stable and in the 38.72-47.22 range - the unstable situation. The probability of bankruptcy is small but not excluded.

With the change of the importance of the b_i-coefficient in terms of the role, it has been calculated for years with the desired value of Z. The calculations are as follows:

In the table "The values of the coefficients used in the table" column "value" click the highest-weight coefficient - the value of the b4 coefficient in our case. The message will appear /Figure 5/

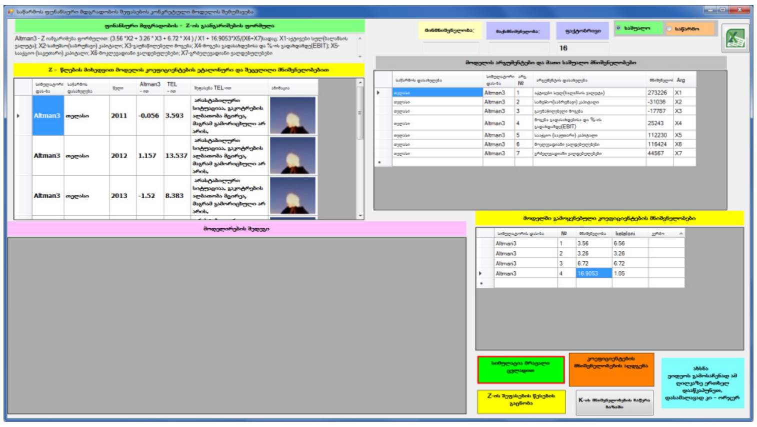


დაწერეთ Z-ის სასურველი მნიშვნელობა მთელი დადებითი რიცხვის სახით

Figure 5. Z's value pointing window

After writing Z value and pressing OK button the values of the Z will be calculated according to the system by years, their assessment will be presented, and the results will be reflected in the table " /Figure 6/.

Third step of specific model developing According to the values of the coefficients selected on the second step of the modeling: first - With a certain set of Z changes, many variables of Z are calculated. second - the variants of the b_i-coefficients will be selected from the variants which the value of Z is the adequate factor of the enterprise. For this purpose:



მოდელის არგუმენტები და მათი საშუალო მნიშვნელობები

საწარმოს დასახელება	სიმულატორის დას-ბა	არგ. №	არგუმენტის დასახელება	მნიშვნელობა	Arg
თელასი	Altman3	1	აქტივები სულ(ბალანსის ვალუტა)	273226	X1
თელასი	Altman3	2	საბრუნავი(საბრუნვო) კაპიტალი	-31036	X2
თელასი	Altman3	3	გაუნაწილებელი მოგება	-17787	X3
თელასი	Altman3	4	მოგება გადასახადებისა და %-ის გადახდამდე(EBIT)	25243	X4
თელასი	Altman3	5	სააქციო (საკუთარი) კაპიტალი	112230	X5
თელასი	Altman3	6	მოკლევადიანი ვალდებულებები	116424	X6
თელასი	Altman3	7	გრძელვადიანი ვალდებულებები	44567	X7

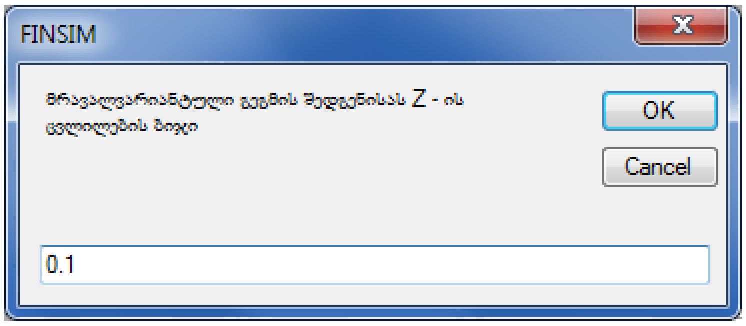
მოდელში გამოყენებული კოეფიციენტების მნიშვნელობები

სიმულატორის დას-ბა	№	მნიშვნელობა	ketaloni	კურსი
Altman3	1	3.56	6.56	
Altman3	2	3.26	3.26	
Altman3	3	6.72	6.72	
Altman3	4	16.9053	1.05	

			Altman3	TEL
Altman3	თელასი	2011	-0.056	3.593
Altman3	თელასი	2012	1.157	13.537
Altman3	თელასი	2013	-1.52	8.383

Figure 6. Z’s calculated values by changing b4 coefficient and pointing Z=16 by years

Click the button "Simulate multiple variables". The window will change the Z to change the step. For example, write 0.1 /Pic. 7/



FINSIM

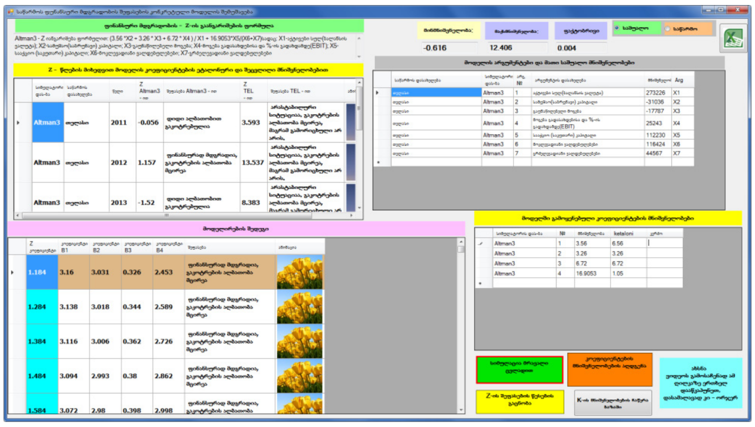
მრავალვარიანტული გეგმის შედგენისას Z - ის ცვლილების ბიჯი

0.1

Pic. 7. Set step of changing of Z coefficient

After pressing OK: First – model arguments’ average values are calculated; second - system will set as default model coefficient’s minimal value as 0,001, and maximal values – values, which were got on the second step of calculations, in particular: b_1=3,56; b_2=3,26; b_3=6,72; b_4=16,9053. third – Z’s actual, minimal and maximal values are calculated. We will get, Z_{actual} =0,004, Z_{min} = -616, Z_{max} =12,406. fourth – Z’s many variants are calculated. Calculation results will appear in the table “Results of modeling” /Figure 8/;

Variants differ by coeeficients’ values. For each variant, the value of Z is calculated and evaluated enterprise’s financial stability according to the Altman model.



Altman3	თელასი	2011	-0.056	დიდი ალბათობით გაკოტრებულია	3.593	არასტაბილური სიტუაციაა, გაკოტრების ალბათობა მცირეა, მაგრამ გამორიცხული არ არის,
Altman3	თელასი	2012	1.157	ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა	13.537	არასტაბილური სიტუაციაა, გაკოტრების ალბათობა მცირეა, მაგრამ გამორიცხული არ არის,
Altman3	თელასი	2013	-1.52	დიდი ალბათობით გაკოტრებულია	8.383	არასტაბილური სიტუაციაა, გაკოტრების ალბათობა მცირეა,

მოდელირების შედეგი

Z	B1	B2	B3	B4		
1.184	3.16	3.031	0.326	2.453	ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა	
1.284	3.138	3.018	0.344	2.589	ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა	
1.384	3.116	3.006	0.362	2.726	ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა	
1.484	3.094	2.993	0.38	2.862	ფინანსურად მდგრადია, გაკოტრების ალბათობა მცირეა	
1.584	3.072	2.98	0.398	2.998	ფინანსურად მდგრადია, გაკოტრების ალბათობა	

-0.616	12.406	0.004

		№			Arg
თელასი	Altman3	1	აქტივები სულ (ბალანსის ვალუტა)	273226	X1
თელასი	Altman3	2	საბრუნავი (საბრუნავი) კაპიტალი	-31036	X2
თელასი	Altman3	3	გაუნაწილებელი მოგება	-17787	X3
თელასი	Altman3	4	მოგება გადასახადებისა და %-ის გადახდამდე (EBIT)	25243	X4
თელასი	Altman3	5	სააქციო (საწესდებო) კაპიტალი	112230	X5
თელასი	Altman3	6	მოკლევადიანი ვალდებულებები	116424	X6
თელასი	Altman3	7	გრძელვადიანი ვალდებულებები	44567	X7

მოდელში გამოყენებული კოეფიციენტების მნიშვნელობები

	№		ketaloni	
Altman3	1	3.56	6.56	
Altman3	2	3.26	3.26	
Altman3	3	6.72	6.72	
Altman3	4	16.9053	1.05	

Figure 8. Result of modeling with many variables

In the table “modeling results” Z’s values, calculated by arguments’ average values, JSC “Telasi” is stable in range 1.184-2.584. Each variant’s b_i coefficient value is assigned to a certain set of values.

Fourth step of specific model developing: Our goal is to select from range of variants that one, which is based on the factual data according to the fact that the value of the Z is adequately reflected in the financial position of the enterprise. In order to achieve this goal it is enough to indicate the result in the table "modeling result". After selecting the row selected result will appear in the table “Coefficient values changed by Z years” /Figure 9/. We can print this table by pressing button after what our table will be exported in Excel /Table 1/.

The table 3.1 shows the estimates obtained by the selected coefficients according to years.

Table 1 shows that from 7 cases in 5 of them, the TEL1 and Altman3 calculations do not match. The enterprise is bankrupt by Altman3, in fact the enterprise is financially stable, the probability of bankruptcy is small. Stable, it seems, the TEL1 calculations make an adequate picture of reality. We can express the opinion that the model of the financial position and bankruptcy forecasts of JSC "Telasi", which we received, is a reflection of the TEL1 reality. At any time we can adjust the TEL1 coefficients.

Table 1. Evaluation of the finerness obtained by the equations of the coefficients and the values obtained by modeling

Simulator’s name	Name of Enterprise	Year	Z by Altman3	Rate Altman3	Z by TEL	Rate by TEL
Altman3	Telasi	2011	-0.056	Very likely is bankrupt	2.83	The unstable situation. The probability of bankruptcy is small but not excluded.
Altman3	Telasi	2012	1.157	Financially sustainable, the probability of bankruptcy is small	10.552	The unstable situation. The probability of bankruptcy is small but not excluded.
Altman3	Telasi	2013	-1.52	Very likely is bankrupt	5.999	The unstable situation. The probability of bankruptcy is small but not excluded.
Altman3	Telasi	2014	1.081	Very likely is bankrupt	10.333	The unstable situation. The probability of bankruptcy is small but not excluded.
Altman3	Telasi	2015	-0.388	Very likely is bankrupt	9.021	The unstable situation. The probability of bankruptcy is small but not excluded.
Altman3	Telasi	2016	1.395	Financially sustainable, the probability of bankruptcy is small	14.846	The unstable situation. The probability of bankruptcy is small but not excluded.
Altman3	Telasi	2017	0.914	Very likely is bankrupt	12.378	The unstable situation. The probability of bankruptcy is small but not excluded.

In Figure 10 the table with coefficients, which were got after selection of variant, is shown

Fix calculation results – inserting data in database is implemented by pressing button „Insert K values in database“

Based on actual data, TEL1 can provide a report for any t year on the financial sustainability of an enterprise with actual data. For this:

- After completing multivariate calculation field “Enterprise” will be automatically checked and the window for writing year will pop out. The same result will be achieved after checking field “Enterprise”;
- Write a year and click "Financial sustainability evaluation" button. The result of calculation will be opened in Excel as a table;
- To get a report according to all the years on the enterprise database, the "Year" field should be empty. By clicking on the " Financial sustainability evaluation " button, the report will be appeared for every year.

According to years of different models, the report of assessing the financial sustainability of the enterprise after the completion of the program.

FINSIM1_PRO2019 is written on VB.NET, database is organized on SQL Server.

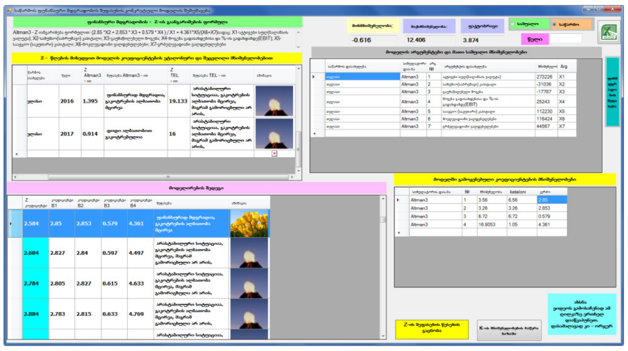
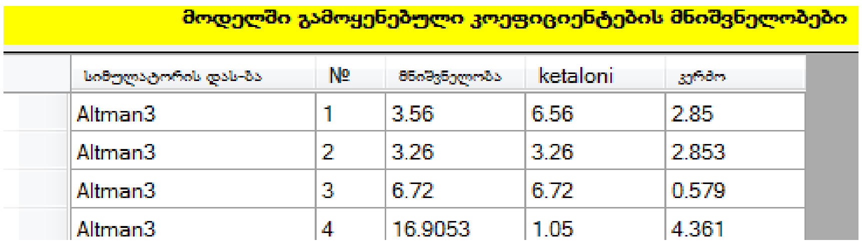


Figure 9. Variant selection



მოდელში გამოყენებული კოეფიციენტების მნიშვნელობები

სიმულატორის დას-ბა	№	მნიშვნელობა	ketaloni	კერძო
Altman3	1	3.56	6.56	2.85
Altman3	2	3.26	3.26	2.853
Altman3	3	6.72	6.72	0.579
Altman3	4	16.9053	1.05	4.361

Figure 10. Coefficients, got after selecting variant

Conclusion

1. Enterprise's bankrupt and financial sustainability forecast in any existing economic-mathematical model uses country and industry's long-term statistical data. In countries with the transition to the market econom, such as Georgia, long-term statistical data of bankruptcy does not exist, so the determination of coefficients is not possible by using stastistic methods, and the use of existing models does not make the desired result.
2. It is justified that the rational way of solving the problem is simulation modeling.
3. Our simulator FINSIM_PRO2019 allows:
   - On the basis of a general model that can be considered as a reference model, a particular economic mathematical model of the financial sustainability assessment of the enterprise can be utilized in the actual activity of the problematic area;
   - To be used by different countries with a single legal space, including Georgia, to prepare specialists in different universities;
   - Specialists employed in management area: economists, financers and others.

Bibliography

[Altman, 2009] Altman Z-score. https://en.wikipedia.org/wiki/Altman_Z-score

[Munjishvili, 2018] Tea Munjishvili. Estimation and Predicting of Enterprise's Financial Stability by Simulation Model. (St. Petersburg) 2018. https://elibrary.ru/item.asp?id=35506557

[Munjishvili, 2018] Tea Munjishvili. Predicting of Enterprise's Financial Stability by Simulation Models. Globalization and business, #5 / 2018 http://www.eugb.ge/uploads/content/N5/5-31.pdf

http://www.statisticshowto.com/probability-and-statistics/z-score/

Authors' Information

Tea Munjishvil - *Iv.Javakhishvili Tbilisi State University; Chavchavadze Av.1, 0129, Tbilisi, Georgia.*
e-mail : tmunjishvili@gmail.com , tea.munjishvili@tsu.ge

TABLE OF CONTENTS