Contents

1. **Abstract……...1**
2. **Introduction and Literature Review……..……..……....1**
   - 2.1. **Atomistic simulation ...……..1**
   - 2.2. **Genetic Algorithm……..2**
   - 2.3. **Predator-Prey Algorithm……..3**
   - 2.4. **Pareto Rankings……4**
   - 2.5. **Evolutionary Neural Nets (EvoNN).....5**
   - 2.6. **Tree Based Genetic Programming……...6**
   - 2.7. **Bi-Objective Genetic Programming (BioGP).....7**
3. **Objectives……8**
4. **Experimental / Modelling work performed……8**
   - 4.1. **Atomistic simulations……8**
   - 4.2. **EvoNN Algorithm……..9**
   - 4.3. **BioGP Algorithm……10**
   - 4.4. **Single Variable response……11**
5. **Results and Discussions……...11**
6. **Conclusion……..…….....20**
7. **References……..…….……..21**

1. Abstract

Fe-Cr alloys have been exceptional for scientific and industrial reasons. They show a wide spectrum of interesting properties and consequently, they have been regarded as good model alloys for testing various models and theories. The industrial importance of the alloys stems from their low ductile-to-brittle transition temperatures, strong resistance to swelling and bubble promotion as well as to high-temperature corrosion. Fe-Cr alloys have wide application in the steelmaking industry. Fe-Cr-based steels are suitable for the design of various structural components in advanced nuclear energy installations. They are also extensively used for various accelerating various surface reactions due to their excellent catalytic properties.

Two genetic algorithm based optimization algorithms, EvoNN and BioGP were utilised in the training process, using surface energy and cohesive energy calculated from the study of co-working group on the atomistic simulations of Fe-Cr alloying system nanoparticles with varying radius and Cr concentration. A predator–prey algorithm efficiently performed the optimization task and several important trends were observed. We used Genetic Algorithm to optimize the Surface energy and Cohesive energy of the Fe-Cr alloy for optimal performance of Fe-Cr alloy in catalytic reactions. As these are conflicting objectives, instead of getting one optimal solution, we obtain a set of pareto-optimal solutions by minimizing the training error along with the network and tree size (complexity).

2. Introduction and Literature Review

2.1. Atomistic simulation

Atomistic simulation models at the nanoscale provide information about the crystal and particles about the physical movements and interaction behaviour. Atomistic simulation can be categorised into Molecular Statics and Molecular Dynamics[1], In the case of Molecular Statics (MS), the relaxed configuration of atoms is found using conjugate gradient or some similar (constrained) minimization of the total energy. This provides information about crystal lattice structure in different phases and under different conditions. In the case of Molecular Dynamics (MD), the actual motion of the atoms is simulated by evolving the atomic configuration in time according to Newton's equation (F=ma)[2]. This allows the direct study of the dynamical and thermodynamical evolution of the system. We used Atomistic simulations to generate the binary alloy of Fe-Cr with varying Cr concentration from 5% to 50% with steps of 0.5%. Calculations of corresponding lattice parameters was done. Further, particles of varying radius were scooped out from the centre of simulated crystal lattices and these particles were simulated for calculation of surface energy and cohesive energy of nanoparticle.

2.2. Genetic Algorithm

The Genetic Algorithm (GA) is a heuristic search algorithm that mimics the process of natural evolution. This algorithm is inspired by the Darwinian Theory of evolution. Genetic algorithms belong to the bigger class of evolutionary algorithms (EA)[3], that generate solutions to optimization problems with techniques on the premises of natural evolution processes, such as inheritance, mutation, selection, and crossover. A population representing possible solutions is bred in the process and depending upon their fitness values while the selection, crossover and mutation operations are performed on the entire pool of solutions and at the end of the GA run, the individual with best fitness according to some standard rankings is picked up as an optimized solution. The combinations of various potential solutions forms a population[4]. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. The population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. Selection of the individuals to undergo crossover and mutation operation is generally done by Roulette wheel selection and tournament selection[5].

Outline of Basic Genetic Algorithm :

1. **[Start]** Generate random population of n chromosomes (suitable solutions for the problem)

2. **[Fitness]** Evaluate the fitness f(x) of each chromosome x in the population

3. **[New population]** Create a new population by repeating following steps until the new population is complete

    3.1 **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

    3.2 **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

    3.3 **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).

    3.4 **[Accepting]** Place new offspring in a new population

4. **[Replace]** Use new generated population for a further run of algorithm

5. **[Test]** If the end condition (for example number of populations or improvement of the best solution) is satisfied, stop and return the best solution in current population

6. **[Loop]** Go to step 2

2.3. Predator-Prey Algorithm

Predator Prey Algorithm a multi-objective evolution strategy (ES) which is radically different from any other evolutionary algorithms. Preys represent a set of solutions which are placed on the vertices of a undirected connected graph. First, each predator is randomly placed on any vertex of graph. Each predator is associated with a particular objective function. This predator catches a prey with worst value of its associated objective. Once a prey is caught it is removed from the vertex and a new solution is obtained. After this the predator takes a random walk to any of its neighbouring vertices. The above procedure continues in parallel for all predators until a pre-specified number of iterations have elapsed[6]. The simultaneous presence of predators favouring each objective allows trade-off solutions to co-exist in the graph. Instead of associating one objective to each predator, we associate different weight vector with each predator to maintain diversity in solution and to obtain a nicely separated solutions in the Pareto front.

The preys in this algorithm are initiated as a family of randomly generated computer programs from a user defined function set and a terminal set, which constituted an initial population in the usual genetic algorithms sense. The exact rules of PPGA are:

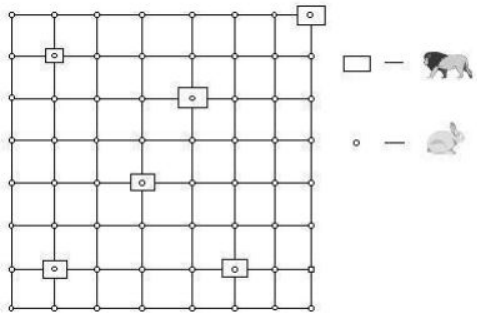


Figure 1: Placement of predators and preys on a toroidal grid (original model).

Initialization:

1. Define parameters such as the lattice size, number of prey, number of predators, number of generations, and probabilities of crossover and mutation.
2. Generate random individuals (models or solutions) and place them randomly in the lattice.
3. Generate predators with a certain fitness criteria based upon objectives, and place the predators randomly.

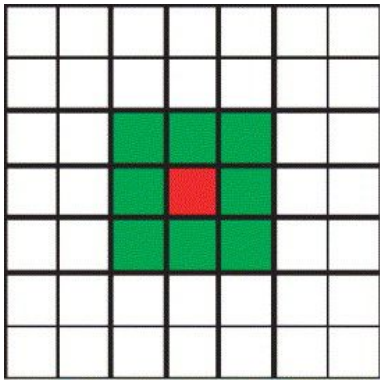


Figure 2: Moore neighbourhood

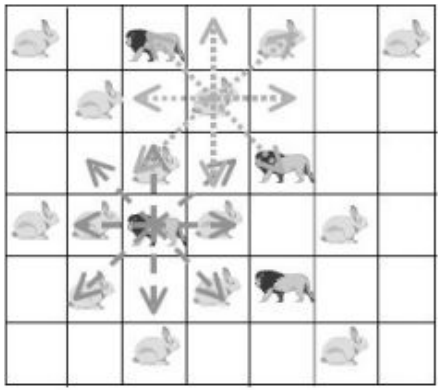


Figure 3: Predator-prey model proposed by Li[6]

After the above initialization, the predator-prey model proceeds in the following steps:

1. Each prey is allowed to move in a random direction, i.e., one of the eight cells in an 8-cell Moore neighbourhood (north, south, east, west, and plus the four diagonal neighbours), to move into. They then attempt to move. If the cells they are attempting to move into are occupied by other prey or predators, then they try again. If the prey is still unable to find a place to move, it remains where it is.

2. After the prey have moved they are then allowed to breed. Space plays a critical role in this model as each prey can only breed with another prey within its neighbourhood (excluding itself). If the prey has no neighbours it is not allowed to breed. Otherwise the prey is allowed to breed with another randomly selected neighbour to produce an offspring using real crossover and mutation operators.

3. The prey population is under constant threat from the predators, pressure is exerted upon the prey population through the predator-prey interaction, that is, predators are given the task of killing the leastfit prey in their vicinity.

4. Go back to step (1), if the number of generations required is not reached.

In order to prevent predators from completely wiping out the entire prey population, the following formula is adopted to keep the prey population at an acceptable level:

$$iterations = \left\lfloor \frac{numPreys_{Actual} - numPreys_{Preferred}}{num_{Predators}} \right\rfloor$$

where iterations is the number of moves the predators may take before the prey can make their moves.

2.4. Pareto Rankings

Ranking of the population individuals is an important element in Multiobjective Evolutionary Algorithms in order to establish later the probabilities of survival that are necessary for the selection process[7]. This process determines which elements of the population are selected to be members of the next generation. It is done according to how fit the individual is. The better it is, the higher probability of survival it has, and so, it has a higher probability of being selected for the next generation. In the scalar case, it is obvious: the better objective value it has, the better a solution is. In the multiobjective case, there is not only one criterion to conclude whether one solution is better than another, because no total order exists in the set of solutions. The definition

of a dominating solution is : One solution a ∈ A is said to be Pareto optimal or efficient with regard to a set B ⊂A if and only if there does not exist any à ∈B such that à dominates a.
The three common methods used for ranking solution in multi objective algorithm are :

1. **Belegundu's ranking:** All non dominated individuals are assigned rank 0 (or 1) and the dominated ones rank 1 (or 2).
2. **Goldberg's ranking:** It assigns equal probability of reproduction to all non dominated individuals in the population. The method consisted of assigning rank 1 to the non dominated individuals and removing them from contention, then finding a new set of non dominated individuals, ranked 2, and so forth.
3. **Fonseca and Fleming's ranking:** An individual's rank corresponds to the number of individuals in the current population by which it is dominated. Non dominated individuals are, therefore, all assigned the same rank equal to zero, while the dominated ones have values between 1 and k − 1, where k is the population size.

2.5. Evolutionary Neural Nets (EvoNN)

Evolutionary Neural Network (shorthand as EvoNN) consists of population of neural network with different set of hyperparameters and best amongst these is chosen by optimisation techniques such as `Ant-Colony optimisation` or `Prey-Predator optimisation`(used for current purposes)[8][9]. Here, Prey population : Artificial Neural Network(as shown below). Each ANN is identified with a unique set of hyperparameters which are

- Different number of nodes in the hidden layer.
- Since it is not fully connected neural net, hence configuration of edges between 2 layers are different.

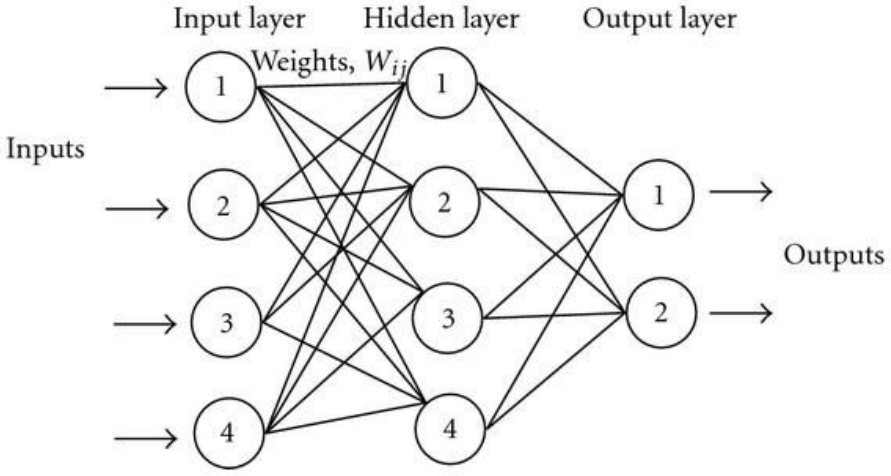


Figure 4: A typical Artificial Neural Network

A population of these neural nets are optimised using genetic algorithms, we use Prey-Predator optimisation algorithm technique to choose the best hyperparameters. The lower level connections are optimised using Predator–Prey Multiobjective Genetic Algorithm (or PPGA) while upper level connections are obtained by simple regression using least squares estimation.
As a result, a Pareto front is created between the two objective functions consisting of the most optimal ANNs, amongst these the best ANN is chosen using the Corrected Akaike Information Criterion(AICc)[10].

2.6. Tree Based Genetic Programming

In GP, programs are expressed as syntax trees rather than as lines of code. The variables and constants in the program are leaves of the tree which are called terminals, the arithmetic operations are internal nodes called functions[11]. These sets of functions and terminals together form the primitive set of a GP system. In more advanced forms of GP, programs can be composed of multiple components. In this case the representation used in GP is a set of trees (one for each component) grouped together under a special root node. These are (sub)trees branches. It is common in the GP literature to represent expressions in a prefix notation similar to that used in scheme. For example, max(x+x,x+3*y) becomes (max (+ x x) (+ x (* 3 y))). Below figure shows the tree representation of the program : max(x+x,x+3*y).

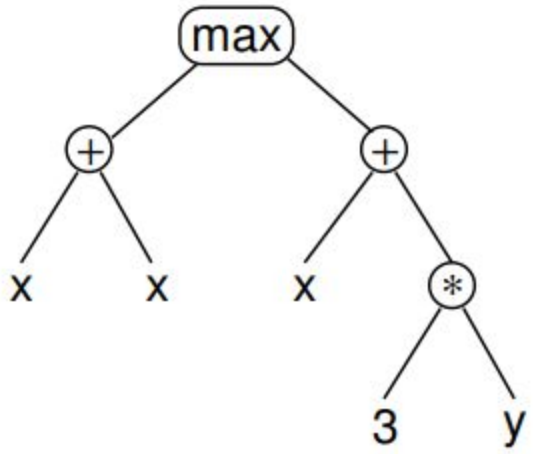


Figure 5: GP syntax tree representing max(x+x,x+3*y)

GP departs significantly from other evolutionary algorithms in the implementation of the operators of crossover and mutation. Crossover happens as subtree crossover and mutation happens as subtree mutation. Given two parents, subtree crossover randomly (and independently) selects a crossover point (a node) in each parent tree[12]. Then, it creates the offspring by replacing the subtree rooted at the crossover point in a copy of the first parent with a copy of the subtree rooted at the crossover point in the second parent.

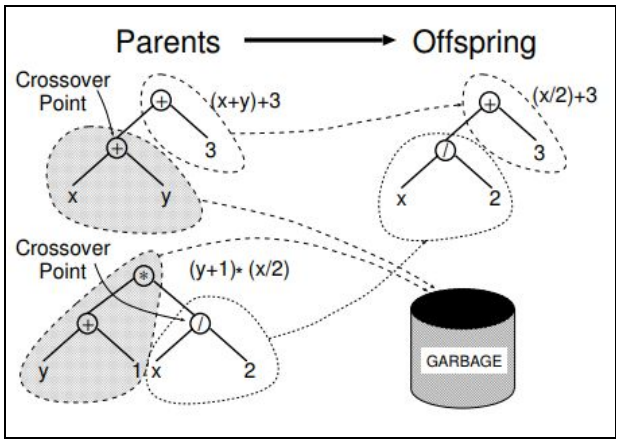


Figure 6: Example of subtree crossover.

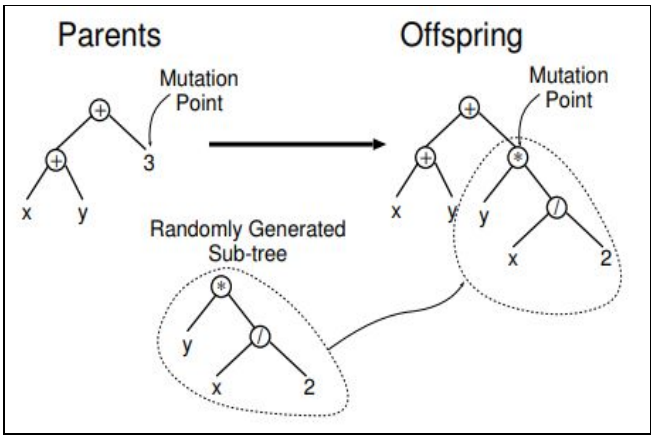


Figure 7: Example of subtree mutation.

The termination criterion may include a maximum number of generations to be run as well as a problem-specific success predicate. The single best individual is harvested and designated as the result of the run.

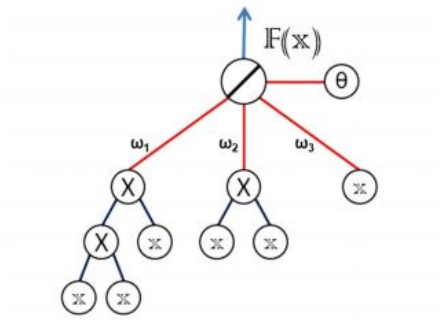
2.7. Bi-Objective Genetic Programming (BioGP)

Conventional Genetic Programming technique which aims to generate a tree with the lowest `Mean Square Error`(MSE) value given by:

$$\mathrm{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i(\boldsymbol{X}) - \Phi_i)^2},$$

GP tends to minimise the MSE values which results in overfitting of the tree. BioGP contours the problem of overfitting. This technique generates a trade-off between bi-objectives i.e `complexity` and `error` i.e minimize$\{\zeta(x_{gp}) \cdot \xi(x_{gp})\}$.

This results in a number of equally optimal solutions together represented as Pareto Front[11]. The maximum number of roots of tree is defined during initialization. The roots passes the output to the successive node which weighted sums all its input, also adds a bias value and outputs the final value as input to its successive node. The process continues until the final outcome is reached. The weights and the bias value are calculated by the linear least square technique[12]. Moreover the complexity of the tree is calculated as the weighted sum of the total number of the function nodes and tree depth[13]. Hence the final optimal solutions are generated as pareto front.



$$F(x) = \omega_1 * x^3 + \omega_2 * x^2 + \omega_3 * x + \theta$$

Figure 8: BioGP Tree

3. Objectives

A. Atomistic Simulations for generation of Fe-Cr alloy nanoparticles, and simulation of lattice parameters, cohesive and surface energies : Done by co-working group
B. Collection of relevant data points from the simulations : Done by co-working group
C. Fonseca Ranking of above data points to obtain optimal points within dataset.
D. Training the model for different input parameters and Fe-Cr alloy parameters
E. Optimization of conflicting variables by using enhanced Evolutionary algorithm: EvoNN and BioGP
F. SVR on data models to study effect of individual variables on objectives
G. Finding tradeoff of radius of particles with cohesive energy and bulk energy in simulated Fe-Cr alloy system and further discarding of non feasible solutions.

4. Experimental / Modelling work performed

4.1. Atomistic Modeling

We used Atomistic simulations to generate the binary alloy of Fe-Cr with varying Cr concentration from 5% to 50% with steps of 0.5%. Calculations of lattice parameters was done, further particles of varying radius were scooped-out from the simulated models and these particles were simulated for calculation of surface energy and cohesive energy. This was achieved by our co-working group and relevant data points were generated and collected for further optimization studies.

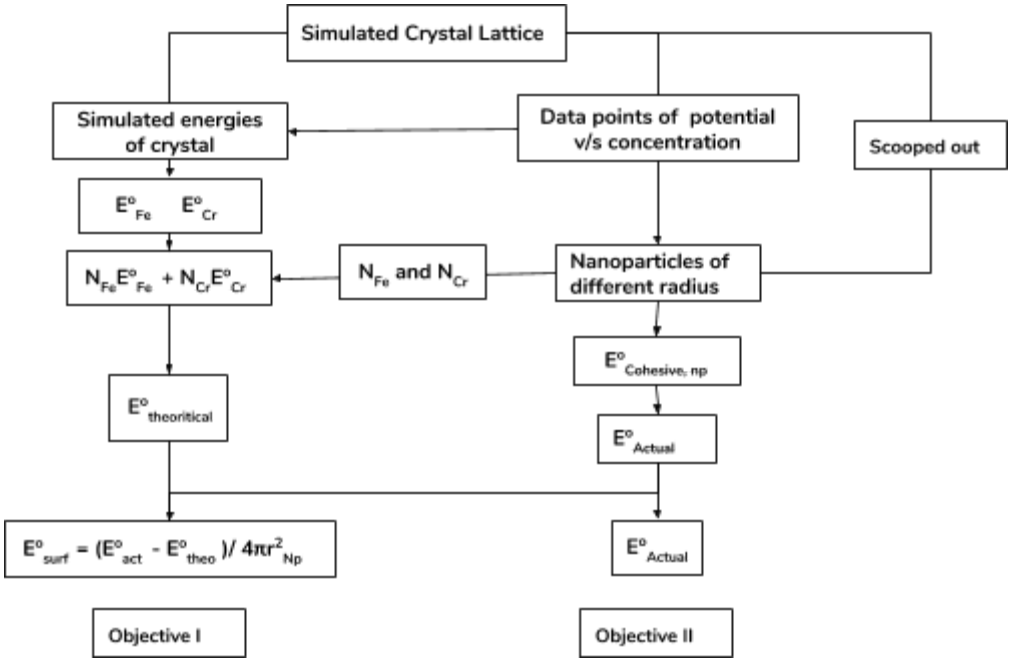


Figure 9: Flowchart of atomic simulations

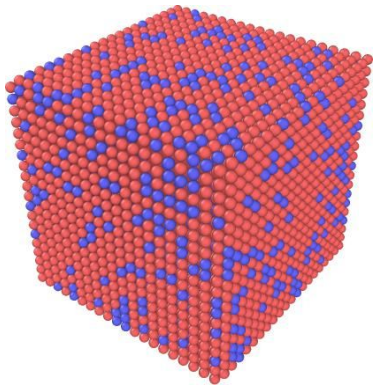


Figure 10: Fe-Cr Crystal, 20% Cr

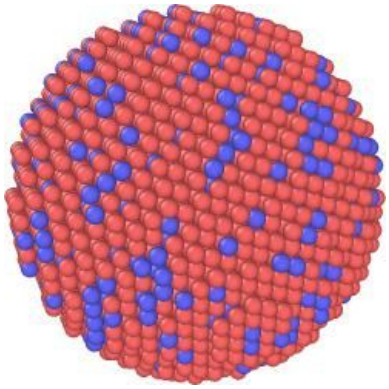


Figure 11: Nanoparticle with 25 Å radius

4.2. BioGP Algorithm[12]

Algorithm 2 Pseudo code of BioGP

begin bi-objective genetic programming
generate random population of each subtree
while (*generation* < *maximum generations*) **do**
perform suitable crossover and mutation for each subtree population
determine *error reduction ratio* for each subtree
if (*error reduction ratio* $\neq$ *viable*) **then**
delete and regrow the subtree
repeat until (*error reduction ratio* = *viable*)
end repeat
assemble viable subtrees using a bias and weights
optimize weights and bias using *LLSQ* procedure
determine *training error* for each tree
while (*generation* $\leq$ single objective runs) **do**
perform *tournament selection* based upon *training error*
create new population
end do
end while
while (*generation* > single objective runs) **do**
determine **complexity**
activate predator–prey GA operators
determine *best tradeoff* between *complexity* and *accuracy*
Pareto frontier = *best tradeoff*
end do
end while
best model = *lowest error model in the Pareto frontier*
end do
end while
end bi-objective genetic programming

4.3. EvoNN Algorithm[12]

Algorithm 1 Pseudo code of EvoNN

begin bi-objective Predator–prey GA for EvoNN training
generate population of random networks
non-dominated front ≠ *Pareto frontier*
repeat until (*non-dominated front* ≠ *Pareto frontier*)
with (network *Neti, Netj* ∈ *population*) **do**
repeat until (*population size* ≠ *new population size*)
% *crossover* in lower parts of *Neti, Netj*
ChildNeti, ChildNetj ← *Neti, Netj*
% mutation of lower part weights in children network
MutChildNeti ← *Neti*
MutChildNetj ← *Netj*
MutChildNeti, MutChildNetj ∈ *new population*
∄ (*Neti, Netj* ∈ *new population*)
pick new random *Neti, Netj* ∈ *new population*
(end repeat)
end do
end with
with (*Neti* ∈ *new population*) **do**
for *all i*
activate *LLSQ* in the upper part
determine *training error*
determine *complexity*
end do
end with
with (*Neti, Netj* ∈ *new population*) **do**
for *all i, j*
check *weak dominance between training error and complexity*
for *new population*
perform predator–prey actions
determine *survivor population*
population ← *survivor population*
determine *non-dominated front*
end do
end with
end repeat
end bi-objective Predator–prey GA for EvoNN training

4.4. Single Variable Response

Generation of variable response on the objective variable with the input parameters gives an analysis of how the objective is affected by the variations in the input parameters. For non-linear systems, the analyses is mathematically computation is complex. In this data-driven model, all every input variables are held at constant, except for one that is allowed to vary following some defined strategy. The variable input factor is allowed to go above and below the base level, following some definite patterns[12]. Corresponding trend of the output variable is determined from the model. If the trend of any model output variable matches the nature of variation provided to the input variable, their interdependence is considered to be direct. If an increase in the input variable causes a decrease in the output space and the converse of it also remains true, then their mutual dependence is taken as of invers. In some cases, the responses could also be mixed and the analysis might also detect no dependence in some cases, which may actually depict any lack of dependence, or could even be the inability of the model to detect a correlation.

5. Results and Discussion

5.1. Scatter plots

5.1.1 Scatter plot between input variables of simulated data

We mapped our input data points obtained by running molecular dynamic simulations. Mapping helps us get a better idea of how our original data spreads across input variables. Datapoints fell into 6 distinct subgroups of different radii, namely, 5, 10, 15, 20, 25 and 30 Å respectively.

Input Variables :
Nanoparticle Radius :
Nanoparticle was simulated by scooping out radius of fixed dimensions from the centre of the crystal lattice.
Nanoparticle Cr concentration :
Nanoparticle Cr concentration = No. of Cr atoms/ (No. of Cr atoms + No. of Fe atoms)

5.1.2. Scatter plot between objectives of input data

We mapped our objectives corresponding to data points obtained by running molecular dynamic simulations.The data points obtained is largely clustered but seems to following an upward trend. Our objective is minimize Cohesive Energy per atom (more negative value preferred) and maximize Surface Energy per Unit Area (higher positive value preferred). This is due to the inherent fact that system tends to minimize its internal energy while preferring higher value for surface energy, a behaviour much desired for excellent catalytic properties.

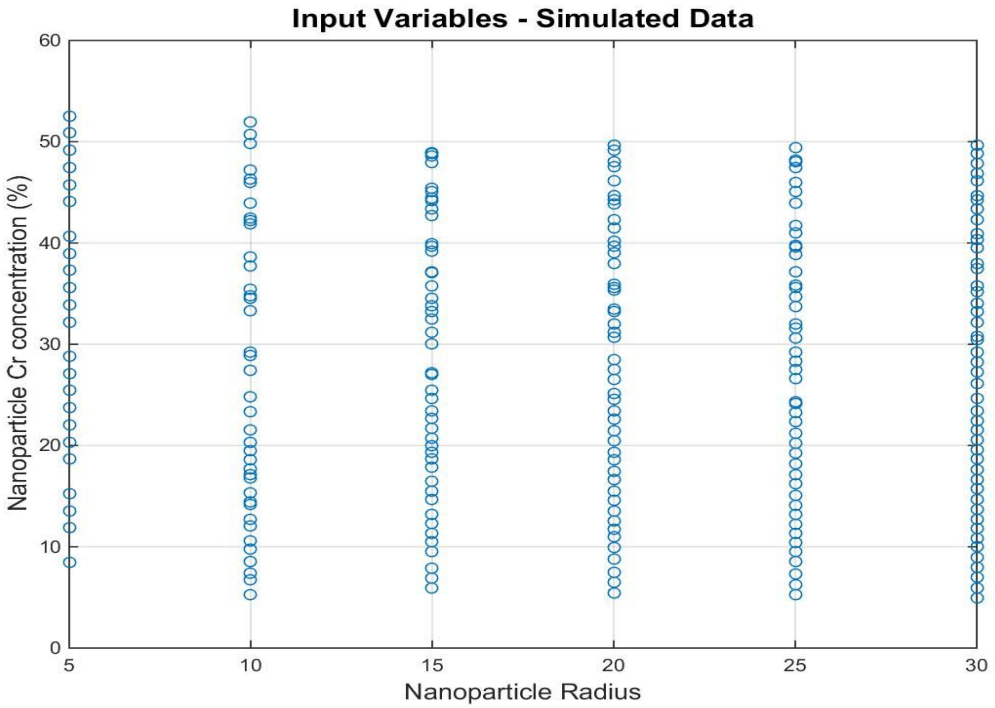


Figure 12: **Scatter plot between input variables of simulated data**

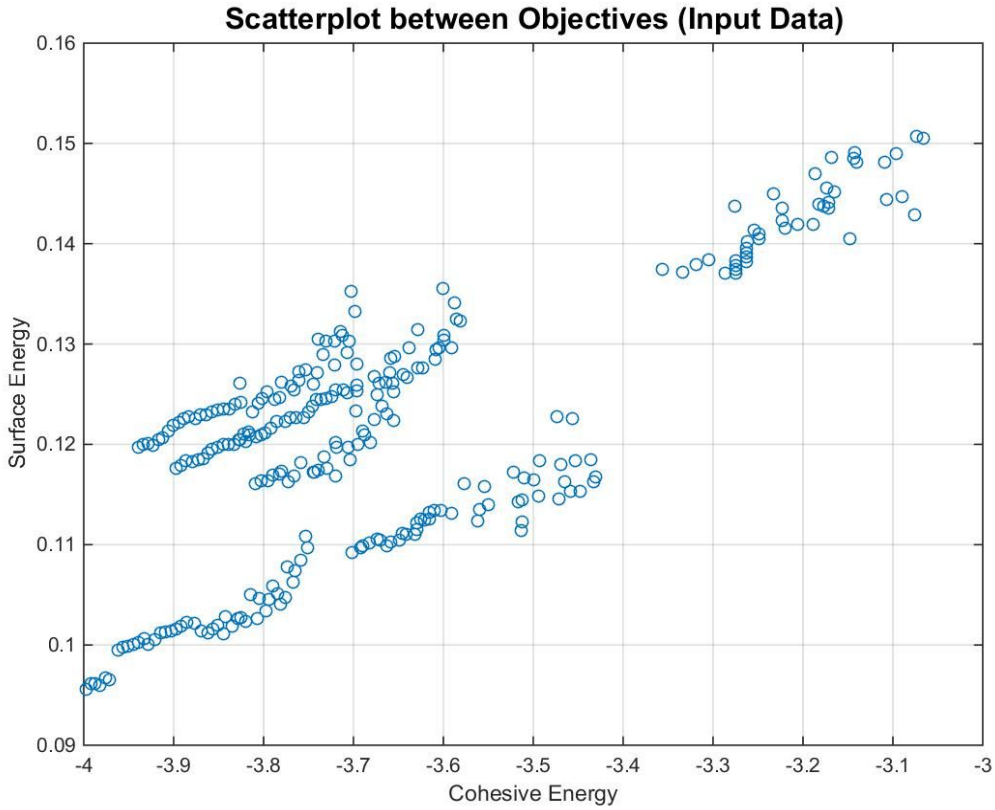


Figure 13: Scatter plot between objectives of input data

5.2. Single Variable Response

Summary of SVR :

The pairwise correlation between input variables and objective functions can be observed from the table given below and the succeeding graphs. While there is little conflict for response obtained between surface energy and input variables, cohesive energy seems to exhibit a bit more relationship with input variables especially nanoparticle chromium concentration leading to conflict between objective functions thereby bringing in the concept of pareto optimality.

Variable	Cohesive Energy (Min)	Surface Energy (Max)
Nanoparticle Radius	Negative	Positive
Nanoparticle Cr concentration	Mixed Response	Positive

5.2.1 Nanoparticle Radius v/s Cohesive Energy

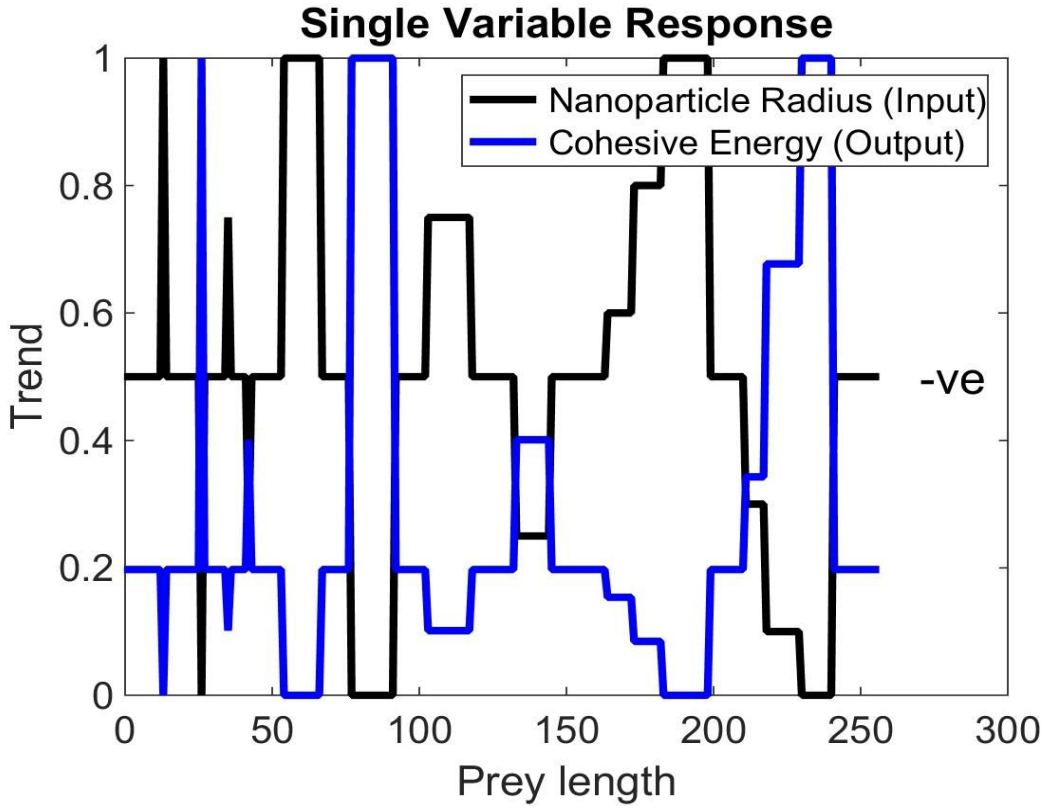


Figure 14: Nanoparticle Radius v/s Cohesive Energy

5.2.2. Nanoparticle Radius v/s Surface Energy

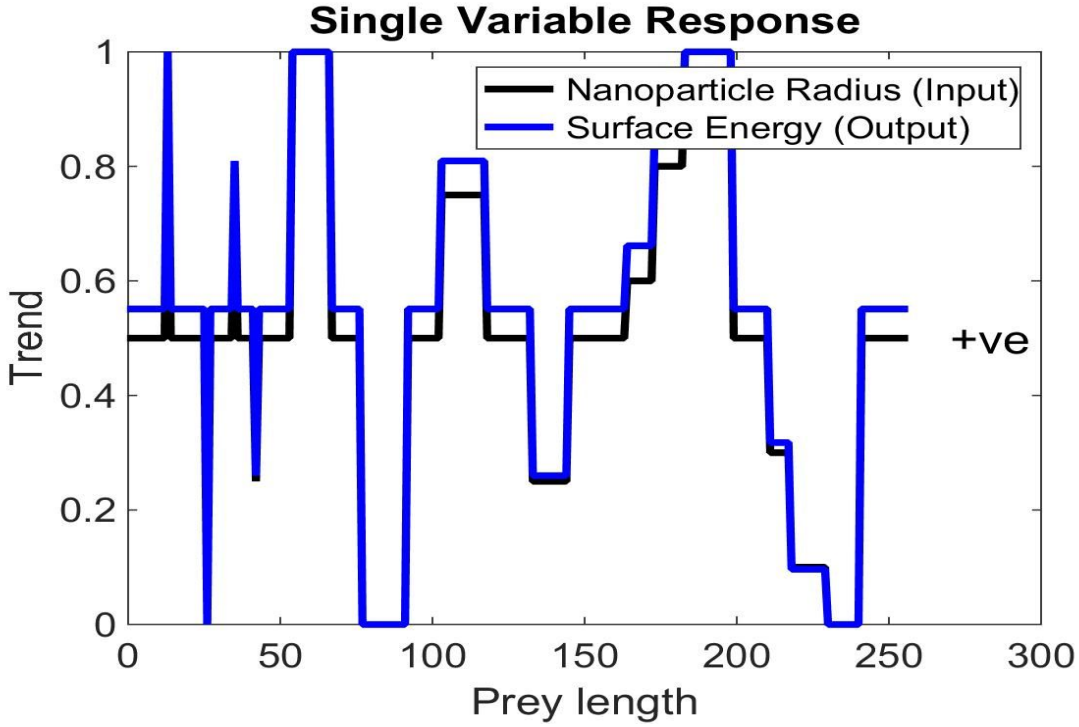


Figure 15: Nanoparticle Radius v/s Surface Energy

5.2.3. Nanoparticle Cr concentration v/s Cohesive Energy

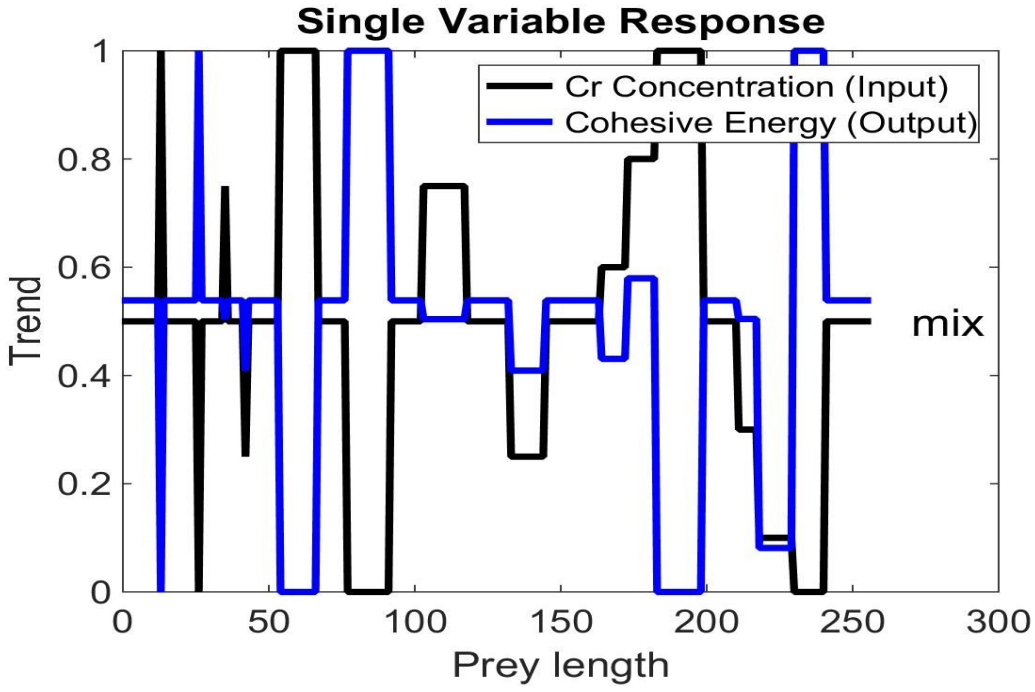


Figure 16: Nanoparticle Cr concentration v/s Cohesive Energy

5.2.4. Nanoparticle Cr concentration v/s Surface Energy

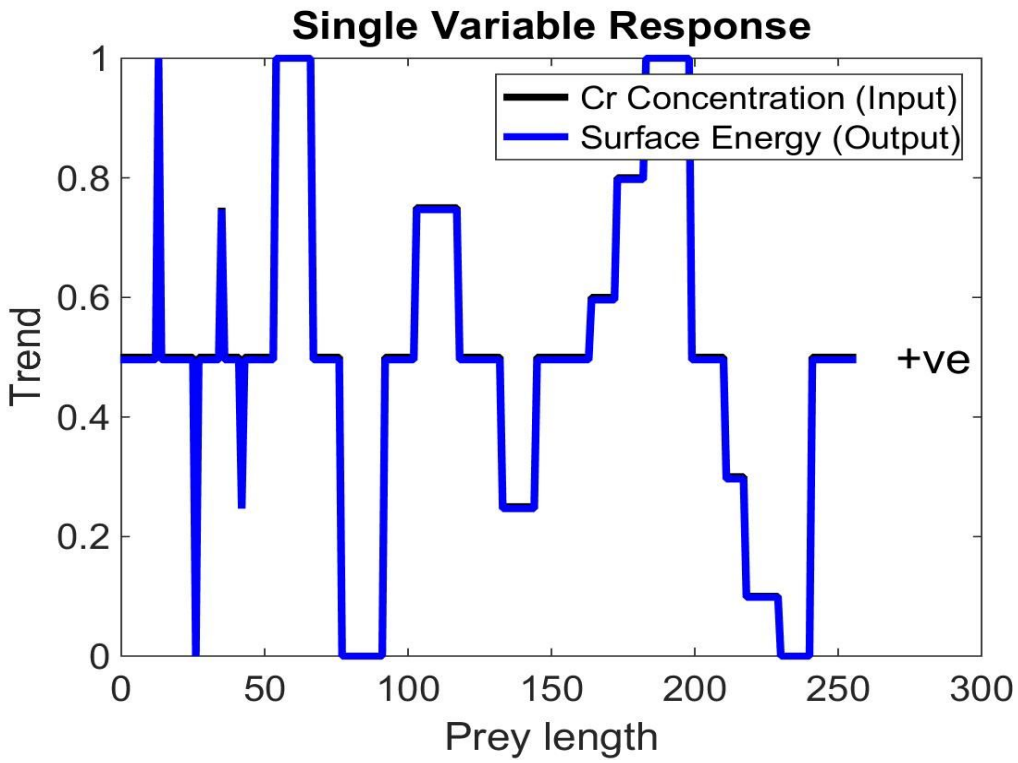
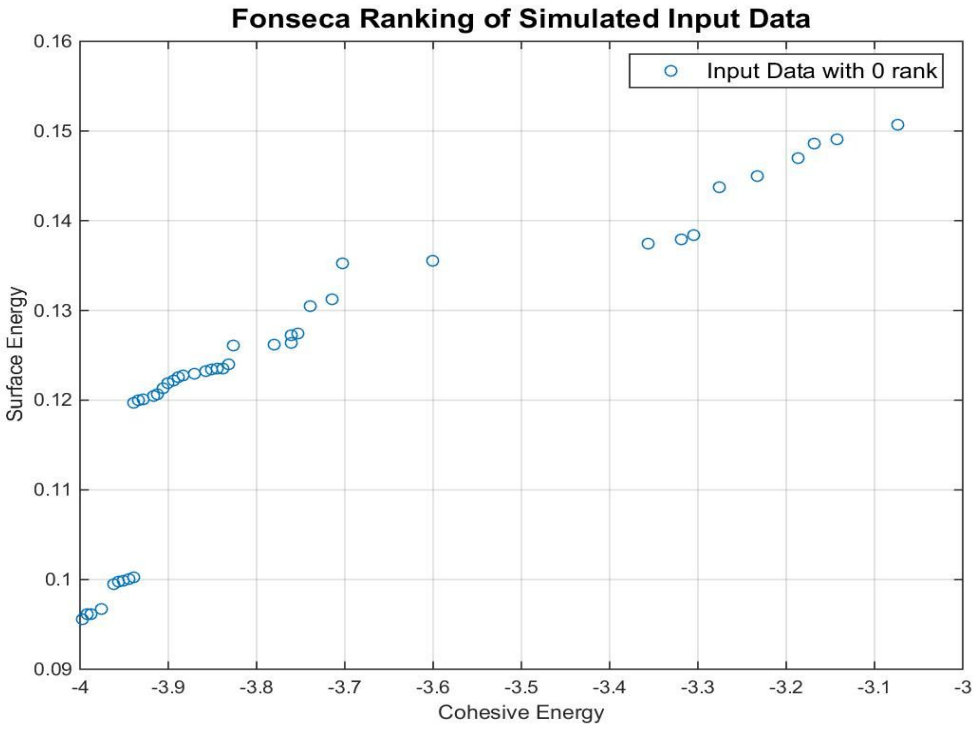


Figure 17: Nanoparticle Cr concentration v/s Surface Energy

5.3. Fonseca Ranking of Input Data Points based on their fitness

Before we proceed for optimization using EvoNN and BioGP, it is worthwhile to have a look at the most optimal data points in our simulated dataset. It was earlier stated that in Fonseca ranking, an individual's rank corresponds to the number of individuals in the current population by which it is dominated. Thus, the optimal solutions are the points with rank 0 obtained from conducting a Fonseca ranking for our data set. The following figures maps the optimal points over objective function space and input variable space. There seems to be a linear upward trend for optimal points in objective function space whereas input data space shows more clustering with very few optimal solutions corresponding to radius 10 and 15. We employ EvoNN and BioGP to check for pareto points in this space given their potential to find diverse solutions using genetic algorithms.



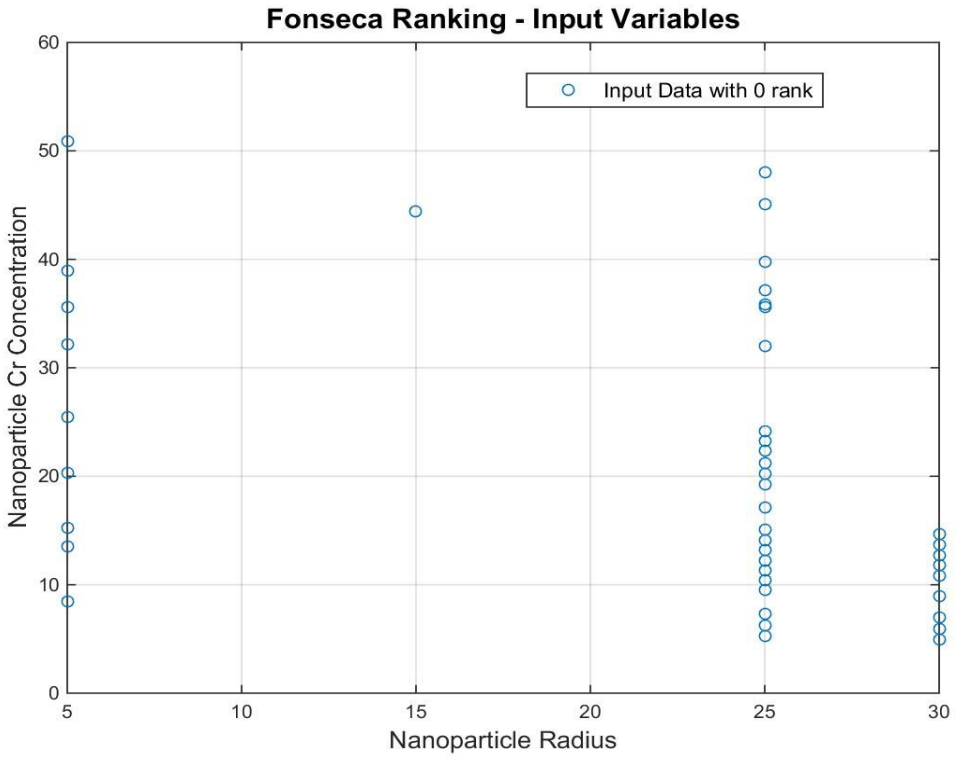


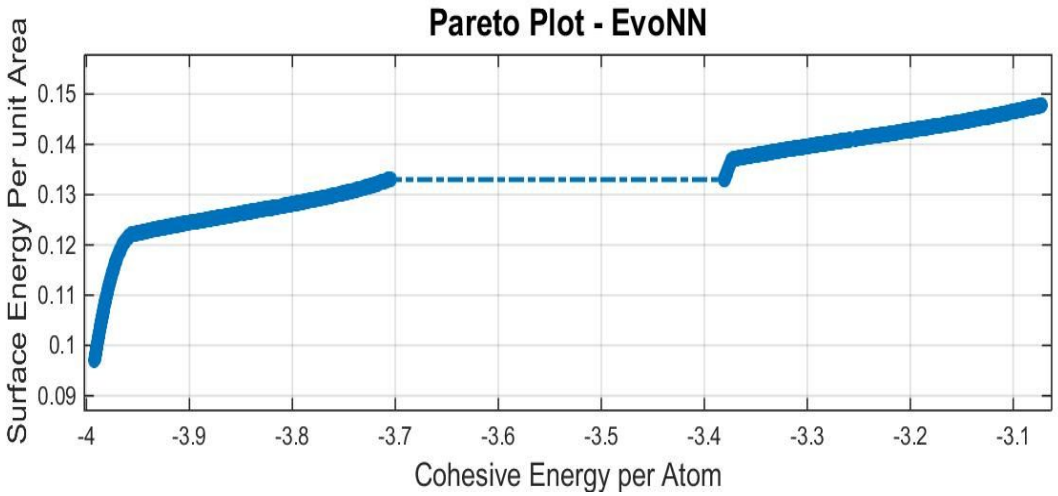
Figure 18: Fonseca Rankings

5.4. Pareto plot obtained using EvoNN algorithm

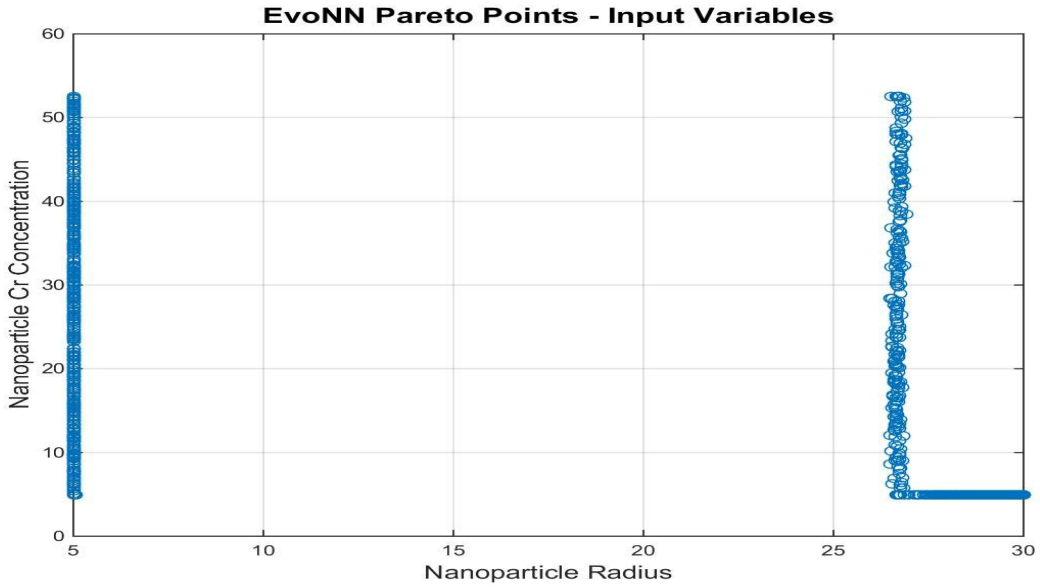
5.4.1. Pareto points distribution over objective functions

The pareto front shows some discontinuities corresponding to the cohesive energy ranges of -3.7 to -3.4. EvoNN algorithms had found optimal solutions in this range as well. However, these points were discarded as their input variable, Cr concentration has values corresponding to 60 which are outside the bounds of the simulated data points.

As can be seen from the input variable space graph, EvoNN extensively explores optimal solutions corresponding to radii below 10 and above 25, however, has very little optimal solutions between these ranges due to the discarding to pareto points.



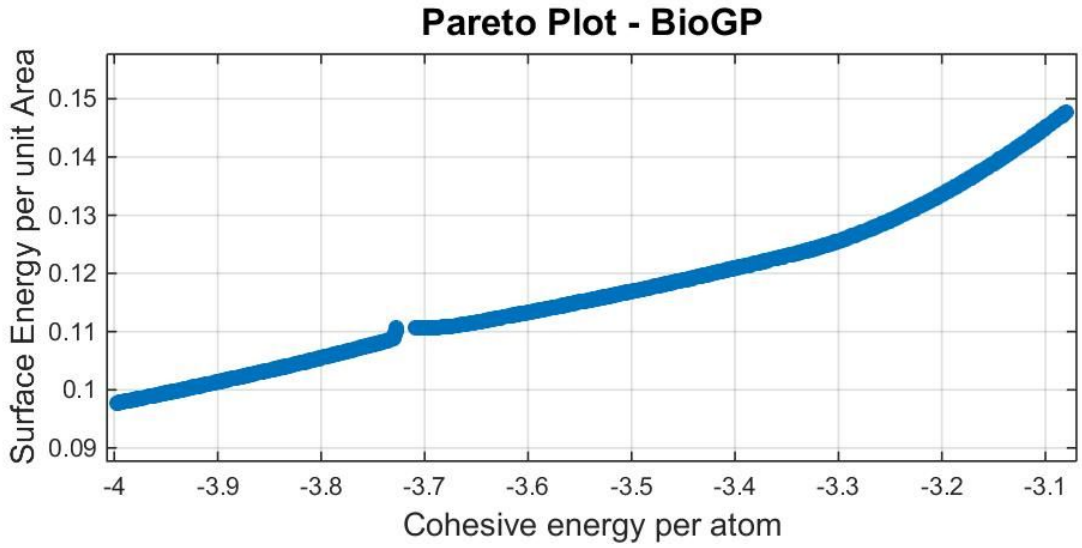
5.4.2. Pareto points distribution over input variables



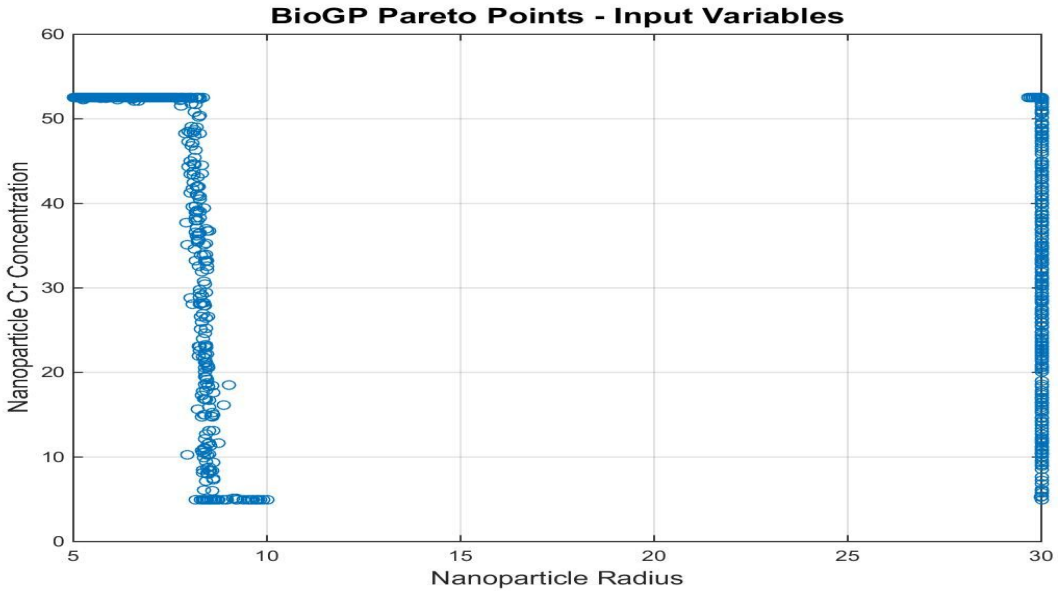
5.5. Pareto plot obtained using BioGP algorithm

5.5.1. Pareto points distribution over objective functions

Mapping done on objective function space indicates BioGP to perform better compared to EvoNN, as there is not much discontinuity in the pareto space. Some discontinuity in pareto plot is however evident in the input variables space where there are hardly any data points in the mid ranges of radii. The reason is not due to inability to BioGP to search the space for solutions, in fact, BioGP explores the ranges as well. This is however due to removal of points lying out of bounds for Cr concentration, thereby rendering them not feasible for further analysis.



5.5.2. Pareto points distribution over input variables



5.6 Comparison between optimal solutions obtained from different methods

The graph below indicates the inherent benefit of diversity obtained by using genetic algorithms in optimization. A whole new range of optimal data points, hitherto unexplored from conventional methods (Fonseca Ranking) was obtained. EvoNN explored pareto points in the higher ranges of dataset while BioGP explored optimal solutions in lower ranges. The large range of pareto points helps a practitioner achieve optimal applications depending on his/her priority for different objectives as well as tolerance levels in fabrication. The ranges at which these optimal solutions can be achieved is obtained from input variables mapping. Ideally, a practitioner should fabricate in the ranges where many pareto points are clustered in limited ranges of input variables , that is, in the ranges of radii near 10 and between 25 and 30.

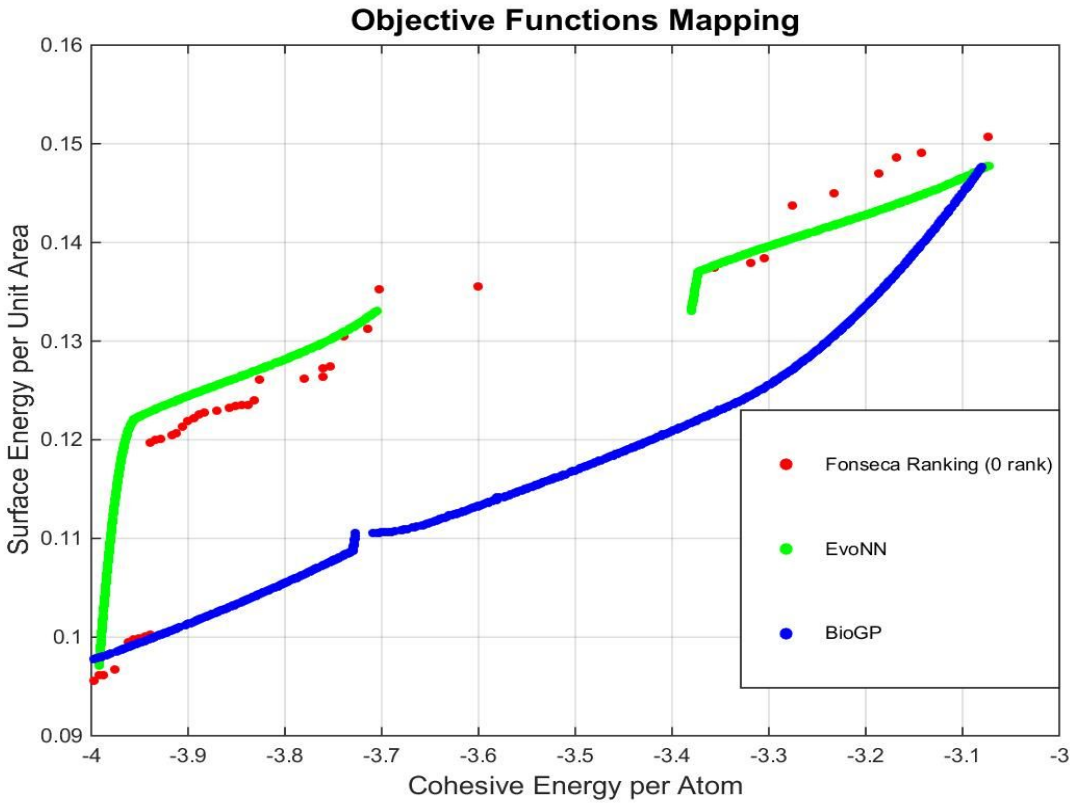


Figure 20: Objective functions mapping

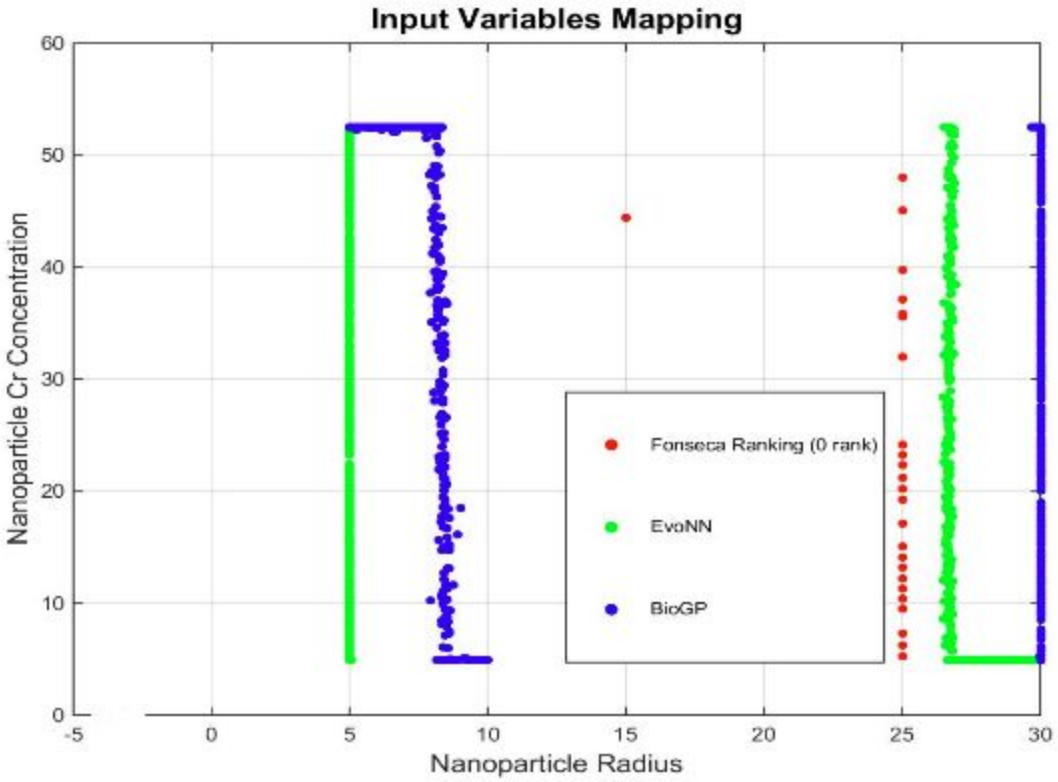


Figure 21: Input variables mapping

6. Conclusion

The benefits of employing genetic algorithms in search of more diverse optimal solutions are quite evident from the above discussion. Employing EvoNN and BioGP have helped explore new optimal data points which could otherwise only be obtained by time computationally expensive molecular simulations. Also, we obtain closely spaced clusters of diverse optimal solutions (near radii ranges of 10 A^o and 25-30 A^o), thereby making it easy for a practitioner to fabricate keeping the objectives in mind as well account for some tolerances in fabrication. This proves to be of extreme use in designing Fe-Cr solid solutions for enhanced catalytic applications (A low cohesive energy and high surface energy are the distinct features of an excellent catalyst). A more robust result can be obtained if we employ the Kimeme algorithm in conjugation with the above methods. The method of k-optimality can be further employed for enhanced selection of pareto points according to practitioner's preference for fabrication ease.

7. References

[1] Allen, M.P., 2004. Introduction to molecular dynamics simulation. *Computational soft matter: from synthetic polymers to proteins*, *23*, pp.1-28.

[2] Olsson, P., Abrikosov, I.A., Vitos, L. and Wallenius, J., 2003. Ab initio formation energies of Fe–Cr alloys. *Journal of Nuclear Materials*, *321*(1), pp.84-90.

[3] Chakraborti, N., Kumar, R. and Jain, D., 2001. A study of the continuous casting mold using a pareto-converging genetic algorithm. *Applied Mathematical Modelling*, *25*(4), pp.287-297.

[4] Bajpai, P. and Kumar, M., 2010. Genetic algorithm–an approach to solve global optimization problems. *Indian Journal of computer science and engineering*, *1*(3), pp.199-206.

[5] Malhotra, R., Singh, N. and Singh, Y., 2011. Genetic algorithms: Concepts, design for optimization of process controllers. *Computer and Information Science*, *4*(2), p.39.

[6] Li, X. and Sutherland, S., 2002. A cellular genetic algorithm simulating predator-prey interactions. In *in Proc. 4th Asia-Pacific Conference on Simulated Evolution And Learning*.

[7] Alberto, I., Azcarate, C., Mallor, F. and Mateo, P.M., 2003. Multiobjective evolutionary algorithms. pareto ranking. *Monografias del Seminario Matematico Garcia de Galdeano*, *27*, pp.25-35.

[8] Pettersson, F., Chakraborti, N. and Saxén, H., 2007. A genetic algorithms based multi-objective neural net applied to noisy blast furnace data. *Applied Soft Computing*, *7*(1), pp.387-397.

[9] Agarwal, A., Tewary, U., Pettersson, F., Das, S., Saxen, H. and Chakraborti, N., 2010. Analysing blast furnace data using evolutionary neural network and multiobjective genetic algorithms. *Ironmaking & Steelmaking*, *37*(5), pp.353-359.

[10] Chakraborti, N., 2014. Critical assessment 3: The unique contributions of multi-objective evolutionary and genetic algorithms in materials research. *Materials Science and Technology*, *30*(11), pp.1259-1262.

[11] McPhee, N.F., Poli, R. and Langdon, W.B., 2008. Field guide to genetic programming.

[12] Chakraborti, N., 2014. Strategies for evolutionary data driven modeling in chemical and metallurgical Systems. In *Applications of Metaheuristics in Process Engineering* (pp. 89-122). Springer, Cham.

[13] Giri, B.K., Hakanen, J., Miettinen, K. and Chakraborti, N., 2013. Genetic programming through bi-objective genetic algorithms with a study of a simulated moving bed process involving multiple objectives. *Applied Soft Computing*, *13*(5), pp.2613-2623.