CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Bachelor Thesis

Jiří Pytela

Implementation and Testing of Social Optimization Algorithms

Implementace a testování algoritmů sociální optimalizace

Department of Cybernetics

Thesis supervisor: **Ing. Martin Macaš**

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Jiří Pytela
Study programme:	Cybernetics a Robotics
Specialisation:	Robotics
Title of Bachelor Project:	Implementation and Testing of Social Optimization Algorithms

Guidelines:

1. In JAVA environment, implement an application for running and testing of selected social approaches to optimization.
2. A comprehensive documentation must be part of the implementation.
3. Include binary particle swarm optimization and simple genetic algorithm into the application.
4. Perform testing and comparison on at least five testing functions with binary inputs.

Bibliography/Sources:

[1] Wu, X., Zhao, M., & Qu, Y. (2010). Particle Swarm Optimization Programming. 2010 International Conference on Computational Aspects of Social Networks CASoN (pp. 397-400). IEEE.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5600263
[2] Macas, M.; Lhotska, L.: Optimizers derived from human opinion formation. Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on, Issue Date: 19-21 Oct. 2011 pp.359-364.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6089618&isnumber=6089255

Bachelor Project Supervisor: Ing. Martin Macaš

Valid until: the end of the winter semester of academic year 2012/2013

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 9, 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Jiří P y t e l a
Studijní program:	Kybernetika a robotika (bakalářský)
Obor:	Robotika
Název tématu:	Implementace a testování algoritmů sociální optimalizace

Pokyny pro vypracování:

1. V prostředí JAVA naimplementujte aplikaci pro spouštění a testování vybraných sociálních přístupů k optimalizaci. Soustřeďte se na budoucí rozšiřitelnost vaší aplikace.
2. Součástí aplikace musí být podrobná a přehledná dokumentace.
3. Do aplikace naimplementujte také binární particle swarm optimalizaci a případně jednoduchý genetický algoritmus.
4. Proveďte testování a porovnání na minimálně pěti testovacích funkcích s binárními vstupy.

Seznam odborné literatury:

[1] Wu, X., Zhao, M., & Qu, Y. (2010). Particle Swarm Optimization Programming. 2010 International Conference on Computational Aspects of Social Networks CASoN (pp. 397-400). IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5600263
[2] Macas, M.; Lhotska, L.: Optimizers derived from human opinion formation. Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on, Issue Date: 19-21 Oct. 2011 pp.359-364. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6089618&isnumber=6089255

Vedoucí bakalářské práce: Ing. Martin Macaš

Platnost zadání: do konce zimního semestru 2012/2013

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S. 4

[illegible]

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2012

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Dne: 25. 5. 2012

Podpis: Jiří Pytela

Acknowledgements

First of all I would like to thank my supervisor Ing. Martin Macaš, who offered me to work on this interesting subject and patiently helped me with any problems and questions that arose during the work on the thesis. I much appreciate all his help with LaTeX syntax.

I also wanted to express my gratitude to my parents for all their encouragement and support in my studies and special thanks to my father for his help with English language in this thesis.

Abstrakt

Optimalizace je problém, který je třeba řešit v mnoha odvětvích lidské činnosti. Tato práce se zabývá implementací nové optimalizační metody, založené na teorii sociálního vlivu, do Java programu SITO_tester. Program SITO_tester je Java projekt pro NetBeans IDE. Funkčnost vytvořeného programu byla ověřena úvodními experimenty a výsledky sociální optimalizace byly porovnány s binární optimalizací hejnem částic. Pro porovnání bylo použito pět testovacích funkcí: Onemax, ECC, Bipolar-6, PPeaks a Binary. V závěru jsou nastíněny možnosti dalšího rozvoje programu

Klíčová slova: sociální optimalizace, teorie sociálního vlivu, implementace SITO, SITO_tester, optimalizace hejnem částic

Abstract

Optimization tasks can be found in many disciplines. This thesis is about an implementation of a novel optimization method, Social Impact Theory based Optimization (SITO). The method has been implemented within a Java program called SITO_tester, the program is in form of a Java project for NetBeans IDE. Functionality of the SITO_tester was proved in the preliminary tests. The performance of the SITO algorithm was compared to the peformance of Binary Particle Swarm Optimization algorithm (BPSO). Five testing functions were used for the preliminary experiments: Onemax, ECC, Bipolar-6, PPeaks and Binary. Some possibilities for further development of the program were suggested in the conclusions.

Keywords: Social Impact Theory based Optimization, SITO implementation, SITO_tester, Binary Particle Swarm Optimization

Contents

1 Introduction 1
1.1 Goals 1
1.2 Motivation 2
1.3 State Of the Art 2
1.3.1 Implementations of SITO 2
1.3.2 Nature Inspired Algorithms 3
1.4 Thesis Organization 6

2 Opinion Formation based Optimization 7
2.1 Opinion Formation Model 7
2.2 Social Impact Theory based Optimization 8
2.3 Binary Particle Swarm Optimization 10

3 Java Implementation 11
3.1 Program Structure 11
3.2 Description of the Classes 11

4 Preliminary Experiments 14
4.1 Fitness Functions 14
4.2 Parameter Settings 15
4.3 Experiments Results 15
4.4 Population Size 18

5 Conclusions 20
5.1 Future Work 21

List of Figures

1 Performance of SITO and BPSO algorithms on ONEMAX fitness function 16
2 Performance of SITO and BPSO algorithms on ECC fitness function . . . 16
3 Performance of SITO and BPSO algorithms on PPEAKS fitness function . 17
4 Performance of SITO and BPSO algorithms on BIPOLAR-6 fitness function 17
5 Performance of SITO and BPSO algorithms on BINARY fitness function . 18

List of Tables

1 BPSO parameter settings 15
2 SITO parameter settings 15
3 Number of iterations required to find optimum with different population sizes 18
4 Contents of the CD 24

List of algorithms

1 Pseudocode for SITO algorithm 9

1 Introduction

1.1 Goals

The purpose of this thesis is to develop a program simplifying the work with Social Impact Theory based Optimization algorithm (SITO) (3.2). The program is named SITO_tester and the exact goals of the thesis are:

Implement a java application for running and testing of selected social approaches to optimization. Comprehensive documentation must be part of the application

The main goal of this thesis is developing a tool for testing SITO on various testing functions. It is essential that all the parameters defining SITO algorithm, the parameters of the testing functions and the testing functions themselves should be easily changeable. If the SITO_tester program is to be useful, it needs to allow future users to add and modify the testing functions and the optimization algorithms.

There are many approaches to binary optimization, some of them are described in section 1.3, and it is not possible, and neither is it the goal of this thesis, to implement all of them. Instead the program SITO_tester implements only modules for SITO algorithm. However different optimization algorithms can be implemented, as is shown with Binary Particle Swarm Optimization algorithm (BPSO)[1], which was used for comparison with SITO performance. As long as the newly implemented algorithm has some required basic structure its results can be plotted in graphs and it is also possible to compare its performance with other optimization algorithms inside the program. Or, if preferred, its results are saved to text file and can be later used in other programs.

It is clear that modularity and openness to future extensions is one of the most important features of SITO_tester. To achieve it, the program is not a standalone application, instead it is a java project for NetBeans IDE.

Include binary particle swarm optimization

In order to evaluate the performance of SITO algorithms there needs to be other similar algorithm working on the same problem. In this thesis we compare SITO with BPSO algorithm, mainly because BPSO is easy to implement and reliable in finding the global optimum in wide range of optimization problems.

Perform testing and comparison on at least five testing functions with binary inputs

To evaluate an optimization algorithm, it is necessary to have a comparison with other algorithms and measure the quality of solution for each algorithm as well as the time needed to found the solution. Often, an algorithm is outstanding on one kind of problems and performs poorly on others. In this thesis it is tested whether SITO can be successfully applied to different optimization problems by comparing its performance on several binary optimization benchmark functions to BPSO.

1.2 Motivation

The main goal is to implement the SITO algorithm and provide a tool for parameter testing and modifications. SITO algorithm is relatively new method, first published in 2007, created as an alternative for popular Binary Particle Swarm Optimization method. While BPSO is inspired by social interactions of animals living in a group, SITO takes inspiration in social behavior of humans. So far, SITO method has been successfully applied to the problem of feature selection ([2][3]), which is an important part of design of classifiers or other machine learning systems. The feature selection reduces computation time and also improves the efficiency of the classifier. In 2011 research team at Central Scientific Instruments Organisation, Chandigarh in India applied SITO method to optimization of an impedance-Tongue [4] for classifying samples of black tea. In this case, SITO approach outperformed Genetic Algorithm and Binary Particle Swarm Optimization. The same team used SITO in 2012 to create methods for performance enhancement of an electronic nose [5], the new methods proved more efficient and could lead to the development of new control systems for tea production.

As noted in Section 1.3.1 - Implementations of SITO, there are currently two programs for testing and modifying SITO and both are implemented in Matlab. Some people prefer Java over Matlab, but so far there has been no program for working with SITO in Java. The purpose of SITO_tester is to allow more people to explore and to use this new optimization method.

The application is open to modifications, extensions and many new interesting and more advanced functions can be added into the program. Because of the vast space for further development, the program can be useful as a base for individual projects or theses for students interested in SITO algorithm or its applications.

1.3 State Of the Art

1.3.1 Implementations of SITO

SITO algorithm is a new method and up to now, there have been only two programs enabling work with this algorithm. SIFS Tool [6] was developed at Czech Technical University in Prague. During my work on this thesis, Bhondekar et al. published SITO Library [7]. It was created after a successful application of SITO approach in their research. Both SIFS Tool and SITO Library are implemented as Matlab toolboxes.

1.3.2 Nature Inspired Algorithms

Optimization as defined by Encyclopædia Britannica is collection of mathematical principles and methods used for solving quantitative problems in many disciplines, including physics, biology, engineering, economics, and business. In the last century many new difficult optimization problems emerged in those and other fields and with them also various techniques to solve them. In general, the techniques, or algorithms, can be exact or stochastic.

Exact algorithms, for example A* search, are effective and guarantee finding optimal solution of the problem. However, when facing too complex problems the use of exact algorithms is not possible and instead it is necessary to use stochastic algorithms. Stochastic algorithms do not guarantee that the solution they find is the optimum but it is an acceptable solution that takes acceptable amount of time to get. Many algorithms that efficiently explore a search space have been found by observation of processes in nature, for example social structures, or behaviour patterns of animal and insect species, and so they are called Nature Inspired Algorithms. They are subject of ongoing research and often the algorithms are much improved since they were introduced for the first time. Below is a list of some of the most used Nature Inspired Algorithms for optimization.

Particle Swarm Optimization

Particle Swarm Optimization algorithm (PSO) [8], first introduced in 1995, is population based method inspired by flocks of birds and fish schools. The swarm is made of randomly initiated candidate solutions, or particles, that move in multidimensional search space. Each particle has a velocity and information about the best position particles in the neighbourhood found so far, also called gbest, and its own best position, called lbest. In time, particles change their velocities towards gbest and lbest positions assuming that they will reach even better solutions. The particles move in a swarm-like group and keep diversity in their population even after they found the best possible solution.

Ant Colony Optimization

Ant Colony Optimization algorithm (ACO) [9], introduced in 1996, is based on behaviour of ants in their search for food. In nature, ants move at first randomly searching for food. When an ant finds food, he returns to the colony and lays down pheromone trail. If other ants find this path they follow it until they also find the food, on their return to the colony they enforce the path with their own pheromones. Because the pheromones evaporate, the more time it takes to follow a path to a source of food and back to the colony, the less attractive is the path. This allows further exploration even after a local optimum has been found. ACO algorithm uses a set of agents, called artificial ants, to find best paths on a graph. Pheromone distribution model is part of the graph and it is modified by the artificial ants moving through the graph, much like real ants do in real world. One of advantages of the ant colony is that even dynamically changing graphs can be explored. ACO has been used to find near-optimal solutions to the traveling salesman problem and it can be useful for network routing or transportation systems.

Simulated Annealing

Simulated annealing [10] was first described in 1983, the method is analogical to annealing in metallurgy. When material is heated its atoms are freed from the positions they were locked in are able to randomly change their positions in search for configuration with lower internal energy than that of the initial state. The ability of atoms to change position decreases with the temperature until all the atoms stay in their fixed positions. The algorithm works the same. Every iteration new position is found by changing the current position, if the new one is better, it is accepted. When new position is worse than the current one, it is still accepted with probability depending on the system's temperature. The temperature decreases over time and so poorer positions are accepted with high probability at the beginning of the search, allowing exploration of large part of the search space. Later the search is focused on small part of the search space and only changes that improve the position are accepted. Simulated annealing can be useful for example for structural optimization or optimization of water distribution or transportation systems.

Genetic Algorithm

The first Genetic Algorithm (GA) [11] was used in 1975 by J. H. Holland. It is based on idea of evolution through random mutation and natural selection. The research in this field started with effort to better understand and model processes of natural evolution but soon it became clear that the method could be applied to solve artificial problems. There are many variations and modifications of the basic GA but in general it uses population of individuals, usually character strings of fixed length are used to represent their genetic information. Over iterations the population changes by two main methods of change – mutation and crossover. Mutation is random and small modification of an individual's genetic information. Crossover is combination of two individuals in order to produce their offspring. The child's genetic information is some combination of its parent's genetic information. Original population size is maintained by removing poor individuals from the population. Different GAs have been successfully used for machine learning, circuit design, neural network design, robots control and wide range of other optimization problems.

Differential Evolution

Differential Evolution (DE) [12], introduced in 1997, is population based Evolutionary algorithm used for optimization of real functions or real parameters. The population of candidate solutions is used for the optimization, new candidate solutions are created by combination of other candidate solutions from the population according to simple formula and the best solutions are kept in the population while others are removed. In tests DE proved to perform better than genetic algorithms or simulated annealing. Other advantages of DE are easy implementation and minimal parameter tuning requirements.

Artificial Immune Systems

Artificial Immune Systems (AIS) [13] were originally models of the immune system, helping to better understand its mechanisms. Since then, research has focused also on solving computation problems by imitating the biological processes than simulating the immune system. Although, even today, AIS are closely connected with immunology, because the better the understanding of the biological immune systems the more possibility to apply the knowledge into creating more powerful and more efficient artificial systems.

Decentralized Cellular Evolutionary Algorithm

Decentralized Cellular Evolutionary Algorithms [14] are very similar to Evolutionary or Genetic algorithms. In GAs every individual in the population can interact with any other while in Decentralized Cellular Evolutionary Algorithms there is always defined structure in the population. Individuals form groups, or cells, and only individuals in the same cell can interact. This structure helps maintain diversity .

Brain Storm Optimization

Brain Storm Optimization algorithm (BSO) [15] is a new method, introduced in 2011, and it is similar to SITO algorithm in the sense of taking inspiration from human behavior, in the case of BSO the human brainstorming process. It is suggested that since humans are more intelligent than animals or insects, an optimization algorithm inspired by humans should be superior to those inspired by other species. The effectiveness of the algorithm was proved in the preliminary experiments, detailed analysis and experimental tests are now subjects of further research.

1.4 Thesis Organization

The thesis is organised into five sections. Section 1 - Introduction - contains details about goals and motivation of this thesis as well as current state of research in the field of Nature Inspired Algorithms. This work is about implementing algorithm for Social Impact Theory based Optimization, so in section 2 - Opinion Formation based Optimization - you can find information about the algorithm and its background and section 3 - Java Implementation - contains details about the implementation of the algorithm in Java and organization of the SITO_tester program. Section 4 - Preliminary Experiments - describes experiments that compare SITO and BPSO using SITO_tester, details of the testing functions are included. Section 5 - Conclusion - summarizes the work that has been done and suggests possibilities for future development.

2 Opinion Formation based Optimization

SITO algorithm is population based optimization method and takes inspiration in social psychology. The main part of the algorithm is simulated society. Fitness function is used to evaluate each individual, individuals with higher fitness value are preferred.

2.1 Opinion Formation Model

The model of Opinion Formation in society is based on Bibb Latané's Dynamic Theory of Social Impact [16]. This theory suggests that behaviour and opinions of single individual depend on society to which he belongs and if the same individual is introduced into another social environment he will adapt to the new society and in turn affect its other members. The Dynamic Theory of Social Impact successfully explained some of the phenomena studied in social psychology.

Any individual changes his opinions accordingly with social impact of his group. The opinion, or attitude, of the individual can take one of two possible values. The values may be interpreted as people for or against some idea, people belonging or not belonging to given group etc. but the interpretation is not important for the model itself. Social impact is any influence on the individual in his social environment, its magnitude depends on three variables:

Strength is an attribute of each individual and it determines how much can he affect others. It corresponds to credibility or attractiveness of the individual. In real world people with higher social status, experience or better public skills are better and more efficient in communicating their opinion to others. The model distinguishes two different forms of strength, one for individuals who share the same opinion and one for those who oppose each other. *Persuasiveness* is the ability to convince someone with opposing opinion to change it and *supportiveness* is the ability to help those sharing the same opinion to resist influence from others. Persuasiveness and supportiveness are not necessarily correlated.

Immediacy is a term the social impact theory uses for group structure. Immediacy is obviously not attribute of an individual but of a pair of individuals. It can be viewed as relations in the society or physical distance of the individuals. Generally it expresses the possibility of communication between the two individuals.

Number of other individuals simply expresses the number of individuals that contribute to the social impact on given individual. Usually not all individuals in the population are considered, but only those whose immediacy to the given individual is above some threshold value.

2.2 Social Impact Theory based Optimization

Social Impact Theory based Optimization (SITO)[17] takes advantage of the opinion formation model described in previous section. With three modifications, it introduces optimization ability into the simulation.

- **Multiple attitudes:** To make more-dimensional optimization problems possible each individual is represented by a binary vector instead of a binary number.
- **Replacement of persuasiveness and supportiveness by a single parameter:** As described in previous section 2.1, in Latané's Dynamic Theory of Social Impact persuasiveness and supportiveness are different but not independent. To simplify the computation process, SITO algorithm uses only one parameter, strength. Persuasiveness and supportiveness are both equal to strength.
- **Introduction of a fitness function:** The strength factors (persuasiveness and supportiveness) in the opinion formation simulation were random numbers assigned to each individual on initialization and changed only with change of attitude. This randomness modelled some unknown phenomena in the simulation, which was the point of the simulation, but in SITO algorithm the point is to implement optimization ability into the simulation. The behaviour is changed by adding one parameter to each individual, the value of fitness function, which is used to compute the strength factor of the individual. Fitness function evaluates each individual in terms of a binary optimization problem, the best individual in the population is the one with highest fitness value. Strength factor is proportional to fitness value of given individual, so more successful individuals have more influence on the rest of the population.

The mechanisms of SITO algorithm are explained for its particular instance, Simplified SITO (sSITO) [3], which uses different equations for strength and impact functions and different population topology than Original SITO [17] in order to simplify the computational process. The algorithm's main part is population of individuals. Each individual is candidate solution represented by binary vector $\mathbf{s_i} = (s_i^1, ..., s_i^d)$, where $s_i^k \in \{1, 0\}$. The binary vector is evaluated by the fitness function $f_i = f(s_i^1, ..., s_i^d)$ in terms of maximizing f.

The topology of the population is important, because contribution of individual i to the impact on individual j depends on his position in relation to j. All individuals who contribute to impact on individual j are called neighbors of j. sSITO implements random topology, meaning that neighbors of individual i are each iteration chosen randomly from all individuals in the population. Social strength q_{ij} is strength by which individual i affects j. There can be many ways to compute the social strength [3], for example:

$$q_{ij} = \max(f_i - f_j, 0) \tag{1}$$

f_i and f_j are fitness values of individuals i and j. This equation means that each individual is affected only by those who are better than him. Social impact $I_j = (I_j^1, ..., I_j^d)$ is vector of impact values on each bit of candidate solution j and it is computed from social strengths of j's neighbors toward him. To compute overall impact on bit m of individual j we can use following equation:

$$I_j^m = \sum_{i \in \mathbb{P}_i^m} q_{ij} - \sum_{i \in \mathbb{S}_i^m} q_{ij} \tag{2}$$

Neighbors of j are on each dimension divided into two subgroups, persuaders $\mathbb{P}_j^m$ and supporters $\mathbb{S}_j^m$. The equation expresses overall social impact on individual i as sum of overall persuasive minus overall supportive impact. If any of the two subgroups is empty, its overall impact is defined as zero. Simple update rule generates new state of s_j^m:

$$s_j^m(t+1) = \begin{cases} 1 - s_j^m(t) & \text{if} I_j^m > 0 \\ s_j^m(t) & \text{otherwise} \end{cases} \tag{3}$$

If impact function has positive value the individual changes his opinion. Random element is added to the update rule in form of mutation. After new state was generated, every bit of jth candidate solution changes its value with probability $\kappa << 1$, κ is the mutation rate.

The pseudocode of sSITO is shown in algorithm 1. First, all individuals in the population are initialized as random candidate solutions with uniform distribution of values. Candidate solutions are evaluated and all strength values are computed. Value of the impact function is computed for each dimension of each candidate solution and the candidate solutions are updated accordingly, with the possibility of random mutation. When all candidate solutions have been updated, the process is repeated until a stopping condition is met.

```
initialize all $\mathbf{s_i(0)}$
while stop condition not met do
  for all $i$ do
    evaluate $f_i(t) = f(\mathbf{s_i(t)})$
  end for
  for all $i$, $j$ do
    compute strength values $q_{ij}$ using equation 1
  end for
  for all $i$, $d$ do
    compute $I_i^d$(t) using equation 2
    compute $\tilde{s}_i^d$(t+1) using equation 3
  end for
  for all $i$, $d$ do
    compute $s_i^d$(t+1) by random mutation of $\tilde{s}_i^d$(t+1)
  end for
  $t \leftarrow t + 1$
end while
```

Algorithm 1: Pseudocode for SITO algorithm

2.3 Binary Particle Swarm Optimization

Binary Particle Swarm Optimization (BPSO) ([18][1]) is similar to Particle Swarm Optimization, but its particles move only in binary space of attitudes instead of continuous state space. Each particle is represented by binary vector of position $\mathbf{s}_i(t)$ and velocity vector of real values $\mathbf{v}_i(t)$ and has information about the best position the particle achieved so far, we can call it experience, and information about the best position of the best particle in its neighborhood, a kind of social knowledge. Following equations describe rules for velocity update and position update.

$$\mathbf{v}_i(t+1) = \omega\mathbf{v}_i(t) + \varphi_1 R_1(\mathbf{p}_i - \mathbf{s}_i(t)) + \varphi_2 R_2(\mathbf{p}_l - \mathbf{s}_i(t)) \tag{4}$$

$$s_{i,j}(t) = \begin{cases} 1 & \text{if} R_3 < \frac{1}{1+e^{-v_{i,j}(t)}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$\mathbf{p}_i$ is the best position of particle i, the part of the formula represents experience of the particle, φ_1 is constant of experience weight. $\mathbf{p}_l$ is the best position of any particle in the neighborhood, it is the social knowledge of the particle, φ_2 weight of the social knowledge. The parameter ω is called inertia weight, modifying it affects the exploration behavior of particles. R_1 and R_2 are diagonal matrices of random values from a uniform distribution between 0 and 1, R_3 is only one number generated the same way.

New values for every component $\mathbf{v}_i$ are assigned, values higher than V_{max} are replaced by V_{max}, values lower than $-V_{max}$ are replaced by $-V_{max}$. The position of each particle is updated according to its velocity. For the purposes of position update velocities are normalized using sigmoid function.

3 Java Implementation

For reasons we already explained in Introduction section (1.1), the program SITO_tester is not standalone application but a java project for NetBeans IDE.

Before starting the work on the program itself it was needed to think about the structure of the program. The structure is important for good functioning and is hard to change later during the programming. To allow users to easily modify the program it is also necessary for the structure to be easy to understand, intuitive.

3.1 Program Structure

The pseudocode of SITO algorithm (1) shows there are four main actions that together make the optimization process. They are evaluation, strength computation, impact computation and update. These four actions are repeated until the optimization is completed. It seems only natural that each action should be implemented as java class encapsulating its sub-processes. Names of the classes are *Fitness*, *Strength*, *Impact* and *Update*. SITO is population based algorithm, relations among candidate solutions play important part in the optimization. To store the population, information about it and its properties another java class called *Topology* was used. The optimization process information, in fact the core of SITO algorithm, is inside java class *Optimizer*. The six classes are enough to perform SITO optimization, but it proved to be useful to create one more class, i.e. *Results*, to handle results of the optimization, as plotting graphs and comparing or saving the results.

At initialization all the other classes are input parameters of *Optimizer*. The optimization process then takes place within *Optimizer* class, the output is class *Results* containing the results of the optimization.

When testing the SITO on different fitness functions we noticed that assigning input parameters of the *Optimizer* class manually each time is not efficient and so the last class in the project was created - *Cycling*. Included are methods for setting up the optimization automatically as well as for performing multiple runs of optimization and comparing them.

3.2 Description of the Classes

The classes are implemented as abstract and serve as templates for various range of specific classes that fulfil their purpose. The abstract classes are stored in own packages with corresponding name, the specific classes are in ”corresponding package”.custom (e.g. abstract class *Fitness* is in package named ”fitness” and specific fitness functions are in package ”fitness.custom”). In following list each class is explained further. Additional information can be found in SITO_tester documentation.

- **Fitness**
  Fitness function is in fact a description of the problem, the subject of optimization. The optimization process is a search for candidate solution with the highest value of fitness function. Some of the fitness functions already implemented are described in section 4.1.

- **Strength**
  Strength function computes social strength from fitness values of candidate solutions, equation (1) can be used for this purpose. Should the optimization be successful, the social strengths have to be proportional to fitness values. The action of computing strength produces 2-D matrix of social strength values mapping the relations of all individuals in the population, indexes of columns correspond to indexes of individuals that are affected while indexes of rows to individuals that are affecting others.

- **Impact**
  Overall impact on given candidate solution is computed inside this class based on partial impacts of all neighbors. Each neighbor contributes to the overall impact according to the social strength he has toward target candidate solution. The choice of exact equation used to compute overall impact is important and can change quality of the optimization ability, for this purpose it is common to use equation (2) as it leads to good results.

- **Update**
  Candidate solutions in the population are updated to create new candidate solutions. The change depends on overall impact on each candidate solution and on the update equation. The formula (3) shown above, proved to be effective. However other, more probabilistic, formulae can be used as well. Mutation, if present, is part of the update function, the mutation rate κ is usually proportional to dimension of the optimization.

- **Optimizer**
  This class is a template for any optimizer, in the SITO_tester program SITO and BPSO are already implemented. This class implements functions like stopping condition or results processing while minimizing computation time for the optimization. Any new optimizer algorithm, similar to SITO and BPSO, is easy to implement, because the only thing required is defining instructions for one iteration. Results of the optimization are obtained as the output of the optimization process and can be saved to txt file or further used in the program.

- **Results**
  Factically, there are two classes involved in processing the results - *Results* and *ResultsModule* - but they are strongly related and thus their functions can be described together. Every *ResultsModule* class is stored in "results_modules.custom" package. In each iteration the *ResultsModule* scans the population for previously selected characteristic and stores it (e.g. best fitness value so far or average fitness value). *Results*

class contains a set of *ResultsModule*s so it is possible to monitor a number of characteristics during one optimization process. After the optimization, the results can be plotted in graphs (using free java library JFreeChart([19])), saved to files or compared to other results.

- **Cycling**
  Functions that simplify testing are implemented inside *Cycling* class along with settings of the algorithm. It is easy to automatically perform the same optimization many times, to change the fitness functions or to compare results from different optimization processes.

4 Preliminary Experiments

To compare SITO and BPSO methods we use five fitness functions, each defining different optimization problem. Every optimization process is repeated 30 times, average values are compared in graphs.

4.1 Fitness Functions

Performance of SITO and BPSO algorithms is tested on these five fitness functions:

1. ONEMAX:
   One maximum function is simple function counting number of positive values in the binary vector. Dimension $D = 300$, fitness value of optimum is $f(\mathbf{s}^*) = 300$.

2. ECC:
   Error Correcting Code Design Problem is search for codewords of alphabet, their minimal Hamming distance is maximized. In every candidate solution there are 24 codewords each containing 12 bits. Fitness value is computed using following formula: $f(\mathbf{s}) = 1/(\sum_{i=1}^{D} \sum_{j=1}^{D} \delta_{Hij}^{-2})$ where δ_{Hij} is Hamming distance between codewords i and j. Dimension of the problem is $D = 288$, fitness value of optimal solution is $f(\mathbf{s}^*) = 0.0674$.

3. PPEAKS:
   The function generates P random strings representing optima, that create peaks in the fitness landscape. Every candidate solution is evaluated by the number of bits in common with the closest optimum divided by the dimension of the problem. Dimension $D = 300$, fitness value of optimal solution is $f(\mathbf{s}^*) = 1$.

4. BIPOLAR-6:
   The fitness value is computed by applying function f_6 on six-bit substrings of the evaluated binary vector, which is defined as $f_6(\vec{\mathbf{s}_i}) = 1, 0, 0.4, 0.8, 0.4, 0, 1$ for $|\vec{\mathbf{s}_i}| = 0, 1, 2, 3, 4, 5, 6$. Dimension $D = 120$ corresponds to 20 subproblems so the fitness value of optimal solution is $f(\mathbf{s}^*) = 20$.

5. BINARY:
   Input binary vectors are evaluated as binary representations of numbers. Fitness value of each is difference between its decimal value and goal number. Dimension $D = 100$, fitness value of optimal solution is $f(\mathbf{s}^*) = 0$.

BPSO	
Population size	400
Neighborhood size	4
Inertia weight	1
Experience weight φ_1	2
Individual weight φ_2	2
Max. velocity V_{max}	5

Table 1: BPSO parameter settings

SITO	
Population size	400
Neighborhood size	5
Mutation rate κ	$\frac{0.8}{D}$

Table 2: SITO parameter settings

4.2 Parameter Settings

Parameters for BPSO and SITO are shown in table 1 and 2, respectively. The selected parameters for BPSO [1] are those commonly used, they are known to provide good results. Besides the population size, SITO algorithm requires only two parameters: mutation rate κ and number of neighbors. The value of κ was obtained by simple testing. The number of neighbors suggested in reference [3] is six, but in this work it was changed to five so it would be closer to the same parameter of BPSO. The size of population is discussed in section 4.4.

4.3 Experiments Results

Performance of both optimization techniques is shown in graphs in figures 1, 2, 3, 4 and 5, values in the graphs corresponds to fitness values of best candidate solutions found so far averaged over the 30 runs of the optimization. Blue line marks values achieved by SITO algorithm, red line is for BPSO values. In most of the graphs the blue line seems to disappear when the optimal solution is found, that is because the red line is painted over the blue line so when values of SITO are the same as values of BPSO only the red line is visible.

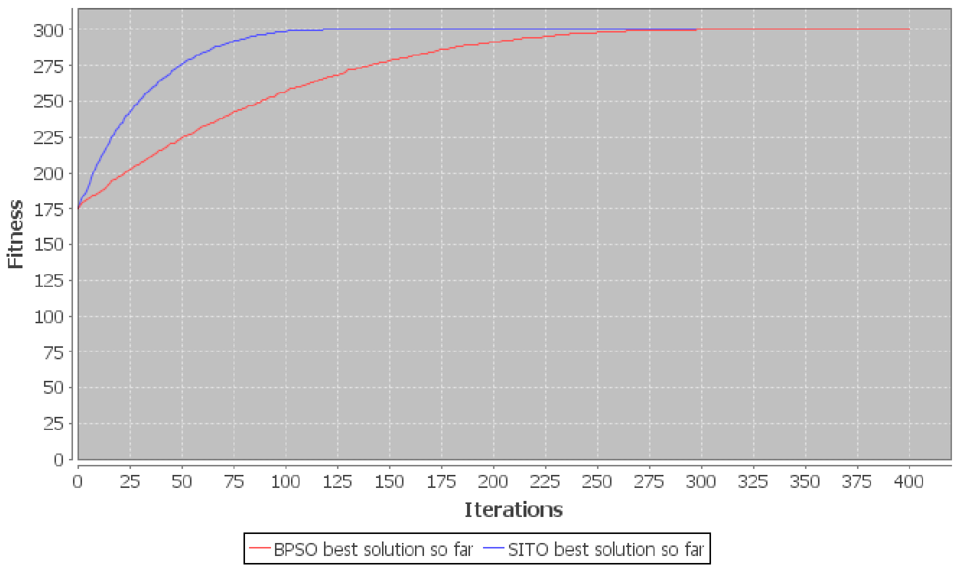

Figure 1: Performance of SITO and BPSO algorithms on ONEMAX fitness function

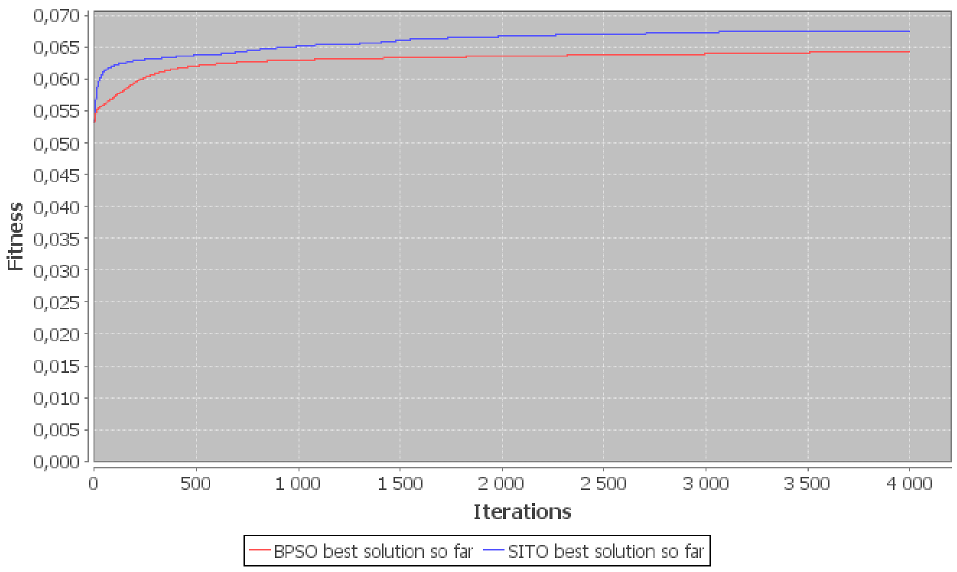

Figure 2: Performance of SITO and BPSO algorithms on ECC fitness function

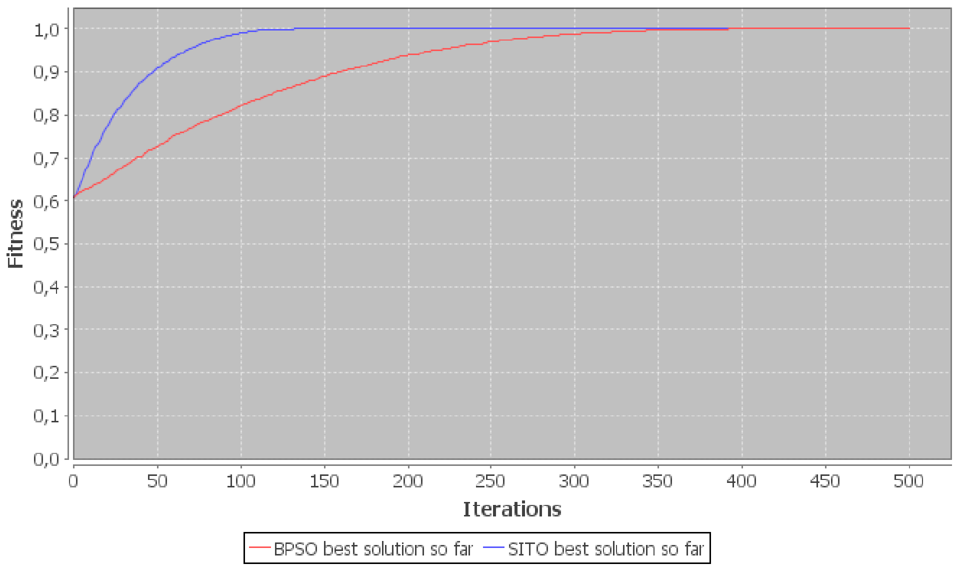

Figure 3: Performance of SITO and BPSO algorithms on PPEAKS fitness function

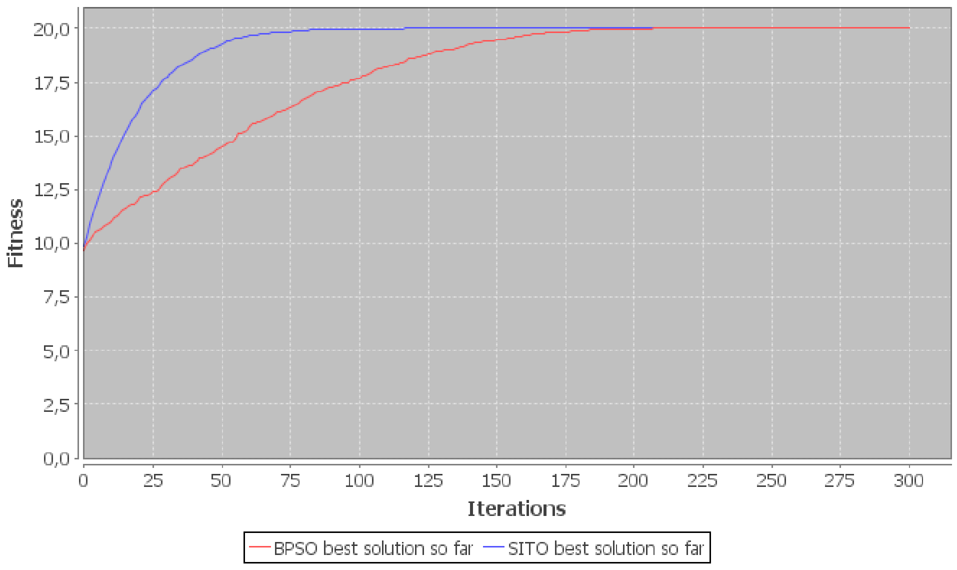

Figure 4: Performance of SITO and BPSO algorithms on BIPOLAR-6 fitness function

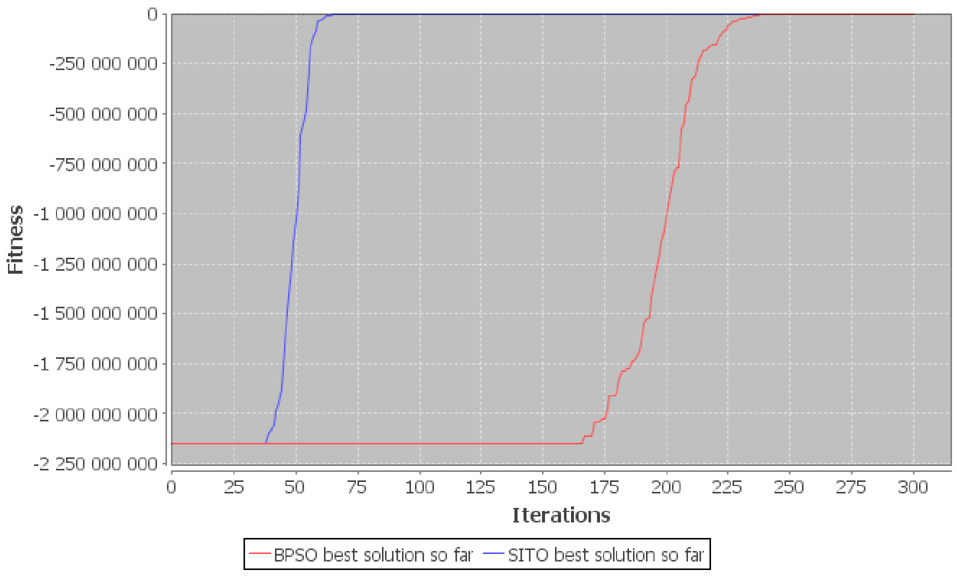

Figure 5: Performance of SITO and BPSO algorithms on BINARY fitness function

population	BPSO	SITO
10	891	3667
20	547	1968
30	427	938
50	244	562
100	211	378
200	207	113
300	201	85
400	182	90

Table 3: Number of iterations required to find optimum with different population sizes

4.4 Population Size

SITO and BPSO are both population based algorithms and the size of the population strongly affects the efficiency of the algorithms. A simple test was used to determine how the performance of each algorithm changes with the population size. Each algorithm was run five times and the best fitness value achieved so far was noted in each iteration, average value of the five runs was taken as the result. The number of iteration was set high in order to provide enough time for the algorithms to find the optimal solution. The test was performed on a problem defined by fitness function BIPOLAR-6, which was described in section 4.1.

The results are shown in table 3. The number of iterations needed to reach the optimum as well as the population size are given for both algorithms. It seems that the efficiency of the BPSO algorithm does not strongly depend on the size of the population and it can yield relatively good results even for small population size. In contrast, the SITO algorithm becomes efficient only when the population size is around one hundred candidate solutions or more.

5 Conclusions

The implementation of Social Impact Theory algorithm (SITO) into a Java application and creating a useful tool for further work and research of the algorithm is the most important contribution of this thesis. The program is called SITO_tester and its purpose is to simplify work for people interested in the SITO algorithm. SITO_tester is open to modifications and extensions so that it can meet the requirements of its users. In the development of the program the priority was to keep the structure as intuitive and understandable as possible and the single parts of the program easily changeable. To help understand the inner functionality of the program, javadoc documentation containing description of all classes and methods within the program is included.

The functions of the program were successfully tested in the preliminary experiments. The Binary Particle Swarm Optimization algorithm (BPSO) was also implemented into the SITO_tester and even though it does not share the structure of the SITO algorithm it can still use the functions for processing the results.

Initially it was assumed that the results of a single optimization processes will be saved to text files and later processed in other programs, i.e. MS Excel or Matlab. Later, it became clear that a simple results processing inside the SITO_tester would be useful and so the possibility to compute the average value from multiple runs of the optimization was included. A free java library JFreeChart made possible plotting graphs inside the program. SITO_tester can compare, plot to graphs or save to text files the results from multiple optimizations. If no advanced functions are required to process the results of the optimization, no other programs are needed to process the data.

Five testing functions were chosen to compare performance of SITO and BPSO algorithms: Onemax, Bipolar-6, PPeaks, ECC, Binary. These testing functions are implemented within SITO_tester program. It was unexpected that the preliminary experiments showed SITO algorithm exceeding BPSO algorithm on every one of the five used testing functions. Several other simple tests were performed to determine the cause and they proved that the size of the population in order of hundreds candidate solutions is better for SITO algorithm than for BPSO. However for small population sizes BPSO algorithm strongly outperforms SITO. It might be caused by a lost of diversity in the population and the SITO algorithm needs long time to achieve the optimum by random mutations.

5.1 Future Work

SITO_tester could be used for individual projects or theses for students interested in this field. Many new functions could be added to make the program more usable and could extend its applicability. Some of the possibilities for further development of the program are suggested:

1. Extending the mechanisms for processing results or adding advanced functions for plotting graphs.
2. Further minimizing the computation cost of the optimization process.
3. Including Graphical User Interface would make the program more attractive and easier to use.
4. Adding the possibility to load data from files.
5. Including visualization of the optimization process could make the program useful tool for better understanding the advantages and disadvantages of the optimization method.
6. Implementation of other optimization methods within the SITO_tester.

References

[1] Russell C. Eberhart, Yuhui Shi, and James Kennedy. *Swarm Intelligence (The Morgan Kaufmann Series in Evolutionary Computation)*. Morgan Kaufmann, 2001.

[2] M. Macaš, L. Lhotská, and V. Křemen. Social Impact based Approach to Feature Subset Selection. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, 2008.

[3] M. Macaš and L. Lhotská. Simplified Social Impact Theory Based Optimizer in Feature Subset Selection. In *Nature Inspired Cooperative Strategies for Optimization*, pages 133–147, Heidelberg, 2011. Springer.

[4] Amol P. Bhondekar, R. Kaur, R. Kumar, R. Vig, and P. Kapur. A novel approach using dynamic social impact theory for optimization of impedance-Tongue (iTongue). 2011.

[5] Amol P. Bhondekar, R. Kaur, A. Kumar, R.and Gulati, C. Ghanshyam, and P. Kapur. Enhancing electronic nose performance: A novel feature selection approach using dynamic social impact theory and moving window time slicing for classification of Kangra orthodox black tea (Camellia sinensis (L.) O. Kuntze). 2012.

[6] SIFS Tool. *bio.felk.cvut.cz*.

[7] SITO Library. *www.sitolib.org*.

[8] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS 1995*, pages 39–43, 1995.

[9] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 1996.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[11] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press, 1975.

[12] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. 1997.

[13] Leandro N. de Castro and Fernando J. Von Zuben. Artificial Immune Systems - Part I: Basic Theory and Applications. Technical report, 1999.

[14] E. Alba, B. Dorronsoro, M. Giacobini, and M. Tomassini. Decentralized Cellular Evolutionary Algorithms. 2004.

[15] Yuhui Shi. Brain storm optimization algorithm. In *Proceedings of the Second international conference on Advances in swarm intelligence - Volume Part I*. Springer-Verlag, 2011.

[16] A. Nowak, J. Szamrej, and B. Latane. From Private Attitude to Public Opinion: A Dynamic Theory of Social Impact. 1990.

[17] M. Macaš and L. Lhotská. Social Impact Theory based Optimizer. In *Advances in Artificial Life*, pages 635–644, Heidelberg, 2007. Springer.

[18] J. Kennedy and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics*, 1997.

[19] JFreeChart. *www.jfree.org/jfreechart/*.

Appendix

CD Content

Contents of the CD are listed in table 4.

Name	**Description**
thesis	bachelor thesis in pdf format.
SITO_tester	source codes of the program
experiments	data from the preliminary experiments
documentation	documentation for the program
instructions	simple guide for the SITO_tester

Table 4: Contents of the CD