Machine Learning Algorithms for QSPR/QSAR Predictive Model Development Involving High-Dimensional Data

by

Shounak Datta

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 4, 2019

Keywords: machine learning, cheminformatics, hybrid algorithms, predictive model

Approved by

Mario R. Eden, Chair, Professor of Chemical Engineering
Allan E. David, Associate Professor of Chemical Engineering
Elizabeth Lipke, Associate Professor of Chemical Engineering
Thomas Vincent Gallagher, Professor of Forestry and Wildlife Sciences

Abstract

With advancements in fields such as computational chemistry, computer-aided molecular design and chemoinformatics, the scientific community has now become inundated with a very large set of molecular descriptors. The advantage of availability of large set of descriptors is that computational modelers can now capture different characteristics of molecules of varying sizes in different solvent/reaction mediums. However, the drawback is that during model development, the number of descriptors can exceed the number of instances in a dataset. Such datasets are known as high-dimensional data matrix. This is especially the case when the process of data generation is complex, time-consuming and/or resource intensive. Apart from these reasons, this can also happen when a specific product needs to be developed for a very specific use (e.g. drugs for a specific physical condition, polymers of a specific property, reaction in a specific environment). These cases tend to be very condition-specific, e.g. type of chemical species, activities or responses in specific environment, temperature, pressure, etc. The challenges of modeling such cases include but are not limited to; difficulty of generating a generalizable model, large model uncertainty and overfitting of model(s) generated. To address the aforementioned drawbacks and ensuing challenges, in this work, we have developed hybrid algorithms which are efficient and can generate generalizable models. These algorithms overcome the disadvantage of traditional modeling techniques that break down when the number of descriptors exceed the sample size. The developed algorithms, in our work, can be incorporated in software platforms, useful for automated design of product-centric industrial processes. Such software should be capable of analyzing experimental data and generating the best possible molecular structure for the specific constraints and objectives. It is also required to

be fast and accurate at the same time. In the past, such situations were tackled with ab initio calculations, later replaced by DFT (Density Function Theory) based calculations. Apart from being computationally expensive, such methods include problems of manual handling of data for molecular design operations. To address such limitations, molecular descriptors (0D-7D) became attractive alternatives. However, the complexity of the calculation of descriptors increases with the complexity of the molecular structure. 2D (2 dimensional) descriptors, such as connectivity index descriptors, have been proven to be efficient in model generation with significant accuracy. Also, the design calculation steps are not computationally expensive. For these reasons, in this work, the generated models are based on 2D molecular descriptors. In this work, two unique condition-specific situations have been discussed. Case 1 encompasses relating reactant and solvent structures to the reaction rate constants for Diels Alder reactions. As reaction rates are more prone to depend of inter-atom connectivity, connectivity index descriptors were used to develop this model. A hybrid GA-DT (Genetic Algorithm-Decision Tree) algorithm was developed to select features and for model development. This case is unique as it involves the study of three different chemical species while generating the predictive model, and hence a challenge for both traditional and newly developed hybrid algorithms. Further improvements for the model were proposed using Multi-Gene Genetic Programming (MGGP) algorithm to derive non-linear models. Case 2 is based on developing a model to relate structures of 9-Anilinoacridine derivatives with respective DNA-drug binding affinity values. Although this case has only one group of chemical species under consideration, challenges emerge when two or more models with similar metrics are generated. Although the genetic algorithm was used for feature selection, initially, a novel adaptive version of LASSO (Least Absolute Shrinkage and Selection Operator) algorithm was developed. This adaptive correlation-based LASSO

(CorrLASSO) was used to perform regression and shrinkage calculations. To evaluate model fitness, R^2 and Q^2 values were calculated that represent model internal and external validation respectively. For the second case, mean square error (MSE) was also calculated to compare the performances of LASSO and CorrLASSO algorithm.

Acknowledgements

This astonishing journey of last five years, in many ways, has occurred in debt of many amazing persons, both in Auburn and back home. I was fortunate to have acquaintance with people who admired a poetic soul in STEM; people who saw a philosopher in me and supported that ideology and philosophy while blossoming through scientific research. Above all, I was fortunate to have acquaintance with people who cared to know, understand, and be part of this journey, making these five years to be the best part of my life so far.

First and foremost, my heartfelt gratitude to Dr. Mario R. Eden, my thesis supervisor. None can ask for a better supervisor. This part of my life has been very enjoyable due to his guidance, friendship, and endless support. He has provided me opportunities not only to pursue my passion for research, but also to present my work and connect with my peers both in US and abroad. Nothing can be truer than the fact that the advisor, particularly while pursuing a doctorate, can make all the difference in growing as a researcher, and a wondering soul. Dr. Eden did everything humanly possible to provide me with the environment to grow as a researcher, a professional, a problem solver, and a seeker of knowledge. For that, I am forever in debt of his gracious nature.

I would also like to take the opportunity to acknowledge the support provided by my committee members, Dr. Allan David, Dr. Elizabeth Lipke, and Dr. Thomas Gallagher. Their feedback, support, and constant interest in my work can be considered of utmost importance for development of my dissertation, as well as the quality of my work. To add, I have been the most

fortunate to have the following people as my peers and groupmates. The list includes, but is not limited to, Dr. Vikrant Dev, Dr. Narendra Sadhwani, Dr. Anjan Kumar Tula, Dr. Sawitree Kalakul, Dr. Robert H. Herring III, Mr. Bernardo Lausada, and Mr. Pengcheng Li. The bond of friendship, companionship, and camaraderie I have shared with each and every one of them is priceless. These incredibly smart and wise peers of mine have always been able to help me with professional and academic advice, suggestions, and comments. Special thanks to both Dr. Robert H. Herring III, and Dr. Vikrant Dev for always collaborating with me in my projects.

In life, it's not only people around you, but also people you never meet, that sometimes end up leaving something behind, making your life a better one. This section is for the people I've never met, but who inspired or helped me get through the challenges of this phase of my professional journey. With that spirit, I'd like to thank John Lenon (Imagine), Frank Sinatra (My Way), Edith Piaf (Rien de Rien), Nada (Senza Un Perche), Johhny Cash (Hurt), 4 Non Blondes (What's Up), Queen (Break Free), Bee Gees (Stayin Alive), Franz Lehar (The Merry Widow Waltz), Claude-Michel Schönberg (One Day More, Les Mis), and last but not the least, Ludovico Einaudi (Divenire, Experience, Prima Verra, Newton's Cradle). To add, a special gratitude for the unnamed and unknown heroes who have discovered palatability of coffee, boiled eggs, and bananas.

A special thanks to all the friends I've made during my stay in Auburn. The list of their name and their contributions in my life deserve a thesis dissertation of its own. However, to keep it short, I would like to thank the graduate recruits of Fall 2013. In a literal sense, "we had joy, we had fun, we had seasons in the sun" (Terry Jacks). Thanks to all my friends in Miller Writing Center, specially David Vinson, Phineas Dowling, Katie Kirk, James Truman, Savanna Davey, and Jacob Dylian Geiger.

Additionally, special thanks for my parents, who supported me through the decision of leaving a lucrative job back home and pursuing my passion for research. Also, I would like to express my gratitude to my parents-in-law, my sister-in-law, for always concerning themselves with my worries and wellbeing. And in the end, to the girl who I saw distributing food in a carnival, fell in love with, and who chose be the most important part of my life, four years ago; accepting the hardships of starting a family abroad all by herself, Joyeeta Das. Challenges came in bundles, storms brewed, both personal and professional, and all were much easier to deal with due to her elegant presence, endless support, encouragement, and love.

Table of Contents

List of Figures

List of Tables

List or Abbreviations

AM1	Austin Model 1
CAMD	Computer Aided Molecular Design
CART	Classification and Regression Trees
CI	Connectivity Index
DFT	Density Function Theory
DT	Decision Tree
ESCAPE	European Symposium of Computer Aided Process Engineering
FOCAPD	Foundation of Computer Aided Process Design
GA	Genetic Algorithm
GA-DT	Genetic Algorithm- Decision Tree
GP	Genetic Programming
HF	Hartree- Fock
ID3	Iterative Dichotomiser 3
LASSO	Least Absolute Square and Shrinkage Operator
LCAO	Linear Combination of Atomic Orbitals
MD	Molecular Descriptor
MGGP	Multi-Gene Genetic Programming
MLR	Multiple Linear Regression
MPPSE	Multi-scale Product and Process System Engineering
MSE	Mean Squared Error
PCA	Principal Component Analysis

PE	Prediction Error
PM	Property Model
PM3	Parameterized Model 3
PSE	Process Systems Engineering
QM	Quantum Mechanics
QSAR	Quantitative Structure-Activity Relationship
QSPR	Quantitative Structure-Property Relationship
RMSE	Root Mean Squared Error
SAR	Structure-Activity Relationship
SVR	Support Vector Regression
TI	Topological Index
TSP	Travelling Salesman Problem

1. Introduction

According to Staphanopoulos and Reklaitis (2011), chemical industries transformed to become more product-centric than process-centric in the period of 1980-2000. In the process-centric period, products were considered to be molecules that can be used for a specific function and R&D was more focused on increasing process efficiency. This period produced the famous notion, “Chemical Engineers either make money, or save money.” The product-centric period required the chemical industries to focus more on development and sale of value-added materials that can be marketed based on performance (Hill, 2009). These products came to be known as chemical products. Grossman and Westerberg (2000) have also discussed this shift in focus by the chemical industries. They predicted that, demands for improved earnings performance from commodity and specialty product manufacture will become a significant method of attracting investors in the industries. According to their predictions, such driving forces were expected to lead to process design expanding to accommodate product design; where the particular emphasis would be on designing new molecular structures of chemical species. This shift in process design has influenced Process System Engineering (PSE) to add branches to its process roots; and include project management, multi-scale operations, even whole supply chain (Sargent, 2005). This expansion of PSE community also attracted and forced chemical engineers to collaborate with other disciplines like material sciences, computational chemistry, electrochemistry, etc. to pursue product design (Klatt and Marquardt, 2009). Klatt and Marquardt also suggested widening the scope of PSE into multi-scale product and process system engineering (MPPSE) to address product design in an integrated fashion. Adjiman and Galindo (2011) suggested the term Molecular Systems Engineering to formalize the recognition of designing molecules and materials as an integral part of designing and optimizing processes and products. Glavic (2012)

also mentioned that in the period of 1985-2006, integrated product development and design became one of the significant parts of PSE conferences. Certainly, there has been a successful attempt to illustrate and elevate the relevance of product design over the years in the PSE and the chemical engineering community.

As stated in the article of Stephanopoulos and Reklaitis (2011), in the international PSE conference, named Foundations of Computer Aided Process Design (FOCAPD), held in 2005 at Princeton University, was themed to be “Discovery through Product and Process Design”. This conference made an important contribution to the field of product design as it promoted a very broad range of issues, ideas and fields related to product design. These ranged from designing simple small molecules, functional molecules such as dyes to structured products which perform certain functions such as batteries and products closely connected to emotional disposition of humans such as clothing. A significant amount of contributions from PSE community on product design focused on optimal generation of molecular structures that can satisfy specific needs. These structure development works are highly dependent on deriving models to assume a specific property or activity of a molecular structure using physico-chemical properties of the molecular structures under study. Although Stephanopoulos and Reklaitis (2011) expressed their doubts towards such operations due to reduction in reliability of mathematical models relating structures to properties, Mlinar (2015) has expressed that almost over 151 new products have been design in recent times using Computer Aided Molecular Design (CAMD) approaches. These products varied from polymers to pharmaceutical drugs to corrosion inhibition fluids. However, Segall (2012) raised his skepticism in the abilities of computer aided drug design (CADD) processes. He expressed that the prediction in drug discovery is not yet sufficient to permit a design paradigm, as demonstrated by the large number of compounds that must be

synthesized and tested to identify a successful drug. However, many drug-like index descriptors have been utilized for studies in present times. He also did not comment on whether choosing proper descriptors and model boosting could lead to successful drug design.

1.1. Scope and Objectives

In the present era of CAMD, a vast majority of the molecular design problems are addressed using computationally expensive approaches, such as quantum mechanics, molecular mechanics, density function theory, and electromagnetic valence bond theory. These approaches, although promising high accuracy, come with problems including time consumption, computational power consumption, and geometric evaluation restrictions. Geometric evaluation restrictions are generally caused by algorithms used to optimize the geometry. They can surely generate models with higher accuracy, but the consumed time and computational energy is exponentially high. Also, the product design operation is rarely an automated one using these models. Even if such system is automated, reaching an optimum result can consume much higher computational energy and time. Also, such developed models, although published worldwide, can be rarely used by communities not having the privileges of high computational powers provided by well-known HPC (High Power Computer) centers. For these reasons, a better approach of developing predictive models using lesser computational power and time is of paramount importance. QSAR (Quantitative Structure-Activity Relationship), in this situation, provides a lesser time consuming option. However, the model applications are dependent on descriptors used, and the size of the dataset available for model development.

With the increase in demands of personalized product development, there is high uncertainty that a high volume of experimental data for a specific product design problem may

be available. Although the PSE focus has been on developing such predictive models for a while now, the future can demand a software platform for further analysis of these product development questions. This requires development of algorithms for QSAR generation and hybrid machine learning algorithms can serve an important purpose here.

This project, aiming at developing predictive models using generic algorithms, can obviously show promise in dealing with such technical questions. The project used descriptors that are computationally easy to calculate but capture a significant amount of molecular information. The focus was to develop models with higher predictive accuracy. While doing so, caution was exercised to not overfit the training set. It was assumed that the models are applicable within a limited chemical space. For that reason, the focus was put on developing generic algorithms in addition to generalizable models. The proposed algorithms showcase two unique situations. One case involved multiple classes of chemical species influencing the particular property; and the second case involved generation of multiple predictive models with similar accuracy. Both involve questions not been answered significantly using QSAR approaches. The project also briefly describes the hybrid algorithms developed, used, and modified, and the considerations required for such alterations.

1.2. Research Significance

Although, the end of Moore's law can be noticed as present technologies have met their limits in producing smaller transistors at present, chip manufacturing companies are investing extensively to develop post-Moore's law devices (Pavlus, 2015). These chips, if developed, can play a crucial part in significantly improving the current computational efficiency. Strategies such as 'heterotic computing' that involve usage of combination of two or more computational

systems are also emerging to help accelerate progress in a post-Moore's law world (Kendon et al., 2015). Due to the recent developments and improvements in computer architecture and distribution techniques, Cedar and Persson (2013) suggest that our next step might be a giant leap towards a golden age of materials science. Until that time approaches, the smartest way of dealing with the increased diversity of product requirements is developing algorithms with high efficiency. The more condition-specific modeling problems that are being dealt with, the greater the need has become for efficient modeling and molecular design algorithms. The challenge, however, is to develop hybrid algorithms that can boost the modelling efficiency without increasing modelling cost. Such models can assist in developing software platforms for automating model development and optimum molecular structure generation. This important problem is being addressed in our work.

Additionally, many of the algorithms developed in the discipline of computer science are not necessarily geared towards QSAR development. Hence, the algorithms may not be efficient for QSAR development. This issue has been further discussed in Section 2.3, while describing different varieties of machine learning and evolutionary algorithms used in this project. This results in cases where an algorithm needs to be modified to perform well for QSAR predictive model development purposes.

Finally, every problem in QSAR has unique features, so needs different hybrid algorithms. Focus should be directed on the type of problem, chemical space, and chemical descriptors involved. This situation needs delicate consideration while developing hybrid algorithms. However, a good understanding of algorithms is also a prerequisite of developing efficient hybrid algorithms. In theory, a hybrid algorithm's aim is to overcome the limitations of the individual algorithms and produce better result. This work aims at providing case studies for

understanding algorithm limitation, pros and cons, and choosing a perfect combination of two or more algorithms to overcome these limitations. While doing so, modifications and tuning approaches have also been proposed to keep a check on model overfitting. For these reasons, it can be argued that, a crucial part of CAMD research for the next decade at least will be highly dependent on developing hybrid algorithms to generate diverse and reliable predictive models and product design solutions.

1.3. Cases presented

Over the years, significant amount of efforts concerning computer-aided product design (CAPD) have been devoted to computer-aided molecular design (CAMD). However, there are problems involving CAMD that are yet to be substantially addressed. For example, processes that involve reactions involving multiple classes of chemical species need attention. Firstly, in terms of developing property models (PMs) attention is required; and second, in developing methodologies to utilize these PMs to design molecules and processes. So far, in processes involving a reaction, which is one of the focuses of our work, CAMD of solvents, catalysts, reactants and products has been carried out.

The search for QSPR models to predict influence of structures of both reactants and solvents on reaction rate constants has long been a challenge. According to Roy et al. (2015a), QSPR (Quantitative Structure Property Relationship) models are generally linear or non-linear mathematical relationships that correlate a particular property or activity of a chemical species with their structure. Such structures are generally represented numerically by descriptors, which can be determined experimentally or theoretically as per the definition. Early attempts to develop QSPR models for the prediction of rate constant of a reaction have been restricted. Either the rate

constant was studied as a function of structures of reactants while keeping the solvent structure constant or the solvent structures were varied but the reactants' structures were kept constant. With regards to the study of the influence of reactants' structures, Chaudry and Popelier (2003) developed a property model to predict the rate constant of hydrolysis of esters by utilizing quantum chemical descriptors. Estrada and Matamala (2007) have used generalized topological indices to predict the gas phase reaction rate constants of alkanes and cycloalkanes with OH, Cl and NO_3 radicals. With regards to studying the effect of solvent structures, not only have property models been derived but CAMD of solvents to enhance reaction kinetics has been pursued by researchers. Recently, Struebing et al. (2013) developed a methodology to design solvents by utilizing surrogate models and quantum chemical calculations. In the past, Stanescu and Achenie (2006) have presented a theoretical study of solvent effects on Kolbe-Schmitt reaction rates. There is a need for QSPR models that capture the influence of reactants' and solvent structures. Such models will serve two purposes: The first would be to quickly predict the rate constant without relying on experiments, while the second purpose will be the simultaneous design of reactants, products and solvents. With regards to CAMD of reactants/products, Dev et al. (2015) have proposed a methodology to design reactions based on thermodynamic properties of reactions. Thus, there is scope for a methodology which designs reactants and solvents such that the rate of reaction is optimized.

Diels-Alder reaction is a famous and well-studied organic chemical reaction involving a conjugate diene and an alkene, which is also termed as dienophile. Evans and Johnsons (1999) have considerably discussed this reaction in their work. This reaction involves cycloaddition of two reactants in the presence of a solvent. Rideout and Breslow (1980) have presented the hydrophobic acceleration of Diels-Alder reaction. Their work focused on discussing the

influence of hydrophobic cavity in organic structures for acceleration of the reaction rate. In both of the aforementioned works, the impact of the solvent on the rate constant of the reaction has been observed. This feature of this reaction makes it an appealing choice for the aimed study. Figure 1-1 represents a sample of Diels-Alder reaction.

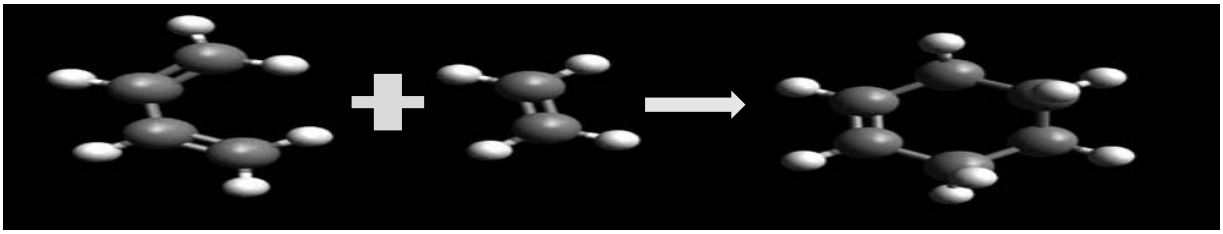


Figure 1-1: Sample Diels-Alder reaction

With respect to QSPRs that capture reactant and solvent influence, Nandi et al. (2013) developed a quantitative structure-activation barrier relationship for Diels-Alder reaction that utilizes quantum chemical descriptors. Their aim was to construct a relationship between the activation energy and the structures of the utilized reactants and solvent. However, their data set lacked solvent variety. Recently, Zhou et al. (2014) have studied a variety of solvents for the Diels-Alder reaction in their search for new solvent descriptors though they only used one set of reactants. Thus, we have combined the data sets utilized by Nandi et al. (2013) and Zhou et al. (2014) and created a set which offers more diversity in terms of the solvents utilized. We have utilized this more diverse data set to develop a rate constant model in terms of connectivity indices. It is worth noting that Nandi et al. (2013) relied on the data set utilized in the work of Tang et al. (2012). In addition to improvement in the data set, we have also proposed an efficient hybrid GA-DT algorithm for model development which utilizes the “divide and conquer” strategy in combination with principal component analysis (PCA). Both internal and external validations were performed separately to determine model confidence. R^2 and Q^2 values in case

of internal and external validation were calculated respectively as they describe model fitness of data.

In this work, additionally, we have proposed an efficient multi-gene genetic programming (MGGP) algorithm using initialization by a modified DT algorithm for model development which utilizes the "divide and conquer" strategy in combination with principal component analysis. This DT algorithm checks if the addition of branched gene improves model fitness. The MGGP algorithms hold promise as they possess the ability of developing models using a wide variety of nonlinear mathematical basis functions. Both internal and external validations were performed separately to determine model confidence. Additionally, model RMSE and R^2 values in case of both external and internal validation were calculated as they describe model fitness of data.

For the second case, the aim is to develop a QSPR model describing the DNA binding properties of 9-anilinoacridines in order to assist in their design and property prediction. Baguley et al. (1981) extensively studied the drug binding abilities of such chemical species and evaluated the drug-DNA association constants (*K*). For their evaluation, they studied the fluorescence of drug-Ethidiam-DNA mixtures to determine C_{50} values at pH 5. A C_{50} value denotes the micromolar drug concentration required to reduce the fluorescence of initially DNA-bound ethidium by 50%. Baguley et al. (1981) also included the calculation of the extent of reduction of quenching (C_q) to finally calculate the *K* values for 9-anilinoacridines. Including all the expressions of C_{50} and C_q, *K* can be defined by Eq. (1.1).

$$K = \frac{D_b}{D_f S_f} \tag{1.1}$$

Where D_b is the DNA-bound drug concentration, D_f is the free concentration of drug, and S_f is the effective free DNA site concentration. All of these values were experimentally derived by equilibrium dialysis or spectrophotometric titration. Recently, Chtita et al. (2016) chose some of the values provided by Baguley et al. to develop a QSPR model to model *K*. In their work, Chtita et al. (2016) utilized DFT (Density Functional Theory) based descriptors to develop the models. Although their proposed model shows great promise for studying the response of such chemical species, the provided model is computationally expensive when used in an automated molecular design system. Also, they formed different descriptors based on free energy descriptors that made the molecular descriptors being dependent on each other to some extent. To ensure drug design efficiency, only topological index, ring index, conventional index, and connectivity index descriptors have been used in our work. For developing the model, a correlation-based LASSO has been combined with GA. Among the 65 chemical species studied by Baguley et al. (1981), 31 have been selected in our work to generate a linear model which has good predictive ability.

1.4. Organization

For truly evaluating the thesis, it is of paramount importance that readers should know about the background of this work. The background plays a significant role in helping the reader skeptically study the proposed methods to disregard any bias towards them. For such purposes, the background is provided in chapter 2. This gives the readers a brief idea about molecular descriptors, types of the descriptors, pros and cons of different classes of descriptors. Next, the section expands into description of various CAMD approaches, including molecular mechanics, quantum chemical methods, and QSAR/QSPR approaches. The sections provide ample amount of details on equations used, approaches developed, modifications made for computational

simplicity, and advantages and disadvantages of these approaches. Also, the chapter includes description of feature selection algorithms, different versions of such algorithms, and their potential uses. Afterwards, regression algorithms are also discussed in the same manner.

Chapter 3 includes the methodology of the proposed methods. A significant portion of this chapter is divided into two parts, depending on the cases analyzed. The chapter presents a deeper look into the approaches used, modification of the algorithms introduced earlier, and reasons for the assumptions made during such modifications. The chapter places high importance on describing each algorithm, their general use, and their roles in this work.

Chapter 4 presents the results observed using the approaches discussed earlier. As per the two cases, chapter 4 has also been divided in two sections. First one describes the observed results of the first case, the second part describing the observed results for the second case. The chapter contains algorithm and proposed model performances. The chapter also, in brief, discusses the probable limitations of the proposed models and methods. Finally, appendices have been provided to detail the codes prepared for the algorithms, the models, and to describe the descriptors in the models.

The work in section 3.1 has been presented in the ESCAPE (European Symposium of Computer Aided Process Engineering) conference in 2016. It was also published in Computer Aided Chemical Engineering (Datta et al., 2016). An expanded version of this work also appears in work of Datta et al. (2017) in the journal, Computers and Chemical Engineering. An improvement of the approach has been presented in the PSE (Process Systems Engineering) conference in 2018. This work appears as a peer reviewed presentation publication (Datta et al., 2018). The work discussed in Section 3.2, has been published in the journal, Computers & Chemical Engineering (Datta et al., 2018).

2. Background

This chapter presents the theoretical ideas and techniques of CAMD required to understand the fundamentals of the project. Another crucial part of this section is to present comparisons of different elements of CAMD approaches through examples, discussion of benefits and shortcomings, and suggestions made to overcome such shortcomings until now. The section begins with describing the viability of using molecular descriptors, the properties of molecular descriptors, and classes of molecular descriptors. While discussing the different classes of the descriptors, both pros and cons of these algorithms are reported. In addition, the applicability of such classes has also been discussed in this section. In the second section, CAMD approaches, quantum mechanics method, molecular mechanics method, and QSAR/QSPR methods are discussed. Attempts were made to describe these methods with as much clarity possible, given their mathematical complexity. Also, as in previous section, advantages and disadvantages of these methods are presented with details. The third section is dedicated to discussion of the algorithms used in the project. While doing so, a deeper analysis of these algorithms is presented to learn about the different versions employed in this project.

2.1. Molecular Descriptors

Molecular Descriptors (MDs) are the numerical values related to the chemical constitution for correlation of chemical structure with various physical properties, chemical reactivity or biological activity (Roy et al., 2015a). Property models (PMs) utilize descriptors to represent the chemical structures by expressing relationships between properties and chemical structures of molecules. PMs are a means of developing a quantitative relationship between properties and structures of molecules. Hence they are also known as Quantitative Structure-

Property Relationships (QSPRs). In cases when the property is the biological activity of a molecule, the QSPR is then known as a Quantitative Structure-Activity Relationship (QSAR). According to Todeschini and Consonni (2000), “the molecular descriptor is the final result of a logical and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number.”

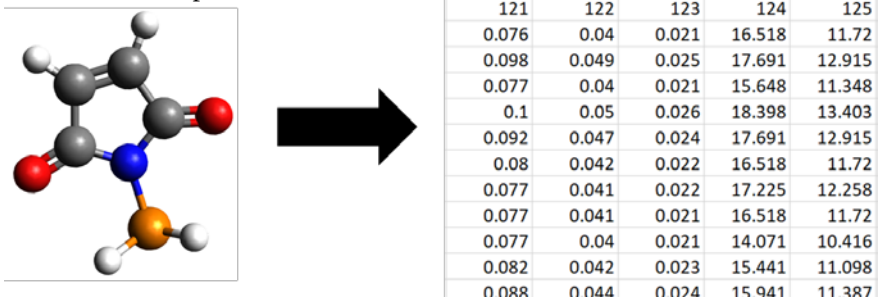
Thus, we can say that there are two types of descriptors: theoretical descriptors and experimental measurements. Theoretical descriptors are numerical values that are obtained analyzing symbolic representation of molecules while experimental measurements are values of physico-chemical properties like polarizability and dipole moment. Theoretical descriptors are more advantageous over experimental measurements because the error associated with experimental noise can be avoided. In addition, the practice of expressing the PMs in terms of other physicochemical properties is an older and obsolete one. These properties themselves can now be expressed in terms of theoretical descriptors. A wide variety of theoretical descriptors have been developed in terms of which different properties can be expressed. Although no set of rules or criteria could be found that dictates the development of new theoretical descriptors for various property models (Hong et al., 2012), some general guidelines have been listed by Roy et al. (2015b) as follows:

- A descriptor must be correlated with the structural features.
- A descriptor shows negligible correlation with other descriptors.
- A descriptor should be applicable to a broad class of compounds.
- A descriptor should be calculated rapidly, not depending on experimental properties

- A descriptor should generate dissimilar values for structurally different molecules, even if the structural differences are small. In other words, the descriptor should show minimal degeneracy.
- In addition to degeneracy, a descriptor should be continuous. Small structural changes should lead to small changes in the value of the descriptor.
- A descriptor should have some form of physical interpretability to encode the query features of the studied molecules.
- A descriptor should have the ability to map descriptor values back to the structure for visualization purposes.

Apart from the classification used by Todeschini and Consonni (2000), there are other types of classifications of molecular descriptors. For example, descriptors can be classified based on origin. Based on origin, MDs can be classified as topological (graph theory based), constitutional (functional group count), geometrical (distances, valence angles, surfaces, etc.), quantum-chemical (charge distribution related), and thermodynamic (heat of formation, entropy, etc.) descriptors (Hong et al., 2012). However, it is much easier to communicate the type of developed PMs when MDs are classified based on their dimensionality (Roy et al., 2015b). The MDs can be classified as zero-dimensional (0D), one dimensional (1D), two dimensional (2D), and so on. MDs with up to seven dimensions (7D) have been developed so far. It is worth noting that descriptors with up to two dimensions are the most commonly utilized ones due to ease of calculation. However, when large molecules are involved (e.g. protein, DNA structure), descriptors with more than 2 dimensions are also utilized in property models. Descriptors with more than 3 dimensions are geared for more sophisticated applications. Due to the complexities

involved in calculating such high dimensional indices, these descriptors are lesser used. Figure 2-1 illustrates a simple example of descriptors generated from molecular structure.



X3Av	X4Av	X5Av	X0sol	X1sol
121	122	123	124	125
0.076	0.04	0.021	16.518	11.72
0.098	0.049	0.025	17.691	12.915
0.077	0.04	0.021	15.648	11.348
0.1	0.05	0.026	18.398	13.403
0.092	0.047	0.024	17.691	12.915
0.08	0.042	0.022	16.518	11.72
0.077	0.041	0.022	17.225	12.258
0.077	0.041	0.021	16.518	11.72
0.077	0.04	0.021	14.071	10.416
0.082	0.042	0.023	15.441	11.098
0.088	0.044	0.024	15.941	11.387

Figure 2-1: Generation of Descriptors from Molecular Structures

2.1.1. 0D Descriptor

Molecular descriptors that are derived from the molecular formula fall in the category of 0D descriptors. Since while writing the molecular formula we are not concerned with the arrangement of molecules but only the composition, the descriptors are derived from a zero dimensional representation of the molecule. Thus the descriptors are referred to as 0D descriptors. Examples of 0D descriptors include atom counts, charge, molecular weight, etc.

In the example shown in Figure 2-2, carbon count, a 0D descriptor, was utilized in creating a linear property model for the boiling point of a series of alkanes (C1-C7). Although, the proposed model was able to account for 97% of the variance of data presented, more information is required to understand the structure of the molecule.

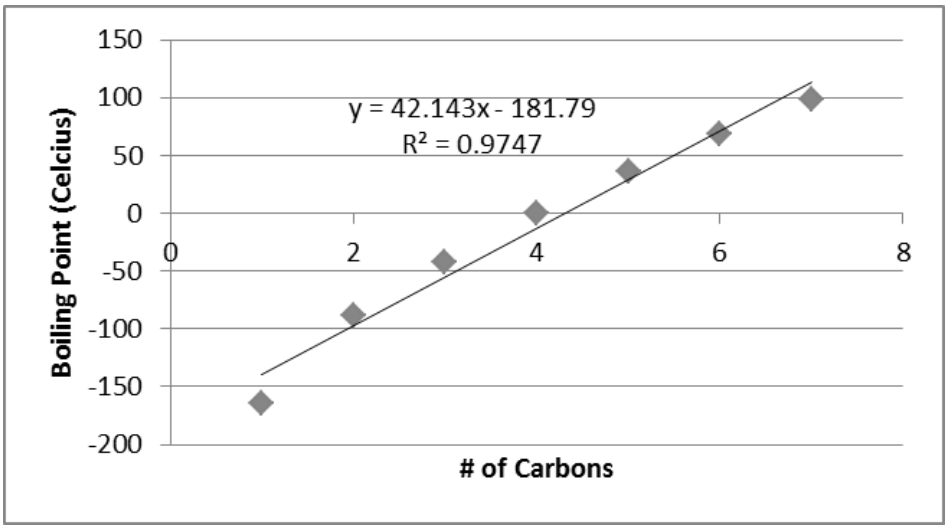


Figure 2-2: Example of constitutional descriptor correlation.

2.1.2. 1D Descriptor

If fragments (e.g. substructural fragments or functional groups) of a molecule are used for molecular representation then 1D descriptors are obtained, as only one dimension is required to depict the type of substitution or fragments present. 1D descriptors can be useful to quickly scan the chemical space for candidates based on some established similarity criteria with respect to a reference molecule. These have been used to filter out structures in the early stages of drug design. An example of such a descriptor is carbon count.

2.1.3. 2D Descriptor

2D descriptors are calculated from 2D representation of molecular structure, taking into account the types of atoms, their number and their connection pattern with each other. Examples of 2D descriptors include chiral center count, providing the number of chiral centers, and rotatable bonds count, the number of bonds capable of rotation (Roy et al., 2015a). The descriptors derived from the graphical representation of molecules are categorized under 2D

descriptors. In the graphical representation, the molecule is referred to as a molecular graph. A molecular graph, G, consists of atoms which form the vertices of the graph and the covalent bonds which form the edges in the graph. Thus atoms that have at least one bond between them are connected by an edge. The various fragments that can be obtained from G can be represented as sub-graphs. The sub-graphs thus consist of subset of edges belonging to the edge set, E, and subset of vertices belonging to the vertex set, V. The descriptors obtained from the graphical representation are termed as topological indices (TIs). These are the most widely used descriptors in model development and hence in computer-aided molecular design. TIs are very convenient to use because they can be easily computed and analyzed. As isomorphic graphs have identical values for a given TI, TIs are graph invariants i.e. their values are independent of labeling of the molecular graph. In the following subsections some details on the most widely used TIs both in modeling and CAMD algorithms are being provided.

2.1.3.1.Connectivity Indices

The connectivity index (CI) was introduced by Randic (1975) and since then has been modified into different forms. The connectivity index is usually denoted by the symbol χ (X in Dragon 6). One usually finds 2 superscripts and one subscript assigned to the CI (Sabljic, 1990). The superscript on the left is a non-negative number and reflects the order of the CI and the superscript on the right, *v,* denotes that a valence delta value has been utilized for calculation. The CIs are in general divided into 4 sub-classes: path (denoted by subscript *p*), cluster (denoted by subscript *c*), path/cluster (denoted by subscript *pc*) and chain (denoted by subscript *ch*). These subclasses are describing the substructural units considered while calculating the CIs. For example, the path-based CI is calculated using paths. A path is a sequence of edges from one vertex to another end vertex, ensuring that the edges are not repeated while traversing this

sequence of edges. In most general cases, the subscript *p* is removed and path type is considered as a default. CIs are usually calculated from hydrogen suppressed graphs. In such molecular graphs, the hydrogen atoms are not considered. Consider the example of the mth order valence connectivity index ${}^{m}\chi_k^{v}$. It is defined as follows (Mu and He, 2011):

$$ {}^{m}\chi_k^{v} = \sum_{j=1}^{n_m} \left(\prod_{i=1}^{m+1} \delta_i^{v} \right)_j^{-0.5} \tag{2.1} $$

$$ \delta_i^{v} = (Z_i^{v} - H_i) / (Z_i - Z_i^{v} - 1) \tag{2.2} $$

$$ \delta_i = (Z_i^{v} - H_i) \tag{2.3} $$

Where, k denotes a contiguous path type fragment, which is divided into paths (*p)*, clusters (*c*), paths/clusters (*pc*) and chains (*ch*). n_m is the number of relevant path type fragments. δ_i^{v} is the valence delta value calculated as shown in Eq. (2.2). In Eq. (2.2), Z_i^{v} is the number of valence electrons, H_i is the number of hydrogen atoms connected to atom *i*, Z_i is the number of electrons of atom *i*. If we calculate the m^{th} order connectivity index ${}^{m}\chi_k$, then δ_i will be substituted instead of δ_i^{v} in Eq. (2.1) to obtain ${}^{m}\chi_k$. δ_i is the degree of the atom *i* obtained from the hydrogen supressed graph. Hence H_i is subtracted from Z_i^{v} in Eq. (2.3). Consider the 3-methyl hexane molecule shown in Figure 2-3. The degree values, δ_i, of each of the atoms have also been displayed.

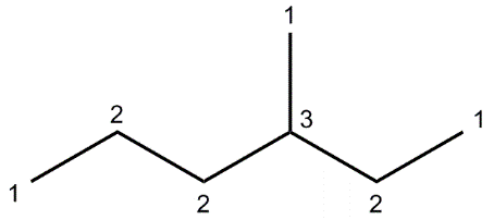


Figure 2-3: 3-methyl hexane molecule

The ${}^{1}\chi$ value of the 3-methyl hexane molecule can be calculated as:

${}^{1}\chi = (1x2)^{-0.5}+(2x2)^{-0.5}+(2x3)^{-0.5}+(3x1)^{-0.5}+(3x2)^{-0.5}+(2x1)^{-0.5} = 3.3081$

2.1.4. 3D Descriptors

3D Descriptors, also known as geometrical descriptors, are calculated by representing the molecule in the 3D space. Generally, geometrical descriptors are calculated either by utilizing optimized molecular geometry obtained by computational chemistry methods or from crystallographic coordinates (Cronin, 2010). 3D descriptors obtained by utilizing geometric distances between atoms constitute a special subset known as topographic indices. The geometrical representation is used to capture the relative positions of the atoms in 3D space. Thus, geometrical descriptors usually offer more information and discrimination power for similar molecular structures and molecule conformations than topological descriptors. This power to discriminate, however, is computationally costly in cases of modelling and molecular structure development.

Since 3D descriptors require geometry optimization, a high computational expense is incurred as this involves calculation of optimal geometry, conformation analysis, Gibbs energy of structure, etc. Additionally, if several conformations of the molecules exist, complexities can

become much higher. Also, it may happen that the active conformations of the chemicals being studied for biological applications are unknown. An overview of the various types of geometric descriptors is provided in Figure 2-4 (Herring, 2014).

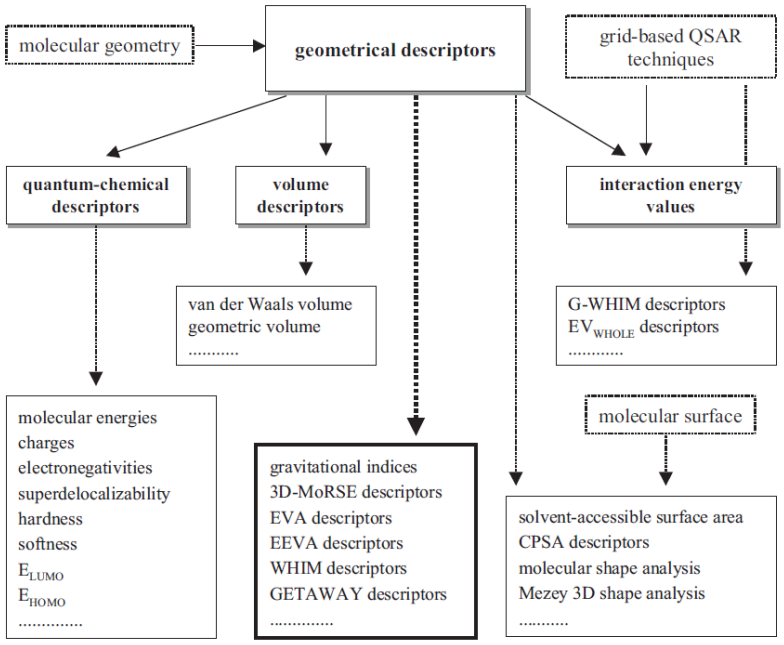


Figure 2-4: Overview of geometrical descriptors (Herring, 2014)

Another concern with 3D descriptors is that there is not a common understanding of the required degree of detail of molecular structure to calculate 3D descriptors reliably (Hechinger et al., 2012). The computational methods utilized for calculation can be anywhere from molecular mechanics to quantum methods which are more rigorous and complex. Due to these reasons, simpler descriptors like TIs (Topological Indices) are usually preferred for the screening of large databases of molecules and CAMD applications. On the other hand, searching for relationships between molecular structures and complex properties, such as biological activities, sometimes consider the use of 3D descriptors.

2.1.5. 4D-7D Descriptors

These descriptors have been utilized the least for CAMD applications as the computational cost incurred is even higher than 3D descriptors. These descriptors consider a variety of factors as dimensions. These include the orientation and the solvation function (Roy et al., 2015a). Such descriptors are beneficial in capturing ligand and receptor interactions.

Although the descriptors mentioned so far have been categorized within whole number dimensions, there are descriptors that are difficult to categorize. For example, between 2D and 3D descriptors, 2.5D descriptors exist as intermediates that tend to incorporate some aspects of the geometrical information contained in a 3D structure that were ignored by a 2D description of the molecule (Doucet and Panaye, 2010).

2.2.Model Development

Molecular modeling encompasses all of the techniques and tools useful for modeling the motions and interaction of molecules. These techniques are used in the fields of computational chemistry, drug design, computational biology, materials science, and now many engineering fields for studying molecular systems ranging from single small molecules in the gas phase to large biological molecules (e.g. receptor ligand complexes) and material assemblies. There are many approaches available for the treatment of molecular structures ranging from modeling atoms as the smallest individual unit (in the molecular mechanics approach) to explicitly modeling the electrons in each atom (in the quantum chemistry approach). The information gained from these techniques is useful in the development of three-dimensional descriptors, which have applications in a wide variety of structure-activity (property) correlations.

2.2.1. Molecular Mechanics

Molecular mechanics stands for using classical mechanics to describe the motions of atoms and molecules. For such approaches, the atoms are treated as point charges with the nucleus and associated electrons. The structure potential energy, that indicates the likelihood of the structural occurrence, is calculated via the means of a force-field. This force-field, also called potential function, uses different terms to summarize the potential energy associated with the structure. The molecule is described using internal and external coordinates of each atom in the structure. The external coordinates are based on Cartesian coordinate system, whether the internal coordinates use the inherent nature of these systems by utilizing bond-lengths, bond-angles, and torsional angles. Additionally, force-field calculation contains non-bonded interaction terms between the atoms as showed in Eq. (2.4)

$$E_{total} = E_{bonds} + E_{angle} + E_{dihedral} + E_{non-bonded} \tag{2.4}$$

$$E_{non-bonded} = E_{electrostatic} + E_{van\ der\ Waals} \tag{2.5}$$

As shown in Eq. (2.5), the non-bonded interactions represent the addition of electrostatic and van der Waals forces. (Leach, 2001)

2.2.2. Quantum Chemical Methods

Quantum mechanics (QM) lay out molecules in terms of interactions between nuclei and electrons. The molecular geometry is determined by determining a minimum energy arrangement of nuclei in a molecule or set of molecules. This process, involving high computational expense, has been made much more reasonable through a series of approximations. These approximations

are generally applied upon the original formulation based on the Schrodinger equation shown in Eq. (2.6).

$$\hat{H}\Psi = \mathrm{E}\Psi \tag{2.6}$$

Where

Ψ = many-electron wave function

$\hat{\mathrm{H}}$ = Hamiltonian operator (Hamiltonian)

$\hat{\mathrm{H}}$ can also be represented as in Eq. (2.7).

$$\hat{H} = -\frac{1}{2}\sum_{i}^{electrons}\nabla_i^2 - \frac{1}{2}\sum_{A}^{nuclei}\frac{1}{M_A}\nabla_A^2 - \sum_{i}^{electrons}\sum_{A}^{nuclei}\frac{Z_A}{r_{iA}} + \sum_{i<}^{electons}\sum_{j}\frac{1}{r_{ij}} + \sum_{A}^{nuclei}\sum_{B}\frac{Z_A Z_B}{R_{AB}} \tag{2.7}$$

Where

Z = the nuclear charge

M_A = the ratio of mass of nucleus A to the mass of an electron

R_{AB} = the distance between nuclei A and B

r_{ij} = the distance between electrons I and j

r_{iA} = the distance between electron I and nucleus A.

This equation is too tedious to be exactly solved for even a simple two-electron system, e.g. helium atom or hydrogen molecule, without introducing some sort of approximation. One such method is known as the Born-Oppenheimer Approximation, which assumes that the motion of the electrons is much faster than that of the nuclei (Born and Huang, 1988). This allows decoupling the two and producing the "electronic" Schrodinger equation. The electronic Schrodinger equation is still intractable after this simplification, and more approximations are required. The Hartree-Fock approximation is based on independent movement of the electrons, meaning, the electrons move independently of each other (Slater, 1930). This turns total wave function in the form of a single determinant, also known as a Slater determinant. This leads to a set of the Hartree-Fock equations that involve the coordinates of a single electron. At this point, although numerical solution to these equations is possible, further approximations have to be introduced to transform them into a set of computationally applicable algebraic equations. The linear combination of atomic orbitals (LCAO) was the next step to develop a better tractable representation of a molecule through the quantum chemical formalism (Clark and Koch, 1999). When the Hartree-Fock and LCAO approximations are applied to the electronic Schrodinger equation, the Roothaan-Hall equations are derived (Roothaan, 1951). The solutions of the Roothaan-Hall equations results in Hartree-Fock models, also known as *Ab Initio* ("from the beginning") models. These models help evaluate first and second derivatives of energy, making both geometry optimization and determination of vibrational frequencies possible.

A point to note, however, is that overestimation of electron-electron repulsion energies is a common phenomenon in the solutions generated using HF models. This is because pair-wise electron interactions are not considered directly in the assumptions for the optimization, causing overlapping of electron positions. This situation can be avoided using electron correlation that

helps accounting for coupling of electron motions, and lessens the electron-electron repulsion energy. One such approach is the density functional theory (DFT) model (Becke, 1988). This model tackles the situation by introducing an approximate correlation term without causing higher computational cost.

For larger problems, semi-empirical models are generally used using more simplifications to the HF models. In these simplifications only valance electrons are considered rather than core electrons. This approximation assumes that atomic orbital on different atomic centers do not overlap. This approximation is known as Neglect of Diatomic Differential Overlap (NDDO) approximation (Pople et al., 1967). For further simplification, additional approximations, and empirical parameters can be introduced. Some common semi-empirical models of such simplification are Austin Model 1(AM1) (Dewar et al., 1985), and Parameterized Model 3(PM3) (Stewart, 1989).

2.2.3. QSPR/QSAR

The aim of a structure activity relationship (SAR) is to develop a mapping between the structural information of a group of compounds and a desired activity/property. Corwin Hansch was a notable pioneer of this field. His work expanded the boundaries of formulating such relationships. Hansch et al. (1962) initially observed that the partition coefficient of various compounds, in combination with other cheminformatics parameters, can be utilized to characterize their relative biological activity. His observation led him to believe that SAR's should not be limited to certain independent variables and fits, and paved the way for a successful merging between the development of these models with various mathematical/statistical techniques (Hansch, 1969). To further the improvements of the SAR

models, this trend has continued to grow and include modern day computational approaches such as pattern recognition, molecular modeling, artificial intelligence, and machine learning. Another key turning point in the development of SAR's was proposed by Kier et al. (1975) by introducing the molecular connectivity index that shows to have strong correlations to physicochemical properties (Hall et al., 1975) as well as biological activities (Kier and Murray, 1975). Thus a separate genre of developing and utilizing new molecular descriptors began, paving way for developing numerous techniques aiming to differentiate molecular structures combining mathematical invariants and formerly used physico-chemical properties. Regardless of the steps of developing and utilizing these models, the process can be divided into three distinct phases: 1) calculating molecular descriptors for structures in the training set 2) choosing the most informative molecular descriptors and 3) utilizing the chosen descriptors as independent variables to create a mapping into property/activity space. This approach is visualized in Figure 2-5 (Dudek et al., 2006).

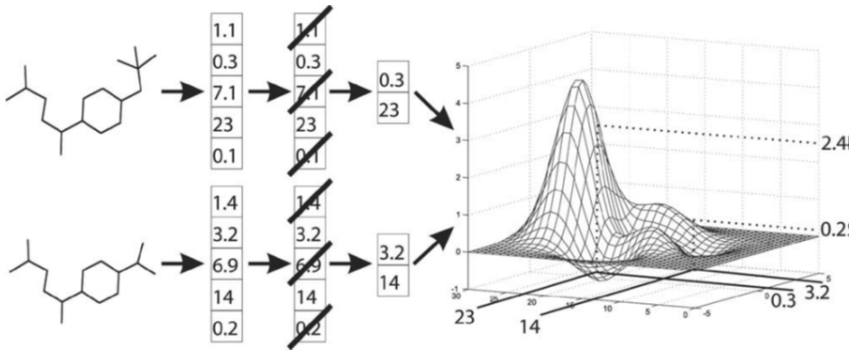


Figure 2-5: Visualization of QSAR approach.

2.3.Variable Selection

There are two ways of developing automated selection of descriptor variables for use in a property model (Guyon and Elisseeff, 2003). One technique, the feature selection approach, involves the identification of an optimal subset of descriptors based on random/guided selection of meaningful descriptors and ranking of formed models based on maximizing/minimizing objective/cost function, which is, in most SAR cases, an error function. The other, known as filtering, does not develop a subset to construct models in the selection process as features are evaluated using other criteria. This is a necessary step in developing of most structure activity relationships as a large number of descriptors are commercially or academically available for correlation with the result of interest.

2.3.1. Genetic Algorithm

Genetic Algorithm (Siedlecki and Sklansky, 1988), stands out for this approach and is an efficient method for sampling large descriptor spaces. Being categorized as stochastic programming, genetic algorithm mimics the process of natural evolution whereby a population is guided towards a higher degree of fitness, as often measured by the error of the model generated, through operations of mutation and crossover. Each member of the population is represented by a chromosome, within which each position usually corresponds to the absence or presence of a specific variable through the binary notation. Individual chromosomes with an increased measure of fitness, typically measured by the prediction capabilities of the model resulting from the descriptors represented within the chromosome, are selected for the conventional operations of crossover and mutation. Mutation typically involved the change of binary variables within the chromosome to either a 0 or 1, the opposite of its initial state; and crossover involves the

selection of two chromosomes which are cut and recombined at one (single-point crossover) or more points. However, the success of a GA relies on the careful tuning of several probability parameters such that the solution space can be effectively explored and early convergence to a homogenous population, occupying a local minimum, is not met. Genetic Algorithm is widely employed in developing QSPR/QSAR models. As Whitley (1994) describes, such an evolutionary algorithm is efficient as a function optimizer and its applications are very diverse. Houck et al. (1996) has presented a basic idea of GA. In their work, they have proposed these basic steps that can be used as a guideline for designing a GA process for function optimization problems.

- Supply an initial population P_0 of N individuals with respective function values and constraints, if any.
- i = 1
- $P_i^{'}$ = selection_function(P_{i-1})
- P_i = reproduction_function($P_i^{'}$)
- evaluate (P_i)
- i =i+1
- Repeat 3rd step until termination condition met
- Print best solution achieved

Here, for a given GA, the *selection_function()* represents the Roulette wheel developed by Holland (1975). For this operation, the N individuals pass through the fitness evaluation that is performed by a fitness function. The individual with best fitness then replaces the individuals with least fitness. The *reproduction_function()* generally consists of both the crossover and mutation operators, also known as genetic operators. These steps generate new population that may produce better individuals. The reproduced population then goes through the *evaluate()*

function, which determines the acceptance of the population. This process is repeated until a better solution is not produced anymore or if any termination condition is met. Finally, the best possible result is printed. A flowchart of genetic algorithm is presented in Figure 2-6. Reeves (1995) has applied GA in case of flowshop sequencing problem. On the other hand, Leardi (2001) discussed its applications in molecular design and modeling. He discussed a variety of cases of property model development which benefited from implementing genetic algorithm.

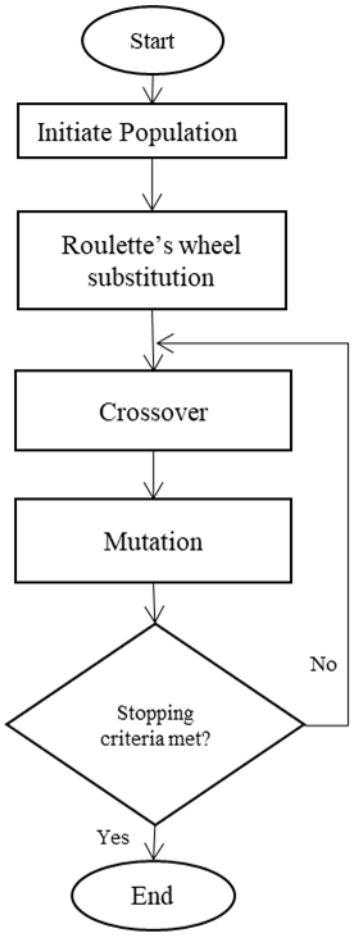


Figure 2-6: Flowchart of Genetic Algorithm

2.3.1.1. Selection Strategies

A Roulette wheel is considered to be the most commonly used selection operation. However, there also exists Tournament based selection strategy (Zhong et al., 2005). The followings will be discussing these selection strategies in details.

2.3.1.1.1. Proportional Roulette Wheel

In case of a proportional roulette wheel, the individuals are selected proportionally based on their fitness values. The name was derived from the instance that a pointer will have higher possibility of choosing an individual with the largest fitness. As it can be seen in Figure 2-7, in case of a proportional roulette wheel, the higher the fitness of an individual, the higher is the probability of it being selected for further operations. The presented example proposes a selection method that only selects individuals with highest fitness in every turn, thus preserving the ultimate requirement of survival of the fittest (Razali and Geraghty, 2011).

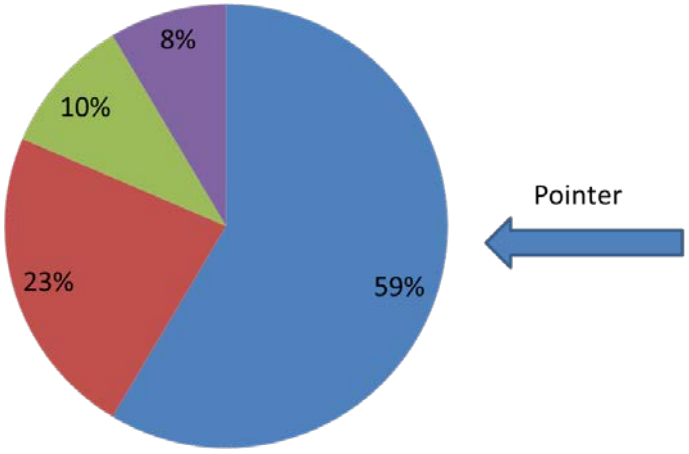


Figure 2-7: Proportional roulette wheel strategy

Although this process has the advantage of preserving diversity in the population by not discarding the individuals with poorest fitness, some major deficiencies cannot be denied. One major problem with this strategy is that it quickly introduces bias of outstanding individuals in the beginning of the search. This can cause premature convergence during the initial rounds, resulting in lack of diversity in further generations. This premature convergence is a hindrance towards achieving diversity in gene pool, resulting in a poor optimization operation. Additionally, a problem occurs in this strategy when a large number of individuals having very similar fitness values are generated in the initial steps. This occurrence can make this strategy fruitless as the algorithm cannot move forward towards a better solution.

2.3.1.1.2. Rank-based Roulette Wheel

Since proportional roulette wheel develops complications during minimization, a solution was advanced through rank-based roulette wheel algorithm (Goldberg and Deb, 1991). This algorithm selects the individuals based on their fitness ranks rather than fitness values, a process also known as elitism. This method first performs a sorting operation over the generated population. As the selection scale shifts the strategy from fitness value dominance to individual fitness ranking, the selection strategy is forced to consider uniform scaling rather than being influenced by only outstanding individuals of the population (Figure 2-8).

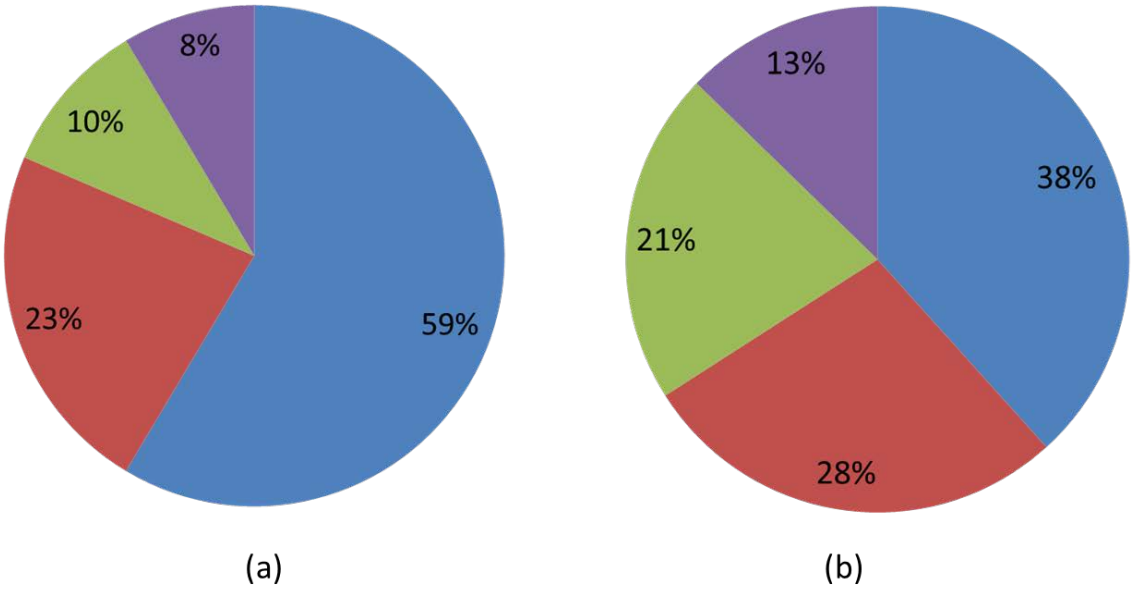


Figure 2-8: Probability of individual selection using (a) proportional and (b) rank-based roulette wheel strategies

The steps in general involve mapping the individuals based on their ranked probability. This mapping can be both linear and non-linear, and both can be used for the same purpose in different types of data matrix. In case of linear mapping, a selective pressure, SP, is introduced to control fitness bias. For a minimization operation, the mapping is done based on Eq (2.8), for $2.0 \geq SP \geq 1.0$. Hence, for the best individual, the expected sampling rate is SP, and for the worst, it is 2-SP. For a minimization problem, pos = 1 for least fit individual, where pos = n fop the fittest individual.

$$Rank(pos) = 2 - SP + (2(SP - 1)\left(\frac{pos - 1}{n - 1}\right)) \quad \text{Eq (2.8)}$$

Compared to the linear process, the non-linear method is less popular as it is based on the exponential function. For instance, in non-linear approach, the best individual is given a ranking

of 1, the second one is given SP, the third being SP^2, and so on to the last one, SP^{N-1}. This method is less popular due to increased need of convergence number to achieve the same goal compared to linear ranking process (Hancock, 1994). However, in case of situations where it is more important to preserve genetic diversity as much possible, this method can play a significant role.

2.3.1.1.3. Tournament Selection

Tournament selection method is quite popular for its efficiency and simplicity of computational implementation (Goldberg and Deb, 1991). The process is based on selecting individuals randomly and putting them into tournaments with each other. The individuals with highest fitness in the group gets selected to move forward in the convergence (Figure 2-9). This is a unique process as these random selection-based tournaments give chance to individuals with different fitness values. The tournament selection process is highly advantageous in the cases of computational efficiency, premature convergence, and preserving diversity. As the selection method doesn't require any probability or ranking calculations, some added burden of roulette wheel calculations get avoided in an effective manner.

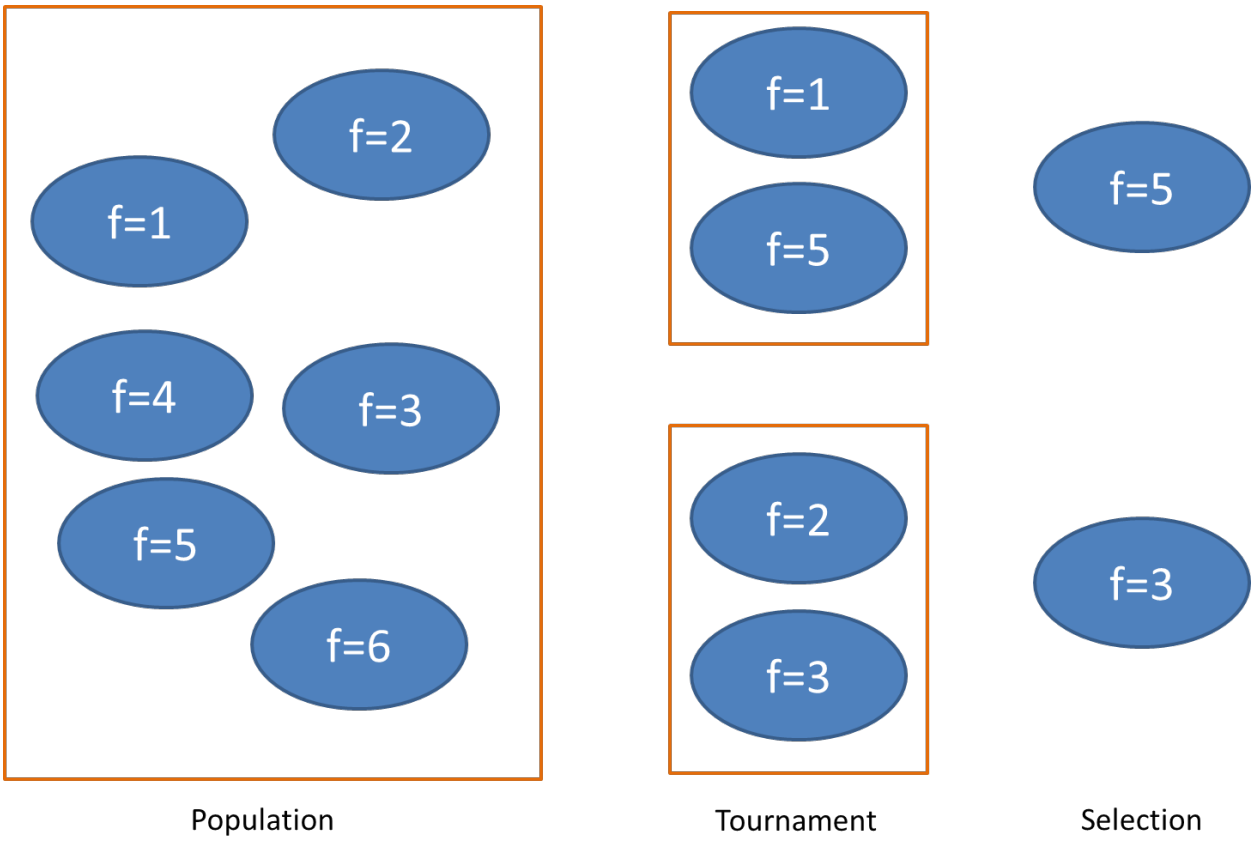


Figure 2-9: Tournament selection strategy

The only drawback, however, is the size of the tournament. Although ideal situation dictates using a binary tournament (2 individuals in each tournament), that is always not practical while using a high number of individuals in the population. And the higher the tournament size (number of individuals in a tournament), the higher the possibility of losing gene diversity. The challenge, in this case, is to find a balance between the tournament size and the population to operate the selection operation on.

2.3.1.2.Crossover Strategies

During last few decades, numerous types of crossover strategies have been developed (Starkweather et al., 1991). The general aim of the crossover operation is to develop new members in the population and mixing the chromosomes of the present individuals (also called reproduction).

2.3.1.2.1. Order Crossover

Davis (1985) developed the order based crossover operator to develop offspring individuals inheriting elements from parents in the same order in preceding individuals. The operation generates new individuals using single/double swap points. In case of the single swap point, one point is used to identify swapping location (Figure 2-10 (a)). The offspring resulting from such operation retain the same chromosome order of the parents, with the change at the swapping point. In the case of the double point operation, however, two points are selected to choose a region to swap (Figure 2-10 (b)).

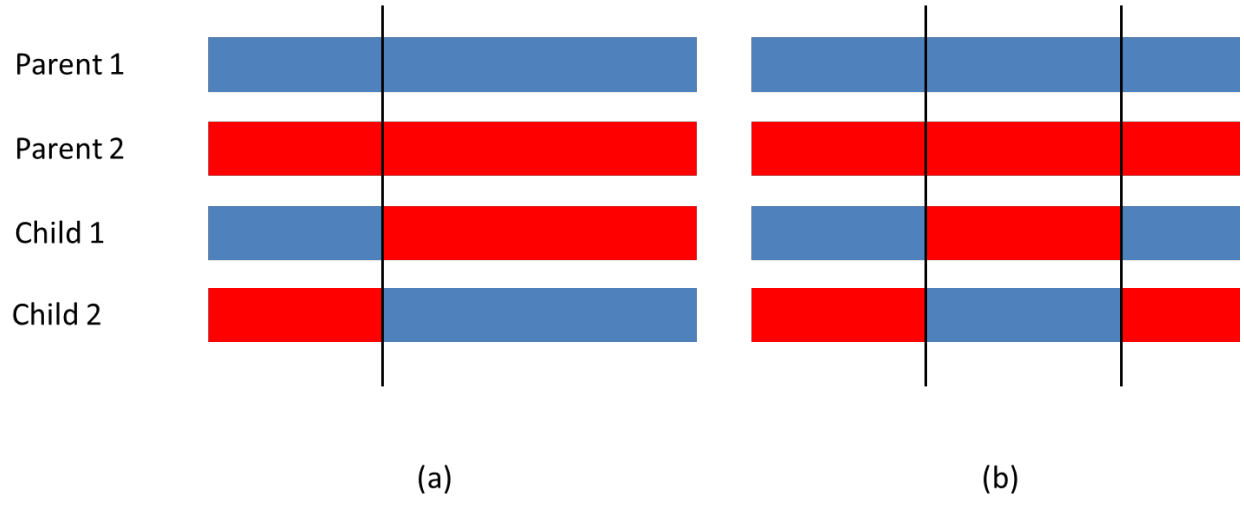


Figure 2-10: Order crossover for (a) single point and (b) double point operation

2.3.1.2.2. Order 2 Crossover

Syswerda (1990) also developed an order based crossover operation with some modifications. This crossover is also called as scatter point crossover. In this operation, several key points are chosen to be swapped to generate offsprings (Figure 2-11).

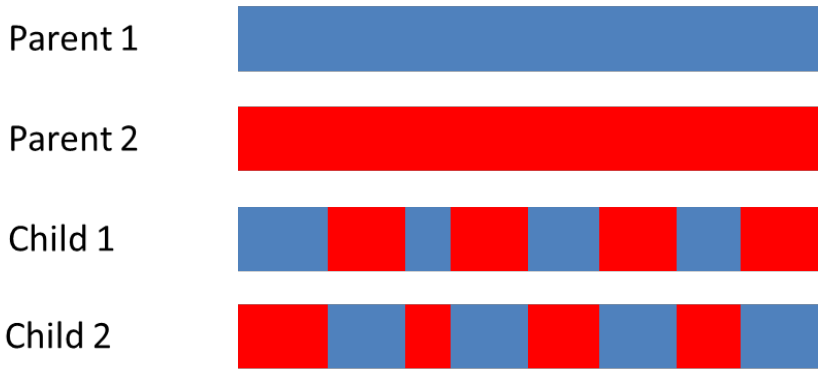


Figure 2-11: Order 2 Crossover (Scatter point crossover)

This operation is highly valuable in cases where a good number of candidates are present in the population. In such situations, using single/double point swaps can end up developing lesser efficient models. However, using scatter point operation in that case can help achieving optimization in shorter number of convergences.

2.3.1.2.3. Cycle Crossover

Oliver et al. (1987) developed this crossover strategy to develop a more efficient solution for the travelling salesman problem (TSP). For this operation, a parental sequence and a cycle starting point has to be selected. Both are selected in a random manner. In this operation, like the single point crossover operation, a swap point is chosen.

The offspring are developed adding one part of the chromosomes in direct order of one parents, and the other part using the cyclic reverse order of the another parent using the cycle starting point (Figure 2-12). However, in present algorithms, without using a cycle point, a more efficient approach is to simply reverse the other set of chromosomes. This is done to make the process computationally efficient. This approach is very useful in case of developing individuals where the sequence of the chromosomes matter (e.g. TSP optimization). In case of predictive model development, such approach rarely has any effect.

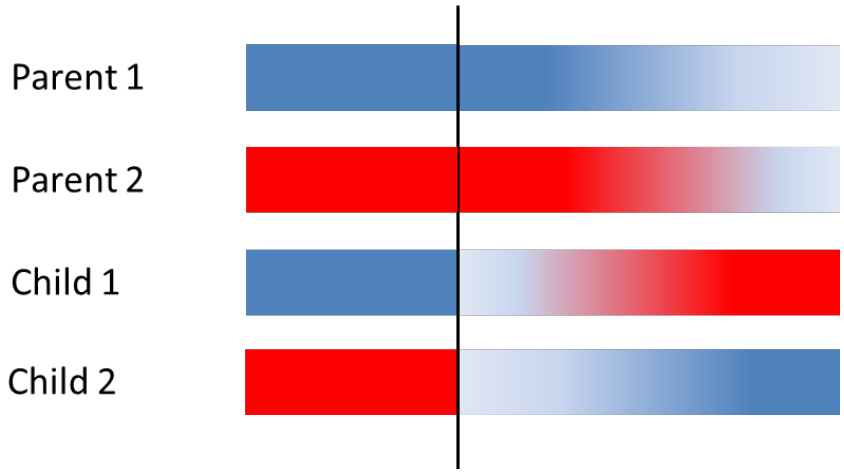


Figure 2-12: Cycle Crossover operation

2.3.1.2.4. Position based Crossover

Also developed by Syswerda (1990), position based crossover operation develops only one new individual using two parent individuals. This is very similar to the scatter swap algorithm, given that the only difference is to determine which of the parents will donate at what points. The example (Figure 2-13), however, makes the operation look almost similar to a multiple double point crossover operation. In practice, such assumption cannot be denied as the

algorithm developed for such strategy resembles the occurrence of multiple subset-based swapping of the chromosomes.

Figure 2-13: position based crossover strategies

This strategy is not as much utilized as the parent to child ratio mandates more number of crossover operations to generate a certain number of offspring. However, in some genetic algorithm approaches, this operation is used to develop new members in the population to replace the worst ranking members (Whitley, 1975).

2.3.1.3.Mutation Strategies

A noteworthy overview, containing examples of different mutation operation strategies, has been provided in the work of Sivanandam and Deepa (2007). In general, the aim of a mutation operation is to randomly select one or more chromosomes in the individuals and alter it. In the basic form of binary mutation, where values of chromosomes are either 0 or 1, the operation's aim is to change one to another (Figure 2-14). In practical use, however, this operation can be completed using various approaches. This section discusses some of such approaches.

Figure 2-14: Mutation operation in binary GA optimization

2.3.1.3.1. Insert Mutation

This is generally used in cases of permutation encoding. The operation begins with the choice of two chromosome locations at random. It then moves the second location value to follow the first one, and rest of the chromosomes get shifted to accommodate accordingly. In this situation, however, no new value is introduced in the subset.

2.3.1.3.2. Inversion Mutation

In case of inversion mutation, however, two random chromosome locations are selected. After that, the substring between these locations gets inverted.

2.3.1.3.3. Swap Mutation

In the swap mutation, two chromosome locations are selected and the values of the chosen locations get swapped.

2.3.1.3.4. Scramble Mutation

In this mutation operation, a subset of genes is selected randomly. After that, the subset is rearranged in those positions in a random manner.

2.3.1.3.5. Reversing Mutation

Reversing mutation is commonly used for binary encoded chromosomes. In this method, a location is chosen, and the bit next to it is reversed to produce the child.

2.3.1.3.6. Creep Mutation

In case of creep mutation, a random gene needs to be chosen. After that, the value of the chosen gene is changed with a random value between user defined upper and lower value. This is the most common approach used for predictive QSAR model development approaches.

2.3.1.3.7. Uniform Mutation

In general, uniform mutation is similar to creep mutation, with the difference being that a uniform random value is chosen, making it usable for integer-based operations.

2.3.2. Decision Tree

Decision tree is an algorithm that represents possible choices and their probable outcomes. This unique algorithm maps the best possible path to follow to reach the desirable result. In practice, decision tree is one of the most commonly used algorithms for data mining, data prescreening, classification and regression analysis (Gupta et al., 2017). Some advantages of this algorithm include easy visualization and interpretation with no requirement of data preparation, and removal of blank values. Additionally, the algorithm can also handle both categorical and numerical data, multiple output problems. Also, this algorithm can be simply explained by Boolean logic of true or false (yes or no). The earliest form of decision tree was introduced by North (1968), where it was introduced for solving an anniversary problem (Figure 2-15).

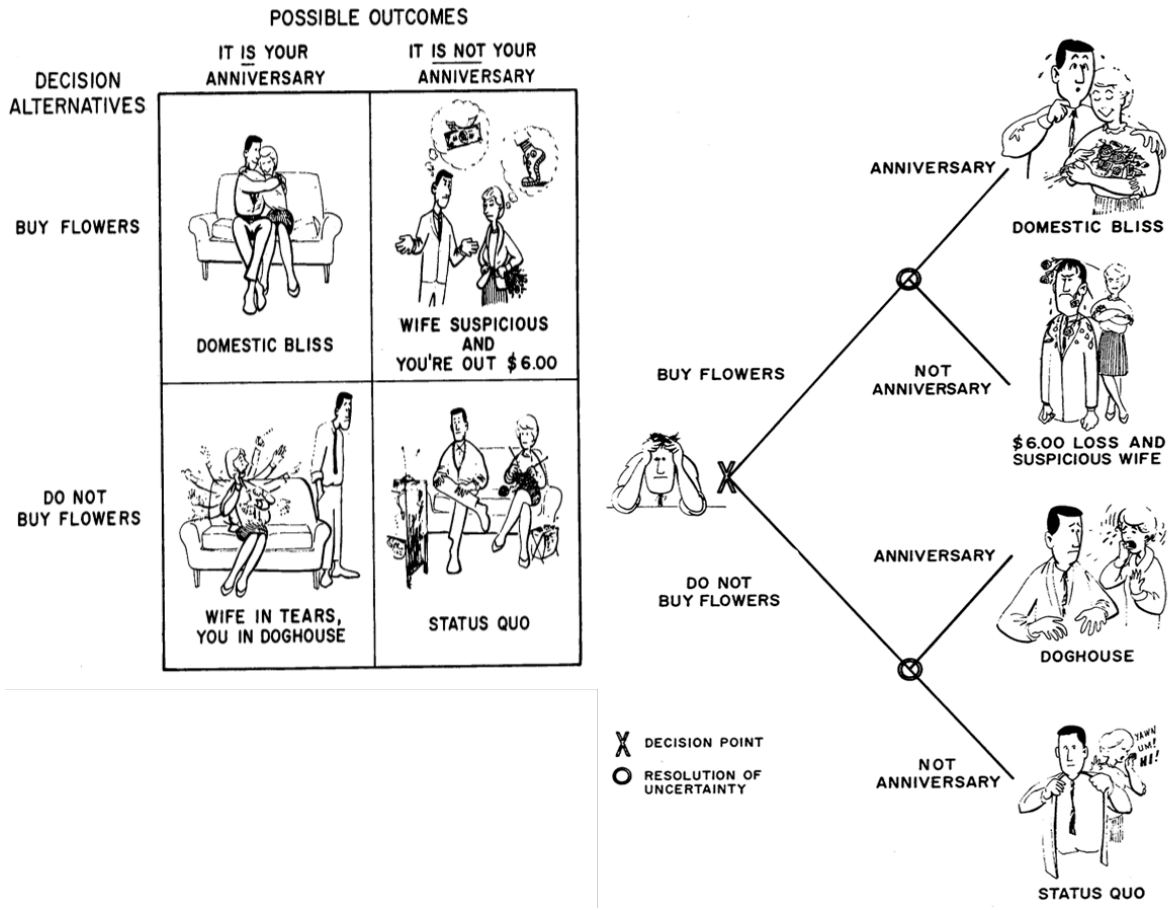


Figure 2-15: Introduction of Decision Tree, North (1968)

However, in present day, various types of decision tree approaches have been developed (e.g. ID3, CART, Random Forest, CHAD, C4.5, Boosted Forest). In this section, the discussion will be limited to the most commonly used ones.

2.3.2.1. Iterative Dichotomiser 3 (ID3)

Developed by Quinlan (1986), the algorithm mostly focuses on classification operations. ID3 approaches the classification problem using a greedy search. Although this algorithm is strongly based on developing smallest decision trees possible, if needed, it ends up developing

larger decision trees for larger dataset analysis. ID3 only analyzed categorical data. However, the algorithm is not much effective against data noise and duplicates. This is due to the greedy approach used in the initial phase.

Some advantages of the ID3 algorithm includes building the fastest and shortest trees possible for a given problem. Moreover, the algorithm searches the whole dataset to develop the overall tree. The training data can be used to develop comprehensible prediction rules. Finally, the calculation time required to develop an ID3 operation is a linear function O(n) of the product of characteristic number and node number.

However, this algorithm comes with some limitations. First of all, due to being a greedy approach, overfitting is a common problem in cases of small datasets. Also, being an O(n) algorithm, classifying data of continuous nature can prove computationally expensive. This is also because the algorithm analyzed only one attribute at an instant. In cases of higher number of input values, the algorithm ends up showing preferences towards features with higher number of values (Gupta, 2017).

2.3.2.2. C4.5 algorithm

C4.5, also developed by Quinlan (1987), is an extension of ID3 algorithm. This algorithm is often referred to as a statistical classifier. The improvement includes ability of handling both continuous and discreet attributes, missing values, and pruning trees after construction. By design, C4.5 is a tree pruning process. For its operation, C4.5 follows ID3 approach in case of categorical attributes, making sure continuous attributes generate binary splits. Next, attributes with highest gain ratios are selected. Gain ratio helps rank the classification based on the

diversity of the data in the classes rather than the count of data in a class. These steps keep repeating until the stopping criterion is met. Use of gain ration rather than number of inputs makes C4.5 less susceptible to data population bias.

The algorithm is very easy to implement and builds models that are easily comprehensible. An added feature is that it can deal with both categorical and continuous values with ease. Also, due to added feature of gain ratio calculation, the algorithm is able to deal with noise and missing value attributes. However, the algorithm tends to not work well in case of small training sets. Also, a small variation in data leads to different trees being developed.

2.3.2.3. Classification and Regression Trees (CART)

CART, introduced by Breiman (1984), is equipped to build both classification and regression trees. In case of developing classification tree, CART uses binary splitting using Gini index (Lerman and Yitzhaki, 1984). In its simple form, Gini index can be described as a class ranking strategy, derived from products of coefficient of variation of the considered variables, variable's linear correlation coefficients with ranks, and a constant equal to $1/\sqrt{3}$ (Milanovic, 1997). CART also has a regression feature that can help develop predictive models for a dependent variable given a number of predictor variables. It consumes an average speed, and can deal with both continuous and nominal attribute data.

CART is good with the missing values and combinations of continuous/discreet variables. It can automatically perform variable selection and form interactions between the variables. But it may also develop some unstable decision trees as it is non-parametric. Also, the split happens based on only one variable, which makes it difficult to use for large databases.

2.3.2.4. Random Forest

Random forest, also developed by Breiman (2001), is a collection of simple tree predictors. The simple trees are arranged such that each tree produces a different response based on different predictor value input. Like CART, random forest can also work for both classification and regression operations. Random forest is generally used as a tool to leverage the ability of multiple varied analyses, organization strategies, predictive modeling, machine learning, ranking, and deep data understanding. It can easily recognize outliers and anomalies in knowledgeable data. It is considered one of the most accurate tree-based learning algorithms available. It can perform classification with significant amount of focus in identifying the important predictor variables. The con, however, is that sometimes the presented classification of random forests may be difficult for human interpretation. Also, in cases of dealing with noisy datasets, overfitting is also a main concern.

2.3.2.5. Decision Tree in QSAR

In case of feature selection for QSAR studies, decision trees are allowed to keep forming branches and following paths that produce better model fitness. DT starts with an initial node (O), and keeps presenting options for further study. In the example, the options (A, B, C, D, …L) can be considered to be the descriptors (Figure 2-16). The numbers can be considered the R^2 value that will result if the user opts to choose the next descriptor. In other terms, these numbers are generally the outcome of selecting the nodes. The example needs to select a path to achieve highest R^2 value in the end. Here, path O-B-F-J-L gives maximum 0.413, which provides an idea about the path to follow to obtain the best result. But it also provides other options that can be taken into consideration (Path O-A-C-H). Izrailev and Agrafiotis (2001)

have presented through their work that such an algorithm can be successfully applied for regression problems. They have utilized Artificial Ant Colony System for this purpose. On a different application in field of property modeling, Andres and Hutter (2006) used DT algorithm to predict drug properties.

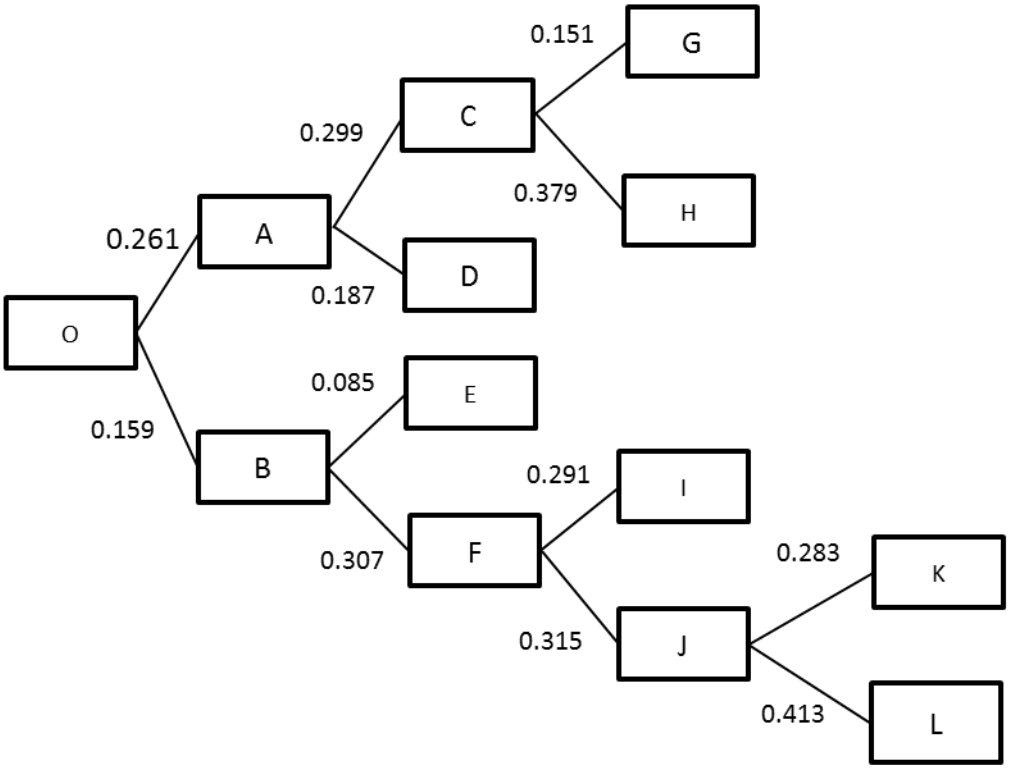


Figure 2-16: Decision Tree Algorithm

2.3.3. Hybrid Algorithms

According to Grosan and Abraham (2007), hybridization of evolutionary algorithms produces a different algorithm which can improve the algorithm performance and overall result. A hybrid algorithm is an algorithm that is generated combining two or more algorithms. The hybridization of the algorithms is performed such that two or more separate algorithms systematically perform

their desired tasks. In this process, the hybridization is meaningful only if the work-load of the algorithms is distributed in an efficient manner. This suggests level of hybridization is to be carefully determined for generating more efficient algorithm producing better results than the singular algorithms. In this respect, Loukas (2000) has developed an adaptive Neuro-Fuzzy inference algorithm to develop better QSAR models. The aim was to develop a QSAR model to calculate the apparent inhibition constant. This project used Gaussian Member Functions of Fuzzy system trained with hybrid back-propagation for property model development. Later, Goodarzi et al. (2009) developed a hybrid GA based Support Vector Regression (SVR) method. In their work, they used GA to optimize parameter values they received from SVR to improve the prediction efficiency of the initial model. On the other hand, Jun et al. (2010) developed two separate hybrid algorithms combining GA with Support Vector Machine learning and RBF Neural Networks respectively. Their generated models also provided better prediction in case of developing QSAR of aqueous solubility of polycyclic aromatic hydrocarbons. In the aforementioned works, hybrid algorithms have produced better QSAR models than the case where no hybridization was involved. These works also draw attention to the possibility of model generation requiring lesser computational time if the hybridization is performed properly.

2.3.4. Multi Gene Genetic Programming (MGGP)

In a feature selection approach using Genetic Algorithm, the chromosomes of the population generally contains position of variables (descriptors) that can be used to develop a property prediction model. The difference in MGGP is, instead of such variable positions, the chromosomes contain genes developed using GP tree (Gandomi and Alami, 2011).

Multigene symbolic regression is used to develop the genes (Searson et al., 2010). A symbolic regression begins with developing a GP tree, and the final model is the linear combination of these trees. The development of these initial genes requires identification of root node, functional nodes, and terminal nodes (Figure 2-17). The root nodes and functional nodes are populated by the arithmetic operators, geometric operators, Boolean logic functions, and other mathematical operators. The terminal nodes, on the other hand, contain the logical constants, numerical constants, and variables to populate the rest of the tree (Gandomi et al., 2010). For selection process, tournament approach is used.

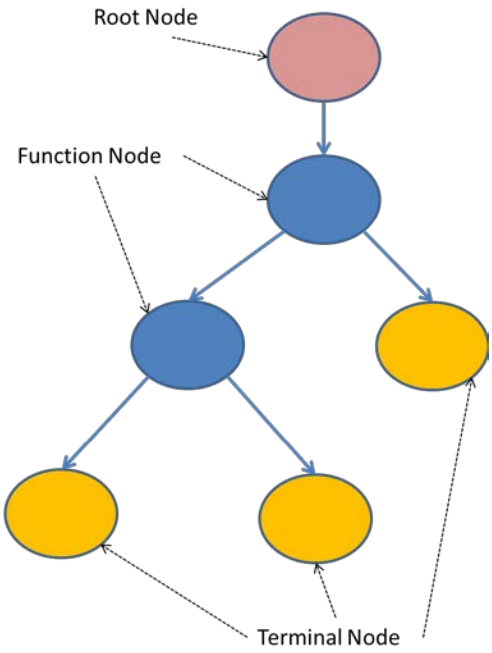


Figure 2-17: GP tree with symbolic regression

The crossover operation, in case of MGGP operation, has some significant difference than general GA operation. In MGGP approach, the crossover operation not only can swap a particular set of genes of the parents, but also can choose if swapping a particular section will be

more advantageous than swapping the whole gene (Figure 2-18). During this operation, however, precautions are required to prevent developing genes that may produce unacceptable mathematical tree. A way to prevent such situation is to perform the crossover at the same depth point for both parents, as can be noticed in Figure 2-18. This is chosen to be a safe process as performing crossover at the same depth has been seen to prevent development of bizarre symbolic trees in case of model development using higher gene depths. For such operation, two different crossover operations are performed. The high level crossover decides total swapping of the genes, whether low level crossover decides the subtree crossover, swapping of the particular section of the genes to create new individuals. However, a cancellation process is also maintained in case the newly formed individual ends up developing higher gene depth than allowed.

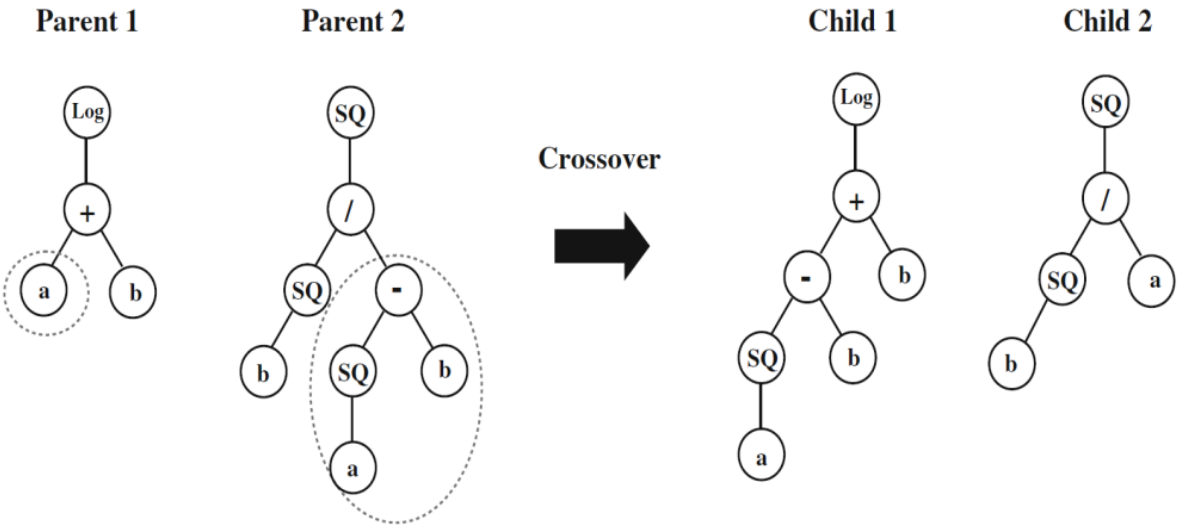


Figure 2-18: MGGP crossover operation (Gandomi et al., 2011)

The mutation operation can be applied in six different ways. First, it can be a sub tree mutation, changing a section of the tree. Second, the constant values can be mutated using an

additive Gaussian perturbation. Third, functional node can be substituted by another randomly chosen functional node. Fourth, a randomly chosen constant can be set to zero. Fifth, the variable nodes can be substituted by a different variable node. Finally, some random constant values can be changed to one. Figure 2-19 shows an example of mutation operation of third kind.

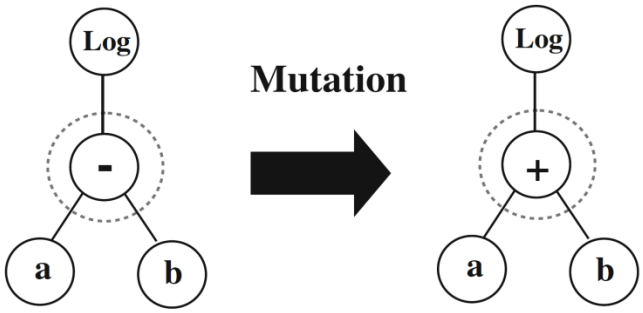


Figure 2-19: MGGP mutation operation (Gandomi et al., 2011)

Weight of each genes are calculated and evolutionary algorithms (EA) are used to tune the values of d0, d1, d2, d3, and d4 (Riolo and Worzel, 2003). In Figure 2-20, a small example of model development in both GA(a) and MGGP(b) are presented. The figure also shows that the development of final model from the individual developed in MGGP.

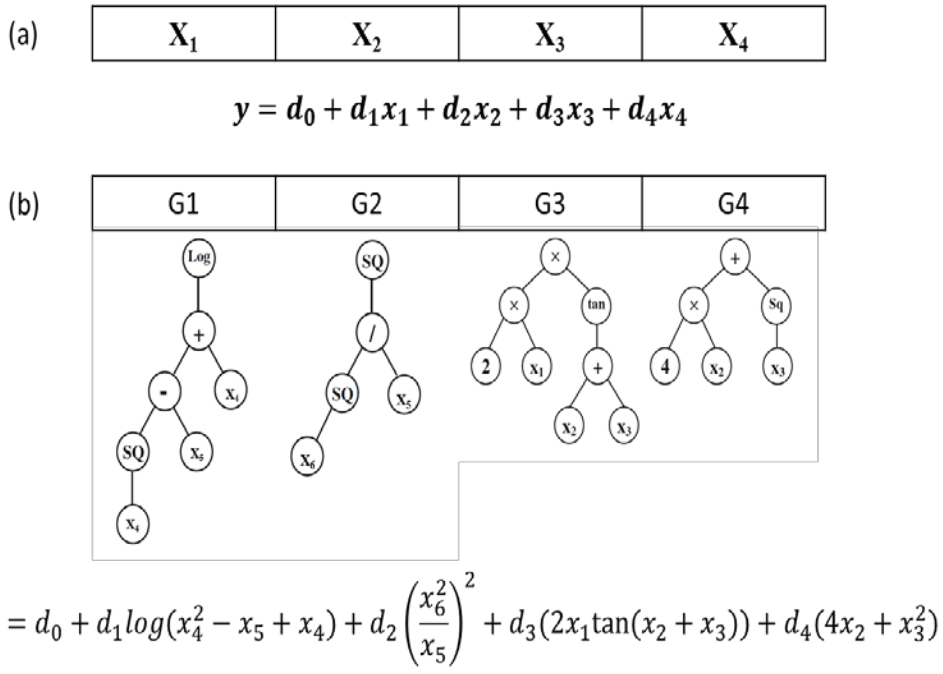


Figure 2-20: Individual development of (a) Genetic Algorithm, and (b) Multi-gene genetic Programing

2.4. Coefficient Generation

In general, model comprises of the features selected and the coefficient related to the features. These coefficients dictate the influence of a certain feature (e.g. descriptors) on the model. There are various ways of generating coefficients of features to develop the model.

2.4.1. Multiple Linear Regression (MLR)

To use MLR for coefficient calculation of linear models, it is important to ensure that the variables (x) are independent. According to Geladi and Kowalski (1986), MLR can be used to derive the coefficients of Eq. (2.9) provided that y is related to x_j in a linear manner, where e denotes the error of the equation.

$$y = \sum \beta_j x_j + e \tag{2.9}$$

For understanding the mathematical process of deriving β, let us assume a case with n numbers of y and m number of x. Figure 2-21 can be used to visualize this case based matrix setup. Although solving such system can be much easier when m = n as β can be simply derived by Eq (2.10), in most cases of QSPR/QSAR development, it is possible to see either cases with m > n or m < n.

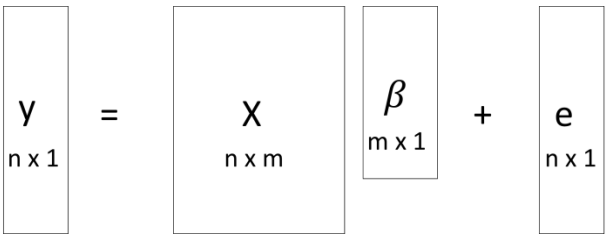


Figure 2-21: Sample Multiple Linear Equation

In case of m > n, there are more variables than samples. Such case causes generation of infinite number of solutions for β, and all of them are applicable. For this reason, MLR operation is never performed for cases where there are more variables than samples.

For cases where m < n, it is difficult to get exact solution for β by using Eq (2.10). However, one can use Eq (2.11) to generate a least-squares solution.

$$\beta = X'y \tag{2.10}$$

$$\beta = (X'X)^{-1}X'y \tag{2.11}$$

2.4.2. LASSO Regression

LASSO regression was performed using descriptors generated by the genetic algorithm. LASSO regression has been discussed in detail in the work of Tibshirani (1996). LASSO attempts to shrink some coefficients of the models and sets others to zero. In this way, LASSO retains the beneficial features of subset selection and ridge regression. Eq. (2.12) shows that LASSO works to minimize MSE, and based on that, reduces the number of predictors required to generate the model. The final set of coefficients (β^{LASSO}) is derived at the point of minimum MSE (Eq. (2.13)). Here, λ (lambda) is a user-defined constant. The higher the value of λ, the higher are the number of descriptors with zero coefficient.

$$\min \sum_{i=1}^{N} (y_i - \sum \beta_j x_{ij})^2 + \lambda \sum |\beta_j| \tag{2.12}$$

$$\beta^{LASSO} = argmin \left|\left|y - X\beta\right|\right|_2^2 + \lambda \left|\left|\beta\right|\right|_1 \tag{2.13}$$

Recently, Algamal et al. (2015) used an adaptive version of LASSO to generate a QSAR model of anticancer potency of imidazo derivatives. According to his report, LASSO can be greatly useful for model regression as well as model development algorithm when feature cancelation approach is followed. A simple representation of LASSO process to determine coefficients is presented in Figure 2-22. At this point, it is important to notice that derivation of LASSO coefficients depend on deriving a constant t such that $\left|\left|\beta\right|\right|_1 \leq t$. The function values (f1-f4) presented in the figure can be assumed to be MSE (mean square error) values of the model under development due to different values of $\beta 1$ and $\beta 2$. LASSO tries to shrink the coefficients of the descriptors to push them towards zero values.

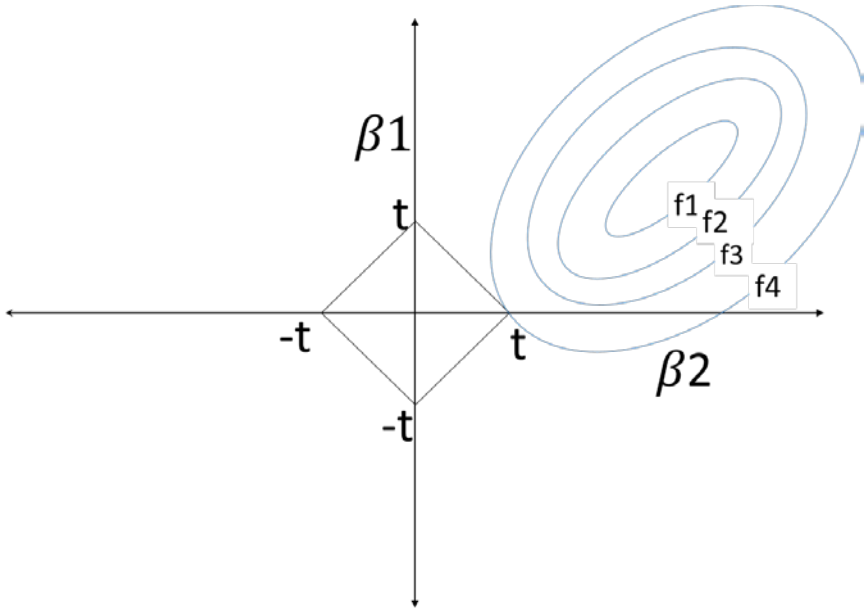


Figure 2-22: LASSO Coefficient shrinkage operation

For this 2-coefficient system, LASSO pushed the values of the coefficients until summation of their absolute values become equal to or less than t. For that reason, LASSO will suggest f4 value of MSE to be most applicable value with $\beta 1 = 0$ and $\beta 2$=t. This process is not that simple when more than two variables are involved in the model. For that, a mathematical approach is required.

So far, it has been established that regression process using LASSO highly depends on selection of λ (penalty) and t (tuning). These constants can also be called the controlling parameters of LASSO. As λ can be any positive real value in the range of $0 \rightarrow \propto$, an automated computational method can be developed to determine value of λ.

Determining the right value for t requires few more steps. There are three methods that can be used to determine t: cross-validation, generalized cross-validation and an analytical

unbiased estimate of risk. According to Tibshirani (1996), the most conventional method is the first one due to its flexibility in terms of X-Y relationship complexity and decreased computational effort.

$$Y = \eta(X) + \epsilon \tag{2.14}$$

$$MSE = \sum\left(\hat{\eta}(X) - \eta(X)\right)^2 \tag{2.15}$$

$$PE = \sum\{Y - \hat{\eta}(X)\}\text{^}2 = MSE + \sigma^2 \tag{2.16}$$

$$MSE = (\hat{\beta} - \beta)'V(\hat{\beta} - \beta) \tag{2.17}$$

For this part, relationship of X and Y has been supposed to be as Eq. (2.14). Here, σ^2 can be defined as variance of ϵ. The prediction error for LASSO procedure (PE) is produced by at least fivefold cross-validation. A normalized parameter $s = t/\sum|\widehat{\beta_J}|$ is used to index LASSO coefficients. Value of s can be varied from 0 to 1 and $\hat{\beta}$ can be determined for lowest PE. Finally, Eq. 2.17 was used to determine MSE in simpler manner once $\hat{\beta}$ exists. Here, V is a population covariance matrix of X, expressed by X'X. The whole process has been presented in the flowchart of Figure 2-23.

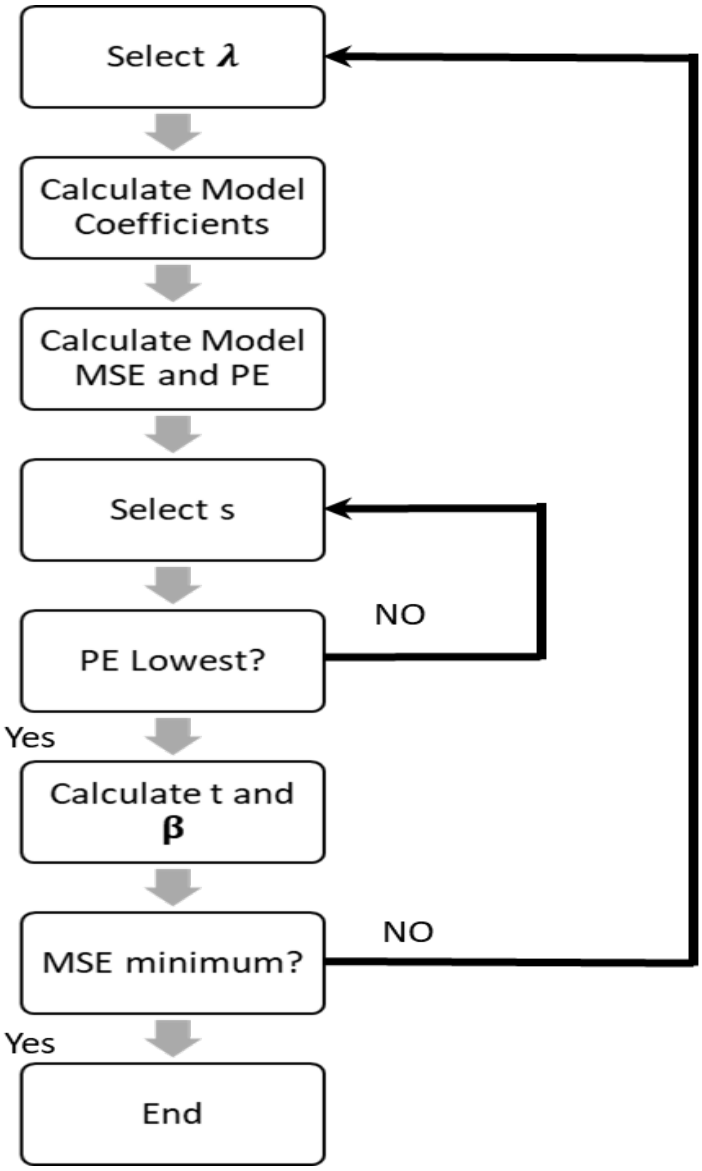


Figure 2-23: LASSO algorithm flowchart

3. Methodology

Here, we present the methodology performed to develop models for both cases. Although the method applied to develop models from descriptors for the cases are different, some initial steps remain same. The first step is to develop a dataset that is suitable for the case. Based on the dataset, molecular structures are developed using Avogadro© software. Avogadro© is a free software platform to design molecules which is very user-friendly. It is very easy to structures of varying complexities using Avogadro© as various fragments can be called in as per requirement. Figure 3-1 shows some sample molecular structures drawn by Avogadro© software. These structures are saved as .MOL files, which can be used to develop descriptors.

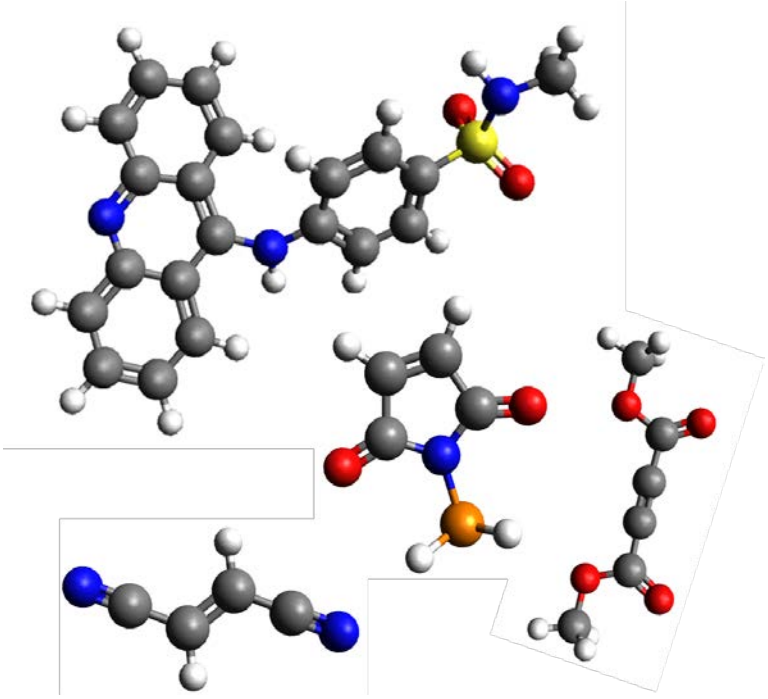


Figure 3-1: Molecular Structures Developed in Avogadro Platform

After this step, the .MOL files are collected and used in Dragon© 6 software to calculate the descriptors. Dragon© 6 can calculate 4885 different descriptors that include 1D to 4D type of descriptors. These descriptors are saved in a .TXT file and processed for feature selection operations. For our case, we have limited our investigation to 2D descriptors.

3.1.Case 1: Reaction Rate Constant of Diels-Alder Reaction

From the work of Tang et al. (2012) and Zhou et al. (2014, 2015), we generated a diverse data set of 72 reactions that consisted of 38 different dienophiles, 19 dienes and 10 solvents. These reactions, along with respective experimental and predicted reaction rates, are presented in Appendix A. All chemical species were designed using Avogadro software. The structures were optimized using MMFF94s, a built-in geometry optimization algorithm of Avogadro software, as suggested by Datta et al. (2015). The optimized geometries were saved as .MOL files. These files were then used as input for Dragon 6 software to calculate descriptors. Following convention, one sixth of the reactions were separated for external validation and all other reactions were used to train the model. Keeping reaction design simplicity in mind, only connectivity descriptors were used for model development.

3.1.1. Divide and Conquer Algorithm

This method has been previously utilized in many studies that required handling complicated datasets. Bentley (1980) has expressed the significance of multidimensional divide and conquer algorithm in his work. He has presented some classic point-based problems, where the main goal is to determine the domination of a point over other points present in dataset. This domination is determined by ranking the points, which refers to influence or significance of the point in dataset. Figure 3-2 shows a simple representation of the algorithm.

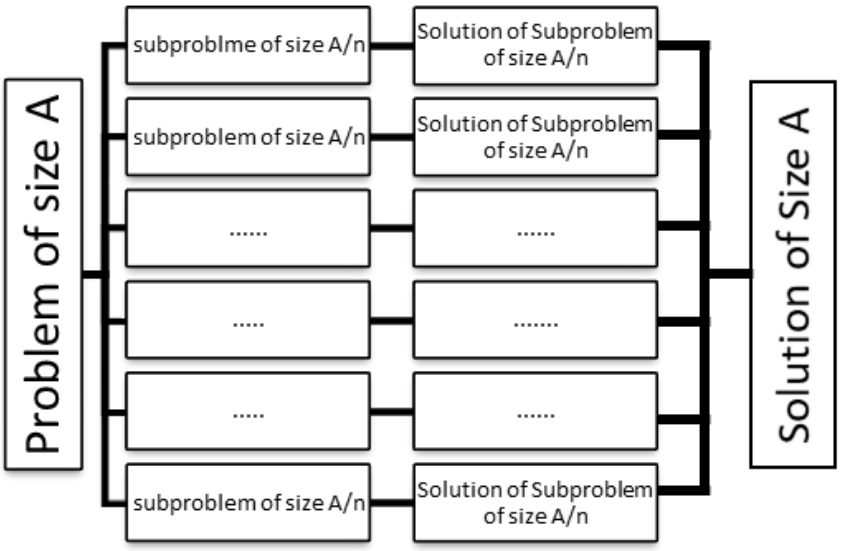


Figure 3-2: Problem size reduction by Divide and Conquer Algorithm

Zhang (2004) has showed application of this algorithm in his peptide sequencing algorithm. Cheng et al. (2012) have used this technique to improve their template-base modeling for their protein modeling project. The concept has also been discussed in detail in the work of Hemmateenezad et al. (2004). They have used PCA on different classes of descriptors to minimize the workload of GA. However, in this work, subsets of chemical species were subject to this strategy. The three subsets used were dienophiles, dienes and solvents. However, we checked for normalized standard deviation for all the descriptors of the subsets. After analyzing that, the descriptors with higher standard deviation were chosen to develop the dataset. For using this step, the number of meaningful descriptors was reduced to 32. These descriptors were then used as input to the DT algorithm for generating initial population of GA.

3.1.2. Decision Tree Algorithm

After the dataset was properly reduced to generate a dataset with only meaningful descriptors, the dataset was analyzed using a DT algorithm. As discussed in Section 2.3.2, DT is a good algorithm that can be used both for feature selection and coefficient generation. It is worth noticing that in the previous cases and in the example provided, possible solutions were generated using unaltered initial node. This project used a different approach of DT where its aim was to develop best possible initial population for GA. For such a requirement, DT was modified to alter its initial node every time it generates a member of the initial population. This ensures uniqueness of the developed members in the population.

3.1.3. Modified Genetic Algorithm

In most cases, GA works with generating an initial population. This population then goes through Roulette wheel elitism, crossover and mutation in each generation to develop better population. However, mostly single point swap is used for the process of crossover. Single point swap, in general, chooses two parents for the crossover. This process is also known as paring. The children are developed by swapping the parent data from a chosen single point. This is the most widely used process of crossover. In this work, scattered approach is used to perform the crossover operation. In scattered approach (Figure 3-3), multiple points of swaps are randomly selected. The swap occurs to the selected cells containing descriptors and the rest remains the same. The reason for choosing scatter swap over single point swap is to prevent over-adulteration of initial population while seeking better solution causing surgical changes.

	Single Point Swap				Scatter Swap			
Parent 1	X1	X2	X3	X4	X1	X2	X3	X4
Parent 2	Y1	Y2	Y3	Y4	Y1	Y2	Y3	Y4
Swap Point	0	1	0	0	0	1	1	0
Child 1	X1	Y2	Y3	Y4	X1	Y2	Y3	X4
Child 2	Y1	X2	X3	X4	Y1	X2	X3	Y4

Figure 3-3: Comparison of single point swap and scatter swap in crossover operation

3.1.4. Hybrid GA-DT Algorithm Development

In this step, the GA algorithm was modified using DT approach for generating a better initial population. A modified version of DT was used to choose meaningful descriptors that will increase model fitness. As shown in Figure 3-4, the initiation of GA uses DT. The DT algorithm controls the selection of the descriptors so that they increase the R^2 value of the candidates of the population. In this way, each member of the population represents a potential model. The first descriptor is a random selection. From the second descriptor and onwards, DT tries to include descriptors such that the R^2 value keeps getting better. This ensures not only a good initial population but also a good final model using the fewest number of generations.

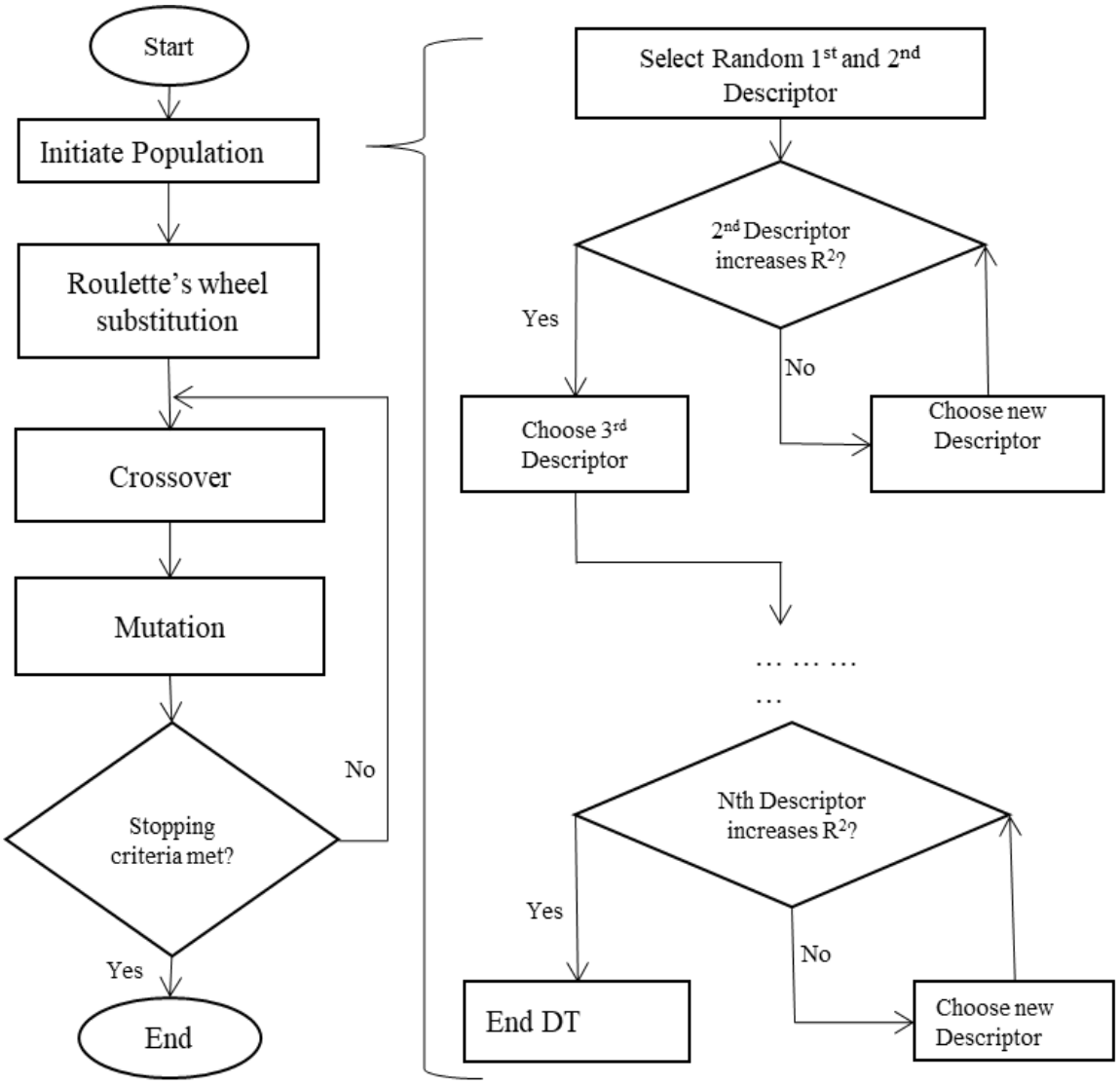


Figure 3-4: Developed hybrid GA-DT algorithm flowchart

After obtaining a good initial population, the population goes through Roulette's wheel population substitution. For the purpose of this work, any model with a R^2 less than the mean of all R^2 values were substituted by the best model value achieved in the generation. After that, Crossover and Mutation brings changes to populations. Crossover probability was selected to be

0.6 and mutation probability was assumed to be 0.02. Mutation, in this algorithm, randomly selects chromosomes and replaces them with descriptors not existing in the current population. After these steps, if the achieved generation does not produce at least one better model, the changes in the generation are rejected. And the system goes through crossover and mutation process again until the best possible model is identified. The operations of DT were confined to generating the initial population. The hypothesis behind this was that a more viable initial population can generate a potential model faster than one obtained from a randomly developed initial population. The entire algorithm was manually coded and executed using MATLAB scripts.

3.1.5. Multi-Gene Genetic Programming (MGGP)

To further investigate the effect of non-linear models on data prediction and representation, MGGP algorithm was used. For execution of this operation, the GPTIPS 2.0 (Searson et al., 2010) toolbox of MATLAB was used due to its wide variety of flexibility presented. It is important to note that the initial node of developing an MGGP individual is always an arithmetic operator. The GPTIPS 2.0, to determine the overall coefficients for genes, uses simple MLR assuming the model structure can be considered linear. For our work, although we used MLR for initial determination of the overall coefficients, a genetic algorithm was used to tune the coefficients further for better result. There needs to be defined some parameters to use the toolbox, and such parameters are recorded in Table 3-1. It is worth noticing that various combinations of gene numbers, gene depth, high and low level crossover probabilities were used to develop acceptable models. After event numbers are decided by the percentage of crossover events, the numbers of events are divided into two groups, the low and high level crossover

operations. It is of prime importance to remember that total probability of high and low level crossover operation must result to 1.

Table 3-1: MGGP parameters used for model development

Parameter	Settings
Population size	50
Number of generations	250
Maximum number of genes	4-7
Maximum gene depth	3-6
Tournament size	12
Crossover events	0.85
Mutation events	0.02
Subtree mutation	0.9

3.2.Case 2: Predicting DNA Drug Binding Affinity of 9-Anilinoacridine Derivatives

Three principal classes of descriptors (Conventional, 2D and Connectivity Index) were generated using the Dragon© 6 software. A total of 205 descriptors were generated using the software. An initial screening was performed to reduce the number of descriptors by eliminating those with zero values. 185 descriptors were obtained after this initial screening. GA was used to perform feature selection operation in combination with MLR. Finally, a correlation-based adaptive LASSO was used to further decrease the number of descriptors required and generate a model with enhanced prediction ability. During model development, one sixth of the molecules (5 out of 31) were used to test the model and the rest of the molecules (26 out of 31) were used to train it. For this purpose, the descriptors were used to generate an initial population of 50 rows and 13 columns. This initial population underwent the processes of roulette-wheel selection, single point swap crossover (with probability 0.6), and mutation (with probability 0.02). The objective was to select descriptors such that a model with the best R^2 and Q^2 values are generated. R^2 and Q^2 values describe the fitness of generated model using the training and test sets respectively. But the problem occurred was presence of multiple models with same R^2 and Q^2 values. It was necessary to analyse the final models and decide which one can be used as a property model.

Here, we have developed a correlation-based adaptive LASSO algorithm to direct the algorithm shrinkage towards descriptors in the model that show reduced correlation with the association constants. From previous discussions on LASSO, it was evident that correlations of target-descriptor values are never checked while shrinking the coefficients. A problem arises when LASSO tries to shrink descriptors with lowest values to zero and there are two of them. In such situations, LASSO automatically reduces both descriptors to zero. However, one can argue

that reducing one descriptor's coefficient to zero will change the coefficients of the rest of the descriptors. In that case, one of the two "zeroed-out" descriptor might become a descriptor with higher coefficient. To evaluate this assumption and check the effect, one way is to allow only one coefficient to be "zeroed-out" while saving another. As the coefficients of the models are checked for every new $\boldsymbol{\lambda}$ value using MLR, it is important to save the descriptor that shows comparatively higher correlation although having lowest-most coefficient. We have named this algorithm as CorrLASSO. To validate the superiority of the proposed algorithm, the results from the basic LASSO algorithm (using *lasso()* function of MATLAB) are compared with the correlation-based CorrLASSO algorithm. Finally, an overall R^2 value is calculated for the model with the lowest MSE to provide the best model possible for the available dataset. Figure 3-5 presents an overview on the process of CorrLASSO regression algorithm. This minor tuning of the algorithm is assumed to improve the algorithm efficiency as descriptor-property relationship is being utilized in model shrinkage and selection.

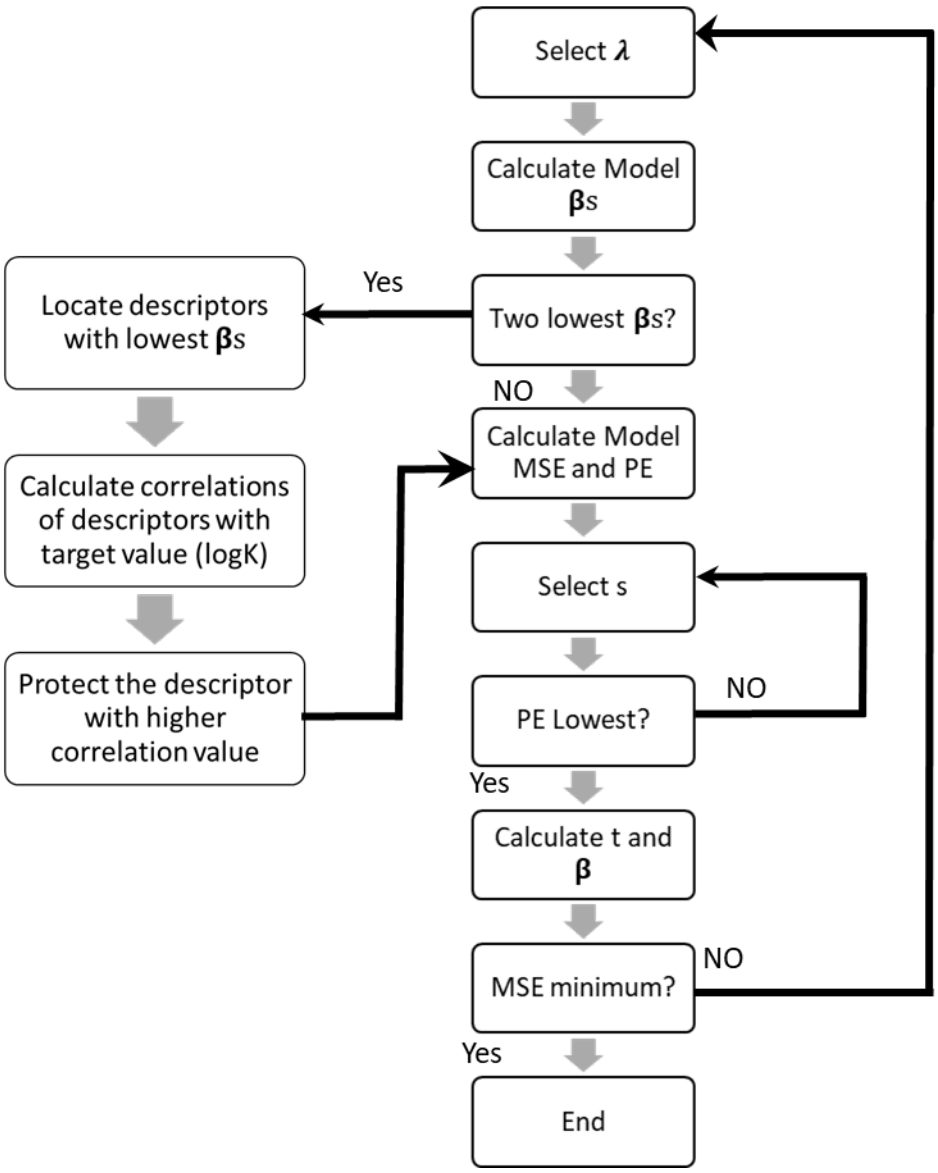


Figure 3-5: Flowchart of CorrLASSO regression algorithm

4. Results

To discuss and analyse the effectiveness of the methods proposed, results based on both cases have been discussed in this section.

4.1. Case Study 1

Using modified DT to develop initial population for GA plays a vital role in efficiency of model development. As shown in Table 4-1, the difference of developed models with and without DT modification in GA shows significant difference in model confidence.

Table 4-1: Improvement of initial model confidence with addition of descriptors in GA and GA-DT method

Descriptor addition	GA method		GA-DT method	
	R^2	Q^2	R^2	Q^2
1	-0.3035	-1.0256	-0.3035	-1.0256
2	0.0136	-0.546	0.0169	0.0089
3	0.2085	0.1029	0.0825	0.0153
4	-10.52	-15.35	0.1253	0.0328
5	-2.155	-4.081	0.3576	0.1284
6	0.3025	0.0158	0.3661	0.1153
7	0.1582	0.0379	0.4011	0.2086
8	0.0631	-.5379	0.4583	0.4105

The negative values might have been generated due to selection of descriptors which generated a model that doesn't follow the trend of the data. The table clearly demonstrates the influence of DT modification in this case. To make the comparison clearer, same seed number of random selection was used in both GA and GA-DT method.

It has also been noticed that this method can generate a suitable model much faster than simple use of GA algorithm. As suggested in Figure 4-1, fewer generations are required to develop the best model possible.

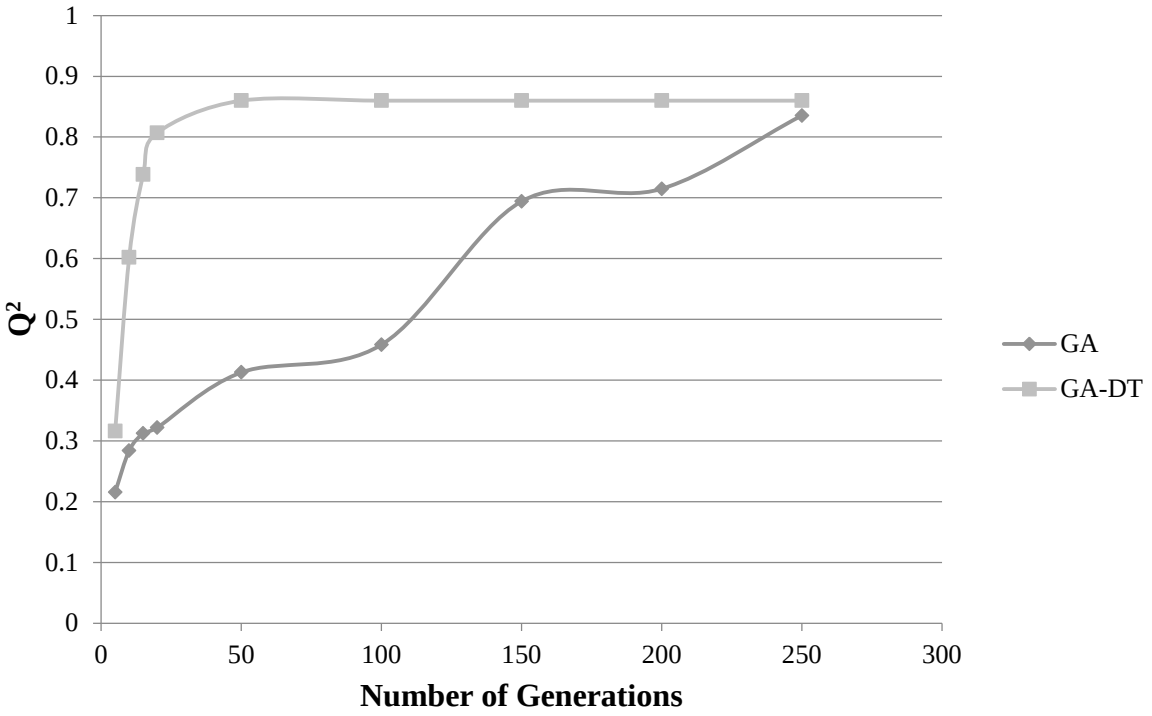


Figure 4-1: Q^2 Value improvement of developed models with number of generations

It can be seen that approximately 50 generations are sufficient for the hybrid GA-DT approach to develop the best model, while 250 generations of the GA approach was required to get a model with a confidence to that of the GA-DT model. In both cases, MLR was used as regression tool.

Figure 4-2 presents a comparison between the predicted and observed –log(k) values of the reactions used in model development. It can be seen that the model tends to have some difficulties predicting reaction rates in negative values. These values are related to reactions that were performed with water as solvent. So we can assume that this model is more applicable to reaction systems with organic solvents.

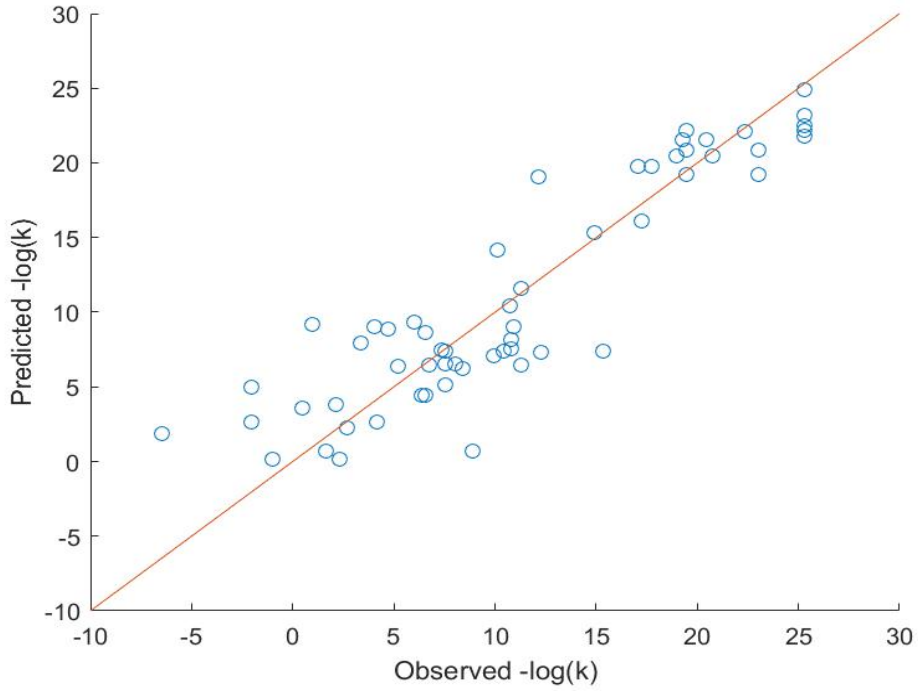


Figure 4-2: Observed vs predicted -log(k) values using hybrid GA-DT algorithm

It should be noted that the model is developed based only on connectivity descriptors of all chemical species. Eq. (4.1) shows the developed model:

$$-\log(k) = -57.0184 + 9.193891\ X3A_{R-1} - 57.7238\ X1Kup_{R-1} - 7.33585\ X2_{sol} + 23.46769\ X0Av_{R-1} + 96.15087\ X3Av_{R-1} + 3.094669\ X4_{R-2} + 72.42918\ X1Per_{R-1} + 66.62089\ X0A_{R-2} \quad (4.1)$$

Here,

$X3A_{R-1}$ - Average Connectivity Index of Order 3-Dienophile

$X1Kup_{R-1}$ - Kupchik Connectivity Index-Dienophile

$X2_{sol}$ - Connectivity Index of Order 2-Solvent

$X0Av_{R-1}$ - Average Valence Connectivity Index of Order 0-Dienophile

$X3Av_{R-1}$ – Average Valence Connectivity Index of Order 3-Dienophile

$X4_{R-2}$ - Connectivity Index of Order 4-Diene

$X1Per_{R-1}$ - Perturbation Connectivity Index-Dienophile

$X0A_{R-2}$ - Average Connectivity index of Order 0-Diene

k – reaction rate constant for second order reaction

The subscript R-1 denotes Dienophiles, while R-2 denotes Dienes and Sol denotes solvent. The model includes descriptors from every subset of chemical species used to develop the model. This gives reaction designers an opportunity to simultaneously evaluate effects of both reactants and solvents on the reaction rate constant. The model has R^2 value of 0.81 and Q^2 value of 0.86. As genetic algorithm is an evolution-based algorithm that locates maximum or minimum value of a given setup, this model can be considered to have the best predictability possible for the reaction-solvent system using the mentioned descriptors. However, there is always a possibility that the model might not be the only one with same predictability. In reality, one more model with almost same R^2 and Q^2 value did form. However, that model was rejected as it did not include effect of solvent structure on the target property.

However, better models were generated using MGGP algorithm. As GPTIPS 2.0 gives the opportunity for using various numbers and depths of the genes, the objective was to develop

a better model than Eq (4.1), presented by Datta et al. (2017), using a nonlinear model that contained the minimum number of descriptors possible. The population size and number of iterations were kept constant respectively to 50 and 250. In different settings of number of genes, gene depths, and high and low crossover possibilities, three models possessed better performance metrics than the model presented in the work of Datta et al. (2017). For convenience, they are identified as M1, M2, and M3. More details on these models have been presented in APPENDIX A. It should be noted that, all the selected models represent the influence of all the chemical classes involved in the reactions. However, the numbers of descriptors used to develop the models are different. M1 required 13 descriptors, where as M2 and M3 required 10 and 18 descriptors respectively. Here, R^2 and Q^2 represent model fitness for training set and test set data respectively, and RMSE expresses overall root mean squared error for the model.

Table 4-2: Properties of non-linear models developed and parameters used

Model Number	M1	M2	M3
R^2	0.9825	0.9011	0.9538
Q^2	0.8943	0.8213	0.9316
RMSE	0.5471	0.8410	0.2918
Gene number	4	5	7
Gene depth	3	4	5
Low level crossover	0.8	0.7	0.6
High level crossover	0.2	0.3	0.4

As presented in Table 4-2, M3 model shows the best fit compared to the other models. It can also be noticed that, in case of M2 and M3, decreased lower level crossover possibility, and increased number and depth of gene was able to improve fitness. This helps draw the conclusion that overuse of low level crossover can result in lower quality of predictive model. Although all

the models have significant level of nonlinearity, M3 was more successful in data fitness as the model showed highest nonlinearity among them due to comparatively higher numbers of gene number and depth. This also supports the assumptions made while developing model to describe reaction rate constants.

From the analyses of observed vs predicted –log(k) values in Figure 4-3, Figure 4-4, and Figure 4-5, it is clear that the non-linear models have much better fit than the linear one presented in Figure 4-2.

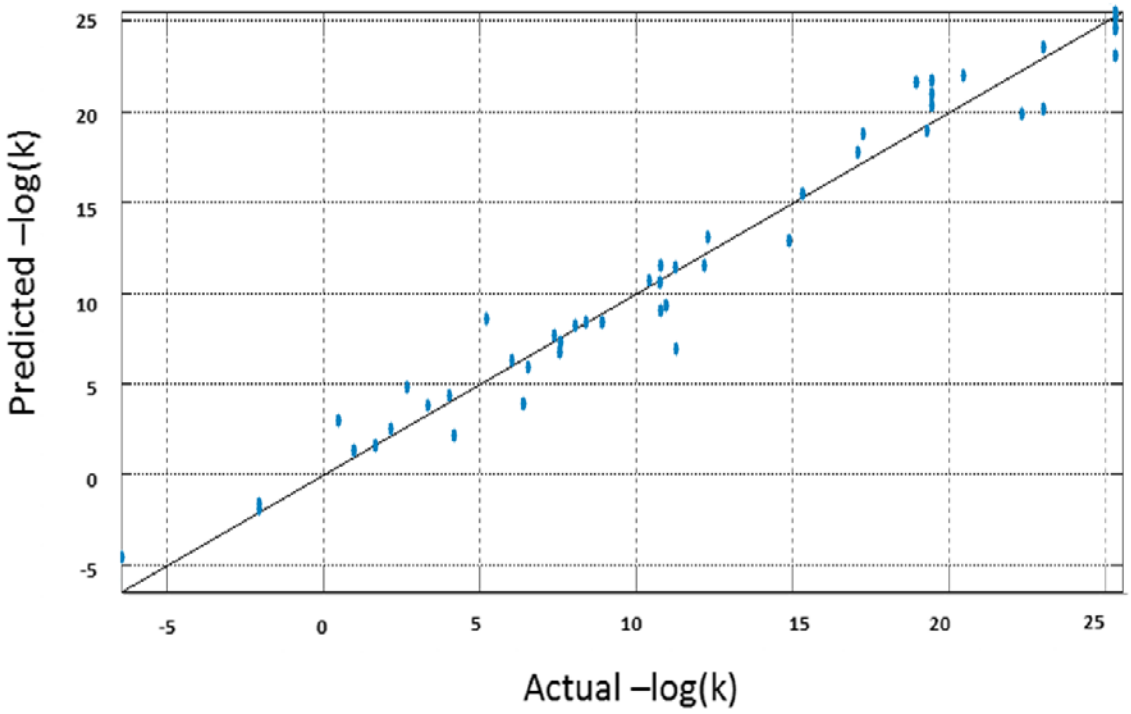


Figure 4-3: Observed vs predicted -log(k) values using M1

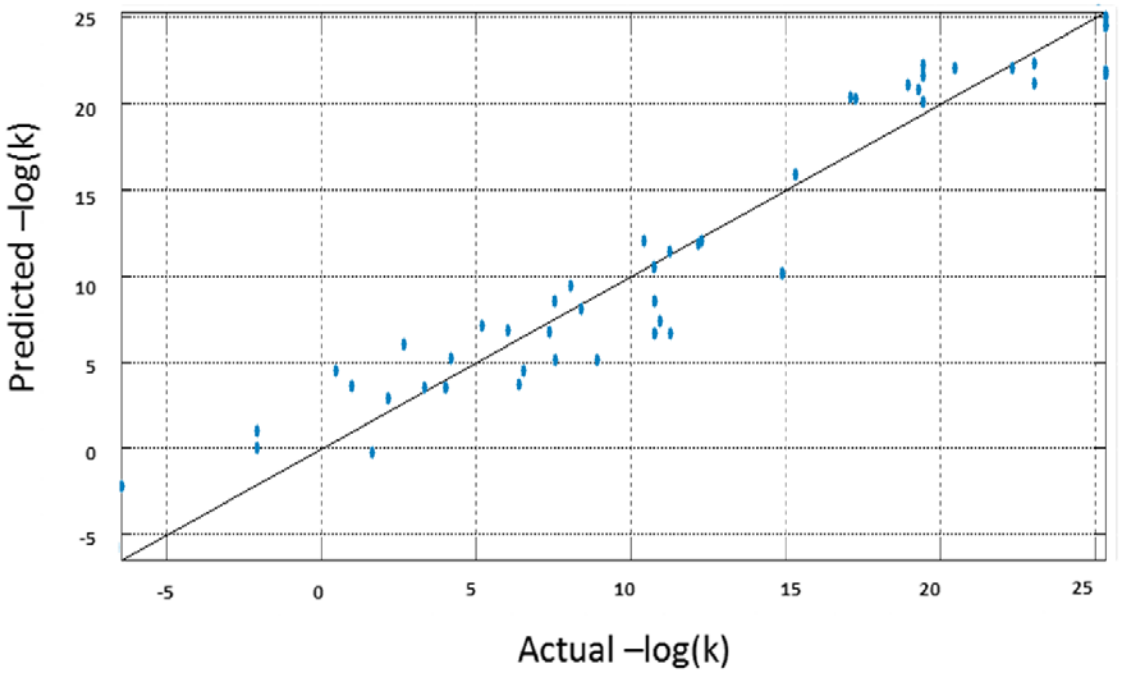


Figure 4-4: Observed vs predicted -log(k) values using M2

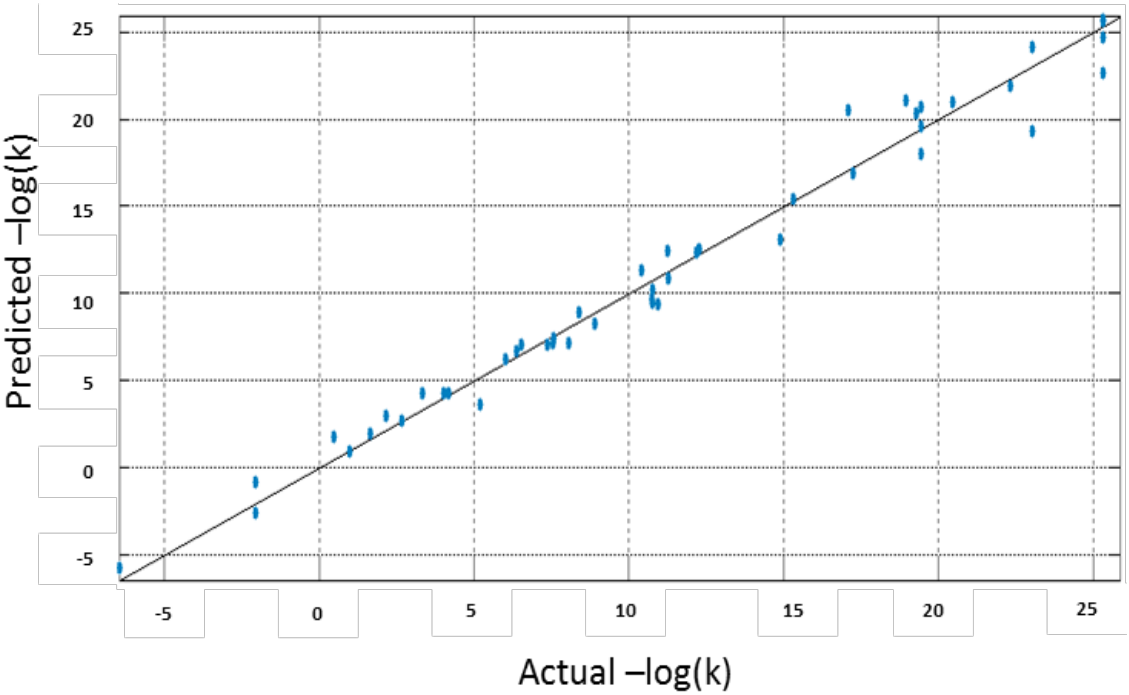


Figure 4-5: Observed vs predicted -log(k) values using M3

Observing these figures, it also becomes definite that M3 has the best fit, as suggested by Table 4-2. It is notable that M1 and M3 have significantly similar fit. However, careful observation dictates that the values involving aqueous solvent shows tighter fit for M3 than M1. These values show anomalies due to the fact that using water as solvent causes a vacuum of descriptor values, and that causes the prediction errors in linear models. However, the non-linear models tend to address this matter better than the linear models, producing better fit. Among the non-linear models developed, M3 also shows the best fit for this region. For these reasons, and the minimum model RMSE value, M3 can be considered a great candidate to be used as a predictive model.

The most important matter of this model is the descriptors are computationally easy to calculate from the molecular structures. This model gives an opportunity for studying not only the roles of the reactants but also solvents on the reaction rates of Diels-Alder reactions. A limitation, however, is that this model is only applicable for reactions performed at 25^0C (298K). A question may be raised about whether this model is suitable for reaction design at industrial level where temperatures being dealt with are much higher.

4.2.Case Study 2

From genetic algorithm, three acceptable sets of descriptors were generated. As the MLR (multiple linear regression) models generated from GA have very close R^2 (0.87) and Q^2 (0.92) values, more investigation was required to determine a unique model with the highest data fitness. In such a case, LASSO can be used to determine the best possible model. Figure 4-6 demonstrates the observed vs. predicted values of log(*K*) for models generated by the genetic algorithm. Although there are some points which fall on the line, some severe anomalies can be

noticed. To generate a better model, traditional LASSO regression was performed on the set of descriptors generated from GA.

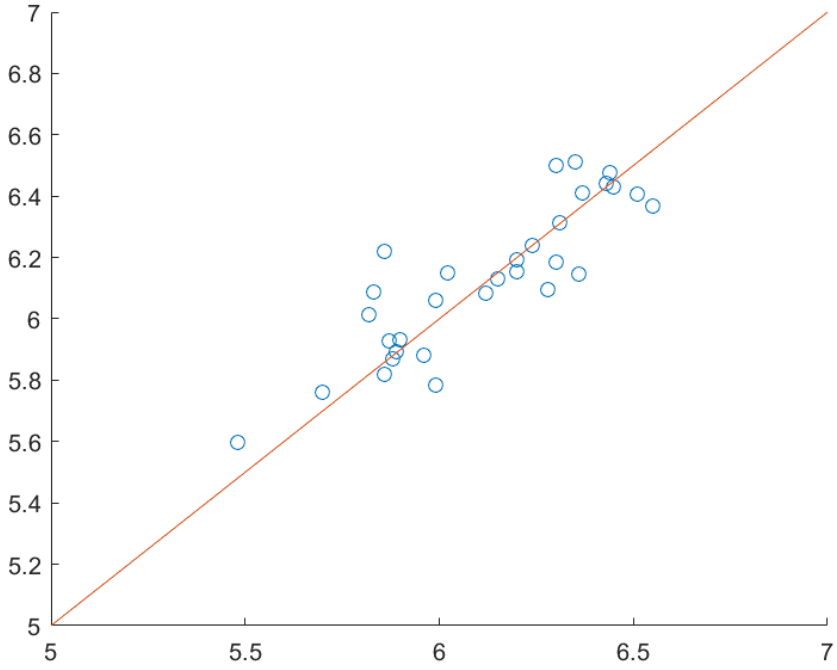


Figure 4-6: Observed vs. predicted log(K) values using genetic algorithm

It can be seen in Figure 4-7 that the traditional LASSO model reduced the MSE value, but the R^2 (0.89) and Q^2 (0.9) values suggest that the overall improvement was not significant. The observed vs predicted log(K) values (Figure 4-8) also show that there still remain some severe anomalies. However, the MSE value dropped significantly from around 0.05 to around 0.027. The aim of LASSO is to minimize the MSE of the model through removal of descriptors by reducing their coefficients to zero. It is possible that a descriptor having good correlation with the target value may be suppressed to zero.

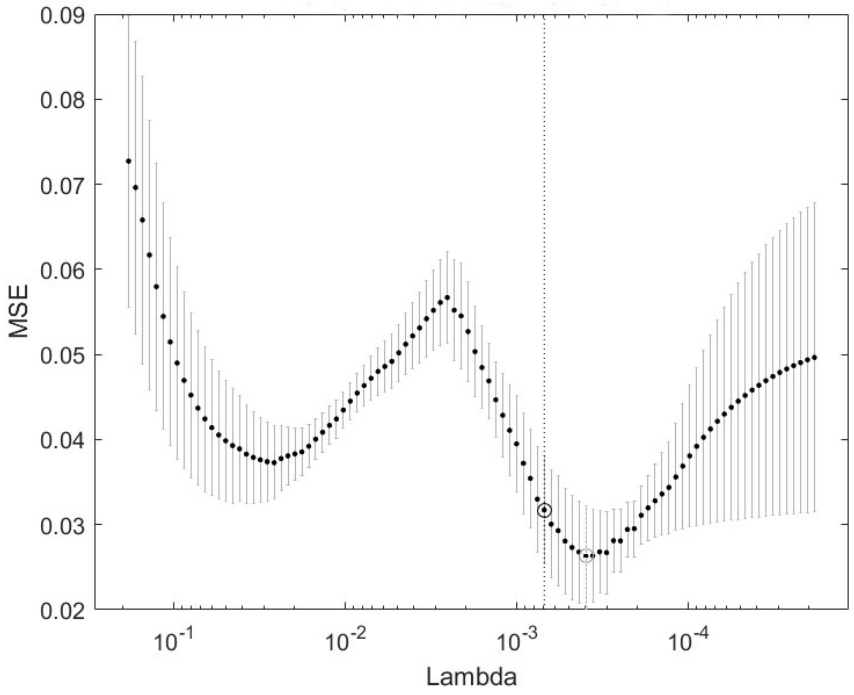


Figure 4-7: MSE analysis of model for log(K) using basic LASSO regression

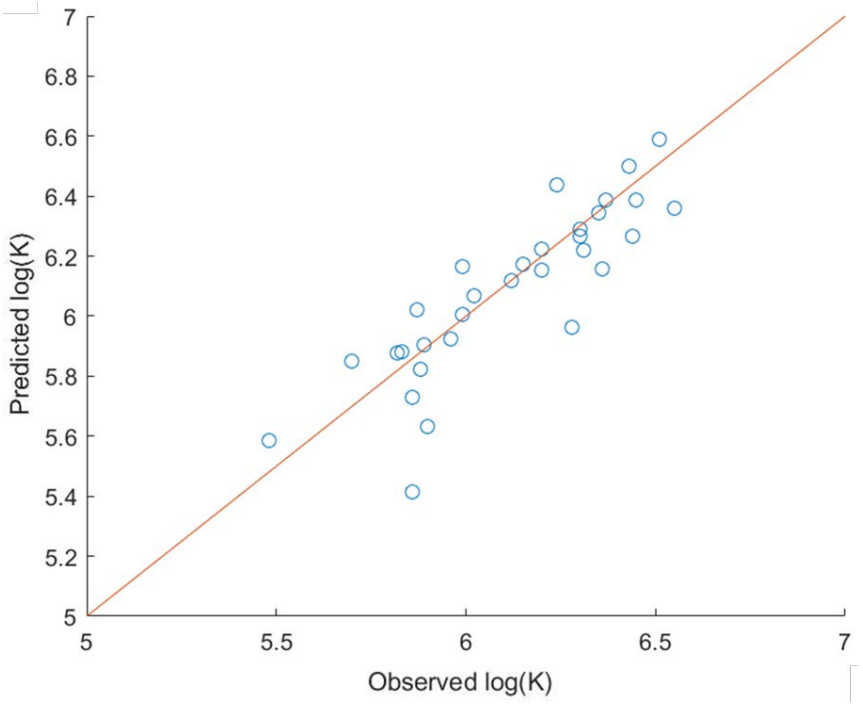


Figure 4-8: Observed vs predicted log(K) values for model generated using LASSO regression

In Table 4-3, coefficients of X3A are significantly different in LASSO and CorrLASSO evaluations. Presence of X3A in the model improved MSE value of the model as it has higher correlation than ZM2Mad and RBF with the target property. But its coefficient was suppressed to zero by LASSO algorithm based on its MLR coefficient.

Table 4-3: Coefficients of descriptors based on MLR, LASSO and CorrLASSO regression

Method		**MLR**	**LASSO**	**CorrLASSO**
Intercept		16.90	-12.09	-24.56
Descriptors	ZM2Mad	**0.01**	**0.00**	**0.00**
	IC0	-45.25	0.30	0.18
	RBF	**0.02**	**0.00**	**0.00**
	nCsp3	-7.26	-0.03	-0.01
	TIE	0.06	-0.05	-0.08
	PW2	0.03	3.40	7.54
	nDB	0.50	-0.33	-0.33
	TIC0	4.21	0.02	0.03
	CIC1	-0.02	-0.06	-0.01
	X5sol	-0.68	0.37	0.57
	Psi_e_A	-2.84	1.92	3.03
	X3A	**0.01**	**0.00**	**-22.58**
	H%	0.03	0.10	0.11
Model MSE		0.05	0.027	0.0126

To address this shortcoming, correlation values of two descriptors with lowest absolute coefficient values ($|b_j|$) were calculated. The descriptor with the lowest correlation with log(*K*) values was allowed to be suppressed to zero while the other one was protected to be evaluated further in the CorrLASSO.

When the CorrLASSO is used, it improves the model fitness (Figure 4-9). It can be seen that the MSE value has dropped to almost 0.0126 and the fitting curve (Figure 4-10) has also improved. It can be seen that X3A is improving the model if that is protected for future calculation. To do that, correlation values of two descriptors (ZM2Mad, X3A) with lowest absolute coefficient values ($|b_j|$) were calculated.

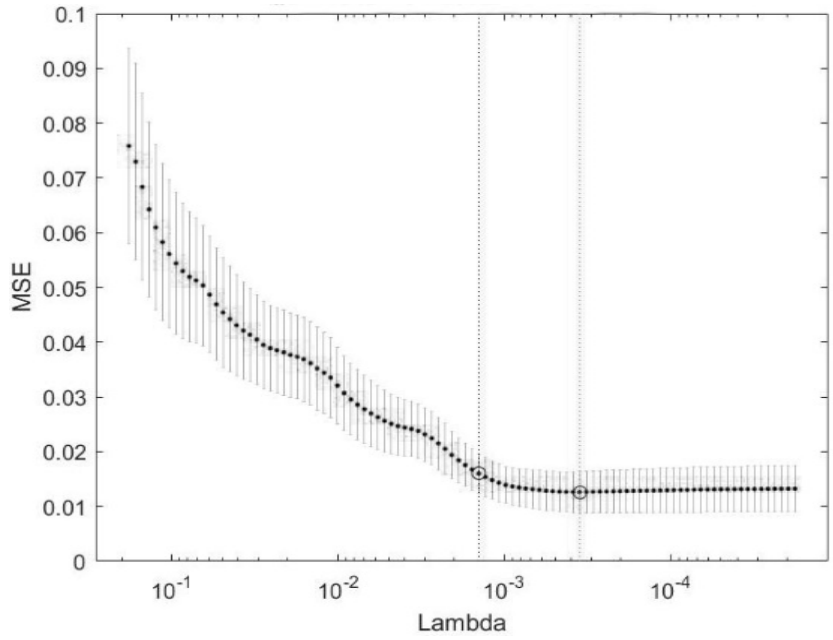


Figure 4-9: MSE analysis of the model using CorrLASSO regression

The descriptor with lowest correlation with *log(K)* values was allowed to be suppressed where the other one was protected to be tested further. When the CorrLASSO is used, it improves the model. Fitness values for both internal and external validations have improved. For

the achieved model, R^2 value is 0.9 and Q^2 value is 0.989. The overall R^2 value for this model is 0.947, which is better than the linear model proposed by Chtita et al. (2016) with R^2 value of 0.873. For the given model, the Q^2 value is significantly high. This means that this model can be used with high confidence in case of developing and studying derivatives that were not part of the training set. It can also be argued that the proposed model is not a significant improvement from the previous work. However, the descriptors used to develop the model are significantly easier to calculate for a given structure within the limits of the chemical space. Information about the molecular descriptors listed can be depicted from Talete information website (talete, 2018). From Table 4-3, it can be noticed that, logK values are highly dependent on three descriptors; X3A (average connectivity index of order 3), PW2 (path/walk 2 - Randic shape index), and Psi_e_A (intrinsic state pseudoconnectivity index - type S average). This proves the importance of CI descriptors in the developed model.

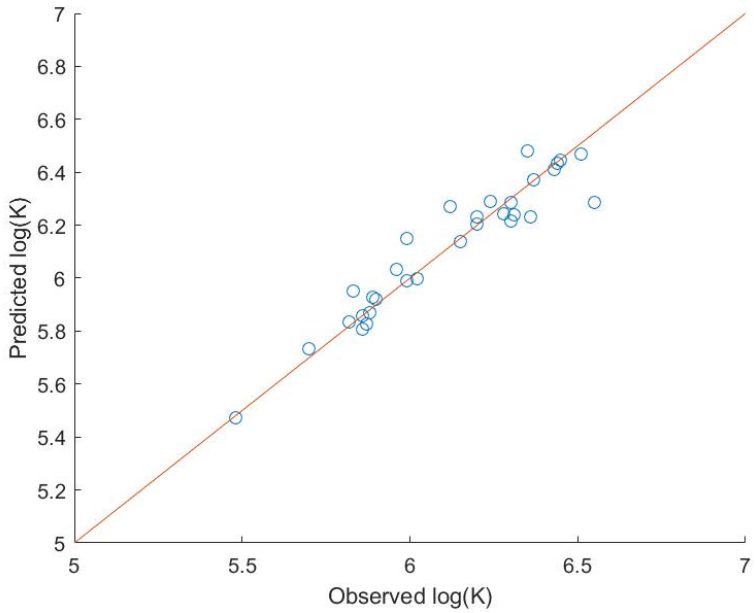


Figure 4-10: Observed vs predicted log(K) values using CorrLASSO algorithm

The new proposed method of regression analysis has proven to be very efficient. The best feature about this change is this only takes effect in cases where two lowest coefficients exist. A significant fact to consider for such cases is that it is not a good idea to let an algorithm reject two descriptors in a single step, as happens in cases of LASSO.

5. Conclusions and Future Directions

In this work, two condition-specific cases have been presented. The focus of the work was to develop hybrid algorithms that are computationally efficient and less expensive. Developing these systems can be utilized in case of developing software that tackles such situations. Overall, this project is dedicated to develop efficiency of GA-based hybrid approaches that has far more efficiency than the genetic algorithm alone. From their results, it could be also noticed that such algorithms can become efficient and less costly means of developing models in such condition-specific cases. It is very important to realize that such condition-specific models are likely to have a limited database to work on. These algorithms can be used to design a software platform for such CAMD investigations that will be user-friendly as the calculations can be performed in a simple computer system of present days. From the cases analyzed, it is not difficult to foresee that necessity of such platforms will emerge shortly to help our peers who are performing the experimental part of such studies.

In the first case, we found that by modifying GA through inclusion of DT enhances model generation capabilities. The model generated in this case through the hybrid GA-DT approach has a very good confidence level for both describing and predicting Diels-Alder reaction kinetics with solvent influence. However, some limitations need to be mentioned. The model assumes that a linear relationship exists between the logarithm of the rate constant and the molecular descriptors. Conducting a non-linear analysis might produce a better model, which will be a topic of future study. Including other 2D descriptors might also affect model confidence However, for the second part of this case was focused on developing a better model than that previously proposed for the same property using QSPR analysis (Datta et al., 2017). From the work of Datta et al. (2017), it was clear that the hybrid GA-DT method was very efficient in

developing linear model of such kind. As Dev et al. (2017) also concluded that the hybrid GA-DT method provided the best possible model, developing nonlinear model was the only option left to develop better property model. From the results it can be noticed that M1 and M3 models have similar metrics. However, in case higher level of accuracy is required, use of model M3 is advised. It can be noticed that Model M3 used five more descriptors than M1, but as connectivity index descriptors are very easy to calculate, no severe rise in computational expense is expected. Finally, it can be noticed that the models developed had better quality than it was in the case of linear model presented for same property by Datta et al. (2017). The model generated in our work relates the rate constant to the structures of reactants and solvent at a temperature of 298 K. Including temperature effects can be a good way to make the model more applicable for industrial process design. It should also be noted that although we have increased the diversity of the solvents in our study, in general, we need more data points in our study. We will explore the possibility of using quantum chemical calculations as a source of data to expand our data set. This approach has, for example, been investigated in the work of Sumathi et al. (2002). A concern can arise that the model generated in our study is only applicable for the Diels Alder reaction. Although this limits the application of the model generated, the process of model generation is a general one that can be used for any other reactant-solvent pair QSPR/QSAR study.

The second study aimed at developing a new approach of combining evolutionary algorithm with a feature-selection-based regression algorithm. Although model optimization using traditional LASSO algorithm is strictly based on eliminating coefficients, this work has proposed a correlation based LASSO algorithm which checks for property-feature correlation information to determine feature shrinkage and cancellation. This modification is highly useful

for evaluating models derived from evolutionary algorithms such as genetic algorithm; which tend to produce optimum models that may not be unique as per statistical fitness values. In such cases, regression algorithms like LASSO can be very efficient to evaluate the models for better fitness values. Although we have proposed a way of generating a highly feasible model, more data points are required to include diversity in the processed data. Another point to notice is the avoidance of using 3D descriptors. It is possible that using 3D descriptors may generate a model with better fitness. However, the aim of this work is to develop a model useful for setups with lower computational power. Molecular design using some of the 3D descriptors can result in a process requiring higher computational power that is almost similar to using DFT descriptors. Also, the model generated in this process is only valid for evaluating drug-DNA binding constant of 1' substituted 9-anilinoacridine derivatives, and effect of using 2' and 3' substituted derivatives can be a matter of future studies. However, the focus of this project is to develop an algorithm that can be utilized for any given chemical space. From that point of view, the proposed process is a universal one and can be used to generate models for predicting such drug-related properties.

5.1. Future Directions

The work aimed at developing algorithms to develop predictive models with high probability of predictability. However, the scope of developing models using larger chemical space is always an option. It certainly presents with it some degree of uncertainty; such can be dealt with cross validation. This was not dealt with in these cases due to not having a large chemical space consisting different types of chemical structures.

Additionally, as the goal was to develop predictive models using algorithms with universal acceptability and descriptors that are lesser dependent on high performing computers,

more case studies can be performed using these algorithms to check the viability of these algorithms in different cases with similar situations but different chemical spaces.

Finally, studies needed to be performed to validate the applicability of these models. These models need to be applied to product or research design and present novel solutions for the given problems. It is required to check the model performance in experimental setup before comments can be made on the effectiveness of these models in predicting required molecular structures.

6. References

C.S. Adjiman, A. Galindo, 2011, Front Matter. Process System Engineering, Wiley Online Library, I – XVII. .

Z.Y. Algamal, M.H. Lee, A.M. Al-Fakih, M. Aziz, 2015, High-dimensional QSAR prediction of anticancer potency of imidazo[4,5-b]pyridine derivatives using adjusted adaptive LASSO, J. Chemometrics, 29, 547-556

E.A. Amin and W.J. Welsh, 2001, Three-dimensional Quantative Structure-Activity Relationship (3D-QSAR) models for a movel class of Piperazine-based Stromelysin-1 (MMP-3) inhibitors: Applying a “Divide and Conquer” strategy, J. Med. Chem, 44, 3849-3855

C.M. Anderson-Cook , 2005, Practical Genetic Algorithms, Journal of the American Statistical Association, 100:471, 1099-1099

C. Andres and M.C. Hutter, 2006, CNS permeability of drugs predicted by a Decision Tree, QSAR Comb. Sci., 25, 305-309

A. Becke, 1988, Density-functional exchange-energy approximation with correct asymptotic behavior. Physical Reviews A, 3098-3100.

J.L. Bently, 1980, multidimensional Divide-and-Conquer, Communications of the ACM, 23(4), 214-229

M. Born, & K. Huang, 1988, Dynamical Theory of Crystal Lattices. Oxford: Clarendon Press.

L. Breiman, H. Friedman, R.A. Olshen, and C.J. Stone, 1984. Classification and Regression mes, Wadsworth International Group, Belmont, California, 356-358.

L. Breiman, 2001, Random Forests, Machine Learning, 45, 1, 5-32

B.C. Bugaley, W.A. Denny, G.J. Atwell, B.F. Cain, 1981, Potential antitumor agents. 34. Quantitative relationships between DNA binding and molecular structure for 9-anilinoacridines substituted in the aniline ring, J. Med. Chem, 24, 170-177

G. Ceder, K. Persson, 2013, How Supercomputers Will Yield a Golden Age of Materials Science, Sci. Am.

U.A. Chaudry, P.L.A. Popelier, 2003, Ester Hydrolysis Rate Constant Prediction from Quantum Topological Molecular Similarity Descriptors, J. Chem. Phys. Chem. A, 107, 4578-4582.

J. Cheng, J. Eickholt, Z. Wang, X. Deng, 2012, Recursive protein modeling: A divide and conquer strategy for protein structure prediction and its case study in CASP9, J Bioinform. Comput. Biol., doi:10.1142/S0219720012420036

S. Chtita, R. Hmamouchi, M. Larif, M. Ghamali, M. Bouachrine, T. Lakhlifi, 2016, QSPR studies of 9-anilinoacridine derivatives for their DNA drug binding properties based on density functional theory using statistical methods: Model, validation and influencing factors, J of Taibah University for Science, 10, 868-876

T. Clark, & R. Koch, 1999, Linear Combination of Atomic Orbitals. Berlin: Springer.

V. Consonni, R Todeshini, 2008, Handbook of Molecular Descriptors. Wiley Online Library, I – XXI.

M.T.D Cronin, J.S. Jaworska, J.D. Walker, M.H.I. Comber, C.D. Watts, A.P. Worth, 2003, Use of QSARs in international decision-making frameworks to predict health effects of chemical substances. Environ Health Perspect, 111(10), 1391–1401.

S. Datta, R.H. Herring III, M.R. Eden, 2015, Data Mining and Regression Algorithms for the Development of a QSPR Model Relating Solvent Structure and Ibuprofen Crystal Morphology, Computer-Aided Chemical Engineering, 37, 1439-1444.

S. Datta, V.A. Dev. M.R. Eden, 2016, Relating Reaction Rate Constant to Structures of Reactants and Solvent Using a Hybrid GA-DT Approach, Computer-Aided Chemical Engineering, 38, 2049-2054.

S. Datta, V.A. Dev, M.R. Eden, 2017, Hybrid genetic algorithm-decision tree approach for rate constant prediction using structures of reactants and solvent for Diels-Alder reaction. Computers and Chemical Engineering, 106, 60-698

S. Datta, V.A. Dev, M.R. Eden, 2018a, Developing Non-linear Rate Constant QSPR using Decision Trees and Multi-Gene Genetic Programming, Computer-Aided Chemical Engineering, 44, 2473-2478

S. Datta, V.A. Dev, M.R. Eden, 2018b, Using correlation based adaptive LASSO algorithm to develop QSPR of antitumor agents for DNA-drug binding prediction, Computers and Chemical Engineering, https://doi.org/10.1016/j.compchemeng.2018.08.039

L. Davis, 1986, Applying adaptive algorithms to epistatic domain, 9th joint conf on AI, 162-164

V.A. Dev, N.G. Chemmangattuvalappil, M.R. Eden, 2015, Designing Reactants and Products with Properties Dependent on Both Structures, Computer-Aided Chemical Engineering, 37, 1445-1450.

V.A. Dev, S. Datta, N.G. Chemmangattuvalappil, M.R. Eden, 2017, Comparison of Tree Based Ensemble Machine Learning Methods for Prediction of Rate Constant of Diels-Alder Reaction, Computer Aided Chemical Engineering, 40, 997-1002.

M. Dewar, E. Zoebisch, E. Healy, & J.Stewart, (1985). Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanics molecular model. Journal of the American Chemical Society, 3902-3909.

A. Dudek, T. Arodz, J. Galvez, 2006, Computational Methods in Developing Quantitative Structure-Activity Relationships (QSAR): A Review. Combinatorial Chemistry & Hight Throughput Screening, 213-228

E. Estrada, A.R. Matamala, 2007, Generalized Topological Indices. Modeling Gas-Phase Rate Coefficients of Atmospheric Relevance, J. Chem. Inf. Model., 47, 794-804.

D.A. Evans and J.S. Johnson, Diels-Alder Reactions, 1999, Chapter 33.1, Comprehensive Asymmetric Catalysis I-III, Springer-Verlag, Berlin Heidelberg, 1178-1235

A.H. Gandomi and A.H. Alavi, 2011, Multi-stage genetic programming: A new strategy to nonlinear system modeling, Information Sciences, 181,23, 5227-5239.

A.H. Gandomi and A.H. Alavi, 2012, A new multi-gene genetic programming approach to nonlinear system modeling. Part I: materials and structural engineering problems, Neural Computation and application, 21, 171-187

P. Geladi, B.R. Kowalski, Partial Least-Squares Regression: A Tutorial, 1986, Analytica Chimica Acta, 186 (1986), 1-17.

P. Glavič, 2012, Thirty Years of International Symposia on Process Systems Engineering. Curr Opin Chem Eng, 1(4), 421–429.

M. Goodarzi, P.R. Duchowicz, C.H. Wu, F.M. Fernandez, E.A. Castro, 2009, New hybrid genetic based support vector regression as QSAR approach for analyzing Flavonoids-GABA(A) complexes, J. Chem. Inf. Model, 49, 1475-1485

C. Grosan and A. Abraham, 2007, Hybrid evolutionary algorithms: Methodology, architectures, and review, Studies in Computational Intelligence, 75, 1-17

I.E. Grossmann, A.W. Westerberg, 2000, Research challenges in Process Systems Engineering. AIChE J, 46(9), 1700–1703.

D.E. Goldberg and K. Deb, 1991, A comparative analysis of selection schemes used in genetic algorithms, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, Los Altos, 69–93.

B. Gupta, A. Rawat, A. Jain, A. Arora,, A. Dhami, 2017, Analysis of Various Decision Tree Algorithms for Classification in Data Mining, International Journal of Computer Applications, 163,8, 15-19

I. Guyon, A. Elisseeff, 2003, An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, 1157-1182.

L. Hall, L. Kier, W. Murray, 1975, Molecular connectivity II: Relationship to water solubility and boiling point. Journal of pharmaceutical sciences, 1974-1977.

P.J.B. Hancock, 1994, An empirical comparison of selection methods in evolutionary algorithms, Selected Papers from AISB Workshop on Evolutionary Computing, Springer–Verlag, London, UK (1994), 80-94

C. Hansch, 1969, A Quantitative Approach to Biochemical Structure-Activity Relationships. Acc. Chem. Res., 232-239.

C. Hansch, P. Maloney, T. Fujita, R. Muir, 1962, Correlation of biological activity of phenoxyacetic acids with Hammett substituent constants and partition coefficients. Nature, 178-180.

B. Hemmateenejad, M.A. Sarafpour, R. Miri, F. Taghavi, 2004, Application of Ab Initio Theory to QSAR Study of 1,4-Dihydropyridine-Based Calcium Channel Blockers Using GA-MLR and PC-GA-ANN Procedures, Computational Chemistry, 25, 12, 1495-1503

R.H. Herring III, 2014, Computer aided molecular design with multi-dimentional characterization (Doctoral dissertation), Retrieved from http://hdl.handle.net/10415/4352

M. Hill, 2009, Chemical Product Engineering—The third paradigm. Comput Chem Eng, 33(5), 947–953.

J.H. Holland, 1975, Adaptation in natural and artificial systems, The University of Michigan Press, Ann Arbor

H. Hong, S. Slavov, W. Ge, F. Qian, Z. Su, H. Fang, 2012, Molecular Descriptors for QSAR. Stat Model Mol Descriptors QSAR/QSPR, WileyOnline Library, 65–109.

C.R Houck, J.A. Joines, M.G. Kay, 1996, A Genetic Algorithm for function Optimization: A MATLAB implementation (accessed 02.05.2017) https://www.researchgate.net/publication/2386612_A_Genetic_Algorithm_for_Function_Optimization_A_MATLAB_implementation

S. Izralev and D. Agrafiotis, 2001, A novel method for building regression tree models for QSAR based on artificial ant colony systems, J. Chem. Inf. Comput. Sci., 41,176-180

Q. Jun, S. Chang-Hong, W. Jia, 2010, Comparison of Genetic Algorithm based Support Vector Mahine anf Genetic Algorithm based RBF Neural Network in Quantitative Structure-Property Relationship models on aqueous solubility of polycyclic aromatic hydrocarbons, Procedia Environmental Sciences, 2, 1429-1437

V. Kendon, A. Sebald, S. Stepney, 2015, Heterotic computing: exploiting hybrid computational devices, Philos Trans R Soc London A Math Phys Eng Sci, 373 (2046).

K.U. Klatt, W. Marquardt, 2009, Perspectives for process systems engineering—Personal views from academia and industry. Comput Chem Eng , 33(3), 536–550.

L. Kier, W. Murray, 1975, Molecular Connectivity. 4. Relationships to Biological Activity. Journal of Medicinal Chemistry, 1272-1274.

L. Kier, W. Murray, M. Randic, L. Hall, 1975, Molecular connectivity. I. Relationship to nonspecific local anesthesia. Journal of Pharmaceutical Sciences, 1971-1974.

R. Leardi, 2001, Genetic algorithms in chemometrics and chemistry: a review, Journal of Chemometrics, 15, 559-569

R.I. Lerman, S. Yitzhaki, 1984, A note on the calculation and interpretation of the Gini index, Economic letters, 15, 363-368

Y.L. Loukas, 2001, Adaptive Neuro-Fuzzy infrence system: An instant and architecture –free predictor for improved QSAR studies, J. Med. Chem., 44, 2772-2783

B. Milanovic, 1997, A simple way to calculate the gini coefficient, and some implications, Economic letters, 56, 45-49

V. Mlinar, 2015, Utilization of inverse approach in the design of materials over nano- to macro-scale. Ann Phys., 527(3-4), 187–204.

S. Nandi, A. Monesi, V. Drgan, F. Merzel, M. Novic, Quantitative structure-activation barrier relationship modeling for Diels-Alder ligations utilizing quantum chemical structural descriptors, Chemistry Central Journal, 7, 171

I. M. Oliver, D. J. Smith and J. R. C. Holland, “A study of permutation crossover operators on the travelling salesman problem,” Proc. Second Int. Con. Genetic Algorithms, 1987, pp. 224-230.

J. Pavlus, 2015, The Search for a New Machine. Scientific American, 312, 58-63.

Pople, J., Beveridge, D., & Dobosh, P. (1967). Approximate Self-Consistent MolecularOrbital Theory. V. Intermediate Neglect of Differential Ocerlap. The Journal of Chemical Physics, 2026-2033.

J.R. Quinlan, 1986, Induction of Decision Trees; Kluwer Academic Publishers: Dordrecht, The Netherlands, 81–106.

J.R. Quinlan, 1987, Generating Production Rules from Decision Trees, In Proceedings of the International Joint Conference on Artificial Intelligence, Cambridge, MA, USA, 304–307.

M. Randic, 1975, Characterization of molecular branching. J Am Chem Soc. American Chemical Society, 97(23), 6609–6615.

N.M. Razali, J. Geraghty, 2011, Genetic algorithm performance with different selection strategies in solving TSP, Proceedings of Workld Congress on Engineering 2011, Vol II, ISBN: 978-988-19251-4-5

S. Reddy, S. Kumar, R. Garg, 2010, hybrid-genetic algorithm based descriptors optimization and QSAR models for predicting the biological activity of Tipranavir analogs for HIV protease inhibition, Journal of Molecular Graphics and Modelling, 28, 852-862

C.R. Reeves, 1995, A genetic algorithm for flowshop sequencing, Computers Ops. Res., 22, 5-13

D.C. Rideout and R. Breslow, 1980, Hydrophobic acceleration of Diels-Alder reactions, J. Am. Chem. Soc., 102, 7817-7818

R. Riolo and B. Worzel, 2003,Classification of gene expression data with genetic programming, Genetic Programming Theory and Practice, New Yor, USA.

Roothaan, C. (1951). New Developments in Molecular Orbital Theory. Reviews of Modern Physics, 69-89.

K. Roy, S. Kar, R.N. Das, 2015a, QSAR/QSPR Modeling: Introduction. A Prim QSAR/QSPR Model SE - 1. Springer International Publishing, 1–36.

K. Roy, S. Kar, R.N. Das, 2015b, Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment, 47–80.

A Sabljic, H Guesten, J Schoenherr, M Riederer, 1990, Modelling plant uptake of airborne organic chemicals. 1. Plant cuticle/water partitioning and molecular connectivity. Environ Sci Technol, American Chemical Society, 24(9):1321–1326.

R. Sargent, 2005, Process systems engineering: A retrospective view with questions for the future, Comput Chem Eng., 29(6):1237–1241.

D.P. Searson, D.E. Leahy, M.J. Willis, 2010, GPTIPS: an open source genetic programming toolbox for multigene symbolic regression, Proceedings of International Multiconference of Enginners and Computer Scientists 2010, Hong Kong.

M. Segall, 2012, Can we really do computer-aided drug design? J Comput Aided Mol Des. Springer Netherlands, 26(1):121–124.

W. Siedlecki, J. Sklansky, 1988, On Automatic Feature Selection. International Journal of Pattern Recognition and Artificial Intelligence, 197-220

S.N. Sivanandam and S. N. Deepa., 2007, Introduction to Genetic Algorithms, Springer, ISBN 9783540731894

N. Soni and T. Kumar, 2014, Study of Various Mutation Operators in Genetic Algorithms, International Journal of Computer Science and Information Technologies, 5 (3), 4519-4521

T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, & D. Whitley, (1991). A Comparison of Genetic Sequencing Operators. In Belew, R. & Booker, L. (eds.) Proceedings on the Fourth International Conference on Genetic Algorithms, 69-76. Los Altos, CA: Morgan Kaufmann Publishers.

I. Stanescu, L.E.K. Achenie, 2006, Atheoretical study of solvent effects on Kolbe–Schmitt reaction kinetics Chemical Engineering Science, 61, 6199-6212

G. Stephanopoulos, G.V. Reklaitis, 2011, Process systems engineering: From Solvay to modern bio- and nanotechnology. : A history of development, successes and prospects for the future. Chem Eng Sci, 66(19):4272–4306.

J. Stewart, (1989). Optimization of parameters for semiempirical methods I. Method. The Journal of Computational Chemistry, 209-220.

H. Struebing, Z. Ganase, P.G. Karamertzanis, E. Siougkrou, P. Haycock, P.M. Piccione, A. Armstrong, A. Galindo, C.S. Adjiman, 2013, Computer-aided molecular design of solvents for accelerated reaction kinetics, Nature Chemistry, 5, 952–957

R. Sumathi, H. Carstensen, W.H. Green Jr., 2002, Reaction Rate Predictions Via Group Additivity. Part 3: Effect of Substituents with CH2 as the Mediator, J. Phys. Chem. A, 106, 5474-5489

G. Syswerda, 1991, Schedule optimization using genetic algorithms, L Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York (1991), pp. 332-349

S. Tang, J. Shi, Q. Guo, 2012, Accurate prediction of rate constants of Diels–Alder reactions and application to design of Diels–Alder ligation, Org. Biomol. Chem., 10, 2673

R. Tibshirani, 1996, Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society, 58, 267-288

D. Whitley, 1994, A Genetic Algorithm Tutorial, Statistics and Computing, 4,65-85

Z. Zhang, 2004, De novo peptide sequencing based on a divide-and-conquer algorithm and peptide tandem spectrum simulation, Anal. Chem., 76, 6374-6383

J. Zhong, X. Hu, M. Gu, J. Zhang, 2005, Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms, Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05), doi: 10.1109/CIMCA.2005.1631619

T. Zhou, K. Mcbride, X. Zhang, Z. Qi, K. Sundmacher, 2014, Integrated solvent and process design exemplified for a Diels–Alder reaction, AIChE Journal, 61, 1, 147-158

T. Zhou, Z. Lyu, Z. Qi, K. Sundmacher, 2015, Robust design of optimal solvents for chemical reactions-A combined experimental and computational strategy, Chemical Engineering Science, http://dx.doi.org/10.1016/j.ces.2015.07.010

Appendix A- Case 1 supplementary information

A.1 – MATLAB Code for Divide and Algorithm

```
clc
clear
load solvents.txt
load regsolv.txt
xint=solvents;
yint=regsolv;
[n,p]=size(xint);
[pload,score,var]=pca(xint,'Economy',false);
%internal
crint=regress(yint-mean(yint),score(:,1:30));
crint=pload(:,1:30)*crint;
crint=[mean(yint)-mean(xint)*crint;crint];
yfitint=[ones(n,1) xint]*crint;
plot(yint,yfitint,'bo')
tss=sum((yint-mean(yint)).^2);
rss=sum((yint-yfitint).^2);
rsqr=1-(rss/tss);
disp(rsqr)
% external
yfitext=[ones(6,1) xext]*crint;
```

```
figure
plot(yext,yfitext,'bo')
tssex=sum((yext-mean(yext)).^2);
rssex=sum((yext-yfitext).^2);
rsqrex=1-(rssex/tssex);
press=sum((yfitext-y).^2);
qsr=1-(press/tss);
disp(qsr)
disp(rsqrex)
```

A.2 - MATLAB Code for Decision tree function

```
function [a] = decision_tree( x1,x2,y1,y2)
[row,column]=size(x1);
ini_table=randi(column,1);
for column_rand=1:7
     [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
     if column_rand>1
        if rsqrex(column_rand)<rsqrex(column_rand-1)
         [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
     elseif rsqrex(column_rand)<rsqrex(column_rand-1)
         [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
     elseif rsqrex(column_rand)<rsqrex(column_rand-1)
         [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
    elseif rsqrex(column_rand)<rsqrex(column_rand-1)
```

```
            [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
        elseif rsqrex(column_rand)<rsqrex(column_rand-1)
            [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
        elseif rsqrex(column_rand)<rsqrex(column_rand-1)
            [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
        elseif rsqrex(column_rand)<rsqrex(column_rand-1)
            [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
        elseif rsqrex(column_rand)<rsqrex(column_rand-1)
            [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,ini_table,column,y1,y2,x1,x2);
        elseif rsqrex(column_rand)<rsqrex(column_rand-1)
            [rsqr(column_rand) rsqrex(column_rand)
ini_table]=regression(row,column_rand,ini_table,column,y1,y2,x1,
x2);
            end
        end
```

```
end

a=ini_table;

end
```

A.3 - MATLAB Code for regression function

```
function [ r rex table ] = regression(
row,column_rand,ini_table,column,y1,y2,x1,x2)
random=randi(column,1);
    ini_table(column_rand+1)=random;
    [row_gen,column_gen]=size(ini_table);
     %initialize table
     x=ones(row,1);
      xe=ones(46-row,1);
      %column adding
     for column_iter=1:column_gen
         x=[x x1(:,ini_table(column_iter))];
     end
     for col_ex_iter=1:column_gen
         xe=[xe x2(:,ini_table(col_ex_iter))];
     end
%coeff calculation
     coeff=regress(y1,x);
     yfitint=x*coeff;
     yfitext=xe*coeff;
     %internal validation
     tss=sum((y1-mean(y1)).^2);
     rss=sum((y1-yfitint).^2);
```

```
    r=1-rss/tss;
    %external validation
    tssex=sum((y2-mean(y2)).^2);
    rssex=sum((y2-yfitext).^2);
    rex=1-rssex/tssex;
    table=ini_table;
end
```

A.4 – MATLAB code for Hybrid GA-DT Algorithm

```
clear
clc
load pcaint.txt
load regint.txt
load pcaext.txt
load regext.txt
yint=-log10(regint);
yext=-log10(regext);
%seed fixing
rng(9);
s=rng;
rng(s);
xint=pcaint;
xext=pcaext;
count=0;
[row,column]=size(xint);
for population=1:40
gen_table_initial(population,:)=decision_tree(xint,xext,yint,yex
t);
end
[row_gen,column_gen]=size(gen_table_initial);
loged_columns=randperm(column_gen,3);
```

```
for loging=1:3


xint(:,loged_columns(loging))=log10(xint(:,loged_columns(loging)
));
xext(:,loged_columns(loging))=log10(xext(:,loged_columns(loging)
));
end
coeff_best=0;
coeff_prev=0;
cross_probable=0.4;
mutation_probable=0.02;
%starting GA operations
for generation=1:550
 %reggression calculation
 for row_iter=1:row_gen
     x=ones(row,1);
      xe=ones(46-row,1);
     for column_iter=1:column_gen
         x=[x xint(:,gen_table_initial(row_iter,column_iter))];
     end
     for col_ex_iter=1:column_gen

         xe=[xe
xext(:,gen_table_initial(row_iter,col_ex_iter))];
```

```
    end
    coeff=regress(yint,x);
    yfitint=x*coeff;
    yfitext=xe*coeff;
    %internal validation
    tss=sum((yint-mean(yint)).^2);
    rss=sum((yint-yfitint).^2);
    rsqr(row_iter)=1-rss/tss;
    %external validation
    tssex=sum((yext-mean(yext)).^2);
    rssex=sum((yext-yfitext).^2);
    rsqrex(row_iter)=1-rssex/tssex;
    if row_iter>=2
        if rsqrex(row_iter)>rsqrex(row_iter-1)
        coeff_best=coeff;
        best_x=[x; xe];
        coeff_prev=coeff_best;
        else
            coeff_best=coeff_prev;
        end
    else
        coeff_prev=coeff;
    end
end
```

```
%Roulette's wheel starts
 for best=1:row_gen
     if rsqrex(best)==max(rsqrex);
         best_row=best;
     end
 end
 for pop=1:row_gen
     if rsqrex(pop)<mean(rsqrex)
       gen_table_initial(pop)=gen_table_initial(best_row);
         rsqr(pop)=rsqr(best_row);
         rsqrex(pop)=rsqrex(best_row);
     end
 end
 new_table=gen_table_initial;
 r_new=rsqr;
 q_new=rsqrex;
    %crossover starts
   cross_connection=floor(cross_probable*row_gen);%populations
to change in crossover
cross_relative_1=randi(floor(row_gen/2),1,floor(cross_connection
/2));%first set of populations to change
cross_relative_2=randi([floor(row_gen/2),row_gen],1,floor(cross_
connection/2));%second set of population to change
   [row_cross,column_cross]=size(cross_relative_1);
```

```
cross_point=randi(column_cross,floor(cross_connection/2),1);%col
umn where crossover happens
   for marriage=1:floor(cross_connection/2)%crossover operation
relative1temp=gen_table_initial(cross_relative_1(marriage),cross
_point(marriage):column_cross);
relative2temp=gen_table_initial(cross_relative_2(marriage),cross
_point(marriage):column_cross);
new_table(cross_relative_1(marriage),cross_point(marriage):colum
n_cross)=relative2temp;
new_table(cross_relative_2(marriage),cross_point(marriage):colum
n_cross)=relative1temp;
   end
%    mutation starts
      mut_cells=mutation_probable*row_gen*column_gen;
      mut_cells=floor(mut_cells);
      cell_row=randi(row_gen,1,mut_cells);
      cell_column=randi(column_gen,1,mut_cells);
      for mutation=1:mut_cells
new_table(cell_row(mutation),cell_column(mutation))=randi(column
,1,1);
      end
     %regression calculation on new_table
   for row_iter=1:row_gen
```

```
x=ones(row,1);
 xe=ones(46-row,1);
for column_iter=1:column_gen
    x=[x xint(:,new_table(row_iter,column_iter))];
end
for col_ex_iter=1:column_gen

    xe=[xe xext(:,new_table(row_iter,col_ex_iter))];

end
coeff=regress(yint,x);
yfitint=x*coeff;
yfitext=xe*coeff;
%internal validation
tss=sum((yint-mean(yint)).^2);
rss=sum((yint-yfitint).^2);
r_new(row_iter)=1-rss/tss;
%external validation
tssex=sum((yext-mean(yext)).^2);
rssex=sum((yext-yfitext).^2);
q_new(row_iter)=1-rssex/tssex;

if row_iter>=2
    if q_new(row_iter)>q_new(row_iter-1)
```

```
          coeff_best_new=coeff;
          best_x=[x; xe];
          coeff_prev=coeff_best_new;
          else
              coeff_best_new=coeff_prev;
          end
      else
          coeff_prev=coeff;
      end
    end
    %checking if the change is better
 if max(rsqrex)<max(q_new)
      count=count+1;
      improved_rsqr(count)=max(q_new);
gen_table_initial=new_table;
rsqr=r_new;
rsqrex= q_new;
coeff_best=coeff_best_new;
 end
end
 plot(1:count,improved_rsqr)
 if max(rsqrex)>.80
      disp('good model found')
      disp('R-sqr value')
```

```
    key=find(rsqrex==max(rsqrex),1);
    disp(rsqr(key))
    disp('Q-sqr value')
disp(rsqrex(key))
disp('Descriptor positions')
disp(gen_table_initial(key,:))
disp('Coefficients')
disp(coeff_best)
end
```

A.5 – Modified MATLAB functions of GPTIPS 2.0 for MGGP algorithm

A.5.1- evalfitness function

```
function gp = evalfitness(gp)
%EVALFITNESS Calls the user specified fitness function.
%
%   GP = EVALFITNESS(GP) evaluates the the fitnesses of
individuals stored
%   in the GP structure and updates various other fields of GP
accordingly.
%
%   Copyright (c) 2009-2015 Dominic Searson
%
%   GPTIPS 2
%
%   See also TREE2EVALSTR, EVALFITNESS_PAR

%check parallel mode.
if gp.runcontrol.parallel.enable && gp.runcontrol.parallel.ok
    gp = evalfitness_par(gp);
    return;

    %regular version
else
```

```
    for i = 1:gp.runcontrol.pop_size

        gp.state.current_individual = i;

        %retrieve values if cached
        if gp.runcontrol.usecache && gp.fitness.cache.isKey(i)
            cache = gp.fitness.cache(i);
            gp.fitness.complexity(i,1) = cache.complexity;
            gp.fitness.values(i,1) = cache.value;
            gp.fitness.returnvalues{i,1} = cache.returnvalues;

        else
            %preprocess cell array of string expressions into a
form that
            %Matlab can evaluate
            evalstr = tree2evalstr(gp.pop{i},gp);

            %store complexity of individual (either number of
nodes or tree
            %expressional complexity)
            if gp.fitness.complexityMeasure
                gp.fitness.complexity(i,1) =
getcomplexity(gp.pop{i});
```

```
            else
                gp.fitness.complexity(i,1) =
getnumnodes(gp.pop{i});
            end

            [fitness,gp] = feval(gp.fitness.fitfun,evalstr,gp);
            gp.fitness.values(i) = fitness;

        end
    end
end
```

A.5.2- gpmodelfilter Class

```
classdef gpmodelfilter
    %GPMODELFILTER Object to filter a population of multigene
symbolic regression models.
    %
    %   Usage:
    %
    %   First, create a default filter object F
    %
    %   F = GPMODELFILTER
    %
    %   Next, set the properties of the filter. E.g. to keep
only models
    %   that have an R^2 >= 0.7 (training data) but contain no
more than 3
    %   input variables use:
    %
    %   F.MINR2TRAIN = 0.7; F.MAXVARS = 3;
    %
    %   Finally, apply the filter to the population of models in
the GP
    %   struct:
    %
```

```
    %   GPF = F.APPLYFILTER(GP);
    %
    %   This returns a structure GPF which is functionally
identical to GP
    %   except that that models not meeting the filter
specifications have
    %   been removed.
    %
    %   It also removes duplicate models whose genotypes are
identical. All
    %   the usual GPTIPS functions such as POPBROWSER, RUNTREE,
GPPOPVARS,
    %   GPPRETTY etc. can be applied to the filtered data
structure GPF.
    %
    %   Remarks:
    %
    %   The filter has the following settings and defaults:
    %
    %      MINR2TRAIN = 0 (keeps models attaining this R2 on the
    %      training data).
    %
    %      MAXCOMPLEXITY = Inf (keeps models that have this
level of
```

```
    %      expressional complexity or lower).
    %
    %      PARETOFRONT = FALSE (true to keep only models on the
Pareto
    %      front of performance and expressional complexity).
Note that
    %      'expressional complexity' is used to compute the
front even if
    %      the GPTIPS run was actually performed using 'node
count' as the
    %      measure of tree complexity.
    %
    %      MAXVARS = Inf (keeps models containing this max
number of input
    %      vars).
    %
    %      MINVARS = 0 (keeps models containing this min number
of input
    %      vars).
    %
    %      INCLUDEVARS = [] (keeps models that include these
input variables
    %      - a row vector containing the input indices).
    %
```

```
    %      EXCLUDEVARS = [] (keeps models that do not contain
these
    %      input variables - a row vector containing the input
indices).
    %
    %      REMOVEDUPLICATES = TRUE (removes duplicate genotypes
from the
    %      population).
    %
    %   Hence, the default filter object only removes
duplicates.
    %
    %   [GPF,MODELINDS] = F.APPLYFILTER(GP) does the same but
also returns
    %   a Boolean vector MODELINDS which refers to the
population indices
    %   in GP that survived the filtering process.
    %
    %   Copyright (c) 2009-2015 Dominic Searson
    %
    %   GPTIPS 2
    %
    %   See also mergegp, genefilter, popbrowser, paretoreport
```

```
    properties (SetAccess = public)
        minR2train = 0.9; %the minimum R2 on the selected
dataset
        minR2test = 0.8; %the minimum R2 on the test dataset,
modification in the code
        maxComplexity = Inf; %the maximum complexity of models
to retain
        paretoFront = false; %true to select only models on the
pareto front
        maxVars = Inf; %selects models containing a max number
of input vars
        minVars = 0; %selects models containing a minimum number
of input vars
        includeVars =[]; %row vector of inputs that the models
must contain
        excludeVars = []; %row vector of inputs that the models
must not contain
        removeDuplicates = true; %true to remove duplicate
genotypes from population
    end

    methods

        %set removeDuplicates property
```

```
        function obj = set.removeDuplicates(obj, bool)
            if ~islogical(bool)
                disp('Error: removeDuplicates must either be set
to true or false');
                return;
            end
            obj.removeDuplicates=bool;
        end

        %set excludeVars property
        function obj = set.excludeVars(obj,varList)

            if isempty(varList)
                obj.excludeVars = varList;
                return;
            end

            if size(varList,1) > 1
                disp('Error: supplied list must be a row vector
of input variable numbers.');
                return;
            end

            if any(find(varList <= 0))
```

```
                disp('Error: 0 or negative numbers are not valid
input variable numbers.');
                return;
            end

            if numel(varList) ~= numel(unique(varList))
                disp('Error: supplied list must not contain
duplicate input variable numbers.');
                return;
            end

            if ~isempty(intersect(varList,obj.includeVars))
                disp('Error: supplied exclude list contains
variables on the include list.');
                return;
            end

            obj.excludeVars = varList;

        end%includeVars

        %set includeVars property
        function obj = set.includeVars(obj,varList)
```

```
            if isempty(varList)
                obj.includeVars = varList;
                return;
            end

            if size(varList,1) > 1
                disp('Error: supplied list must be a row vector
of input variable numbers.');
                return;
            end

            if any(find(varList <= 0))
                disp('Error: 0 or negative numbers are not valid
input variable numbers.');
                return;
            end

            if numel(varList) ~= numel(unique(varList))
                disp('Error: supplied list must not contain
duplicate input variable numbers.');
                return;
            end

            if numel(varList) > obj.maxVars
```

```
                disp('Error: supplied list must not exceed the
maxVars filter property.');
                return;
            end

            if ~isempty(intersect(varList,obj.excludeVars))
                disp('Error: supplied include list contains
variables on the exclude list.');
                return;
            end

            obj.includeVars = varList;

        end%includeVars



        %set R2min property
        function obj = set.minR2train(obj,r2min)

            if ~isa(r2min,'double')
                disp('Error: minimum R^2 training must be
between 0 and 1.');
                return;
            end
```

```
            if r2min < 0 || r2min > 1
                disp('Error: minimum R^2 training must be
between 0 and 1.');
                return;
            end
            obj.minR2train = r2min;
        end



        %set maxVars property
        function obj = set.maxVars(obj,maxvars)

            if ~isa(maxvars,'double')
                disp('Error: max. input vars must be greater
than 0.');
                return;
            end

            if maxvars < 1
                disp('Error: max. input vars must be greater
than 0.');
                return;
            end
```

```
            if maxvars < obj.minVars
                disp('Error: max. input vars must be equal to or
greater than min. input vars.');
                return;
            end

            obj.maxVars = maxvars;
        end

        %set minVars property
        function obj = set.minVars(obj,minvars)

            if ~isa(minvars,'double')
                disp('Error: min. input vars must be 1 or
greater');
                return;
            end

            if minvars < 1
                disp('Error: min. input vars must be 1 or
greater');
                return;
            end
```

```
            if minvars > obj.maxVars
                disp('Error: min. input vars must be smaller
than or equal to max. input vars.');
                return;
            end

            obj.minVars = minvars;
        end

        %set maxComplexity property
        function obj = set.maxComplexity(obj,maxc)

            if ~isa(maxc,'double')
                disp('Error: maximum complexity must be a number
greater than 1.');
                return;
            end

            if maxc < 1
                disp('Error: maximum complexity must be a number
greater than 1.');
                return;
            end
```

```
            obj.maxComplexity = maxc;
        end

        %set pareto front property
        function obj = set.paretoFront(obj,bool)

            if ~islogical(bool)
                disp('Error: paretoFront must either be set to
true or false');
                return;
            end
            obj.paretoFront = bool;
        end

        %function to apply the filter settings to a GP structure
        function [gp,filterInds] = applyFilter(obj,gp)

            if nargin < 2
                error('Usage is APPLYFILTER(GP)');
            end



            if ~isfield(gp.fitness,'r2train')
```

```
                error('GPMODELFILTER cannot find R^2 training
data. GPMODELFILTER is intended for use with populations
containing multigene regression models.');
            end

            if gp.runcontrol.pop_size > 1000
                disp('Please wait, this may take a few
moments...');
            end

            %always do r2 & complexity filter first
            filterInds = (gp.fitness.r2train >= obj.minR2train)
& (gp.fitness.complexity <= obj.maxComplexity);
            locations = find(filterInds);
            numLeft = numel(locations);
            disp([num2str(numLeft) ' models passed R^2 training
(>= ' num2str(obj.minR2train) ') and expressional complexity (<=
' int2str(obj.maxComplexity)  ') filter ...']);

            if numLeft == 0
               gp = [];
               return;
            end
```

```
            %pareto rank 1 filter
            if obj.paretoFront
                disp('Computing pareto front on training
data...');
                paretoInds = ndfsort_rank1([(1-
gp.fitness.r2train) gp.fitness.complexity]);
                filterInds = filterInds & paretoInds;
            end

            %next apply vars filters
            if ~isinf(obj.maxVars) || obj.minVars ||
~isempty(obj.includeVars) || ~isempty(obj.excludeVars)

                locations = find(filterInds);
                numLeft = numel(locations);
                disp(['Applying variable filter to '
num2str(numLeft) ' remaining models ...']);

                for i=1:numLeft
                    hvec = gpmodelvars(gp,locations(i));
                    vars = find(hvec);
                    numvars = numel(vars);
```

```
                    if numvars > obj.maxVars || numvars <
obj.minVars
                        filterInds(locations(i)) = false;
                    else

                        if ~isempty(obj.excludeVars)
                            if
~isempty(intersect(vars,obj.excludeVars))
                                filterInds(locations(i)) =
false;
                            end
                        end

                        if ~isempty(obj.includeVars)
                            intersection =
intersect(vars,obj.includeVars);
                            if  numel(intersection) <
numel(obj.includeVars)
                                filterInds(locations(i)) =
false;
                            end
                        end

                    end
```

```
            end %end of loop through individuals
        end


        %if enabled, loop through remaining genotypes and
remove
        %duplicates.
        if obj.removeDuplicates &&
~gp.info.duplicatesRemoved

            locations = find(filterInds);
            numLeft = numel(locations);
            disp(['Removing genotype duplicates from '
num2str(numLeft) ' remaining models ...']);

            for i=1:numLeft

                for j=1:numLeft

                    if i~=j && locations(i) && locations(j)
                        model_i = gp.pop{locations(i)};
                        model_j = gp.pop{locations(j)};
```

```
                            if numel(model_i) ~= numel(model_j)
                                continue
                            end

                            if
isequal(sort(model_i),sort(model_j))
                                filterInds(locations(j)) =
false;
                                locations(j)=0;
                            end
                        end

                    end

                end

                gp.info.duplicatesRemoved = true;
            end%end of removeDuplicates



            numModels = sum(filterInds);

            if numModels == 0
```

```
                disp('No models matching all filter criteria
were found.');
                gp=[];
                return
            end

            gp.pop = gp.pop(filterInds);
            gp.fitness.returnvalues =
gp.fitness.returnvalues(filterInds);
            gp.fitness.values = gp.fitness.values(filterInds);
            gp.fitness.r2train = gp.fitness.r2train(filterInds);

            if isfield(gp.fitness,'r2val')
                gp.fitness.r2val = gp.fitness.r2val(filterInds);
            end

            if isfield(gp.fitness,'r2test')
                gp.fitness.r2test =
gp.fitness.r2test(filterInds);
            end

            gp.fitness.complexity =
gp.fitness.complexity(filterInds);
```

```
            gp.fitness.nodecount =
gp.fitness.nodecount(filterInds);
            gp.runcontrol.pop_size = numModels;
            gp.info.filtered = true;
            gp.info.lastFilter = obj;
            gp.source = 'gpmodelfilter';
            disp([num2str(numModels) ' models passed the
filtering process.']);
        end %applyFilter
    end %methods
end %classdef
```

A.5.3- Models generated using MGGP algorithm

Model M1:

$$-logk = 6.038\,X1_{R1} + 7.133\,X0Av_{R1} - 5.605X0sol_{R1} - 7.133X1Kup_{R1} + 6.038X1_{R2} + 5.605X4v_{R2} + 1.528XMOD_{R2} + 7.133X1Kup_{R2} + 19.34X0Av_{R1}X1Kup_{R1} - 6.308X1v_{R2}X1sol_s - 7.133\mathrm{X3sol_{R2}X5v_s} + 5.605X0Av_{R1}(\mathrm{X3sol_{R1}} - \mathrm{X4v_{R2}} - X1Kup_{R2} + X1sol_s) + 16.03X3Av_{R2}X1sol_s^2 - 19.34X0Av_{R1}^2X4sol_{R1}X5_s - 6.308X0Av_{R1}X4sol_{R1}X3Av_{R2}X5_sX2sol_s - 33.74$$

Model M2:

$$-logk = 10.89X0Av_{R1} - 0.1703\exp\left(XMOD_s{}^{\frac{1}{2}}\right) - 3.064\sin(\sin(XMOD_s)) - 21.78\exp(X4Av_{R2}) - 3.064\log(X0_{R1}) + 10.89\log(X3A_{R1}) - 3.064\log(X0_{R1})^{\frac{1}{4}} + 10.1\sin(X1Kup_{R1}^{\frac{1}{2}})\sin(\sin(X4v_{R1})) + 20.84|sin(\cos(X3A_{R1}))|\left(X1sol_{R2} + \log(X3A_{R1}) + X1Kup_{R1}^{\frac{1}{4}}\right) + 10.89X1kup_{R1}^{\frac{1}{4}} - 10.89RDSQ_{R2}^{\frac{1}{2}} + 48.94$$

Model M3:

$$-logk = 14.08X1_{R2} - 3.755X1Kup_{R1} - 3.755xX1v_{R2} + 13.63X4Av_{R2} + 13.63X1Mad_s + 3.075\cos(X1v_{R2}\tan(X1sol_{R1})) - 1.246\exp(-\sin(X4_{R1} + XMOD_{R1})) - 35.19\tan(\exp(-X4v_{R1})) + 230\cos(X3Av_{R2}X1Mad_{R2}) - 3.755\log(X2v_{R1}) + 13.84\log(X1Per_{R1}) - 48.83\tan(X3A_{R1}) - 35.19\exp(-X1_{R2}) - 868.5\exp(-X3Av_{R2}) + 3.755X0sol_s(X5Av_{R2} + X0sol_s) + \frac{3.87X1_{R2}}{0.275(X1v_{R1} + X1Mad_s)} + 13.84\tan(X3A_{R1})\,(X3Av_{R2} - 4.932) + \frac{6.587\log(X3A_{R1})\,(X2A_{R1} - 4.714)}{\cos(X3Av_{R2} + X0sol_s)} + 617.70$$

Subscripts:

R1 – Dienophile
R2- Diene
S- Solvent

Appendix B- Case 2 supplementary information

B.1 – MATLAB code for using CorrLASSO algorithm

```
clear
clc
gen_table_initial=[78 117 49  167  180  143  185  153 186 79 94
170 44];
[row_gen,column_gen]=size(gen_table_initial);
load internal.txt
load external.txt
load Yint.txt
load Yext.txt
yint=Yint;
yext=Yext;
xint=internal;
xext=external;
 [row,column]=size(xint);
     x=[];
      xe=[];
     for column_iter=1:column_gen
         x=[x xint(:,gen_table_initial(column_iter))];
     end
     for col_ex_iter=1:column_gen
         xe=[xe xext(:,gen_table_initial(col_ex_iter))];
```

```
 end
real_x=[x; xe];
real_y=[yint; yext];
 [B, fitinfo]=corrlasso(real_x,real_y);
 coeff=B(:,fitinfo.IndexMinMSE);
 yfitint=x*coeff;
 yfitext=xe*coeff;
 %internal validation
 tss=sum((yint-mean(yint)).^2);
 rss=sum((yint-yfitint).^2);
 rsqr=1-rss/tss;
 %external validation
 tssex=sum((yext-mean(yext)).^2);
 rssex=sum((yext-yfitext).^2);
 rsqrex=1-rssex/tssex;
 disp(rsqr)
 disp(rsqrex)
 disp(B(:,fitinfo.IndexMinMSE))
```

B.2 – MATLAB code for CorrLASSO function

```
function [B, fitinfo]= corrlasso(x,y)
lambda=logspace(-5,-1,100);
initial_B=regress(y,x);
maxdex=[];
for i=1:length(lambda)
    B(:,i)=initial_B;
   [m, ind]=min(abs(B(:,i)));
   %check correlation if more than one minimum value exists
   if length(ind)>1
       for mindex=1:length(ind)
          r(mindex)=corrcoef(x(:,mindex),y);
       end
       [k,maxdex]=max(abs(r));
   end
   s_count=0;
   %finding s value for minimum PE
   for s=0:.01:1
   s_count=s_count+1;
   B(:,i)=s*B(:,i);
   if maxdex~=[]
       B(maxdex,i)=initial_B(maxdex);
   end
```

```
   y_check=B(:,i)*x;
   err(s_count)=immse(y,y_check);
   pe(s_count)= err(s_count)+ var((y-y_check));
   end
   [l, mins]=min(pe);
   B(:,i)=0.01*mins*B(:,i);
   mserr(i)=err(mins)+lambda(i)*(sum(abs(B(:,i))));
end
%finding minimum MSE
   [m,minmse]=min(mserr);
%recording min MSE values
    fitinfo.IndexMinMSE=minmse;
    fitinfo.minMSE=m;
    fitinfo.minPE=l;
end
```

Appendix C- List of Descriptors Used

C.1 –Descriptors Name and Description

Name	Description	Indices
MW	molecular weight	Constitutional indices
AMW	average molecular weight	Constitutional indices
Sv	sum of atomic van der Waals volumes (scaled on Carbon atom)	Constitutional indices
Se	sum of atomic Sanderson electronegativities (scaled on Carbon atom)	Constitutional indices
Sp	sum of atomic polarizabilities (scaled on Carbon atom)	Constitutional indices
Si	sum of first ionization potentials (scaled on Carbon atom)	Constitutional indices
Mv	mean atomic van der Waals volume (scaled on Carbon atom)	Constitutional indices
Me	mean atomic Sanderson electronegativity (scaled on Carbon atom)	Constitutional indices
Mp	mean atomic polarizability (scaled on Carbon atom)	Constitutional indices
Mi	mean first ionization potential (scaled on Carbon atom)	Constitutional indices
nAT	number of atoms	Constitutional indices
nSK	number of non-H atoms	Constitutional indices
nBT	number of bonds	Constitutional indices
nBO	number of non-H bonds	Constitutional indices
nBM	number of multiple bonds	Constitutional indices
SCBO	sum of conventional bond orders (H-depleted)	Constitutional indices
RBN	number of rotatable bonds	Constitutional indices
RBF	rotatable bond fraction	Constitutional indices
nDB	number of double bonds	Constitutional indices
nTB	number of triple bonds	Constitutional indices
nAB	number of aromatic bonds	Constitutional indices
nH	number of Hydrogen atoms	Constitutional indices
nC	number of Carbon atoms	Constitutional indices
nN	number of Nitrogen atoms	Constitutional indices
nO	number of Oxygen atoms	Constitutional indices
nP	number of Phosphorous atoms	Constitutional indices
nS	number of Sulfur atoms	Constitutional indices
nF	number of Fluorine atoms	Constitutional indices
nCL	number of Chlorine atoms	Constitutional indices
nBR	number of Bromine atoms	Constitutional indices
nI	number of Iodine atoms	Constitutional indices
nB	number of Boron atoms	Constitutional indices
nHM	number of heavy atoms	Constitutional indices

nHet	number of heteroatoms	Constitutional indices
nX	number of halogen atoms	Constitutional indices
H%	percentage of H atoms	Constitutional indices
C%	percentage of C atoms	Constitutional indices
N%	percentage of N atoms	Constitutional indices
O%	percentage of O atoms	Constitutional indices
X%	percentage of halogen atoms	Constitutional indices
nCsp3	number of sp3 hybridized Carbon atoms	Constitutional indices
nCsp2	number of sp2 hybridized Carbon atoms	Constitutional indices
nCsp	number of sp hybridized Carbon atoms	Constitutional indices
nCIC	number of rings (cyclomatic number)	Ring descriptors
nCIR	number of circuits	Ring descriptors
TRS	total ring size	Ring descriptors
Rperim	ring perimeter	Ring descriptors
Rbrid	ring bridge count	Ring descriptors
MCD	molecular cyclized degree	Ring descriptors
RFD	ring fusion density	Ring descriptors
RCI	ring complexity index	Ring descriptors
NRS	number of ring systems	Ring descriptors
NNRS	normalized number of ring systems	Ring descriptors
nR03	number of 3-membered rings	Ring descriptors
nR04	number of 4-membered rings	Ring descriptors
nR05	number of 5-membered rings	Ring descriptors
nR06	number of 6-membered rings	Ring descriptors
nR07	number of 7-membered rings	Ring descriptors
nR08	number of 8-membered rings	Ring descriptors
nR09	number of 9-membered rings	Ring descriptors
nR10	number of 10-membered rings	Ring descriptors
nR11	number of 11-membered rings	Ring descriptors
nR12	number of 12-membered rings	Ring descriptors
nBnz	number of benzene-like rings	Ring descriptors
ARR	aromatic ratio	Ring descriptors
D/Dtr03	distance/detour ring index of order 3	Ring descriptors
D/Dtr04	distance/detour ring index of order 4	Ring descriptors
D/Dtr05	distance/detour ring index of order 5	Ring descriptors
D/Dtr06	distance/detour ring index of order 6	Ring descriptors
D/Dtr07	distance/detour ring index of order 7	Ring descriptors
D/Dtr08	distance/detour ring index of order 8	Ring descriptors
D/Dtr09	distance/detour ring index of order 9	Ring descriptors
D/Dtr10	distance/detour ring index of order 10	Ring descriptors
D/Dtr11	distance/detour ring index of order 11	Ring descriptors
D/Dtr12	distance/detour ring index of order 12	Ring descriptors
ZM1	first Zagreb index	Topological indices

ZM1V	first Zagreb index by valence vertex degrees	Topological indices
ZM1Kup	first Zagreb index by Kupchik vertex degrees	Topological indices
ZM1Mad	first Zagreb index by Madan vertex degrees	Topological indices
ZM1Per	first Zagreb index by perturbation vertex degrees	Topological indices
ZM1MulPer	first Zagreb index by multiplicative perturbation vertex degrees	Topological indices
ZM2	second Zagreb index	Topological indices
ZM2V	second Zagreb index by valence vertex degrees	Topological indices
ZM2Kup	second Zagreb index by Kupchik vertex degrees	Topological indices
ZM2Mad	second Zagreb index by Madan vertex degrees	Topological indices
ZM2Per	second Zagreb index by perturbation vertex degrees	Topological indices
ZM2MulPer	second Zagreb index by multiplicative perturbation vertex degrees	Topological indices
ON0	overall modified Zagreb index of order 0	Topological indices
ON0V	overall modified Zagreb index of order 0 by valence vertex degrees	Topological indices
ON1	overall modified Zagreb index of order 1	Topological indices
ON1V	overall modified Zagreb index of order 1 by valence vertex degrees	Topological indices
Qindex	quadratic index	Topological indices
BBI	Bertz branching index	Topological indices
DBI	Dragon branching index	Topological indices
SNar	Narumi simple topological index (log function)	Topological indices
HNar	Narumi harmonic topological index	Topological indices
GNar	Narumi geometric topological index	Topological indices
Xt	total structure connectivity index	Topological indices
Dz	Pogliani index	Topological indices
Ram	ramification index	Topological indices
BLI	Kier benzene-likeliness index	Topological indices
Pol	polarity number	Topological indices
LPRS	log of product of row sums (PRS)	Topological indices
MSD	mean square distance index (Balaban)	Topological indices
SPI	superpendentic index	Topological indices
PJI2	2D Petitjean shape index	Topological indices
ECC	eccentricity	Topological indices
AECC	average eccentricity	Topological indices
DECC	eccentric	Topological indices
MDDD	mean distance degree deviation	Topological indices
UNIP	unipolarity	Topological indices
CENT	centralization	Topological indices
VAR	variation	Topological indices
ICR	radial centric information index	Topological indices
SMTI	Schultz Molecular Topological Index (MTI)	Topological indices
SMTIV	Schultz Molecular Topological Index by valence vertex	Topological indices

	degrees	
GMTI	Gutman Molecular Topological Index	Topological indices
GMTIV	Gutman Molecular Topological Index by valence vertex degrees	Topological indices
Xu	Xu index	Topological indices
CSI	eccentric connectivity index	Topological indices
Wap	all-path Wiener index	Topological indices
S1K	1-path Kier alpha-modified shape index	Topological indices
S2K	2-path Kier alpha-modified shape index	Topological indices
S3K	3-path Kier alpha-modified shape index	Topological indices
PHI	Kier flexibility index	Topological indices
PW2	path/walk 2 - Randic shape index	Topological indices
PW3	path/walk 3 - Randic shape index	Topological indices
PW4	path/walk 4 - Randic shape index	Topological indices
PW5	path/walk 5 - Randic shape index	Topological indices
MAXDN	maximal electrotopological negative variation	Topological indices
MAXDP	maximal electrotopological positive variation	Topological indices
DELS	molecular electrotopological variation	Topological indices
TIE	E-state topological parameter	Topological indices
Psi_i_s	intrinsic state pseudoconnectivity index - type S	Topological indices
Psi_i_A	intrinsic state pseudoconnectivity index - type S average	Topological indices
Psi_i_0	intrinsic state pseudoconnectivity index - type 0	Topological indices
Psi_i_1	intrinsic state pseudoconnectivity index - type 1	Topological indices
Psi_i_t	intrinsic state pseudoconnectivity index - type T	Topological indices
Psi_i_0d	intrinsic state pseudoconnectivity index - type 0d	Topological indices
Psi_i_1d	intrinsic state pseudoconnectivity index - type 1d	Topological indices
Psi_i_1s	intrinsic state pseudoconnectivity index - type 1s	Topological indices
Psi_e_A	electrotopological state pseudoconnectivity index - type S average	Topological indices
Psi_e_0	electrotopological state pseudoconnectivity index - type 0	Topological indices
Psi_e_1	electrotopological state pseudoconnectivity index - type 1	Topological indices
Psi_e_t	electrotopological state pseudoconnectivity index - type T	Topological indices
Psi_e_0d	electrotopological state pseudoconnectivity index - type 0d	Topological indices
Psi_e_1d	electrotopological state pseudoconnectivity index - type 1d	Topological indices
Psi_e_1s	electrotopological state pseudoconnectivity index - type 1s	Topological indices
BAC	Balaban centric index	Topological indices
LOC	lopping centric index	Topological indices
X0	connectivity index of order 0	Connectivity indices
X1	connectivity index of order 1 (Randic connectivity index)	Connectivity indices
X2	connectivity index of order 2	Connectivity indices
X3	connectivity index of order 3	Connectivity indices

X4	connectivity index of order 4	Connectivity indices
X5	connectivity index of order 5	Connectivity indices
X0A	average connectivity index of order 0	Connectivity indices
X1A	average connectivity index of order 1	Connectivity indices
X2A	average connectivity index of order 2	Connectivity indices
X3A	average connectivity index of order 3	Connectivity indices
X4A	average connectivity index of order 4	Connectivity indices
X5A	average connectivity index of order 5	Connectivity indices
X0v	valence connectivity index of order 0	Connectivity indices
X1v	valence connectivity index of order 1	Connectivity indices
X2v	valence connectivity index of order 2	Connectivity indices
X3v	valence connectivity index of order 3	Connectivity indices
X4v	valence connectivity index of order 4	Connectivity indices
X5v	valence connectivity index of order 5	Connectivity indices
X0Av	average valence connectivity index of order 0	Connectivity indices
X1Av	average valence connectivity index of order 1	Connectivity indices
X2Av	average valence connectivity index of order 2	Connectivity indices
X3Av	average valence connectivity index of order 3	Connectivity indices
X4Av	average valence connectivity index of order 4	Connectivity indices
X5Av	average valence connectivity index of order 5	Connectivity indices
X0sol	solvation connectivity index of order 0	Connectivity indices
X1sol	solvation connectivity index of order 1	Connectivity indices
X2sol	solvation connectivity index of order 2	Connectivity indices
X3sol	solvation connectivity index of order 3	Connectivity indices
X4sol	solvation connectivity index of order 4	Connectivity indices
X5sol	solvation connectivity index of order 5	Connectivity indices
XMOD	modified Randic index	Connectivity indices
RDCHI	reciprocal distance sum Randic-like index	Connectivity indices
RDSQ	reciprocal distance sum inverse Randic-like index	Connectivity indices
X1Kup	Kupchik connectivity index	Connectivity indices
X1Mad	connectivity topochemical index	Connectivity indices
X1Per	perturbation connectivity index	Connectivity indices
X1MulPer	multiplicative perturbation connectivity index	Connectivity indices