Introduction and Benchmark Result Comparison of Social-Inspired Algorithms

Sosyo-İlham Algoritmalarına Giriş ve Kıyaslama Sonuçlarının Karşılaştırılması

Ayşenur Dalmış* - Fatih V. Çelebi**

Abstract: In the field of research and development, the most valuable source of inspiration comes from nature itself. Animals, insects, microorganisms exhibit some working principles that inspire today's science and to advance and present ideas in this field. These principles are called bio-inspired algorithms. Of course, in the years when a lot of diversity exploded in the suggestions of bio-inspirational algorithms, it did not take long to introduce new techniques, the concept of which is a very special animal; human. And these techniques are known as socio-inspired algorithms that inspire their functioning in behaviors based on human species and can be applied to solve any problems. Socio-inspired algorithms are a particular type of metaheuristic that bases its behavior on human society. It is a simple approach to the complex world of evolutionary computing, with multiple simulations with recognizable patterns day by day. The aim of this study is to check whether these algorithms have the functional capacity to solve optimization problems. Through experimental analysis, the results of these algorithms will be collected for a known benchmark and compared with those of some reference algorithms that allow assessing their potential in this type of problem adequately. According to the results of the comparison, the effectiveness of the algorithms will be evaluated.

Structured Abstract: In recent years, many efficient metaheuristic algorithms have been proposed for complex, multimodal, high dimensional and nonlinear search and optimization problems. Physical, chemical or biological laws and rules have been used as inspiration for these algorithms.

The metaheuristics that have been most successful in this field have usually been bio-inspired. Those that base their functioning on nature, and generally on animals. After the success harvested by these algorithms, new versions are born that, try to emulate them, and perform their operations following a model based on the behavior of society. These algorithms are called socio-inspired. The socio-inspired algorithms, inspired from social evolution observed in humans, are fairly recent developments in the field of nature-inspired metaheuristics. They provide an attractive, innovative, and more straightforward design to understand and apply than the traditional

* Ağ Güvenliği Uzmanı, Sayıştay
Network Security Engineer, Turkish Court of Accounts
ORCID 0000-0003-4122-1013

Aysenur.akyol@sayistay.gov.tr

** Prof. Dr., Yıldırım Beyazıt Üniversitesi, Fen Bilimleri Fakültesi, Bilgisayar Mühendisliği
Prof. Dr., Yıldırım Beyazıt University, Graduate School of Natural Science, Computer Engineering
ORCID 0000-0001-8569-1098

fatihvcelebi@gmail.com

Cite as/ Atıf: Dalmış, A. (2020). Introduction and benchmark result comparison of social-inspired algorithms, *Turkish Studies-Applied Sciences*, 15(1), 39-56. <https://dx.doi.org/10.29228/TurkishStudies.40085>

Received/Geliş: 09 December/Aralık 2020

Checked by plagiarism software

Accepted/Kabul: 25 March/Mart 2020

Published/Yayın: 30 March/Mart 2020

Copyright © INTAC LTD, Turkey

CC BY-NC 4.0

evolutionary algorithms and with continual social evolution across generations improves the learning or convergence resulting in a better cultural species, i.e., the global optimal solution to an optimization problem. And they are the center of the study of this work.

The primary objective of this thesis is to analyze how some of the most exciting versions of socio-inspired algorithms work through a comparative study with twenty-five functions of a benchmark. A total of six socio-inspired algorithms will be studied, namely:

- Anarchic Society Optimization Algorithm (ASO)
- Ideology Algorithm (IA)
- Imperialist Competitive Algorithm (ICA)
- Parliamentary Optimization Algorithm (POA)
- Social Emotional Optimization Algorithm (SEA)
- Soccer League Competition (SLC)

And to make a comparison, Particle Swarm Optimization Algorithm (PSO) selected as the reference algorithm.

The overall and final goal of the study of this work is to analyze how effective these new techniques are and whether they have a gap between the elite of evolutionary computation, as well as their bioinspired analogs. To do this, firstly, a study must be made in the algorithm literature to fully understand how the algorithm behaves in each moment of the optimization process. The knowledge obtained from each one will be documented in its corresponding chapter of the memory.

Once the whole process is understood, the implementation phase takes place. For this purpose, the knowledge acquired in the previous step has been used, being as much as possible to the proposed design. These part has two step first one is implementation of algorithms and the second part is implementation of experimentation. First of all, the algorithms implemented and these will mentioned detaily in coming chapter. After implementation of algorithms, for the experimental part; firstly, 10 scripts were built for each algorithm to experiment. Five for problems of dimension 10 and five for those of dimension 30. With a total of five functions being executed in each of them. Each function has executed a total of 10 times, collecting the average value obtained at the end of the evaluations for each of these executions.

With all the algorithms implemented, the next step, and more critical for the study to be done, is the experimentation. During the application phase; performance of algorithms are tested with a set of standard test fuctions which are the special session on real-parameter optimization of the IEEE Congress on Evaluatory Computation CEC2005.

Finally, the study itself goes through analyzing the results obtained for each algorithm and function. And compare them in two different ways. In the first place, a comparison will be made with a reference algorithm which is PSO to ensure that said socio-inspired algorithms are able to select a solution that is at least as accurate as standards. Secondly, and to finalize the study, the results of all the socio-inspired algorithms will be compared with a reference algorithm which was compared before with same benchmark, the pros and cons of the best solutions will be analyzed, and a global perspective of the position of these algorithms will be given in the world of metaheuristics. These comparison made with a reference algorithm which is EPVE. These algorithm is an elitist algorithm and presented in 2010. In these study result of socio-inspired algorithm compared with these algorithm results. And the result shows that a total of 25 function, there are only 8 in which socio-inspired algorithms (SLC and ASO results) turn out to be better. It should be remembered that these comparative results were presented in 2010. So, it is due to sign that according to CEC2005 function results not even the best results of the socio-inspired are a worthy rival for algorithms a decade ago.

However when looking at algorithms that beat the reference algorithm, it is seen that some algorithms beat it by a large margin. Based on these algorithms, a new hybrid algorithm that will be created may be found to be more effective and successful in this field.

Keywords: Socio-Inspired Algorithms, Bio-Inspired Algorithms, Evolutionary Algorithm, Particle Swarm Optimization, Experimental Analysis, Comparative.

Öz: Araştırma ve Geliştirme alanında, en değerli ilham kaynağı doğanın kendisinden gelir. Hayvanlar, böcekler, mikroorganizmalar günümüz bilimine ilham veren ve bu alandaki fikirleri ilerleten ve sunan bazı çalışma prensipleri sergilemektedir. Bu ilkelere biyo-ilham algoritmaları denir. Tabii ki, biyo-ilham verici algoritmaların önerilerinde çok fazla çeşitliliğin patladığı yıllarda, konsepti çok özel bir hayvan olan yeni teknikler sunmak uzun sürmedi; insan. Ve bu teknikler, insan türlerine dayalı davranışlarda işleyişlerine ilham veren ve herhangi bir sorunu çözmek için uygulanabilen sosyo-ilhamlı algoritmalar olarak bilinir. Sosyodan ilham alan algoritmalar, davranışını insan toplumuna dayandıran belirli bir meta-sezgisel türdür. Evrimsel bilgi işlemin karmaşık dünyasına, her gün tanınabilir kalıplara sahip çoklu simülasyonlarla basit bir yaklaşımdır. Bu çalışmanın amacı, bu algoritmaların optimizasyon problemlerini çözmek için işlevsel kapasiteye sahip olup olmadığını kontrol etmektir. Deneyisel analiz yoluyla, bu algoritmaların sonuçları bilinen bir kıyaslama için toplanacak ve bu tip problemlerdeki potansiyellerinin yeterince değerlendirilmesine izin veren bazı referans algoritmalarla karşılaştırılacaktır. Karşılaştırma sonuçlarına göre, algoritmaların etkinliği değerlendirilecektir.

Anahtar Kelimeler: Sosyo-İlham Algoritmaları, Biyo-İlham Algoritmaları, Evrimsel Algoritma, Parçacık Sürü Optimasyonu, Deneyisel Analiz, Karşılaştırmalı.

1.Introduction

Computer science has evolved a lot since its inception, and solving complex mathematical formulas or extensive computing calculations is no longer a problem. Many algorithms are responsible for performing these tasks, and that any user can use without having expert knowledge. The new challenges of computer science are plagued with problems that do not have a clear and precise solution, among them the problems of optimization, that will be studied in this work. Just as a calculation problem is minimal and a programmer can devise an algorithm that, based on different formulas and operations, always obtains the desired result, the same can not be said of the problems of optimization, which are influenced by a multitude of factors and that they do not have an absolute solution.

This situation becomes manifest when we want to obtain the values that make a function minimal, and its domain is so great that testing with the crude brute force method is not an option. For this, the metaheuristics were designed, algorithms that look for the best solution to a function and that do not need to be raised in a different way depending on each one. Throughout its (still brief) history, there have already been many different algorithms that provide their solution to this problem. However, this work will focus on the new socio-inspired metaheuristics and analyze what they contribute to the current scenario and on what they are based on.

2. Analysis of the Algorithms

In this section, a detailed analysis will be carried out on how each of the algorithms selected for the study works from a theoretical point of view. The techniques that each one uses to arrive at the optimal solution and how these techniques resemble human behavior will be explained, in order to be considered a socio-inspired algorithm. The algorithms will be analyzed in chronological order, from the newest Ideology Algorithm to the oldest Imperialist Competitive Algorithm.

2.1 Ideology Algorithm

Chronologically the newest one socio-inspired algorithm of the study is the Ideology Algorithm. This proposal is based on an idea similar to that of the Parliamentary Optimization Algorithm, previously presented in this same chapter since it also takes place in a political context in which different parties try to gain control of the government. These are represented by a leader, which would be equated to the party president, although the figure of the vice president, or second-best member of the party, also plays a key role.

In this algorithm, the leader has to try to stay ahead of his party while trying to ascend to be placed as president of the government. For this, it goes through three phases, one of introspection, another of local competition, and a third of the global competition.

- During the introspection phase, the party leader looks around his current position to try to improve facing the next phase.
- During the local competition, he looks for the second-best politician of his party, the vice president, to try to maintain his position and anticipate a possible future movement of the latter.
- Finally, in the global competition, the leader calculates a movement similar to the previous ones but for the current president of the government.

Once you have assessed all movements, select one using the “Roulette Wheel Selection” method presented by Rakesh Kumar and Jyotishree (Kumar and Jyotishree, 2012).

When the leader of each party has made his move, and before the rest of the members take their movement in the political debate, the worst member of each party has to assess whether he continues to feel close to the organization or if he prefers to leave. In case their differences with the second-worst member are large enough, the member leaves the party and joins another randomly selected one. At that time, if the movement of the other members occurs, they seek to improve their position in the surroundings. At the same time, they try to look like the leader of their party. After updating his position, if any of the members have become the best politician of the party, he will occupy the position of leader. In short, the party is reorganized according to the new abilities of the members. Finally, the politician who occupies the position of president of the government when the execution of the algorithm is finished will be considered the optimal solution found by the algorithm. In the following pseudocode, this operation can be performed.

Tablo 2.1: Pseudocode of the IA.

Ideology Algorithm

```

parties  $\leftarrow$  initializePopulation()
while not stopCriterion do
    order politicians in function of their fitness
    selfanalysisi  $\leftarrow$  selfanalysis(leaderi)
    compLocali  $\leftarrow$  competitionLocal(leaderi)
    compGlobali  $\leftarrow$  competitionGlobal(leaderi)
    leaderi  $\leftarrow$  selectionMovement(selfanalysisi, compLocali, compGlobali)
    if (lasti – worsti) > T then
        worsti  $\leftarrow$  chooseNewParty()
    for each politician do
        politician  $\leftarrow$  newDisplacement()
return min(weight (leaderi))

```

By ordering the politicians depending on their adjustment or fitness can be obtained in each iteration, which is the member of each party that acts as leader. This will be the one that experiences the three phases previous to the movement of the rest of the members. In the self-analysis phase, a small mutation will be made in the characteristics of its solution vector, in order to explore the search space contiguous to its current position. In the local and global competitions, in which the leader of the party tries to approach the positions contiguous to the second-best solution of his party and the best global position existing in said iteration, respectively, a similar process happens. However, it changes the objective of displacement.

These movements are a local search that takes place always on the best solution for each party, which can help improve performance evaluations devoting more those solutions with the potential to be optimal. And if these movements enhance the exploitation phase, the exploration phase, in turn, is reinforced by the possibility that politicians have to leave a party. The simile with not being related to his party is determined by the distance between the immediately worse representative of the party and himself. If this distance exceeds the limit of desertion established in the algorithm, the politician will

join another random party. This allows new situations to be explored since approaching a political leader can be substantially different depending on the member making the displacement. When all the other members of the parliament have moved, in this case, towards the leader of their party and while exploring their contiguous position, an iteration of the main loop is terminated, and the positions of each party are reordered. Also, if one of the new leaders turns out to be better than the current president (has better fitness value), he will occupy that position.

The ideology algorithm has its roots in a Particle Swarm Optimization, as mentioned before. Each party functions as a small model of Particle Swarm Optimization, in which the political leader of the party symbolizes the best-placed particle towards which the best move. When there is a varied number of small PSOs, the exploration is more significant, but it is even more emphasized by the fact that these groups can interact with each other in the exploration process. Thus, the leader of a party can be influenced by the position of another party, or a member recently affiliated with the party can help direct it to an unexplored area of space.

2.2 Soccer League Competition Algorithm

In this algorithm, the problem is located in a sports setting, in particular, in a football league. The population is constituted by teams, each one of them with different players. At the same time, there is a distinction between main and substitute players, and some logically have more importance than the others. In this scenario, the goal of each player is to become the best in the league, or at least to be the one that is sufficiently good so that the best teams in the competition want to have him in their squad. The teams contest with each other over the seasons, and the players improve based on their outcome in the matches, and the influence they have had on the game. The main player will improve more than a substitute player, and if a holder lowers the level, he can be relegated to the bench. In turn, if a substitute player does not qualify for the team, they can fire him and sign a new replacement for the next match. The movement dictates that the top-level players seek to resemble the "Super Star Player" to choose to play in better teams, while the alternates try to reach the middle level of the holders of their team, to find a position in the rotation. Concerning the losing team, a percentage of its players will try to train some new technique with which to beat that team in the next game. When the season ends, the best teams are made with the best players, leaving the worst for the lowest level teams. Thus, the following season, the most promising players can continue to improve by being surrounded by an environment at their success. And the low level can stand out within their team, being able to get the attention of a more powerful one. The superstar player will be the one that the algorithm returns as a result of all seasons. Here is the pseudocode of the algorithm that explains more technically the details.

Tablo 2.2 Pseudocode of the SLC.

Soccer League Competition

```

league  $\leftarrow$  initializeLeague()
while not stopCriterion do
    for each pair(teami , teamj) do
        winner, loser  $\leftarrow$  game(teami , teamj)
        mainwinner  $\leftarrow$  operatorImitation()
        substituteswinner  $\leftarrow$  operatorProvocation()
    reorder players according to their fitness
    if conditionDescent then
        teamworse  $\leftarrow$  newTeam()
return weight (superstarplayer)

```

In the first place, the population is composed of players. They are the ones who represent the solutions to the problem, and who will evolve as the algorithmic process takes place. This population is

divided into teams, each of which has its players, and this template with regular players and substitute players. The owners will be the best solutions for the team, and the substitutes those not so good. The solutions must be ordered in each season, which corresponds to an iteration of the main loop. So the first team in the league will always start the season with the best players, while the last will be the one with the worst. However, in the middle of each season, it may be the case that a player of a more low-level team comes to give a solution good enough to occupy the best positions in the league. In this case, this player will be signed by the best team when the season ends. The process of the season consists of each team playing a game against the other teams and depending on the result, and their players will improve in one way or another. This match is simulated by calculating the cost of each participating team, estimating the probability of each to win, and then a random value is generated that determines which team wins. The holders of the winning team make a move towards the best player in the league, the "Super Star Player". If this movement does not mean an improvement, they make a move towards the "Star Player" of their team, which means their best player. If improvement is not found, try to make a last random move to a different position. If the result still does not improve, the player does not move as a result of that match.

The substitute players of the winning team must try to look like the main players, hoping to take their place. For this, the gravity operator is introduced, which, when applied to the players holding the team, represents the average position they occupy in the search space. This will be the point at which the substitute player tries to move. In case not get a good result, perform a similar movement in reverse, moving away from that center of gravity. If this movement is not an improvement to the initial position, the player is dismissed from the team, and a new randomly generated substitute takes his place.

For the losing team, the algorithm selects three members of the main staff and changes a percentage of its components. Only if the improvement is better than the original, the mutation is recorded. Then two substitutes of the said losing team are selected, and a genetic cross between them is generated, obtaining two new solutions. The team reorders its staff and stays with the best players of all generated, freeing the worst.

Finally, at the end of the season, the players are again ordered, but this time on a global level, which simulates the signing of the best players by the best team. The substitutes of the first team are better than the holders of the second team, and so on, posing a league where there is a team with enough purchasing power to get all the best players. If the descent component has been specified, when the necessary condition has been met, the worst equipment is eliminated from the competition and filled with new randomly generated players to provide more exploration throughout the entire process.

2.3 Anarchic Society Optimization Algorithm

This algorithm based on an anarchist society in which the figure of the government is not necessary and where the individuals organize themselves. Separately or through groups, they can determine which is the best possible decision in each case based on their own experiences and those of the rest of society. For this, each has to make a decision when moving;

- have to assess whether it is better for you to your situation,
- on the experiences of the rest of today's society
- on what happened in the past.

If you make this comparison in a balanced way and taking advantage of all the information available (it is assumed that in this society, the individual is aware of the decisions that take the rest) can reach a more suitable position or even optimal. Besides, individuals in an anarchic society may experience radical changes of opinion and move towards positions that do not necessarily have to be better. In the words of the original author, "individuals who are fickle, adventurous, dislike stability, and frequently behave irrationally, moving toward inferior positions they have visited during the exploration

phase.". When they have assessed all the possibilities of movement according to the guidelines indicated above, the individuals decide what strategy to follow to take their new position. They can keep the strictly best of the proposed solutions or try to choose the best of each of them. Over time it is expected that at least a good percentage of society has learned the good behaviors and discover the area of the space where they can best be located, although other members are not so bright and wander through the rest of the space exploring different routes. In the end, it is ambitious to expect the whole society to converge on the same point, and, normally, splits occur, as the algorithm well represents. Below is pseudocode that helps express this process in terms of closer to the implementation.

Table 2.3: Pseudocode of the ASO.

Anarchic Society Optimization

```

society ← initializePopulation()
while not stopCriterion do
     $Mp_i^{current}$  ← movementOwnPosition()
     $Mp_i^{society}$  ← movementSociety()
     $Mp_i^{past}$  ← movementHistory()
     $society_i$  ← selectionMovement( $Mp_i^{current}$ ,  $Mp_i^{society}$ ,  $Mp_i^{past}$ )
    update better historical position
return weight (minimumHistory)

```

In the first place, it is convenient to note that the whole weight of this algorithm falls on the decisions of movement since it is what has workload in each iteration. This algorithm should only maintain the best position that each individual has achieved until that iteration and use three mathematical formulas that determine the new position according to each movement. The question is to find out if these formulas are enough to find an optimal solution in the search space. The movement based on its current situation is determined by a parameter defined as "fickleness" or tendency to change. A low value of this attribute will indicate to the individual that s/he should continue to explore the space by her/his surroundings, while a high value will have repercussions on her/him acting individually and radically changing her/his position.

In the movement based on the rest of social influences, of course, the best solution that the individual can find in the current iteration. S/he knows the position of the rest of her/his peers, and s/he knows who s/he should check, but instead of automatically copying it s/he calculates an "index of external irregularity". Again there is a threshold related to this index that determines whether the individual should proceed to move to the best position of the current iteration or on the contrary, it rebels and acts on its own, generating a position away from it.

Finally, the movement based on the historical record is very similar to the one just explained. In this case, the position in which the individual is fixed is not the best current but the best historical, and the calculated index is called the "index of internal irregularity." There is also an analogous threshold, only in this case, the displacement is made logically towards the best position that has been in all iterations. With all the possible movements evaluated, the individual must make a decision. At this point, the selection operator has to choose which movement to keep, and the algorithm provides two possible solutions:

- that is chosen according to an elitist model, staying with the best position of the three generated,
- or that is chosen according to a sequential model, and in each iteration change from one to another following the same series.

Each one has its advantages and disadvantages, which are also mainly excluding the other method. In the elitist case, many evaluations of the objective function are consumed three times more

than in another common evolutionary algorithm. In the following case, only one-third of the evaluations take place, but these do not give the assurance of being the best in each case. As far as the implementation of this study is concerned, it has been decided to develop the elitist option.

2.4 Social-Emotional Optimization Algorithm

This socio-inspired technique is based on the social relationships of people and how they react based on their successes and errors or how to acts in others. To this, we must take into account past actions and how they influenced the rest of society, and in turn, see how an individual influences the behavior of others.

Regarding this, it is expected that if a member of society sees that the rest of society finds positions better than her/himself, her/his index of emotion will decrease. Also, it will increase completely if s/he has placed the best position in all the population. In the beginning, all the individuals begin with the moral at the same level, but this will vary dynamically depending on the results that are obtained. When the time comes for each to make their next move, they have three possible ways of doing it:

- If her/his morale is very high because s/he is aware that s/he has a good position. The next movement leads to move away from the worst positions while at the same time fixing their own best historical position.
- If her/his morale is average because s/he is still somewhat satisfied with her/himself, even though s/he knows s/he can improve, s/he will move away from the worst positions while looking for a movement toward the best position s/he has had and the best position registered by the population so far.
- If his/her morale is very low because s/he is aware of how bad her/his position is, s/he will seek to be inspired by the best solution of the population so far.

In the following pseudocode, see how the algorithm works. We can start the algorithm analysis by understanding how the population and the different components observed in the pseudocode are technically represented, and therefore how they can be related to a socio-inspired technique.

Tablo 2.4: Pseudocode of the SEA.

Social Emotion Optimization Algorithm

```

society ← initializePopulation()
emotioni ← 1.0
societyi ← behavior1()
while not stopCriterion do
    if emotioni > limit1 then
        societyi ← behavior4()
    else if limit2 > emotioni >= limit1 then
        societyi ← behavior3()
    else
        societyi ← behavior2()
    if weight( societyi ) < weight( minimumHistory ) then
        emotioni ← 1.0
    else
        emotioni ← emotioni - Δ
    update better historical position
return weight( minimumHistory )

```

Initially, the population is a society of human beings, where each of the individuals of the same is a solution to the problem, randomly generated for the initial population. In this society, where individuals will grow up trying to be better and be above the rest. The index of the emotion of each will be a real value between 0 and 1, being 1 for all individuals at the beginning of the algorithm. The index function will increase or decrease by the success or failure of the individual. In the first iteration of the algorithm, as all indices are 1, a particular movement that occurs in the rest of the process takes place. This movement moves the individual away from the worst positions that currently exist in the population, as a defense mechanism to avoid ending up in a situation like that. The solutions to which they have fallen in the worst positions will see their indices of emotion reduced until they manage to move to a better area following the path of an individual with success. This algorithm keeps a historical position for each, the best it has achieved throughout the iterations. This position will be used both for the displacement formulas and for saving the best solution so far since this algorithm is not elitist and allows the best solutions to be lost from the current population. Saving them ensures that you can obtain the best solution found throughout the iterations when you have to return the final value of the process. Each time a move is made, the index of the individual's emotion changes. If the movement places her/him in the best historical position, her/his feeling increases to the maximum value. However, if s/he does not achieve, it said index would be reduced in a parameter Δ previously defined with the algorithm. In the function of the emotion of the individual, in each iteration s/he will choose to follow one behavior or another when it comes to moving. Behavior or other limits is determined by dividing a domain index (0,1) into three sections. This requires two boundaries, which are defined as parameters of the algorithm.

The Social Emotional Optimization Algorithm can be interpreted as a Particle Swarm Optimization, to which some modifications have been made, such as the memory of historical positions and the variation of behavior as a function of the emotion index. As a socio-inspired algorithm, it tries to give meaning to the decision-making of individuals, so that each range of values of the emotion index corresponds to one behavior or another.

2.5 Parliamentary Optimization Algorithm

This proposal simulates being within the political framework in parliament. There are several political parties in the government trying to come to power. Within each of them, there will be competition between the candidates to be promoted to the top positions. If anyone presents different and better solutions, they can succeed in this position. Several political debates could lead most of the minority parties to withdraw from parliament or to participate in an agreement to ensure that most of the majority of the votes are in the government. Among the candidates of each party, he is elected as the president of the parliament, considered the best politician. In the event of a single party or merger, the best representative of the group in question is also accepted. This algorithm has a simple operation, but at the same time, it is very intuitive to associate with society since it is located in a pervasive and everyday environment. Politicians are present in practically every act of daily life, and all know the structure of a parliament. In any case, the political debates that take place there are accessible to any citizen, and it is common to understand how they work and how the political parties operate to win seats and win the elections.

Here is a pseudocode algorithm to understand generally what the operation is included regarding steps and procedures. And we will begin to analyze this algorithm by relating the technical concepts presented in this pseudocode.

Tablo 2.5: Pseudocode of the POA.**Parliamentary Optimization Algorithm**

```

parliament  $\leftarrow$  initializeParties ()
candidatespartyi  $\leftarrow$  bestMembers(partyi)
while not stopCriterion do
  for each party do
    memberi  $\leftarrow$  (candidatesparty - memberi)  $\times$  bias
    if weight(memberi) < weight(candidatej,party) then
      candidatej,party  $\leftarrow$  memberi
    powerj,party  $\leftarrow$  calculatePower(party)
  merge the best 2 parties with probability pm
  eliminate the worst party with probability pd
return min(weight (candidate))

```

In the case of this proposal, the initial population is composed of solutions that are interpreted as human individuals. Throughout this algorithm, each of the solutions will be associated with a particular policy, and its evaluation in the objective function of the problem will determine the validity of said policy within the parliament. With the first politicians generated, this algorithm groups them by matches, as many as indicated in the parameters of the same. Besides, so it must select among the best politicians as many candidates as there are, to assign them to the different parliamentary groups. These will be distributed equally among the rest of the members of parliament, to start from a situation of equality. At this point, the initial situation of the problem has already been generated, and the main loop is entered. In each iteration, first, an internal competition takes place, and then cooperation between parties takes place that may or may not come to fruition within the party itself.

The internal competition is to move all politicians to candidates with a certain bias to avoid a gathering of the whole party in a single point. To do this follow the following formula;

$$p' = p_0 + \eta \left(\frac{\sum_{i=1}^{\theta} (p_i - p_0) \times f(p_i)}{\sum_{i=0}^{\theta} f(p_i)} \right)$$

In the above formula, η is the bias that applies to the movement. Another alternative mechanism is to use large values of η at first iterations and then gradually reduce it, perhaps by analyzing variance. If it takes a higher fitness value, a regular member is allowed to change only. After biasing, regular members might have higher fitness values than candidates. In such cases, candidates are reassigned again. In the figure below, the biasing operation is illustrated. P_0 is the current position of the member, and P_i represents each party's candidate. P' is the new position of the regular member.

Figure 2.1 Biasing a member toward candidates.

When the internal competition of the party ends, the cooperation phase takes place. At this moment, there are two parameters **pm** and **pd** that symbolize the probability that the two best matches will join and that the worst match will be expelled from the parliament, respectively. If the first probabilistic condition exists, the top two parties put aside their differences and merge into one, choosing as the best candidates for members between the two parties regardless of their party of origin. On the other hand, if the second case occurs, the party is eliminated from the parliament, and all members disappear from it, so the parliament sees its size reduced, emphasizing the best solutions for the rest of its iterations. This algorithm bases its operation on the Particle Swarm Optimization, behaving in a very similar way but giving a socio-political character to its environment. In this case, each party works as a PSO algorithm, with the politicians being the particulars and the candidates those located in a better position to which the best approach. In each iteration, the possibility of bringing together two parties can allow increasing the exploration, understanding that a politician can choose to approach a candidate who did not know before and on the way to explore an unknown area.

The main novelty of this algorithm contributes concerning the PSO is that of taking as the reference of displacement a more significant number of particles, in particular, the one that is desired and is specified in the call to the algorithm. This allows that they can generate more than one "good position" towards which to go, also exploring the intermediate roads. The criticism of this algorithm is based on the way to perform the displacement, which limits the exploration. By not allowing the particles to pass to occupy a worse position can easily fall into a local optimum if they stabilize in the first iterations. Also, the bias is a fixed parameter as documented by the authors, and that reduces the search randomness and can stop it entirely at a certain point where the same displacement is always made, and no particle improves its position.

2.6 Imperialist Competitive Algorithm

Finally, the oldest socio-inspired algorithm of the study is the Imperialist Competitive Algorithm. This algorithm starts from the idea of a completely colonialist world, plagued by countries that struggle to win victors and conquerors. In this constant war, the countries with more power would become the leaders and could count on numerous colonies at their service. On the other hand, those countries that can not defeat others would be relegated to occupy a colony position forever. During the war, there can also be a revolution or uprising of one of the colonies against their empire. If this colony turns out to be more powerful than the current leader, this position becomes his. It can also be the opposite case; If a colony is weak, an adjoining empire can absorb it and become stronger thanks to it. Furthermore, of course, one empire can be colonized entirely by another, thus taking over its remaining colonies and the imperial country itself.

As you can see from this road map of the algorithm, the relationship with society is evident. This is the operation of the algorithm in broad strokes, but then the most striking features of it will be detailed. The pseudocode of the algorithm is shown to help down below understand each step of the same before the detailed explanation.

Tablo 2.6: Pseudocode of the ICA.

Imperialist Competitive Algorithm

```

empires  $\leftarrow$  initializeEmpires()
while not stopCriterion do
    for each colony do
        colony  $\leftarrow$  colony + (imperialist - colony + rand())
        if weight(colony) < weight(imperialist) then
            imperialist  $\leftarrow$  colony
    empires  $\leftarrow$  competitionImperial()
    if any empire is empty then
        empires  $\leftarrow$  eliminateempires (empire)
return min(weight (imperialist))

```

First of all, it is convenient to talk about the population. Although usually in these socio-inspired algorithms, the population is usually made up of individuals, this being a group of them, in the case of the ICA algorithm, it is not so. In this algorithm, a simile is made with a global society where each of the solutions that make up the population is like a country, or in this case, like a colony. Therefore, the figure of the human being as an individual does not appear in this proposal. Once the first solutions or colonies that occupied the initial population have been instantiated, we must select those that are more powerful to consider them "imperialist"; that is, they will be the leaders of each of the empires into which the population is divided. Of course, both the number of colonies and the number of initial empires are nested in the parameters of the algorithm and can vary if desired. Throughout this algorithm, it will be understood that one country is more powerful than another if its evaluation for the objective function is less, given that it deals with optimization problems seeking the lowest possible value. The allocation of the empires is made concerning the initial colonies with a lower cost for the said function. With the constituted empires and the distributed colonies, the algorithm's main loop begins to be developed, which will end when the stop criterion is met. The implementation of this algorithm originally stopped after many "decades" (iterations of the main loop). However, in order to integrate it with the rest of the algorithms of the study, it was added the possibility of defining a stop criterion by many evaluations of the function objective.

Fig 2.2 Displacement of a colony towards its imperialist, with a component of randomness

The movement produced is left by the following formula

$$r \sim U(0; \beta \times d)$$

where d is the distance between the colony, and the imperialist and β represents the so-called "coefficient of assimilation". That is, it is a displacement towards the imperialist where the distance traveled is defined by a random uniform value between 0 and the maximum distance. In addition, the angle that appears in the figure, which deflects the colony from its direction, is calculated as follows

$$\theta \sim U(-\lambda; \lambda)$$

where λ is the parameter that adjusts the deviation of the direction, and which is called by the authors "coefficient of the angle of assimilation".

The changes produced by these mentioned operators help to diversify the positions of the colonies and not to be grouped automatically around the best solution of the empire. This allows the algorithm to explore the search space properly and not fall into possible local optimum easily in the first iterations, which limit the evolutionary process much. If the new position towards which a colony move turns out to be better than that of the current imperialist of that empire, the posts are exchanged, and the rest of the colonies will try to move towards the new better solution. The imperialist competition that

takes place in each iteration of the main loop brings even more variety to the process, leading the worst of the colonies of the worst empire to the empire that is most likely to acquire it, which is usually the most powerful. This power of the empires is calculated according to the weight of the imperialist and the average weight of their colonies. This decision also brings more variety as well as realism in the imperialist process, because if the country leader of the empire is mighty but has many weak colonies, generally said empire would be less powerful than one that has an imperialist something worse and fewer colonies although stronger. Once an empire runs out of colonies, it becomes absorbed by the strongest empire, and the number of imperialists is reduced. The process is repeated in each iteration of the loop until there is only one empire that contains all the colonies; in such a case, from that iteration in the main loop, only the colonies will move to new positions, trying to explore the surroundings of the imperialist country.

3.Design of Experiments

In this chapter, we will analyze how the study of the algorithms has been carried out. The benchmark of functions that have been used will be commented on, as the experiments have been designed so that they can be executed individually, allowing parallelisms and how the data has been compiled. For this first study of the algorithms has been preferred to use the CEC2005 benchmark that, apart from being a bit easier than the most current, is more intuitive to interpret.

Ten scripts were built for each algorithm to experiment. With a total of five functions being executed in each of them. The objective of dividing the work into several scripts is to take advantage of the parallelism available when using the Hercules cluster since the evaluations of both functions do not depend on each other, and it is not necessary to collect the data with a MapReduce model. Each function has executed a total of 10 times, collecting the average value obtained at the end of the evaluations for each of these executions. In order to make each experiment distinguishable from the rest while reproducible, a seed is set for random values before each function. If the seed is set correctly before the loop running the algorithm is processed 10 times, the same set of results will always be obtained. It is essential not to confuse this with initializing the seed within the loop, which would cause the same result in each of the 10 times, and therefore make it useless. As is usual in the problems of this style, the maximum number of evaluations per execution is determined by the size of the problem, being " $10000 \times \text{dim}$ " the formula in a question. For each algorithm, the convergence data are extracted to show them in a graph. These data will provide knowledge on how necessary all evaluations are, or which algorithms find the optimum faster. For all the algorithms, a size 30 population has been chosen for dimension 30. This limitation in size for the version of dimension 30 will allow seeing in the results that algorithms are better adapted with a smaller scanning margin.

4. Results and Discussions

4.1 Results

In this chapter, the most representative tables and graphs are compiled with the results obtained in the study. A global table will be shown for each dimension, including the average error obtained concerning the optimum by function and algorithm, in addition to a ranking table that helps determine which algorithms behave better in general. In addition to the tables, some convergence graphs will be included to be able to observe in a more visual way which algorithms arrive before the opticians and which ones need more evaluations, as well as which results are better and in what proportion concerning the rest. Finally, an analytical vision will be given about the results, giving rise to different well-founded assessments that can be extracted from the tables to obtain the conclusions of the study. To better assess how the study algorithms work, a reference algorithm has been chosen and have been added to the results tables. It is a Particle Swarm Optimization, an algorithm that was previously described in this document. To compare the results of this algorithm with that of those socio-inspired algorithms which behave in a similar way serve to determine if such modification to the PSO is significant or, on the contrary, does not contribute much.

Table 4.1: Average and ranking results of the CEC2005 benchmark for Dimension 30.

		1st	2nd	3rd	4th	5th	6th	7th
CEC 2005 FUNCTIONS		IA	SLC	ASO	SEA	POA	ICA	PSO
Unimodal functions	<i>F1</i>	1,98E+03	2,30E-01	5,07E+00	1,38E+01	1,46E+04	6,71E+03	2,83E+03
	<i>F2</i>	3,44E+03	1,60E-08	1,98E+02	1,29E+02	1,30E+04	9,93E+03	1,72E+04
	<i>F3</i>	9,48E+06	2,97E+04	1,14E+06	5,51E+06	1,25E+08	5,01E+07	9,03E+07
	<i>F4</i>	4,17E+03	3,59E-05	6,14E+01	9,35E+02	1,37E+04	1,17E+04	7,71E+04
	<i>F5</i>	7,20E+03	4,59E-03	8,04E+02	6,23E+02	1,90E+04	8,68E+03	1,75E+04
Multimodal Basic Functions	<i>F6</i>	6,22E+07	8,80E+01	1,39E+04	2,76E+04	3,38E+09	9,84E+08	3,26E+08
	<i>F7</i>	1,89E+03	5,67E-01	5,06E+00	1,92E+03	3,13E+03	1,26E+03	2,25E+03
	<i>F8</i>	2,04E+01	2,02E+01	2,04E+01	2,03E+01	2,09E+01	2,05E+01	2,02E+01
	<i>F9</i>	6,51E+01	3,88E+00	3,53E+00	3,29E+01	8,87E+01	7,43E+01	1,20E+02
	<i>F10</i>	9,13E+01	4,58E+00	9,49E+00	2,74E+01	1,16E+02	9,88E+01	2,07E+02
	<i>F11</i>	9,46E+00	3,80E+00	7,79E+00	8,26E+00	1,35E+01	1,03E+01	7,54E+00
	<i>F12</i>	2,25E+04	4,39E+02	1,29E+03	2,69E+03	1,11E+05	5,42E+04	5,52E+04
Multimodal Expanded Functions	<i>F13</i>	6,43E+00	5,71E-01	5,36E-01	2,39E+00	9,04E+00	1,01E+01	6,91E+00
	<i>F14</i>	3,93E+00	2,46E+00	3,75E+00	3,61E+00	4,51E+00	4,11E+00	4,85E+00
Multimodal Hybrid Composition Functions	<i>F15</i>	5,27E+02	2,72E+02	2,54E+02	4,28E+02	9,63E+02	6,97E+02	7,50E+02
	<i>F16</i>	2,80E+02	1,11E+02	1,22E+02	1,68E+02	6,12E+02	3,97E+02	5,50E+02
	<i>F17</i>	1,35E+02	3,22E+01	5,55E+01	9,55E+01	1,13E+02	8,38E+01	2,10E+02
	<i>F18</i>	1,09E+03	8,84E+02	7,77E+02	1,04E+03	1,38E+03	1,13E+03	1,25E+03
	<i>F19</i>	1,08E+03	8,37E+02	8,10E+02	8,56E+02	1,38E+03	1,16E+03	1,19E+03
	<i>F20</i>	1,08E+03	8,65E+02	8,16E+02	9,72E+02	1,38E+03	1,16E+03	1,27E+03
	<i>F21</i>	1,25E+03	6,70E+02	7,05E+02	8,99E+02	1,42E+03	1,34E+03	1,38E+03
	<i>F22</i>	9,52E+02	6,62E+02	7,25E+02	8,94E+02	1,35E+03	1,08E+03	1,25E+03
	<i>F23</i>	1,26E+03	9,56E+02	9,59E+02	1,07E+03	1,47E+03	1,32E+03	1,39E+03
	<i>F24</i>	1,13E+03	2,80E+02	2,37E+02	8,66E+02	1,45E+03	1,16E+03	1,46E+03
	<i>F25</i>	1,35E+03	5,20E+02	5,11E+02	5,24E+02	2,12E+03	1,10E+03	0,00E+00

In Table 4.1, ICA column is always less than the PSO column from these functions. SEA is the third algorithm with the best results in general, according to Table 4.1.

If the different groups of algorithms are studied according to their relationship with PSO, interesting conclusions can be drawn. ICA and POA are the ones that most closely resemble this algorithm, in which their most significant innovation is really to manage several structures similar to a PSO within the problem itself and make them interact with each other. SEA and ASO implement a historical component that helps improve solutions since it is also accompanied in both cases by decision-making based on the temporary state of the problem. This bifurcation in the path of the individual seems to be better raised in ASO, which achieves the best results of the algorithms that start from an idea based

on PSO, and that is that SLC, the algorithm that unquestionably finds better overall results, is an algorithm similar to Differential Evolution that uses mutations. Its necessary foundations, somewhat different from the rest of the socio-inspired groups that have been studied, allow it to stand out from the rest and reach these positive solutions in most cases.

If you look closely at the results of the hybrid functions in Table 4.1 and compare those of ASO and those of SLC, you can find a curious pattern that reinforces the idea presented in the previous paragraph. In those functions in which SLC is better, the distance between said solution and that provided by ASO is very small. However, if ASO provides the best result, the SLC solution is much further away. That is the difference that justified that ASO best fulfills the most complex functions are accentuated in the higher dimension, and this reveals that ASO manages diversity better in its search space.

4.2 Discussions

As a conclusion to this analysis, the results may feel like that SLC, which is the one that stands out the most, is an algorithm that could be part of some competition in a CEC. However, the results are still far from optimal for most functions.

To make a comparison, the best results of the each function will be compared with those provided by a elitist algorithm that was presented in 2010 (Maity etc, 2010) and compared with this same benchmark.

As seen in Table 5.3, for each function of the 2005 benchmark, according to Table 5.2 the best algorithms's and EPVE algorithm's mean values are compared for dimension 30. In the table, results colored by the algorithms and also underlined, darkened and framed to indicate the best ones. A total of 25 function, there are only 8 in which socio-inspired algorithms (SLC and ASO results) turn out to be better. It should be remembered that these algorithms are in development phase and presented a few years ago, and these comparative results were presented in 2010 and are already more than surpassed by new proposals. So, it is due to sign that according to CEC2005 function results not even the best results of the socio-inspired are a worthy rival for algorithms a decade ago.

Table 4.2: Comparison of functions results between SCL and EPVE.

Partial Swarm Optimization		Soccer League Competition		Anarchic Society Optimization	
Function & Algorithm	<i>F1</i>	<i>F2</i>	<i>F3</i>	<i>F4</i>	<i>F5</i>
EPVE	<u>6,00 x 10⁻⁸</u>	2,90 x 10 ²	<u>1,47 x 10⁵</u>	1,94 x 10 ²	<u>4,56 x 10²</u>
Socio-Inspired Algorithm	1,04 x 10 ⁰	<u>1,18 x 10²</u>	5,64 x 10 ⁵	<u>4,68 x 10⁰</u>	★ 4,16 x 10 ³
Function & Algorithm	<i>F6</i>	<i>F7</i>	<i>F8</i>	<i>F9</i>	<i>F10</i>
EPVE	<u>3,49 x 10²</u>	<u>4,75 x 10⁻³</u>	<u>2,04 x 10¹</u>	3,08 x 10 ¹	<u>7,09 x 10¹</u>
Socio-Inspired Algorithm	7,57 x 10 ²	3,93 x 10 ⁻¹	2,06 x 10 ¹	<u>2,94 x 10¹</u>	9,00 x 10 ¹
Function & Algorithm	<i>F11</i>	<i>F12</i>	<i>F13</i>	<i>F14</i>	<i>F15</i>
EPVE	<u>1,02 x 10¹</u>	1,08 x 10 ⁵	<u>3,19 x 10⁰</u>	1,23 x 10 ¹	<u>2,55 x 10²</u>
Socio-Inspired Algorithm	2.41 x 10 ¹	<u>9,49 x 10³</u>	★ 3,43 x 10 ⁰	<u>1,21 x 10¹</u>	2,95 x 10 ²
Function & Algorithm	<i>F16</i>	<i>F17</i>	<i>F18</i>	<i>F19</i>	<i>F20</i>
EPVE	9,31 x 10 ¹	1,06 x 10 ²	<u>3,00 x 10²</u>	<u>8,00 x 10²</u>	<u>8,03 x 10²</u>
Socio-Inspired Algorithm	<u>8,46 x 10¹</u>	<u>1,83 x 10¹</u>	★ 9,11 x 10 ²	9,17 x 10 ²	9,09 x 10 ²
Function & Algorithm	<i>F21</i>	<i>F22</i>	<i>F23</i>	<i>F24</i>	<i>F25</i>
EPVE	<u>5,00 x 10²</u>	9,26 x 10 ²	<u>5,40 x 10²</u>	<u>2,00 x 10²</u>	<u>1,17 x 10²</u>
Socio-Inspired Algorithm	6,01 x 10 ²	<u>9,11 x 10²</u>	9,10 x 10 ²	2,02 x 10 ²	3,57 x 10 ²

As can be seen in Table 4.2, F4, F12 and F17 are marked by a star. It shows that the social-based algorithms which has a best result in these functions beat EPVE by a large margin. The following comments can be made here;

For F4 and F17; unlike SLC, EPVE doesn't obtains very good results for functions with noise.

For F12; this function is obtained by moving the Schwefel's Problem, and has multiple local optimal point and one global optimal point. And in this problem the global minimum is hard to find due to the distribution of its local minimum. So can be said that, SLC is better than EPVE to find global minima for this function.

5. Conclusion

Throughout this study, a series of proposals of socio-inspired algorithms have been studied, in order to shed some light on a field of an algorithm that does not yet have enough results. These algorithms have a very obvious interest in the relationship of evolutionary algorithms. Nevertheless, what has not yet been proven is whether this type of algorithm is prepared to face a benchmark of competition and beat other better-known proposals. The study carried out has covered all the phases of a complete experiment, from the review of the literature to the experimentation and the resulting analysis, through a previous theoretical analysis that allowed to understand and assess each one correctly.

In the experimental analysis section, it has been concluded that algorithms based on a group of PSOs, such as ICA and POA, are less robust and tend to give worse results than those that modify even

more the standard behavior of a PSO. In this group, you can enter the algorithm IA, which explores more positions in each iteration and therefore has more room for maneuver to reach a better solution. In the last step of the socio-inspired based on a PSO structure, we would find the algorithms SEA and ASO, two techniques that use historical memory in the problem to make displacement decisions based on it. Thanks to this, the best position does not necessarily have to be always present in the population, which results in a more varied movement. And there is a greater exploration in the search space that avoids stagnation soon in local optimal. Finally, the socio-inspired algorithm that has given the best result is not a variation of a PSO but slightly reminiscent of a genetic algorithm, or a Differential Evolution, and it is the SLC.

However, the fact that the SLC has swept the study does not imply that it is an algorithm that reflects good results. Although all algorithms, to a greater or lesser extent, have ended up beating the PSO, the algorithm that has been used as a reference. This is not a real challenge since it is not a polished version. When facing the results of the SLC against a polished algorithm a decade ago, the results have been well exposed: not even the best of the socio-inspired algorithms is good enough to be worthy of participating in a competition. At least, until a new proposal denies it. And as noted throughout the study, these algorithms are not even difficult to understand, even without having a conceptual basis on the subject. Valuing this, and from teaching, this study encourages them to be used to help understand the most complex algorithms that can be the first puzzle. As possible extensions to this work, it will be possible to create a hybrid the socio-inspired algorithms with EPVE algorithm to reach optimal solutions. Later, depending on the results, a new study can be done to determine what characteristics of these hybrid algorithm that best reflect can be applied to the rest of the algorithms, and to assess its effectiveness.

References

- Altay, E. and Alataş, B. (2018). *Performance Comparisons of Socially Inspired Metaheuristic Algorithms on Unconstrained Global Optimization*. Springer.
- Altunbey, F. And Alataş, B. (2015). *Çok Amaçlı Sosyal Tabanlı Metasezgisel Optimizasyon Algoritmaları ile Sosyal Ağlarda Örtüşen Topluluk Keşfi*. PhD thesis, Fırat University, Turkey.
- Borji, A. and Hamidi, M. (2008). *A New Approach To Global Optimization Motivated By Parliamentary Political Competitions*. ICIC International ISSN 1349-4198.
- Bozorgi, A., Bozorg-Haddad, O. and Chu, X. (2018). *Anarchic Society Optimization (ASO) Algorithm*. Springer.
- Huan, T.T, Kulkarni, A.J, Kanesan, J. and Huang, C.J. (2016). *Ideology algorithm: a socio-inspired optimization methodology*. The Natural Computing Applications Forum.
- Hudaib, A.A. and Hwaitat. (2017). *Movement Particle Swarm Optimization Algorithm*. Canadian Center of Science and Education, 1;2018, ISSN 1913-1852.
- Javid, A. A. (2011). *Anarchic Society Optimization: A Human-Inspired Method*. In 2011 IEEE Congress of Evolutionary Computation, CEC2011, pages 2586;2592.
- Kızılloluk, S. and Alataş, B. (2012). *Sosyal Tabanlı Güncel Sezgisel Optimizasyon Algoritmaları*. C.Ü. İktisadi ve İdari Bilimler Dergisi.
- Kumar, M., Kulkarni, A. and Satapathy, S. (2018). *Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology*. Future Generation Computer Systems, 81:252-272.
- Maity, S., Gunjan, K. and Das, S. (2010). *An Improved Evolutionary Programming with Voting and Elitist Dispersal Scheme*. Springer.

- Moosavian, N. and Moosavian, H. (2017). *Testing Soccer League Competition Algorithm in Comparison with Ten Popular Meta-heuristic Algorithms for Sizing Optimization of Truss Structures*. International Journal of Engineering, ISSN 926-936.
- Special Session on Real-Parameter Optimization at CEC-05, Edinburgh.
http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-05/CEC05.htm.
- Soccer League Competition (SLC) Algorithm For Discrete Problems, File Exchange, MATLAB Central. <https://www.mathworks.com/matlabcentral/fileexchange/56480-soccer-league-competition-slc-algorithm-for-discrete-problems>.
- Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A. and Tiwari, S. (2005). *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*. Technical Report, Nanyang Technological University, Singapore.
- Towsyfyan H., Adnani-Salehi, S.A., Ghayyem, M. And Mosaedi, F. (2013). *The Comparison of Imperialist Competitive Algorithm Applied and Genetic Algorithm for Machining Allocation of Clutch Assembly*. International Journal of Engineering, ISSN 1485-1494.
- Xu, Y., Cui, Z., and Zeng, J. (2010). *Social Emotional Optimization Algorithm for Nonlinear Constrained Optimization Problems*. Springer.