Advances in Intelligent Systems and Computing 512

Natalya Shakhovska *Editor*

Advances in Intelligent Systems and Computing

Selected Papers from the International Conference on Computer Science and Information Technologies, CSIT 2016, September 6–10 Lviv, Ukraine

Springer

Advances in Intelligent Systems and Computing

Volume 512

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing.

The publications within "Advances in Intelligent Systems and Computing" are primarily textbooks and proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

Advisory Board

Chairman

Nikhil R. Pal, Indian Statistical Institute, Kolkata, India
e-mail: nikhil@isical.ac.in

Members

Rafael Bello, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba
e-mail: rbellop@uclv.edu.cu

Emilio S. Corchado, University of Salamanca, Salamanca, Spain
e-mail: escorchado@usal.es

Hani Hagras, University of Essex, Colchester, UK
e-mail: hani@essex.ac.uk

László T. Kóczy, Széchenyi István University, Győr, Hungary
e-mail: koczy@sze.hu

Vladik Kreinovich, University of Texas at El Paso, El Paso, USA
e-mail: vladik@utep.edu

Chin-Teng Lin, National Chiao Tung University, Hsinchu, Taiwan
e-mail: ctlin@mail.nctu.edu.tw

Jie Lu, University of Technology, Sydney, Australia
e-mail: Jie.Lu@uts.edu.au

Patricia Melin, Tijuana Institute of Technology, Tijuana, Mexico
e-mail: epmelin@hafsamx.org

Nadia Nedjah, State University of Rio de Janeiro, Rio de Janeiro, Brazil
e-mail: nadia@eng.uerj.br

Ngoc Thanh Nguyen, Wroclaw University of Technology, Wroclaw, Poland
e-mail: Ngoc-Thanh.Nguyen@pwr.edu.pl

Jun Wang, The Chinese University of Hong Kong, Shatin, Hong Kong
e-mail: jwang@mae.cuhk.edu.hk

More information about this series at http://www.springer.com/series/11156

Natalya Shakhovska
Editor

Advances in Intelligent Systems and Computing

Selected Papers from the International Conference on Computer Science and Information Technologies, CSIT 2016, September 6–10 Lviv, Ukraine

Springer

Editor
Natalya Shakhovska
Lviv Polytechnic National University
Lviv
Ukraine

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-319-45990-5 ISBN 978-3-319-45991-2 (eBook)
DOI 10.1007/978-3-319-45991-2

Library of Congress Control Number: 2016950408

© Springer International Publishing AG 2017
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

An Improved Particle Swarm Optimization Algorithm with Repair Procedure 1
Bożena Borowska

Fuzzy Evaluation Method for Project Profitability 17
Rekik Ali, Gabsi Mounir, Valentina Emilia Balas and Masmoudi Nissen

New Approach for Saving Semistructured Medical Data 29
Iryna Shvorob

Architecture and Conceptual Bases of Cloud IT Infrastructure Management 41
Sergii Telenyk, Eduard Zharikov and Oleksandr Rolik

Generation of the Association Rules Among Multidimensional Data in DBMS Caché Environment 63
Mykola Fisun and Hlib Horban

Threat Assessment Method for Intelligent Disaster Decision Support System 81
Maryna Zharikova and Volodymyr Sherstjuk

Mobile Activation Learning System Using Gamification Approach 101
V.F. Emets, Jan Rogowski and Jacek Krasiukianis

University's Information Image as a Result of University Web Communities' Activities 115
Roman Korzh, Andriy Peleshchyshyn, Yuriy Syerov and Solomia Fedushko

Multi-modular Optimum Coding Systems Based on Remarkable Geometric Properties of Space 129
Volodymyr Riznyk

The Method of Statistical Analysis of the Scientific, Colloquial, Belles-Lettres and Newspaper Styles on the Phonological Level 149
Iryna Khomytska and Vasyl Teslyuk

The Optimal Aggregation of Integrated Regional Systems "Production, Waste Recycling" 165
Taisa Borovska, Pavel Severilov, Irina Kolesnik and Victor Severilov

Intelligent Systems Design of Distance Learning Realization for Modern Youth Promotion and Involvement in Independent Scientific Researches 175
Nataliya Shakhovska, Victoria Vysotska and Lyubomyr Chyrun

An Ontology-Based Approach for User Interface Adaptation 199
Makram Soui, Soumaya Diab, Ali Ouni, Aroua Essayeh and Mourad Abed

Algebraic Framework for Knowledge Processing in Systems with Situational Awareness 217
Khrystyna Mykich and Yevhen Burov

Classification Methods of Text Documents Using Ontology Based Approach 229
Vasyl Lytvyn, Victoria Vysotska, Oleh Veres, Ihor Rishnyak and Halya Rishnyak

The Identification of the Operator's Systems Images Using the Method of the Phase Portrait 241
Natalya Shakhovska, Lilia Nych and Roman Kaminskyj

Concept Implementation of Decision Support Software for the Risk Management of Complex Technical System 255
Victor Boyko, Nicolay Rudnichenko, Sergey Kramskoy, Yevhen Hrechukha and Natalia Shibaeva

The Model of Data Analysis of the Psychophysiological Survey Results 271
Volodymyr Pasichnyk and Tetiana Shestakevych

Two Algorithms Median Filtering to Identify the Time Series Trend 283
Helen Dmitriv and Roman Kaminsky

Use Electric and Acoustic Technologies for Automated Control of Water 293
Maryna Mikhalieva, Nataliya Hots, Mykola Mykyychuk and Yuliia Dzikovska

Analysis of Clustering Algorithms 305
Iryna Zheliznyak, Zoriana Rybchak and Iryna Zavuschak

Semantic Search Personalized Data as Special Method of Processing Medical Information 315
Natalia Melnykova

An Improved Particle Swarm Optimization Algorithm with Repair Procedure

Bożena Borowska

Abstract In this paper a new particle swarm optimization algorithm called RPSO for solving high dimensional optimization problems is proposed and analyzed both in terms of their efficiency, the ability to avoid local optima and resistance to the problem of premature convergence. In RPSO, a repair procedure was introduced the aim of which was to determine new, better velocities for some particles, when their current velocities are inefficient. New velocities are the functions of previous and current velocities. The new algorithm was tested with a set of benchmark functions and the results were compared with those obtained through the standard PSO (SPSO) and IPSO. Simulation results show that new RPSO is faster and more effective than the standard PSO and IPSO.

Keywords Optimization · Particle swarm optimization · Swarm intelligence · Improved particle swarm optimization

1 Introduction

Particle swarm optimization (PSO) is a stochastic, based on the swarm intelligence, optimization method, introduced by Kennedy and Eberhart [1, 2]. Because of its simplicity, a relatively low computational cost and easy implementation, it has been applied to solve many different optimization and engineering problems [3–8]. However, in case complex, multidimensional surface with many local optima, standard particle swarm optimization (SPSO) can encounter some problems in finding an optimal solution. Moving towards an optimum, the algorithm tends to

B. Borowska (✉)
Institute of Information Technology, Lodz University of Technology,
Wólczańska 215, 90-924 Lodz, Poland
e-mail: bozena.borowska@p.lodz.pl

© Springer International Publishing AG 2017

N. Shakhovska (ed.), *Advances in Intelligent Systems and Computing*,
Advances in Intelligent Systems and Computing 512,
DOI 10.1007/978-3-319-45991-2_1

premature converge to one of the points of the search space, can be very slow and requires thousands of iterations. Moreover, the SPSO algorithm can stop optimizing when reaching a near optimal solution or trap into local optima and never escapes. A lot of various attempts have been made to overcome these problems and improve the performance of SPSO. They include:

- adjustment of basic control parameters (such as inertia weight, acceleration coefficients) [9–12],
- modification of the velocity updating equation [12–15],
- division of a population into sub-swarms [16],
- hybrid algorithms, which combine PSO with other methods like GA [17–19] or SA [20, 21],
- application of a fuzzy system [22–26].

This paper presents a novel particle swarm optimization called RPSO for solving high dimensional optimization problems. In RPSO, a repair procedure was introduced, which relies on determination of new, better velocities for some particles when their current velocities are inefficient. The new velocity is a function of previous and present velocities. The new algorithm was tested with a set of benchmark functions [27, 28] and the results were compared with those obtained through the standard PSO and its improved variant IPSO[1] [16] with a population partitioned into sub-swarms that are shuffled at periodic stages in the evolution.

2 The Standard PSO

The standard PSO algorithm is an optimization method based on the behavior of the swarm and its intelligence. It starts with a population of particles, each of which is initialized with a random generated position vector $X_\mathrm{i} = (x_{\mathrm{i}1}, x_{\mathrm{i}2}, \ldots, x_{\mathrm{iD}})$ and a velocity vector $V_i = (v_{\mathrm{i}1}, v_{\mathrm{i}2}, \ldots, v_{\mathrm{iD}})$. In each iteration, particles move within the search space to find the best solution. The quality (fitness) of each particle is evaluated according to the objective function of the optimization problem. The best previously found position of the particle i is remembered in its memory as its personal best position $pbest_i = (pbest_{\mathrm{i}1}, pbest_{\mathrm{i}2}, \ldots, pbest_{\mathrm{iD}})$. The best position of the whole swarm is remembered in memory of the swarm as the global best position $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$. New positions and velocities of the particles are updated according to the following equations:

$$V_i(t+1) = wV_i(t) + c_1 r_1 (pbest_i - X_i(t)) + c_2 r_2 (gbest - X_i(t)) \tag{1}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2}$$

[1] Improved Particle Swarm Optimization [16].

where w is the inertia weight factor. This parameter determines the impact of a particle previous velocity on its current velocity, and affects the ability of global and local exploration. Factors c_1 and c_2 are acceleration coefficients that determine how much the particle is influenced by the memory of its best position and by the rest of the swarm, respectively, r_1 and r_2 represent randomly generated numbers in the range (0,1).

3 The Proposed RPSO Algorithm

In the proposed RPSO algorithm, the population consists of particles, each of which has its own position and velocity randomly generated during initialization. Both particle and its velocity are represented by D-dimensional vectors $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, respectively. The particles search a solution space, and remember the best position found by themselves as well as their last velocity. In each iteration (except the first one), after evaluating the particle position, a repair procedure is applied. First, from the whole swarm, p particles with the worst fitness are chosen. Next, for each p particle, two new velocities are calculated. These new velocities are the functions of their previous and current velocities determined according to the following formula:

$$velocity_i^1(t) = velocity_i(t)/(velocity_i(t) - velocity_i(t-1)) \tag{3}$$

$$velocity_i^2(t) = (velocity_i(t-1) - (velocity_i(t))/velocity_i(t-1)) \tag{4}$$

One of the new calculated velocities replaces the previous velocity but only when the particle's new position is closer to *gbest*, and the distance between the particle and *gbest* (calculated by Euclidean distance) is greater than zero. In this iteration, new positions of the particles, obtained through the new velocities, are not evaluated by means of the fitness function. In the next step, for the remaining particles as well as for those p particles for which the repair procedure was not successful, a new velocity and a new position are calculated according to the Eqs. (1) and (2). Next, the quality of the solutions represented by all the particles is measured by means of the fitness function. For each particle, the best position found so far and the best position within the entire swarm are established. These steps are repeated until the stopping criterion is met.

In this way, RPSO algorithm can improve considerably the performance of the PSO with low computational cost. The algorithm can be applied for solving high dimensional optimization problems. However, in case of small swarms, the algorithm can premature converge to one of the points of the search space.

4 Results

The simulation tests of the proposed algorithms were carried out on the set of benchmark function and the results were then compared with the performance of the standard PSO algorithm, as well as with IPSO.

For all these algorithms, a set of parameters recommended by Trelea [27] with inertia weight $w = 0.6$ and acceleration constants $c_1 = c_2 = 1.7$ were used. For RPSO, the w parameter was linearly decreasing from 0.6 to 0.475. The number of particles with the worst fitness p was set as 3.

For all the functions, the tests with three different dimension sizes $D = 10$, 20 and 30, for $N = 20$, 40 and 80 particles in the swarm, respectively, were performed. A fixed number of maximum iterations 1000 was established for all the algorithms.

The information about the functions, the admissible range of the variable, and the optimum used for the investigation are depicted in Table 1.

The exemplary results (mean function value, minimum, maximum, and standard deviation) of the tests performed for 20, 40 and 80 particles of the swarm are illustrated in Tables 2, 3, 4 and 5. The presented values were averaged over 50 trials.

The average best fitness in the following iterations for both RPSO, IPSO algorithms and SPSO model for 40 particles (swarm size) and 30 dimensions are illustrated in Figs. 1, 2, 3 and 4. The vertical coordinates indicate the average best fitness in the form of logarithm value.

The results of simulations for benchmark test nonlinear function show that the proposed algorithm with repair procedure gives superior optimization performance over the standard PSO and IPSO (with sub-swarms). For all the considered functions, the minimum and mean function values after 1000 iterations found by RPSO are lower than the results obtained for the other algorithms (Tables 2, 3, 4, 5). The standard deviation calculated for the RPSO is also lower what means that the algorithm is more stable. For Ackley and Griewank function, the new algorithms had also faster convergence than SPSO and IPSO as shown in Figs. 3 and 4. In case of Rosenbrock and Rastrigin function, the IPSO algorithm was initially as fast as RPSO (or even faster, Figs. 1, 2). However, after about 100 iterations, IPSO converged slower than RPSO but still better than standard PSO.

It should be noted that when the number of particles in the swarm was increased, the algorithm converged faster and the mean function value after 1000 iterations was closer to optimum. Using too few particles in the swarm gave greater dispersion of the results and higher difference between minimal and maximal values found by the swarm.

Table 1 Optimization test functions

Function	Formula	Minimum	Range of x
Sphere	$f_1 = \sum_{i=1}^{n} x_i^2$	0	(−100, 100)
Rosenbrock	$f_3 = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right.$	0	(−30, 30)
Griewank	$f_4 = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	0	(−600, 600)
Rastrigin	$f_5 = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	0	(−5.12, 5.12)
Ackley	$f_6 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	0	(−32, 32)

Table 2 Performance of the SPSO, IPSO and RPSO algorithms for Rosenbrock function

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
20	10	SPSO	3.9826e+001	1.5539e+001	5.7182e+000	6.9746e+001
		IPSO	1.2857e+001	9.6048e+000	1.0750e−005	2.0095e+001
		RPSO	1.2360e+001	1.1394e+001	1.1808e−006	1.9070e+001
	20	SPSO	8.9422e+001	3.7105e+001	1.5205e+001	1.6432e+002
		IPSO	7.1679e+001	3.2481e+001	1.0980e+000	1.2978e+002
		RPSO	5.0170e+001	2.8733e+001	8.8276e+000	8.3798e+001
	30	SPSO	1.4152e+002	1.7519e+002	2.1259e+001	8.3083e+002
		IPSO	1.3523e+002	9.4150e+001	2.3050e+001	4.7808e+002
		RPSO	1.3058e+002	1.4658e+002	2.4899e+000	6.3090e+002
40	10	SPSO	2.3952e+001	1.4272e+001	3.1325e−003	4.8156e+001
		IPSO	4.6512e+000	1.5996e+000	8.2945e−004	6.2716e+000
		RPSO	3.0685e+000	1.8423e+000	3.9409e−004	6.7658e+000
	20	SPSO	5.8783e+001	4.1653e+001	6.1272e+000	7.4513e+001
		IPSO	2.1732e+001	2.7055e+001	7.2713e−001	6.9242e+001
		RPSO	1.9626e+001	3.1362e+001	4.2012e−001	6.1643e+001
	30	SPSO	7.1643e+001	1.5214e+002	1.0082e+001	6.9345e+002
		IPSO	2.6921e+001	5.1621e+001	5.8379e+000	1.2230e+002
		RPSO	2.5262e+001	4.8905e+001	6.7201e−001	1.1412e+002

(continued)

Table 2 (continued)

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
80	10	SPSO	1.6312e+001	1.1842e+001	1.5491e−001	2.3422e+001
		IPSO	1.9864e−001	2.5293e+000	6.2411e−002	7.8831e+000
		RPSO	1.7732e−001	2.6497e+000	4.5018e−002	9.0105e+000
	20	SPSO	4.8356e+001	3.3905e+001	1.5273e−001	6.7410e+001
		IPSO	1.4514e+001	2.4261e+001	8.9103e−002	5.1302e+001
		RPSO	1.2722e+001	2.3502e+001	5.8073e−001	4.8809e+002
	30	SPSO	6.7341e+001	1.3856e+002	4.2625e−001	5.4703e+002
		IPSO	1.8753e+001	3.6872e+001	4.3502e−002	8.4038e+001
		RPSO	1.6172e+001	3.2847e+001	2.6372e−001	8.3716e+001

Table 3 Performance of the SPSO, IPSO and RPSO algorithms for Rastrigin function

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
20	10	SPSO	6.0861e+000	4.8873e+000	2.9849e+000	2.7859e+001
		IPSO	3.1724e+000	4.1259e+000	2.1322e+000	2.4861e+001
		RPSO	2.8507e+000	3.6813e+000	2.5848e+000	1.5919e+001
	20	SPSO	4.3380e+001	1.2097e+001	2.0894e+001	6.5667e+001
		IPSO	3.5927e+001	1.2113e+001	1.7231e+001	5.9647e+001
		RPSO	3.1411e+001	1.4215e+001	1.8416e+001	6.0533e+001
	30	SPSO	7.8203e+001	1.7954e+001	4.7758e+001	1.2149e+002
		IPSO	6.7182e+001	1.8065e+001	4.0674e+001	9.2319e+001
		RPSO	5.8562e+001	1.4302e+001	8.8749e+000	8.9062e+001
40	10	SPSO	4.0816e+000	4.2259e+000	0.9945e+000	1.4924e+001
		IPSO	2.9841e+000	3.9145e+000	0.4514e+000	1.1052e+001
		RPSO	2.0865e+000	2.8289e+000	0.2717e+000	1.0193e+001
	20	SPSO	2.1590e+001	9.8853e+000	1.0944e+001	3.3828e+001
		IPSO	1.9271e+001	7.5127e+000	0.8912e+001	2.7118e+001
		RPSO	1.7535e+001	9.1631e+000	0.9318e+001	2.9474e+001
	30	SPSO	6.9049e+001	2.9212e+001	3.7808e+001	1.4227e+002
		IPSO	5.6997e+001	2.2354e+001	2.9155e+001	1.0892e+002
		RPSO	4.7706e+001	2.1053e+001	2.7808e+001	1.0049e+002

(continued)

Table 3 (continued)

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
80	10	SPSO	2.3614e+000	4.2286e+000	0.9950e+000	6.9647e+000
		IPSO	1.8102e+000	3.1215e+000	0.2513e+000	5.8912e+000
		RPSO	1.2713e+000	1.8373e+000	0.2216e+000	5.3526e+000
	20	SPSO	2.1103e+001	9.3464e+000	1.2934e+001	3.7808e+001
		IPSO	1.6821e+001	8.1435e+000	0.9142e+001	3.1671e+001
		RPSO	1.5374e+001	7.1919e+000	0.7355e+001	2.8459e+001
	30	SPSO	5.3207e+001	1.4769e+001	4.5768e+001	9.5156e+001
		IPSO	2.4136e+001	1.4111e+001	1.9230e+001	5.7438e+001
		RPSO	2.1198e+001	1.3447e+001	1.7898e+001	5.3868e+001

Table 4 Performance of the SPSO, IPSO and RPSO algorithms for Griewank function

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
20	10	SPSO	9.8191e−002	8.5931e−002	1.7226e−002	2.9240e−001
		IPSO	7.4177e−002	7.6792e−002	7.1179e−003	2.2386e−001
		RPSO	7.1255e−002	8.4535e−002	1.7226e−002	3.1734e−001
	20	SPSO	7.0779e−002	7.0743e−002	1.0031e−005	1.7921e−001
		IPSO	4.5634e−002	6.9889e−002	5.1899e−003	4.6754e−001
		RPSO	3.8430e−002	3.1461e−001	1.2212e−014	1.0766e−000
	30	SPSO	1.4819e−001	2.3667e−001	2.3633e−010	8.1062e−001
		IPSO	3.8124e−002	4.3762e−002	3.0314e−003	5.3174e−001
		RPSO	2.3537e−002	1.9389e−002	8.3105e−005	6.3420e−002
40	10	SPSO	7.5973e−002	7.2345e−002	7.3961e−003	1.1310e−001
		IPSO	6.5308e−002	5.9416e−002	6.0923e−003	2.7154e−001
		RPSO	6.3972e−002	2.6787e−002	9.8572e−003	1.0587e−001
	20	SPSO	5.9014e−002	6.1108e−002	8.1091e−003	2.2167e−001
		IPSO	4.0317e−002	6.3007e−002	5.8925e−003	2.9226e−001
		RPSO	3.7292e−002	2.5899e−002	1.2316e−003	8.6261e−002
	30	SPSO	3.2189e−002	4.6745e−002	7.9265e−003	2.0898e−001
		IPSO	2.4315e−002	4.2167e−002	5.3216e−003	2.7435e−001
		RPSO	1.5262e−002	9.2600e−003	1.8984e−014	2.9541e−002

(continued)

Table 4 (continued)

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
80	10	SPSO	5.4372e−002	3.0411e−002	1.9697e−002	1.0819e−001
		IPSO	4.3093e−002	6.0953e−002	5.4120e−003	4.1728e−002
		RPSO	4.1984e−002	2.6819e−002	2.9562e−002	1.1062e−001
	20	SPSO	1.7980e−002	1.2844e−002	0.0000e−000	3.6931e−002
		IPSO	2.1372e−002	2.6325e−002	0.0000e−000	4.3422e−002
		RPSO	2.1878e−002	2.3196e−002	0.0000e−000	7.3671e−002
	30	SPSO	1.4498e−002	1.7779e−002	0.0000e−000	4.8906e−002
		IPSO	2.1061e−002	2.2341e−002	0.0000e−000	3.8274e−002
		RPSO	1.5740e−002	1.0931e−002	0.0000e−000	3.6769e−002

Table 5 Performance of the SPSO, IPSO and RPSO algorithms for Ackley function

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
20	10	SPSO	3.1684e−001	6.6211e−001	3.9968e−015	2.0133e+000
		IPSO	1.9934e−001	4.3812e−001	3.9853e−015	1.6462e+000
		RPSO	1.7565e−001	3.5341e−001	3.9968e−015	1.1551e+000
	20	SPSO	2.3405e+000	1.4689e+000	8.5709e−014	4.6041e+000
		IPSO	1.8779e+000	5.7719e−001	6.5340e−001	2.7002e+000
		RPSO	1.5389e+000	6.5959e−001	1.7119e−001	2.1712e+000
	30	SPSO	4.2955e+000	1.3552e+000	2.4077e+000	6.0037e+000
		IPSO	2.7830e+000	7.5923e−001	1.5800e+000	3.8577e+000
		RPSO	2.5354e+000	7.1958e−001	1.1206e+000	3.6819e+000
40	10	SPSO	4.5983e−013	1.3556e−012	3.9968e−015	4.5264e−012
		IPSO	0.0000e+000	0.0000e+000	0.0000e+000	0.0000e+000
		RPSO	0.0000e+000	0.0000e+000	0.0000e+000	0.0000e+000
	20	SPSO	9.4644e−001	8.0979e−001	3.9968e−015	2.1697e+000
		IPSO	6.5472e−001	8.4720e−001	3.9968e−015	2.3224e+000
		RPSO	5.4472e−001	6.7807e−001	3.9968e−015	1.6462e+000
	30	SPSO	2.1083e+000	8.9137e−001	1.1552e+000	4.2969e+000
		IPSO	1.6884e+000	8.4243e−001	2.1099e−006	2.8123e+000
		RPSO	1.2989e+000	7.0879e−001	1.9508e−006	2.7368e+000

(continued)

Table 5 (continued)

Population size	Dimension	Algorithm	Mean function value	Standard deviation	Minimum	Maximum
80	10	SPSO	4.0412e−016	1.3322e−016	3.9968e−015	4.4408e−015
		IPSO	0.0000e+000	0.0000e+000	0.0000e+000	0.0000e+000
		RPSO	0.0000e+000	0.0000e+000	0.0000e+000	0.0000e+000
	20	SPSO	2.3103e−001	4.6206e−001	3.9968e−015	1.1551e+000
		IPSO	1.4235e−001	4.2706e−001	3.9968e−015	1.4235e+000
		RPSO	9.2951e−002	2.7886e−001	3.9968e−015	9.3425e−001
	30	SPSO	1.1058e+000	1.0422e+000	3.2419e−014	2.7368e−001
		IPSO	8.1875e−001	1.1504e+000	1.4655e−014	3.3637e−001
		RPSO	5.5280e−001	7.3453e−001	1.4655e−014	2.1586e−001

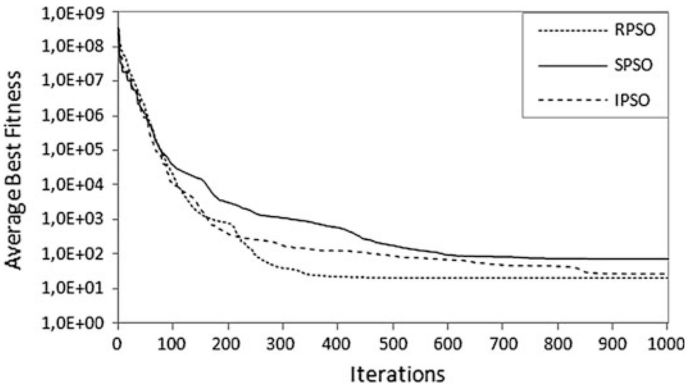


Fig. 1 The average best fitness for Rosenbrock 30 and the population of 40 particles

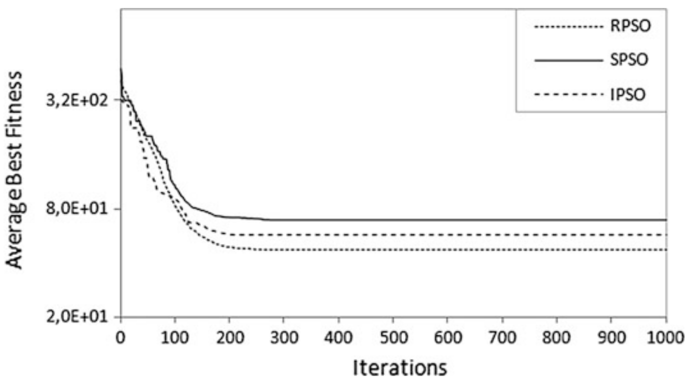


Fig. 2 The average best fitness for Rastrigin 30 and the population of 40 particles

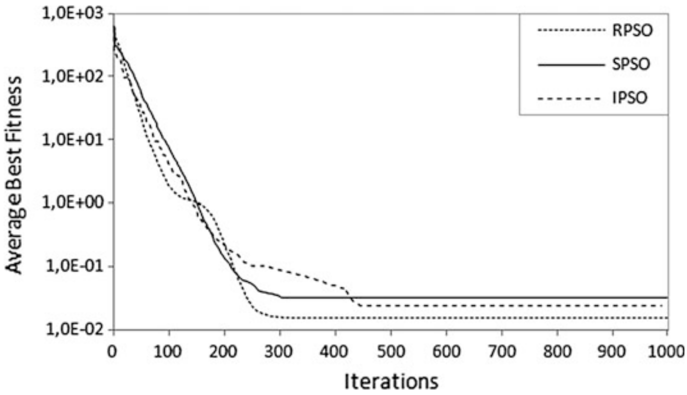


Fig. 3 The average best fitness for Griewank 30 and the population of 40 particles

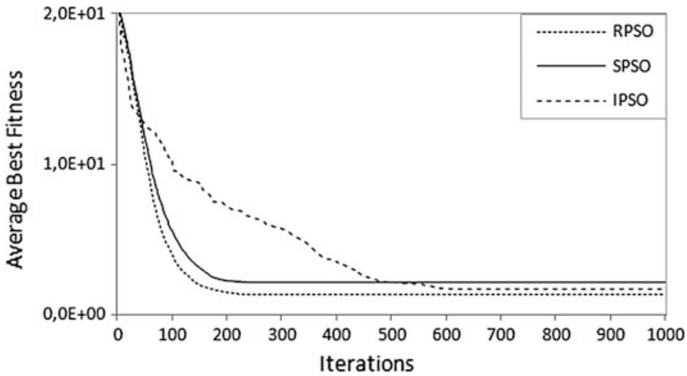


Fig. 4 The average best fitness for Ackley 30 and the population of 40 particles

5 Conclusions

In this paper, an improved particle swarm optimization algorithm called RPSO with a repair procedure has been proposed. The aim of the repair procedure was to determine new, better velocities for some particles when their current velocities are inefficient. New velocities are the functions of previous and current velocities. The new algorithm was tested with a set of benchmark functions, and the results were compared with those obtained through the standard PSO and IPSO. Experimental results have shown that the new algorithm is faster and more effective over the standard PSO and IPSO for all considered functions. It was also noted that convergence speed of proposed algorithm is considerably higher than that of SPSO.

The algorithm can be applied for solving high dimensional optimization problems. In case of small swarms, the algorithm can premature converge to one of the points of the search space.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
2. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
3. Dolatshahi-Zand, A., Khalili-Damghani, K.: Design of SCADA water resource management control center by a bi-objective redundancy allocation problem and particle swarm optimization. Reliab. Eng. Syst. Saf. **133**, 11–21 (2015)
4. Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Joyeux, P.: Particle swarm optimization for solving engineering problems: a new constraint-handling mechanism. Eng. Appl. Artif. Intell. **26**, 1263–1273 (2013)
5. Yildiz, A.R., Solanki, K.N.: Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach. Int. J. Adv. Manuf. Technol. **59**, 367–376 (2012)

6. Guedria, N.B.: Improved accelerated PSO algorithm for mechanical engineering optimization problems. Appl. Soft Comput. **40**, 455–467 (2016)
7. Yadav, R.D.S., Gupta, H.P.: Optimization studies of fuel loading pattern for a typical pressurized water reactor (PWR) using particle swarm method. Ann. Nucl. Energy **38**, 2086–2095 (2011)
8. Hajforoosh, S., Masoum, M.A.S., Islam, S.M.: Real-time charging coordination of plug-in electric vehicles based on hybrid fuzzy discrete particle swarm optimization. Electr. Power Syst. Res. **128**, 19–29 (2015)
9. Eberhart, R.C., Shi, Y.: Evolving artificial neural networks. In: Proceedings of the International Conference Neural Networks and Brain, Beijing, P.R.China, pp. 5–13 (1998)
10. Zheng, Y., Ma, L., Zhang, L., Qian, J.: Empirical study of particle swarm optimizer with an increasing inertia weight. In: Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 221–226 (2003)
11. Han, Y., Tang, J., Kaku, I., Mu, L.: Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight. Comput. Math Appl. **57**, 1748–1755 (2009)
12. Yang, X., Yuan, J., Yuan, J., Mao, H.: A modified particle swarm optimizer with dynamic adaptation. Appl. Math. Comput. **189**, 1205–1213 (2007)
13. Dong, Y., Tang, J., Xu, B., Wang, D.: An application of swarm optimization to nonlinear programming. Comput. Math Appl. **49**, 1655–1668 (2005)
14. Borowska, B.: PAPSO algorithm for optimization of the coil arrangement. Przeglad Elektrotechniczny (Electr Rev) **89**, 272–274 (2013)
15. Clerc, M., Kennedy, J.: The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**, 58–73 (2002)
16. Jiang, Y., Hu, T., Huang, C., Wu, X.: An improved particle swarm optimization algorithm. Appl. Math. Comput. **193**, 231–239 (2007)
17. Robinson, J., Sinton, S., Rahmat-Samii, Y.: Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In: Antennas and Propagation Society International Symposium, vol. 1, pp. 314–317 (2002)
18. Shi, X., Lu, Y., Zhou, C., Lee, H., Lin, W., Liang, Y.: Hybrid evolutionary algorithms based on PSO and GA. In: Proceedings of IEEE Congress on Evolutionary Computation 2003, Canbella, Australia, pp. 2393–2399 (2003)
19. Shi, X.H., Liang, Y.C., Lee, H.P., Lu, C., Wang, L.M.: An improved GA and novel PSO-GA-based hybrid algorithm. Inf. Process. Lett. **93**, 255–261 (2005)
20. Wang, L., Li, L., Liu, L.: An effective hybrid PSOSA strategy for optimization and its application to parameter estimation. Appl. Math. Comput. **179**, 135–146 (2006)
21. Wang, X.H., Li, J.J.: Hybrid particle swarm optimization with simulated annealing. In: Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, pp. 2402–2405 (2004)
22. Shi, Y., Eberhart, R.C.: Fuzzy adaptive particle swarm optimization. In: Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 101–106 (2001)
23. Tian, D., Li, N.: Fuzzy particle swarm optimization algorithm. In: International Joint Conference on Artificial Intelligence, pp. 263–267 (2009)
24. Liu, H., Abraham, A.: Fuzzy adaptive turbulent particle swarm optimization. In: The Fifth International Conference on Hybrid Intelligent Systems, Brazil, pp. 1–6 (2005)
25. Shi, Y.H., Eberhart, R.C.: Experimental study of particle swarm optimization. In: The Fourth World Multiconference on Systemics, Cybemetics and Informatics, USA, pp. 104–110 (2000)
26. Zahiri, S.H., Seyedin, S.A.: Swarm intelligence based classifiers. J. Franklin Inst. **344**, 362–376 (2007)
27. Trelea, I.C.: The particle swarm optimization algorithm convergence analysis and parameter selection. Inf. Process. Lett. **85**, 317–325 (2003)
28. Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Inf. Sci. **176**, 937–971 (2006)

Fuzzy Evaluation Method for Project Profitability

Rekik Ali, Gabsi Mounir, Valentina Emilia Balas and Masmoudi Nissen

Abstract The problem of the project management is performed with the optimization task under uncertainty and subject to real-world constraints. We use the probability theory and insufficiently proved methods, due to unavailable data indeed we need different methods for a best way to evaluate uncertainty. One of these approaches is based on the application of the fuzzy sets theory. Since its inception in 1965, the theory of fuzzy sets has advanced in a variety of ways and in many disciplines. Applications of this theory can be found, for example, in artificial intelligence, computer science, medicine, control engineering, decision theory, expert systems, logic, management science, operations research, pattern recognition, and robotics. This paper proposes a fuzzy decision making approach for project selection problem under uncertainty. An evaluation is provided as an illustration of the proposed approach. In the conclusion, we show how this method can help decision makers in the selection of appropriate project based on their profitability.

Keywords Fuzzy logic · Project management · Project selection · Uncertainty

---

R. Ali (✉) · M. Nissen
Department of Informations Technology, Higher Institue of Technological Studies,
Road Mahdia Km 2.5, BP 88 A, 3099, El Bustan Sfax, Tunisia
e-mail: alirekik1@yahoo.com

M. Nissen
e-mail: nissen.masmoudi@gmail.com

G. Mounir
Department of Informtions Technology, Higher Institute of Technological
Studies of Nabeul, AV: Campus Universitaire Mrezgua, 8000 Nabeul, Tunisia
e-mail: mounirgabsi@yahoo.fr

V.E. Balas
Department of Automation and Applied Informatics, Aurel Vlaicu University
of Arad, B-dul Revolutiei 77, 310130 Arad, Romania
e-mail: balas@drbalas.ro

© Springer International Publishing AG 2017

N. Shakhovska (ed.), *Advances in Intelligent Systems and Computing*,
Advances in Intelligent Systems and Computing 512,
DOI 10.1007/978-3-319-45991-2_2

1 Introduction

Project selection under uncertainty has become an important research topic in project management [1, 6]. In this context, Markowitz [10] based on the variance of project returns as a risk measure for the optimal project choice, introduced the so-called mean-variance model. Companies should be well advised to use the different management project concepts. Today, many organizations are faced with the problem of the project selection and the resources allocation in order to create an optimal decision during the project selection. Among the various models of project selection, we can mention those based on multicriteria decision support system, nonlinear, stochastic [18], linear, dynamic [12], fuzzy programming [14], and fuzzy decision trees [13].

Existing scientific and methodological approaches have the following disadvantages [15]:

- Absence of generic risk assessment model that is invariant to the input parameters;
- The results of mathematical modeling of the risk assessment require clearer graphical interpretation.

The known models that aid in determining the degrees of risk are based on the evaluation of a single parameter (criterion), which leads to the impossibility of comparing the relative risk estimations for two or more parameters simultaneously.

The fuzzy sets theory is used to handle uncertain information in multiple systems, such as planning support systems and the decision support in the project selection management systems. This theory offers an alternative framework for dealing with uncertainty of the selection project parameters. Approximations of these parameters can be estimated by experts based on their skills [5].

Buckley was one of the first authors who used the fuzzy sets in finance [2]. He used them to represent the fuzzy present value, the fuzzy future value, and the fuzzy internal rate of return.

Yu et al. have proposed a decision analysis tool based on several criteria for assessing credit risk from the theory of fuzzy sets [16]. Reveiz and Leon [11] have studied the operational risk in using the fuzzy inference system to take into account the complex interaction and the non-linearity of these elements. Moreover, Leon and Machado [7] have proposed an index established by using an inference system based on fuzzy logic and allowing to make a general assessment of the relative importance of a systematic financial institution.

The objective of this paper is to develop a fuzzy model in order to optimize the innovative project selection under risk. The fuzzy set theory is used with the aim to describe and reduce uncertainty in the information project [14].

Project selection problems have been discussed in a many management tasks such as R&D [8, 9], quality management and environmental management [4].

The objective of this paper is to develop a fuzzy model in order to optimize the project profitability under risk. The fuzzy set theory is used with the aim to describe and reduce uncertainty in the project selection.

The paper is organized as follow: Sect. 2 describes the basic concepts of the fuzzy sets, in this case we introduce the notion of membership functions, the different types of fuzzy numbers and the operations that we can apply on the fuzzy sets. Section 3 represents an application of fuzzy logic to solve a selection project problem by using the inference engine proposed by Mamdani. After that, we introduce the input and output parameters of the proposed approach as well as the membership functions for all model parameters, the simulating results obtained according the inference steps. We analyze the experimental results and discuss the parameters which have an impact on our approach in the Sect. 3.

2 The Basic Concepts of the Fuzzy Logic

The fundamental characteristic of a classical set is the abrupt boundary between two categories of elements: those that belong to the set and those that do not belong to it, since they belong to its complement. In this case, the membership relation is represented by a function which takes μ truth values in the pair $\{0, 1\}$ [17].

Hence, the membership function of a conventional set A is defined by:

$$\mu_A(x) = \begin{cases} 0, & if \quad x \notin A \\ 1, & if \quad x \in A \end{cases} \tag{1}$$

In contrast, a fuzzy set is any set which allows its elements to have different membership grades (membership function) in the interval [0,1]. For a classical set X, a fuzzy set is defined as follows:

$$A = \{(x,\ \mu(x)),\ x \in X\}. \tag{2}$$

The grade of the elements x in relation with the fundamental set X is defined by the membership function $\mu_A(x)$.

For each element having a value of 0 means that the member is not included in the given set, on the contrary if the value is 1 means full member included. Values in the range from 0 to 1 characterize the fuzzy members.

We suppose that, A and B are two fuzzy sets, then we define the membership function as follow:

$$\mu_{A \cup B}(x) = \max(\mu_A, \mu_B) \tag{3}$$

$$\mu_{A \cap B}(x) = \min(\mu_A, \mu_B) \tag{4}$$

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x). \tag{5}$$

when $X = R$ is a set of real numbers, we talk about fuzzy numbers. In the practical field, it is more convenient to work with fuzzy numbers of a special type: triangular and trapezoidal.

The trapezoidal membership function is given by the formula:

$$\mu_A(x) = \begin{cases} 0, & for \quad x < a_1 \quad or \quad x > a_4 \\ \frac{x-a_1}{a_2-a_1}, & for \quad a_1 \le x < a_2 \\ 1, & for \quad a_2 \le x \le a_3 \\ \frac{a_4-x}{a_4-a_3}, & for \quad a_3 < x \le a_4, \end{cases} \tag{6}$$

where $a_1 \le a_2 \le a_3 \le a_4$.

For trapezoidal membership functions, we use the notation: $A = (a_1, a_2, a_3, a_4)$. In the case where $a_2 = a_3$, we obtain a triangular membership function. Let us notice that for triangular membership functions, we use the notation: $A = (a_1, a_2, a_3)$ (Fig. 1).

Let us notice that fuzzy numbers can be added, subtracted, multiplied and divided, as well as ordinary numbers. Moreover, the operations on fuzzy numbers are determined by the following expansion principle.

Let $c = f(a, b)$ be an arbitrary numerical function. For example, concerning the addition operation, $f(a, b) = a + b$. Then, the value of $C = f(A, B)$ of this function with the fuzzy numbers A and B has a membership function which is calculated by the following formula:

$$\mu_C(x) = \sup \min(\mu_A(x), \mu_B(y)), \tag{7}$$

And their α—cuts are deduced according to the following formula:

$$C^\alpha = \{c = f(a, b) | a \in A^\alpha, b \in B^\alpha\}. \tag{8}$$

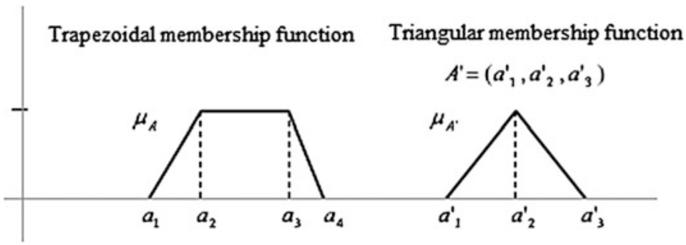


Fig. 1 Trapezoidal and triangular membership functions

3 Application of Fuzzy Logic to the Project Selection

Based on the inference engine proposed by Mamdani, our method represents the certainty degree about the coincidence of metadata elements and user's preferences. The typical structure of our method contains the following units: fuzzification, defuzzification, and an interface system (Fig. 2).

- *Fuzzification interface*: simplify modifies the inputs so that they can interpreted and compared to the rules on the rule base. The fuzzifier determines the degree to which they belong of each input values to each of the fuzzy sets based on the membership functions.
- *Rule base*: holds the knowledge, in the form of a set of rules, of how best to control the system.
- *An inference system*: Inference mechanism allows mapping given input to an output using fuzzy logic. It uses all pieces described in previous sections: membership functions, logical operations and rules. They vary in ways of determining outputs. Each rule is represented in the following form:

$$if\, X_1\, is\, A_1\, and \ldots and\, X_n\, is\, A_n\, then\, Y\, is\, B$$

  with X_i being input and Y output linguistic variables, and with A_i and B being linguistic labels with fuzzy sets associated defining their meaning.
- *A defuzzification interface*: is allowed to find one single crisp value that summarises the fuzzy set. There are several methods to solve this machanizm, and the centroid method is considered as one among them. The centroid method simply the weighted average of the output membership function.
  It can be determined by the following formula:

$$\bar{X}(centroid) = \frac{\int_b^a x\mu(x)dx}{\int_b^a \mu(x)dx}$$

  where [a, b] is the interval of the aggregated membership function.

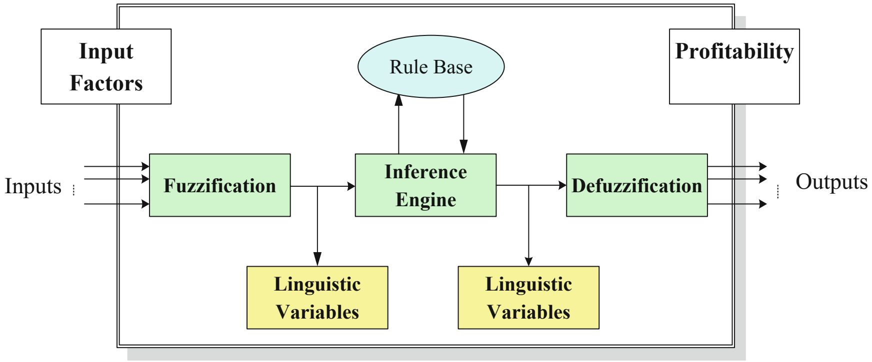


Fig. 2 A typical structure of a fuzzy inference system

3.1 The Proposed Approach

As an uncertainty assessment of the project selection it is advisable to take a parameter of profitability P_{Prof} [3]. The input parameters in this case are obtained by statistical analysis of the average value Q_{RR} (rate of return by introducing an innovative project), economic effects expected using more productive technologies T, for example, the performance degree in the existing equipments, which would produce a greater effect, and as output parameter we have the estimated value of the project's profitability P_{Prof}.

We present the membership functions of the triangular fuzzy numbers $Q_{RR} = [Q_{\min}, Q_0, Q_{\max}]$ and $T = [T_{\min}, T_0, T_{\max}]$ as follow:

$$\mu_Q(x) = \begin{cases} \frac{1}{Q_0 - Q_{\min}} x + \frac{Q_{\min}}{Q_{\min} - Q_0}\,, & Q_{\min} < x < Q_0; \\ \frac{1}{Q_0 - Q_{\max}} x + \frac{Q_{\max}}{Q_{\max} - Q_0}\,, & Q_0 < x < Q_{\max}; \\ 0\,, & (x < Q_{\min}) \vee (x > Q_{\max}). \end{cases} \tag{9}$$

$$\mu_T(x) = \begin{cases} \frac{1}{T_0 - T_{\min}} x + \frac{T_{\min}}{T_{\min} - T_0}\,, & T_{\min} < x < T_0; \\ \frac{1}{T_0 - T_{\max}} x + \frac{T_{\max}}{T_{\max} - T_0}\,, & T_0 < x < T_{\max}; \\ 0\,, & (x < T_{\min}) \vee (x > T_{\max}). \end{cases} \tag{10}$$

The first input indicate the rate of return by introducing an innovative project (Q), his universe of discourse be [0–100]. The second indicate the economic effects expected using more productive technologies (T), its universe of discourse be [0–100]. Both two fuzzy numbers are expressed by a set of terms {low, medium, high}. As result, the output variable characterizes the project profitability (P).

Graphically, the membership functions for each input variable are shown in Figs. 3, 4 and output variable in Fig. 5.

Before the creation of the rule databases, the number of linguistic terms can be changed.

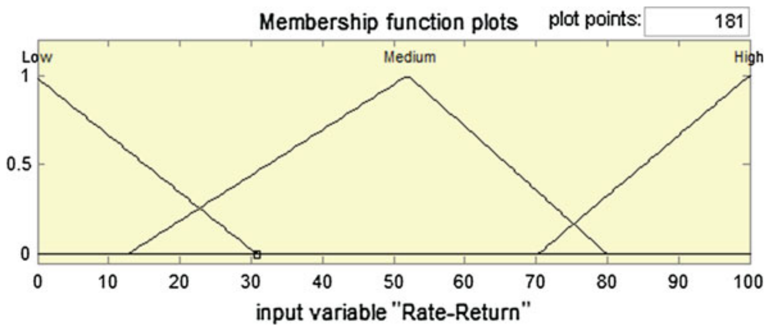


Fig. 3 Membership functions for the rate of return by introducing of an innovative project

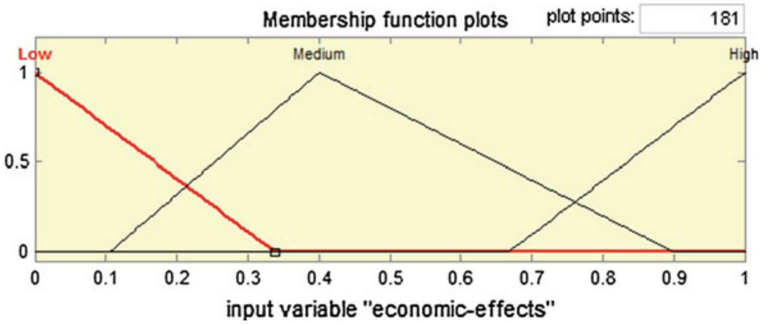


Fig. 4 Membership functions for the economic effects expected using more productive technologies

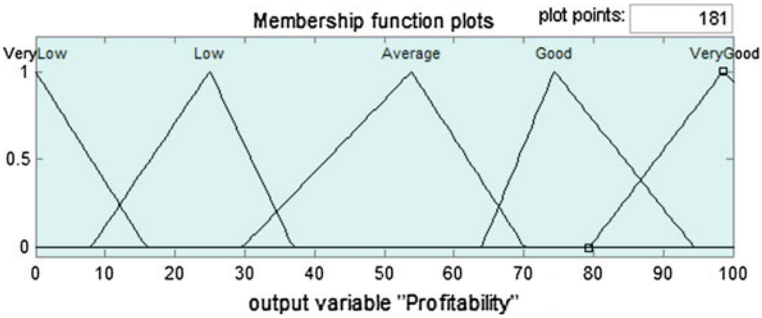


Fig. 5 Fuzzy membership functions for the project profitability

To simulate consolidated factors, the expert must establish a fuzzy knowledge base of Mamdani type. Antecedents may be joined by OR; AND operators (Fig. 6).

In our method we consider the classical engine developed by Mamdani based on the minimum t-norm as conjunctive and implication operators, the defuzzification method is the centroid. The inference engine taking into account the membership functions obtained according the inference steps (Fig. 7).

1. If (Rate-Return is Low) and (economic-effects is Low) then (output1 is VeryLow) (1)
2. If (Rate-Return is Low) and (economic-effects is Medium) then (output1 is VeryLow) (1)
3. If (Rate-Return is Low) and (economic-effects is High) then (output1 is Low) (1)
4. If (Rate-Return is Medium) and (economic-effects is Low) then (output1 is Low) (1)
5. If (Rate-Return is Medium) and (economic-effects is Medium) then (output1 is Average) (1)
6. If (Rate-Return is Medium) and (economic-effects is High) then (output1 is Average) (1)
7. If (Rate-Return is High) and (economic-effects is Low) then (output1 is Average) (1)
8. If (Rate-Return is High) and (economic-effects is Medium) then (output1 is Good) (1)
9. If (Rate-Return is High) and (economic-effects is High) then (output1 is VeryGood) (1)

Fig. 6 Examples of rules

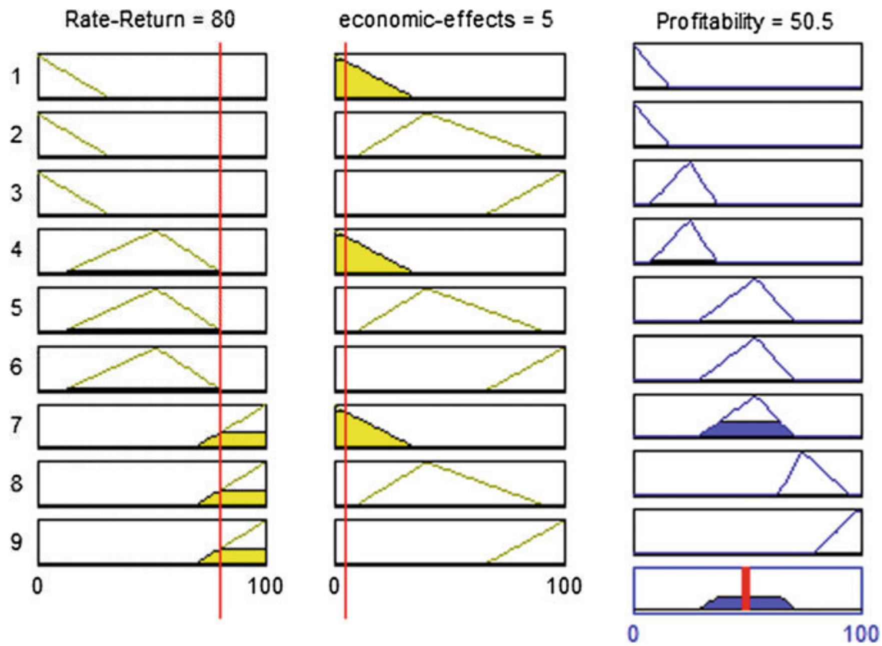
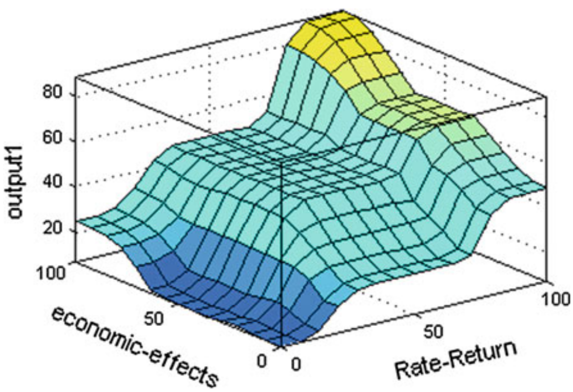


Fig. 7 Operation mode the inference system

Fig. 8 Characteristic surface of the system



In Fig. 8 there is shown a characteristic surface for the rules database of the fuzzy system that characterize the project profitability.

Many tools are used to develop applications based on the fuzzy logic principle. We can mention the MATLAB software package, which is considered the most famous. Also there is a FUZZY-TECH software not yet become frequently used as MATLAB [7].