2 Advanced metaheuristic techniques

Many difficulties such as multi-modality, dimensionality and differentiability are associated with the optimization of large scale problems. Traditional techniques like steepest decent, linear programming, dynamic programming, etc. generally fail to solve such large scale problems especially with non-linear objective functions. Most of the traditional techniques require gradient information and hence it is not possible to solve non-differentiable functions with the help of such traditional techniques. Moreover, such techniques often fail to solve optimization problems that have many local optima. To overcome these problems, there is a need to develop more powerful optimization techniques and research is going on to find effective optimization techniques since last three decades.

Some of the well known population based optimization techniques developed during last three decades are: Genetic Algorithms (GA) (Holland, 1975) which works on the principle of the Darwinian theory of the survival of the fittest and the theory of evolution of the living beings; Artificial Immune Algorithms (AIA) (Farmer et al., 1986) which works on the principle of immune system of the human being; Ant Colony Optimization (ACO) (Dorigo, 1992) which works on the principle of foraging behavior of the ant for the food; Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) which works on the principle of foraging behavior of the swarm of birds; Differential Evolution (DE) (Storn and Price, 1997) which is similar to GA with specialized crossover and selection method; Artificial Bee Colony (ABC) (Karaboga, 2005) which works on the principle of foraging behavior of a honey bee; Biogeography-Based Optimization (BBO) (Simon, 2008) which works on the principle of immigration and emigration of the species from one place to the other; Gravitational Search Algorithm (GSA) (Rashedi et al., 2009) which works on the principle of gravitational force acting between the bodies; Firefly Algorithm (FA) (Yang, 2008) is a metaheuristic algorithm inspired by flashing behavior of firefly insects; Cuckoo search optimization algorithm (Yang and Dev, 2009) are works on the principal based on reproduction strategy of cuckoos; Krill herd (KH) (Gandomi and Alavi, 2012) is based on the simulation of the herding behavior of krill individuals. Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014) mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Symbiotic Organisms Search (SOS) (Cheng and Prayogo, 2014) simulates the symbiotic interaction strategies adopted by organisms
to survive and propagate in the ecosystem; Water wave optimization (WWO) (Zheng, 2015) inspired from shallow water wave models. These algorithms have been applied to many engineering problems and proved effective to solve some these problems. Next section describes advanced metaheuristic techniques implemented in the present work.

2.1 Heat transfer search algorithm (HTS)

Heat transfer search (HTS) is an optimization algorithm inspired from the law of thermodynamics and heat transfer (Patel and Savsani, 2015). The fundamental law of thermodynamics states that any system always tries to achieve thermal equilibrium with its surroundings. Therefore, any system lagging thermal equilibrium always tries to attain thermal equilibrium by conducting heat transfer with surrounding as well as within the different parts of the system. The modes of heat transfer which play a major role in setting thermal equilibrium are conduction, convection, and radiation. Therefore, the HTS algorithm composes with the ‘conduction phase’, ‘convection phase’, and ‘radiation phase’ to reaches at the optimum solution.

HTS is a nature-inspired, population-based algorithm. The algorithm initiates with a set of solutions to achieve the global optimum value. In HTS, a population is akin to molecules of the system, temperature levels of the molecules represent the value of design variables and energy level of the system represents the fitness value of objective function. The best solution treated as the surrounding and rest of the solutions are part of the system. Each phase of the HTS algorithm executes with equal probability during optimization. The equal probability is controlled by a parameter ‘$R$’ in each generation, which is a uniformly distributed random number, varies between 0 and 1.

Working of each phase of HTS algorithm is explained below for minimization problem. Here, the size of a population is represented by ‘$n$’. The number of design variables is denoted by ‘$m$’ and ‘$g$’ represent the generation number.

2.1.1 Conduction phase

Heat transfer between the molecules of the substance is replicated in the conduction phase. In conduction heat transfer, molecules possessing high grade energy transmit heat to adjacent molecules with lower energy level. In the course of optimization with HTS algorithm, higher and lower energy level molecule akin to a population having better and inferior objective function value. During conduction phase, solutions are updated according to the following equations.
\[
\begin{align*}
X_{ji}^{new} &= X_{ji}^{old} - R^2 X_{ji}^{old} & \text{If } f(X_j) > f(X_k) \\
X_{ki}^{new} &= X_{ki}^{old} - R^2 X_{ji}^{old} & \text{If } f(X_k) > f(X_j) ; \quad \text{If } g \leq g_{max}/CDF
\end{align*}
\]

(2.1)

\[
\begin{align*}
X_{ji}^{new} &= X_{ji}^{old} - r_i X_{ki}^{old} & \text{If } f(X_j) > f(X_k) \\
X_{ki}^{new} &= X_{ki}^{old} - r_i X_{ji}^{old} & \text{If } f(X_k) > f(X_j) ; \quad \text{If } g > g_{max}/CDF
\end{align*}
\]

(2.2)

Where, \( R \in [0, 0.3333] \) is the probability for selection of conduction phase. \( j=1,2,\ldots,n, j \neq k, k \in (1,2,\ldots,n) \) and \( k \) is a randomly selected solution from the population. CDF is the conduction factor. \( r_i \in [0, 1] \) is a uniformly distributed random number. Further, \( i \in (1,2,\ldots,m) \) and \( i \) is a randomly selected design variables.

2.1.2 Convection phase

Heat transfer between system and surrounding is replicated in the convection phase. The mean temperature of the system interacts with the surrounding temperature to establish thermal equilibrium with the surrounding. In the course of optimization with HTS algorithm, best solution play a role of surrounding while remaining solution composes the system. Thus, the mean value of any design variable of the system interacts with the design variable of the best solution. In this phase, solutions are updated according to the following equations.

\[
X_{ji}^{new} = X_{ji}^{old} + R \ast (X_s - X_{ms} \ast TCF)
\]

(2.3)

\[
\begin{align*}
TCF &= \text{abs}(R - r_i) & \text{If } g \leq g_{max}/COF \\
TCF &= \text{round}(1 + r_i) & \text{If } g > g_{max}/COF
\end{align*}
\]

(2.4)

Where, \( R \in [0.3333, 0.6666] \) is the probability for selection of convection phase. \( r_i \in [0, 1] \) is a uniformly distributed random number. COF is the convection factor. \( j=1,2,\ldots,n, i=1,2,\ldots,m \). \( X_s \) is the temperature of the surrounding and \( X_{ms} \) is mean temperature of the system.

2.1.3 Radiation phase

The radiation heat transfer between system and surrounding is simulated in this phase. In this phase best solution (i.e. surrounding) is interact with the any other randomly selected solution (i.e. system) to establish a thermal balance. In this phase, solutions are updated as below.

\[
\begin{align*}
X_{ji}^{new} &= X_{ji}^{old} + R \ast (X_{ki}^{old} - X_{ji}^{old}) & \text{If } f(X_j) > f(X_k) \\
X_{ji}^{new} &= X_{ji}^{old} + R \ast (X_{ji}^{old} - X_{ki}^{old}) & \text{If } f(X_k) > f(X_j) ; \quad \text{If } g \leq g_{max}/RDF
\end{align*}
\]

(2.5)
Fig. 2.1 Flow chart of heat transfer search (HTS) algorithm
\[
\begin{align*}
X_{ji}^{\text{new}} &= X_{ji}^{\text{old}} + r_i \cdot (X_{ki}^{\text{old}} - X_{ji}^{\text{old}}) & \text{If } f(X_j) > f(X_k) \\
X_{ji}^{\text{new}} &= X_{ji}^{\text{old}} + r_i \cdot (X_{ji}^{\text{old}} - X_{ki}^{\text{old}}) & \text{If } f(X_k) > f(X_j); \text{If } g > g_{\text{max}} / \text{RDF}
\end{align*}
\] (2.6)

Where, \( R \in [0.6666, 1] \) is the probability for selection of radiation phase. \( r_i \in [0, 1] \) is a uniformly distributed random number. RDF is the radiation factor. \( j=1,2,...,n, j \neq k, k \in (1,2,...,n) \) and \( k \) is a randomly selected solution from the population; \( i \in (1,2,...,m) \). The flow chart of HTS algorithm is shown in Fig. 2.1.

2.2 Tutorial training and self learning inspired teaching-learning-based optimization (TS-TLBO) algorithm

The tutorial training and self learning inspired teaching-learning-based optimization (TS-TLBO) algorithm is modified version of basic teaching-learning-based optimization (TLBO) algorithm. Teaching-learning method is the heart of any culture. Rao et al. (2012) introduced an approach for the optimization called teaching-learning-based optimization algorithm based on the idea of teaching and learning. The algorithm works based on the two fundamental modes: (i) learning through teacher and (ii) learning by interacting with the other learners. TLBO is a population based algorithm where population is composed from the group of students (i.e learner). Further, the different subjects presented to the learners are treated as design variables of the optimization problem. The outcome of a learner in each subject represents a possible solution to the optimization problem (design variables value). The best solution in the entire population is considered as the teacher. The working of TLBO algorithm is explained through two different phase as below.

2.2.1 Teacher Phase

As indicated by the name, in this phase of the algorithm, learners learn through a teacher. The mean result of class depends on the quality of learners. A good teacher tries to uplift the learner’s level up to his or her level as far as knowledge is concern. However, strengthening the knowledge of the learners not depends only on the knowledge of the teacher but also depends on the ability of the learners to understand and grasp the knowledge imparted by teacher. So, this teaching-learning is bilateral process depends on many factors.

To understand the things more precisely, assume that ‘\( n \)’ number of learners compose the class (i.e. population), these learners undergoes ‘\( m \)’ subjects number (i.e.
design variables) in the class, best learner (best solution) of the class treated as teacher ‘T’ and mean results of the learner (i.e. population) is ‘M’. Through teaching-learning process T will try to move M toward his/her level. Further, mean results of the population is changed depending on the parameter called teaching factor ‘TF’ based on the random number varied in the range of 0-1. So, based on the difference between existing and new mean results of the learners’ solutions is modified as below during the teacher phase as below.

\[ A_{ij}^{\text{new}} = A_{ij}^{\text{old}} + r_{ij} \left( T_j - T_F M_j \right) \]  
\[ (2.7) \]

Where, \( i=1,2,...,n, j=1,2,...,m \). \( T_F \) is set to either 1 or 2 and decided randomly with equal probability given as \( T_F = \text{round}[1 + \text{rand}(0, 1)(2-1)] \).

2.2.2 Learner Phase

This phase works based on the knowledge sharing ability between the different learners. Through the mutual discussion with other person (i.e. learner) one can increase his/her knowledge. The mode of discussion between different learners may be presentations, group discussion, one-to-one communication etc. In that way any learners improved their knowledge to a better level than existing. Further, the selection of learner follow the random process based on random number. Let, two random learners (i.e. solutions) \( i \) & \( k \) are selected to update solution in this phase as given below, If \( f(A_i) < f(A_k) \)

\[ A_{ij}^{\text{new}} = A_{ij}^{\text{old}} + r_{ij} \left( A_{kj}^{\text{old}} - A_{kj}^{\text{old}} \right) \text{ where } i \neq k \]  
\[ (2.8) \]

Else

\[ A_{ij}^{\text{new}} = A_{ij}^{\text{old}} + r_{ij} \left( A_{kj}^{\text{old}} - A_{ij}^{\text{old}} \right) \text{ where } i \neq k \]  
\[ (2.9) \]

2.2.3 Tutorial training and self learning

To enhance the exploration and exploitation ability, two additional search mechanism (tutorial training and self learning) are incorporated in the basic TLBO algorithm. The modified version of TLBO algorithm called as ‘tutorial training and self learning inspired teaching-learning-based optimization (TS-TLBO) algorithm’. The tutorial training modification accompany the teacher phase of the of the TLBO algorithm while self learning modification accompany the learner phase of the TLBO algorithm. The idea behind tutorial training modification is that besides the regular teaching, students can also learn through teachers during tutorial/laboratory hours based on
assignment solution. Thus, addition learning can advance the grade of the learner. The improvement in the knowledge of the learner through tutorial training take place with the help of teacher hence it is accompany teacher phase of the TLBO algorithm. Similarly, sometime students’ uplift their knowledge and improve their grade through reading of different books or assignment solution. Further, the improvement in the results due to self learning solely depend on the learner himself/herself hence this modification accompany the learner phase of the TLBO algorithm.

The search mechanism of the teacher phase of the TS-TLBO algorithm is represented as below,

\[ A_{i,j}^{\text{new}} = A_{i,j}^{\text{old}} + r_{ij} \left( T_j - T_i M_j \right) + r_{ij} \left( A_{i,j}^{\text{old}} - A_{k,j}^{\text{old}} \right) \quad \text{If } f(A_i) < f(A_k), \text{where } i \neq k \]  

\[ A_{i,j}^{\text{new}} = A_{i,j}^{\text{old}} + r_{ij} \left( T_j - T_i M_j \right) + r_{ij} \left( A_{k,j}^{\text{old}} - A_{i,j}^{\text{old}} \right) \quad \text{If } f(A_k) < f(A_i), \text{where } i \neq k \]  

Where, the first two terms on the right side of the above equations represent basic TLBO while third term is indicates learning through tutorial training.

Likewise, the search mechanism of the learner phase of the TS-TLBO algorithm is represented as below,

\[ A_{i,j}^{\text{new}} = A_{i,j}^{\text{old}} + r_{ij} \left( A_{i,j}^{\text{old}} - A_{k,j}^{\text{old}} \right) + r_{ij} \left( T_j - E_F A_{i,j}^{\text{old}} \right) \quad \text{where } i \neq k \]  

\[ A_{i,j}^{\text{new}} = A_{i,j}^{\text{old}} + r_{ij} \left( A_{k,j}^{\text{old}} - A_{i,j}^{\text{old}} \right) + r_{ij} \left( T_j - E_F A_{i,j}^{\text{old}} \right) \quad \text{where } i \neq k \]  

Where, \( E_F \) is the exploration factor and its value is decided randomly as: \( E_F = \text{Round} \ (1+r_i) \)

### 2.3 Multi-objective heat transfer search (MOHTS) algorithm

Multi-objective heat transfer search (MOHTS) algorithm is a multi-objective version of the heat transfer search algorithm. The MOHTS algorithm generates simultaneous solutions for each objective, identifies the non-dominated solution and stores those solutions in an external archive. While HTS algorithm is a single objective version where all these things are not required as it handle only single objective. Multi-objective heat transfer search algorithm uses an external archive to store non-dominated solutions for generation of Pareto front. A solution is called non-dominated if none of the objective functions can be improved in value without degrading some of the other objective values. For example if Solution A is better than Solution B in one objective, and
worse in another, then one can say that the two solutions are non-dominated with respect to each other.

Table 2.1: Pseudo-code of MOHTS algorithm

| Set Population size, function evaluation |
| Define objective functions, Minimize/Maximize $f(X) = f_1(X), f_2(X), f_3(X), f_4(X), ..., X=[x_1, x_2, ..., x_k]$ |
| Initialize population, external archives |
| While (Function evaluation < Maximum function evaluations) |
|  | Randomly generate the probability ‘R’ to execute conduction, convection or radiation phase |
|  | Update the solution by any one of the phase depending upon the probability ‘R’ |
|  | Apply the greedy selection between updated solutions and previous solutions |
|  | Initialize grid on the archive |
|  | For each box in the grid |
|  |  | If any box dominate the other boxes |
|  |  | Remove the dominated box and their related solutions |
|  |  | End If |
|  |  | If the box contain more than one solutions |
|  |  | Remove the dominated solution(s) from the box |
|  |  | End If |
|  |  | If the box still contain more than one solution |
|  |  | Keep the solution with less distance from the lower right corner of the box (for minimization problem) or upper right corner of the box (for maximization problem) and remove others |
|  |  | End If |
|  | End for |
| End While |
| Output external archive as Pareto optimal set |

The MOHTS algorithm uses $\varepsilon$-dominance based updating method (Deb et al., 2005) to check the domination of solutions in archive. Pareto front is generated based on the solution kept in external archive. The archiving process of MOHTS algorithm uses grid based approach with fixed size archive. Archive stored the best solutions found
during the update of the solutions. In every generation, archive is update through \( \varepsilon \)-dominance method. \( \varepsilon \)-dominance method presume a space having dimensions equal to the number of objectives of the problem. This space further sliced in \( \varepsilon \) by \( \varepsilon \) sizes which convert the space into the boxes of different shape (e.g square, cube, hyper cubes etc.) for two, three and more objectives respectively. The solutions are hold in these boxes. After that, the boxes and solutions holding in those boxes are removed if they are dominated by other boxes. Then, ensure that remaining boxes contain only one solution. Any boxes contain more than one solution undergoes the process to remove dominated solutions from each box. Thus, only non-dominated solutions are kept in each box and in that way in the archive. The Pseudo-code of MOHTS is shown in Fig.2.2.

2.4 Multi-objective TS-TLBO algorithm

Multi-objective version of the TS-TLBO algorithm is a Pareto based algorithm for two or more than two objectives. The TS-TLBO algorithm generates simultaneous solutions for each objective, identifies the non-dominated solution and stores those solutions in an external archive. While TLBO algorithm is a single objective version. The TS-TLBO algorithm uses \( \varepsilon \)-dominance based updating method (Deb et al., 2005) to check the domination of solutions in archive. Pareto front is generated based on the solution kept in external archive. The archiving process of TS-TLBO algorithm uses grid based approach with fixed size archive. Archive stored the best solutions found during the update of the solutions. In every generation, archive is update through \( \varepsilon \)-dominance method. \( \varepsilon \)-dominance method presume a space having dimensions equal to the number of objectives of the problem. This space further sliced in \( \varepsilon \) by \( \varepsilon \) sizes which convert the space into the boxes of different shape (e.g square, cube, hyper cubes etc.) for two, three and more objectives respectively. The solutions are hold in these boxes. After that, the boxes and solutions holding in those boxes are removed if they are dominated by other boxes. Then, ensure that remaining boxes contain only one solution. Any boxes contain more than one solution undergoes the process to remove dominated solutions from each box. Thus, only non-dominated solutions are kept in each box and in that way in the archive. Implementing steps of the multi-objective TS-TLBO algorithm is given below.

*Step 1:* Initialize algorithm parameters

*Step 2:* Evaluate the initial population.

*Step 3:* Calculate mean of learners in each subject
Step 4: Calculate the difference between the current mean and the corresponding result of the teacher with the help of teaching factor.

Step 5: Update the solutions (Equations (2.10) and (2.11)) according to the teacher phase of the TS-TLBO algorithm.

Step 6: Preserve the modified solutions if it produces better function value.

Step 7: Update the solutions (Equations (2.12) and (2.13)) according to the learner phase of the TS-TLBO algorithm.

Step 8: Preserve the modified solutions if it produces better function value.

Step 9: Initialize the Archive and generate grid on it.

Step 10: Check the solutions in the grid for dominance.

Step 11: Remove the dominated solutions from the grid.

Step 12: Repeat the procedure from step 2 to 11 till the termination criterion is met.

Thus, at the termination of the algorithm, the solution present in the external archive returned as the output and produced the Pareto front.