Molecular Design of Crosslinked Copolymers

by John Eslick

Submitted to the graduate degree program in the Chemical and Petroleum Engineering department and the graduate faculty of the University of Kansas School of Engineering in partial fulfillment for the degree of Doctor of Philosophy.

Dr. Kyle Camarda, Committee Chair

Dr. Margaret Bayer

Dr. Kenneth Bishop

Dr. Stevin Gehrke

Dr. Aaron Scurto

Date defended: December 16, 2008

The Dissertation Committee for John Eslick certifies that this is the approved version of the following dissertation:

Molecular Design of Crosslinked Copolymers

Dr. Kyle Camarda, Committee Chair

Date approved: January 8, 2009

Abstract

A complete methodology for the computational molecular design (CMD) of crosslinked polymers is developed and implemented. The methodology is applied to the design of novel polymers for restorative dental materials. The computational molecular design of crosslinked polymers using optimization techniques is a new area of research. The first part of this project seeks to develop a novel data structure capable of adequately storing a complete description of the crosslinked polymer structure. Numerical descriptors of polymer structure are then calculated from the data structure. Statistical methods are used to relate the structural descriptors to experimentally measured properties. An important part of this project is to show that useful property prediction models can be developed for crosslinked polymers. Desirable property target values are then set for a specific application. Finally, the structure-property relations are combined with a Tabu search optimization algorithm to design improved polymers. Tabu search allows much flexibility in the problem formulations, so a major goal of this project is to show that Tabu search is a effective method for crosslinked polymer design.

To implement the molecular design procedure, a software package is developed. The software allows for easy graphical entry of polymer structures and property data, and contains a Tabu search optimization routine. Since computational molecular design of crosslinked polymers is a relatively new area of research, the software is designed to be easily modified to allow for extensive numerical experimentation.

Finally, the computational design methodology is demonstrated for the design of polymers for restorative dental applications. Using the computational molecular design methodology developed in this project, several monomers are found that may offer a significant improvement over a standard HEMA/bisGMA formulation. The results of the case study show that the new data structure for crosslinked polymers is effective for calculation of topological descriptors and property models can be developed for crosslinked polymers. Tabu search is also shown to be an effective optimization method.

Acknowledgements

This project was funded by NIH grants R13-DK069504-03 and NIH/NIDCR grant DE014392 (PI: Spencer).

I would like to thank Dr. Spencer, Dr. Ye, and Dr. Park for helping me understand dental materials and for providing the experimental data for this project. I would also like to acknowledge Janine Einhellig, Natalia Davydova, and Harrison Davis who worked hard to collect experimental property data.

I would like to thank my associate Sarah Shulda for all her help and for making work more fun. Finally I would like to thank my research advisor, Dr. Camarda, for helping me with his knowledge of molecular design, and for his editing. A good editor is important, especially for me.

Contents

Abstract **ii**

Acknowledgements **iv**

Table of Contents **v**

List of Figures **ix**

List of Tables **xii**

1 Introduction **1**

1.1 Project Overview 2

2 Background **5**

2.1 Structural Descriptors and Property Prediction 5

2.2 Statistical Methods for QSPR Development 9

2.3 Molecular Design 10

2.4 Dental Polymers 12

3 Polymer Structure **14**

3.1 Monomer Graphs 16

3.2 Polymer Graphs 17

3.3 Full Graphs 21

3.4 Software Implementation 21

3.4.1 Vertices and Edges 22

3.4.2 Functional Groups and Graph State 24

3.4.3 Indexing, Adjacency Matrices, and Adjacency Lists 24

4 Structural Descriptors 27

4.1 Graph Algorithms 28

4.1.1 Subgraph Isomorphism 28

4.1.2 Path Finding 31

4.1.3 Shortest Path 32

4.1.4 Cycle Finding 32

4.1.5 Block Finding 33

4.2 Descriptor Calculation 34

4.2.1 Molecular Weight and Atom Type 34

4.2.2 Connectivity Indices 34

4.2.3 Shape Indices 37

4.2.4 Crosslinking 38

4.3 Effect of Graph Size 39

5 Property Models 40

5.1 Multiple Linear Regression 41

5.2 Descriptor Selection 42

5.3 Cross-Validation 43

6 Molecular Design 46

6.1 Problem Formulation 46

6.2 Tabu Search 48

6.2.1 General Method 48

6.2.2 Implementation 50

7 Polymer Designer 54

7.1 Structure Input 55

7.1.1 Monomer Structure 55

7.1.2 Polymer Structure 58

7.2 Property Input 61

7.3 Descriptor Calculation 62

7.4 Structure Optimization 64

7.5 Modification 64

8 Example: Restorative Dental Polymer Design 65

8.1 Physical and Chemical Properties 69

8.2 Initial Descriptor Selection 70

8.3 Graph Size and Descriptor Repeatability 71

8.4 Statistical Analysis 73

8.5 Optimization 83

8.6 Results 84

8.7 Conclusion 86

9 Conclusions and Recommendations 87

9.1 Conclusions 87

9.2 Recommendations 88

Nomenclature 91

References 94

A Polymer Designer Manual 99

A.1 Installing PD 99

A.2 Opening a Database 100

A.3 Adding and Editing Monomers 101

A.4 Adding and Editing Polymers 110

A.4.1 Manual Polymer Entry 110

A.4.2 Automatic Polymer Entry 114

A.5 Adding and Editing Property Descriptions 115

A.6 The Property/Descriptor Spreadsheet 116

A.7 Molecular Design 117

B Group Library 118

C Class Documentation 121

C.1 Tabu Search Implementation 121

C.2 Using the Tabu Search Implementation 133

List of Figures

1.1 Typical Molecular Graph (HEMA) 2

3.1 HEMA Monomer Graph States 16

3.2 BisGMA Monomer Graph States 17

3.3 Polymer Graph Example 18

3.4 Full graph for poly(HEMA) 21

4.1 BisGMA Graph 29

4.2 Benzene Subgraph 29

4.3 Ester Subgraph 29

4.4 Path Algorithm Tree 31

4.5 Naphthalene 33

6.1 Tabu Search Flow Chart 51

7.1 Monomer Data Entry Form 55

7.2 Graphical Molecule Graph Editing 56

7.3 Molecule Graph Vertex Editing 57

7.4 Polymer Data Entry Form 58

7.5 Graphical Polymer Graph Editing 59

7.6 Polymer Building Form 60

7.7 New Property Entry Form 61

7.8 New Property Entry Form 62

7.9 Column Selection 63

8.1 HEMA 66

8.2 bisGMA 66

8.3 TEGDMA 66

8.4 UDMA 67

8.5 bisEMA 67

8.6 PEGDMA 67

8.7 TMPEDMA 67

8.8 MPE 68

8.9 Histograms for ${}^0\xi$ with Different Polymer Sizes 72

8.10 Pairwise Plots of Tensile Strength and Simple Side Chain Connectivity Indices 76

8.11 Pairwise Plots of Tensile Strength and Valence Side Chain Connectivity Indices 76

8.12 Pairwise Plots of Tensile Strength and Simple Crosslink Connectivity Indices 77

8.13 Pairwise Plots of Tensile Strength and Valence Crosslink Connectivity Indices 77

8.14 Pairwise Plots of Tensile Strength and Other Descriptors 78

8.15 Tensile Strength as a Function of ${}^2\xi_s^v$ 79

8.16 Groups Used to Build Monomers in Tabu Search 84

8.17 Candidate Monomer Structures 85

A.1 Monomer Data Browser 101

A.2 Chemical Structure Editor 102

A.3 Monomer Data Entry Form 103

A.4 Toolbox – Mouse Tab 104

A.5 Toolbox – Graph Tab 107

A.6 Atom Properties Form 108

A.7 Bond Properties Form 109

A.8 Periodic Table Dialog Box 109

A.9 Including Monomers 111

A.10 Editing a Vertex 112

A.11 Editing an Edge 113

A.12 Automatic Build Form 114

A.13 Property Description Editor 115

A.14 PD Spreadsheet 116

B.1 Groups 0 to 15 119

B.2 Groups 16 to 24 119

B.3 Groups 25 to 33 119

B.4 Groups 34 to 42 120

List of Tables

3.1 Polymer Composition 20

8.1 Sample Compositions 68

8.2 Target Properties 69

8.3 Experimentally Measured Properties 70

8.4 Standard Deviations for Different Polymer Sample Sizes 71

8.5 Initial Descriptor Set 74

8.6 Connectivity Index Correlation Matrix 75

8.7 Best Combinations of Descriptors for Tensile Strength and Their R^2 . . 80

8.8 Q^2 Values for 3-fold Cross-Validation 81

8.9 Best Tensile Strength Models Based on Monomer Connectivity 82

8.10 QSPR Summary 82

8.11 Predicted Candidate Polymer Properties 85

Chapter 1

Introduction

The goal of this project is to create and implement a methodology for the design of improved crosslinked polymer materials for specific applications.

Researchers often use knowledge of chemistry and experimentation to improve materials in a trial-and-error process, which requires synthesis and testing of many new polymers. Many properties can be important, and changes to the structure of a polymer may improve some properties while degrading others. Therefore it is often difficult to see how structural changes affect the overall usefulness of a product. The time and money spent designing new materials can be reduced by using statistical analysis of empirical data to develop property models, followed by applying optimization techniques to design materials with desirable properties.

Using modeling and optimization to design new chemicals is known as computational molecular design (CMD). Fro this project, the structure of a molecule is related to its properties using statistically derived models called quantitative structure-property relations (QSPRs). CMD has been applied successfully in numerous cases (see Section 2.3). In this project, CMD is applied to the design of crosslinked random copolymers; the structural complexity of these materials adds significant challenges.

Crosslinked random copolymers are made from two or more types of monomers arranged randomly into a large network structure. Crosslinked polymers are used in a number of applications including restorative dental materials, which are used here to demonstrate the methodology.

Chemical product design shares many similarities with chemical process design. Chemical processes and molecules can both be represented by a graph. In the case of process design, vertices can represent unit operations and edges can represent product streams. For a molecule, vertices represent atoms and edges represent bonds. Figure 1.1 gives an example of a molecular graph for 2-hydroxyethyl methacrylate (HEMA). Graphs will be described in more detail in Chapter 3. Structural constraints for a valid process graph are similar to those for a valid molecular graph. The similarities between process design and molecular design allow molecular design problems to be formulated in a very similar way. The vast amount of research in chemical process optimization can be applied to molecular design. d'Anterroches and Gani (2005) even apply group contribution techniques developed for molecular design to process design.

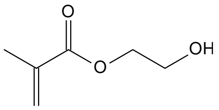


Figure 1.1: Typical Molecular Graph (HEMA)

1.1 Project Overview

Molecular design of crosslinked polymers is an area that has received little if any previous research. The complexity of the crosslinked polymer network makes development of quantitative structure property relations (QSPRs) difficult. Much of the current molecular design research focuses on relatively simple molecules, for which deterministic optimization algorithms can be applied. The purpose of this work is to show that an effective data structure can be developed to store crosslinked polymer structural data; that the data structure can be used to efficiently calculate topological descriptors; that it is possible to develop useful QSPRs for crosslinked polymers; and that a computational molecular design methodology can be successfully implemented for crosslinked polymers.

This project consists of several parts. A literature review is conducted to find relevant background information and methods useful for the design of networked polymers. This background information is provided in **Chapter 2**.

The exact structure of the polymers being studied usually cannot be precisely determined due to its complexity. Therefore, a major part of this project is to develop data structures to store the polymer structures in a way that is useful for calculating structural descriptors in a reasonably efficient manner. The data structures are described in **Chapter 3**.

The second part of the project seeks to provide a means to calculate structural descriptors from the data structures previously mentioned. Many types of structural descriptors are available in the literature and new ones may be created. Several graph theory algorithms are implemented to allow calculation of structural descriptors as described in **Chapter 4**.

Property prediction models are then developed with structural descriptors and experimental data. The property models are often referred to as quantitative structure property relations (QSPRs). Statistical methods are used to create QSPRs and to test their validity. **Chapter 5** describes the statistical methods used.

The QSPRs, structural constraints, and target properties are used to formulate an optimization problem. The problem can be solved using numerous optimization techniques. Tabu Search is used in this project, as described in **Chapter 6**.

Methods for CMD are implemented in the molecular design software developed for this project. An overview of the molecular design software developed for this project can be found in **Chapter 7**, and a detailed manual is provided in **Appendix A**. The exact structure of crosslinked polymers is often complicated and not known. The goal of the software is to provide a simple method of entering and storing polymer structures and to allow many types of structural descriptors to be calculated easily. The software allows many new methods to be tried with minimal effort. Since the types of descriptors and form of the QSPRs are not known beforehand, extensive numerical experimentation may be required; this is facilitated by the software. **Appendix C** provides useful documentation for expanding the software.

The methodology and software were applied to the design of new adhesives for restorative dental applications. **Chapter 8** provides a detailed description of the method and results.

Chapter 9 contains conclusions and recommendations for further work.

Chapter 2

Background

This section provides background information, and a review of relevant literature. Details of calculations will be presented in subsequent chapters.

2.1 Structural Descriptors and Property Prediction

Structural descriptors provide a way to numerically describe the structure of a molecule. The descriptors are used to link the structure of a molecule to its properties. Some simple examples of descriptors are molecular weight and molecular formula.

Many structural descriptors can be obtained from structural graphs of molecules. Graphs consist of a set of vertices and a set of edges. Edges are defined by two endpoints from the vertex set (West, 2001). The structure of a molecule can be represented by a graph in which the atoms are vertices and the edges are bonds. The graphs used for calculating descriptors of organic chemicals usually have the hydrogen atoms removed, and are called hydrogen suppressed graphs (Bicerano, 2002).

The types of descriptors that can be determined from graphs are called topological descriptors. The topology of a molecule describes how the atoms of the molecule are connected (Bicerano, 2002). The actual relative location of the atoms in the molecule is not important in calculating topological descriptors. Descriptors that depend on the locations of the atoms in space are called geometric descriptors. Accurately determining the geometry of large molecules is a difficult computational problem. Although geometry determination is possible, the computation time required makes it unfavorable for inclusion in an optimization routine, which may need to evaluate hundreds or thousands of molecules. This project focuses on topological descriptors.

One of the first molecular descriptors developed that is based on molecular graphs is the Wiener Index (Wiener, 1947a,b). Wiener studied isomers of alkanes, so the effects of number and type of atoms could be removed, and only the arrangement of the atoms needed to be considered. Two structural parameters were used, the polarity number and the path number. The polarity number is the number of pairs of carbon atoms separated by three bonds. The path number is the sum of the distance between every pair of atoms. Distance in terms of molecular a graph is the number of bonds on the shortest path between two atoms. The path number (known as the Wiener Index) provides a measure of the compactness of a molecule, and was shown to be useful in predicting the boiling points of paraffins (Wiener, 1947b). Other studies have have found the Wiener Index to be related to numerous other properties (Rouvray, 1986).

Connectivity indices have been widely used to predict molecular properties (Kier and Hall, 1986; Bicerano, 2002). The definition and methods for calculating connectivity indices are described in Section 4.2.2. The first connectivity index was developed by Randić (1975). Randić developed an index to quantify branching in molecular structures. Like Wiener, Randić found the branching index could be used to predict to the boiling point of alkane isomers.

Kier and Hall (1986) extended Randić branching index ideas into a more general set of connectivity indices and used them to predict properties of organic compounds. A significant part of their work was to relate connectivity indices to properties of interest in pharmaceutical design.

Connectivity indices have also been applied to predicting polymer properties. Bicerano (2002) used zeroth- and first-order connectivity indices to predict several volumetric, thermodynamic, optical, electrical, and mechanical properties of isotropic amorphous linear (not crosslinked) polymers. In addition to connectivity indices, correction terms similar to group contributions were used for some properties. For most properties, between 100 and 200 data points were used and correlation coefficients of greater than 0.99 were obtained. The correlations developed cannot be directly applied to this project because no crosslinking was considered; however, the work of Bicerano (2002) shows that connectivity indices are useful in predicting polymer properties.

Another important topological descriptor is the shape index (Kier, 1986, 1987). The shape index (${}^2\kappa_\alpha$) attempts to quantify the shape of a molecule by comparing the number of two-bond fragments to the maximum number of two-bond fragments in a star shaped molecule having the same number of atoms and the minimum number found in the linear molecule. The shape index contains a correction factor for different atomic sizes by measuring a proportional deviation from the radius of carbon in an sp^3 hybridization state. The shape index was shown to relate to properties in some biological systems.

In addition to the descriptors already mentioned, numerous other topological descriptors exist, many of which may be useful for this project. Todeschini and Consonni (2000) provide an extensive listing of molecular descriptors.

Group contributions are another common way to describe molecular structure. In group contribution methods structures are broken down into fragments. The number of each type of fragment can be used to predict properties. Group contribution methods are commonly used for property prediction. A few notable methods are described here.

A common group contribution method is UNIFAC (UNIQUAC Functional-group

Activity Coefficients) (Fredenslund *et al.*, 1975). UNIFAC breaks chemicals up into a number of small functional groups. Each group has a volume and area parameter associated with it. In addition, group interaction parameters were used to quantify binary interactions between two types of functional group. The UNIFAC model is useful in designing liquid-liquid separation processes, when good experimental data is not available. The UNIFAC groups have been applied in numerous other prediction methods.

González *et al.* (2007) extended the UNIFAC method using connectivity indices to estimate missing group contributions. Their method is used to extend the applicability of UNIFAC in predicting vapor-liquid equilibria. Joback and Reid (1987) used a group contribution method consisting of about 40 simple groups to estimate liquid viscosity and some thermodynamic properties of small molecules. Group contribution methods have been extensively used to predict the properties of pure components and mixtures of small chemicals, but they have also been applied to polymers. Van Krevelen (1997) provides group contribution methods for predicting numerous optical, electrical, magnetic, mechanical, and acoustic properties of non-crosslinked homopolymers.

In this project, crosslinked polymers are studied. The structural descriptors described so far are useful, but crosslinking has a very large effect on the polymer properties. Some measure of the degree of crosslinking must also be considered in the structure description. Bicerano *et al.* (1996) correlated the crosslink density to glass transition temperature in randomly crosslinked polymers. Van Krevelen (1997) also noted the importance of crosslink density on the properties of polymers. Bicerano (2002) states the the correlations based on connectivity indices developed in that work are not applicable to crosslinked polymers due to the overwhelming effect of crosslinking. This work uses a explicit description of crosslinking to develop new QSPRs, so the optimization formulation is capable of designing crosslinked polymer systems.

2.2 Statistical Methods for QSPR Development

This section provides an overview of statistical techniques that may be useful in the development of new QSPRs. For the most part, these methods are quite well known and widely used.

The simplest method used to relate descriptors to properties is multiple linear regression (Draper and Smith, 1966). The descriptors make up a set of predictor variables, and the property is the response variable. The regression equation will be a linear combination of the predictor variables. It is also possible to add predictor variables by transforming some of the existing variables. By adding the square of a descriptor, for example, the regression equation could be a parabola. Exponential functions as well as other functional forms can be fit using linear regression by adding transformed variables. The response variable can also be transformed.

Partial least squares (PLS) is commonly used in computational chemistry (Mevik and Wehrens, 2007). PLS is useful when large numbers of predictor variables are available. The basic idea of PLS is similar to principal component analysis (PCA). The variation in the predictor and response variables is considered and new predictor variables (components) are created using linear combinations of the original predictor variables. The first component is in the direction of the largest variation. The next component is orthogonal to the first and accounts for the second most variation and so on. Regression can then be performed using some number of the first components.

Nonlinear regression (Bates and Watts, 1988) is sometimes useful in the development of QSPRs. Nonlinear regression is also performed by minimizing the sum of the squared residuals, but the parameters being determined are not linearly related, so an iterative optimization method is usually used. Nonlinear regression is most useful to estimate parameters when a theoretical or semi-empirical property model is available.

Once reasonable QSPRs have been found using a regression technique, model validation must be done. Adding parameters to a model will always improve how well a model fits the available data, but may decrease prediction accuracy for new data. A good way to test the validity of a model is through cross-validation, where the data is split into training sets and test sets. In k-fold cross validation, the data is split into k approximately equally sets, and each of the sets is used in turn as the test set (Efron and Tibshirani, 1993). This provides an error estimate for each data point when it is not used in the model data.

Using cross-validation, a statistical value known as Q^2 can be calculated, which is similar to the multiple correlation coefficient (R^2) (Quan, 1988). Q^2 provides some estimate of how well a model works at predicting for new data points not used to make the model. Ideally Q^2 would be close to R^2. A low Q^2 could indicate that the model has too many parameters and is over-fitting the data.

2.3 Molecular Design

This section provides an overview of molecular design techniques and examples of how molecular design has been applied successfully. Molecular design has been applied using both deterministic and heuristic optimization methods. CMD has been applied to polymers, but little information is available for design of general crosslinked polymers. Some examples of CMD studies are provided here.

Gani *et al.* (1991) used group contributions to implement molecular design for chemicals such as solvents for separation processes. The method uses a range of acceptable values for a set of target properties. New molecules were created by attaching the groups from a group contribution method together subject to feasibility constraints. Molecules were then screened to select those that met the target property requirements. Molecules were further evaluated using process simulation.

Sahinidis *et al.* (2003) used a branch-and-bound optimization technique to design new refrigerants. The boiling point, critical temperature, critical pressure, heat capacity, and heat of vaporization were estimated using the Joback and Reid (1987) group contribution method. The problem was formulated as an integer programming problem. Molecules were constructed from the group contribution groups, and a set of constraints was written to ensure molecules were feasible. The problem was formulated such that the global optimum was found. The study identified some novel possible substitutes for Freon 12. The results were sets of groups used to construct a molecule.

Karunanithi *et al.* (2005) designed solvents and solvent mixtures having desirable properties for a number of applications. Solvent properties were estimated using group contribution methods such as UNIFAC. The problem was formulated as a mixed-integer nonlinear optimization problem. The problem was then broken down into subproblems, so that some sets of constraints could be solved to limit the search space before solving the final optimization problem. The method is demonstrated by designing a solvent for a liquid-liquid extraction process.

Allcock (1992) discussed some of the issues associated with the design of polymers for specific applications, and encouraged a rational design approach. The article points out the need to develop structure property relations. Several examples are presented that show how altering various aspects of the structure affect the properties of a polymer. The article discusses backbone and side chain modifications as well as crosslinking.

Venkatasubramanian (1994) and Sundaram and Venkatasubramanian (1998) examined the computational molecular design of linear polymers using a genetic algorithm. The polymers are assembled from groups, and the properties are predicted using Van Krevelen group contributions (Van Krevelen, 1997).

Maranas (1996) used group contribution methods for molecular design of straight-chain polymers. A reformulation of the molecular design problem is used so that the problems can be written as mixed-integer linear programs, and solved using existing optimization software. This allowed the best solutions to be enumerated.

Camarda and Maranas (1999) used zero- and first-order connectivity indices to develop new structure property relations for prediction of the properties of straight-chain polymers. The properties studied were: heat capacity, glass transition temperature, cohesive energy, refractive index, and dielectric constant. The property models where used to formulate a convex mixed-integer nonlinear optimization problem to find new polymers meeting specific target properties. A deterministic algorithm was used to find a provably optimal solution.

In this project, Tabu search is used to solve the optimization problem (Glover, 1990b). Although Tabu search is a well established optimization method in other areas, Lin *et al.* (2005) first applied it to computational molecular design. That work considered the design of transition metal catalysts, and used QSPRs based on connectivity indices. Tabu search is described in detail in Chapter 6.

2.4 Dental Polymers

The molecular design framework developed here was applied to the design of restorative dental adhesives. This section provides a description of the types of materials being used and why improved materials are needed.

Polymer-based materials are increasingly being used in place of dental amalgam. Polymer-ceramic composites are more aesthetically pleasing than amalgam, and there is also concern over the environmental impact of using materials that contain mercury. Dental composites, however, are significantly less durable than dental amalgam

(Bernardo *et al.*, 2007).

Crosslinked methacrylate copolymers, typically containing 2-hydroxyethly methacrylate (HEMA) and 2,2-bis[4(2-hydroxy-3-methacryloyloxy-propyloxy)-phenyl] (bisGMA), are used in dental composites and adhesives. Methacrylate polymers are susceptible to degradation of ester bonds, especially in the presence of esterase enzymes (Finer and Santerre, 2004). Design of a completely new type of material is not desirable due to the vast amount of previous research on use and clinical application of methacrylates. Instead, this project focuses on making improvements to existing materials.

The crosslinked structure of dental polymers presents a major challenge for molecular design. The network structure of crosslinked polymers may change dramatically due to processing conditions, and this structure has a large effect on properties (Ye *et al.*, 2007a). The sensitivity of these polymers to processing conditions means that many property values reported in the literature are inconsistent, and are therefore undesirable for the development of QSPRs. A new set of consistent experimental polymer property data is therefore needed.

The improvement of dental materials is an important area of ongoing research. The application of the molecular design framework presented here can speed development of such materials, so dental materials design provides an excellent test case for this work.

Chapter 3

Polymer Structure

Numerical descriptors of polymer structure need to be calculated for a large number of candidate structures in the CMD process, so a data structure is needed which is capable of storing structural information for crosslinked polymers. The data structure should allow the efficient calculation of many types of topological descriptors. The most basic information needed to define a random crosslinked polymer structure includes: the chemical structure of the monomers that form the polymer, the composition of the resin (material before polymerization), and the degree of conversion of the monomer functional groups. Additional information may also be useful to define the polymer structure more accurately, such as information about intramolecular reactions during polymerization or unequal reactivities of the different functional groups. With this information, a representative section of polymer can be generated and used for calculation of structural descriptors.

Only topological descriptors are considered in this work. Topological descriptors are based on the types of atoms and bonds present and how they are connected. Although atom coordinates are stored, no attempt is made to generate an accurate geometric representation of the molecules, and no geometric descriptors are used. This avoids any requirement to minimize the Gibbs free energy of a structure, which would be too time consuming within a molecular design method. Topological descriptors are often sufficient to predict properties (Van Krevelen, 1997; Bicerano, 2002), and are much more efficient to calculate than geometric descriptors. Calculation efficiency is of particular importance in the optimization phase of molecular design.

In this work, polymer structures are stored using three types of graphs: monomer, polymer, and full. Monomer graphs are used to store the chemical structure of monomers. Each vertex represents an atom or connection point between monomers, and each edge represents a chemical bond. Polymer graphs contain a representative section of the overall polymer structure. Each polymer graph vertex represents a monomer, and each edge represents a bond between monomers. Polymer graphs can be generated systematically using a set of rules. Full graphs describe a representative section of the chemical structure of polymer networks. The full graph is formed by replacing the monomer vertices in the polymer graph with the atoms and bonds of the monomers. Each type of graph is useful for different types of calculations. Monomer graphs are useful in forming the polymer structure, and may be used to predict properties of the unreacted monomers such as viscosity. The polymer and full graphs contain similar information. However, when constructing the polymer structure, it is much easier to deal with monomer vertices than a complete chemical structure. The polymer graph gives a clear picture of the arrangement of monomers. Easy access to the identities of the monomers in the polymer structure is lost in the full graph.

These data structures allow relatively straight-forward calculation of most descriptors, and therefore allow numerical experiments to be performed quickly. Since the types of descriptors and structural information that will be needed for QSPRs is not known beforehand, ease of use is given primary importance at this stage. More efficient methods can be created once QSPRs are obtained for a particular system.

3.1 Monomer Graphs

Monomer graphs have two types of vertices, atom and connection points. Atom vertices simply represent atoms in the chemical structure of the monomer. In the polymerization process, monomers react and bond together. Connection point vertices are dummy atoms that represent an atom in a neighboring connected monomer. Each connection vertex is labeled so that specific bonding patterns can be specified. For example, monomers can be connected head-to-tail.

Monomer graphs are capable of storing the structure of a monomer in various states of reaction. Figure 3.1 shows a monomer graph for 2-hydroxyethly methacrylate (HEMA) and Figure 3.2 shows the graph for 2,2-bis[4(2-hydroxy-3-methacryloyloxy-propyloxy)-phenyl] (bisGMA). The Xx atoms are dummy connector atoms used for polymerization reactions. The vertices and edges of the graphs have labels representing a functional group and state. Each atom and bond is part of some functional group and a state of that functional group. Functional groups can have one or more states which represent the functional group unreacted or reacted in various ways. The system is very useful because it allows the monomer structure to be entered once, but allows potentially numerous reacted structures to be generated. In many cases, monomer properties such as viscosity or solubility are important, so the original unreacted monomer structure remains available.

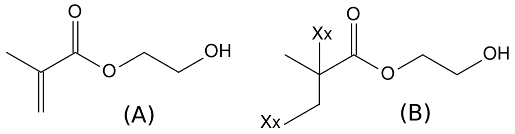


Figure 3.1: HEMA Monomer Graph States

The functional group system has no effect on the efficiency of graph algorithms. The vertices and edges of a graph are re-indexed when a functional group state changes. The indices of the vertices and edges that are in the current state are set lower than

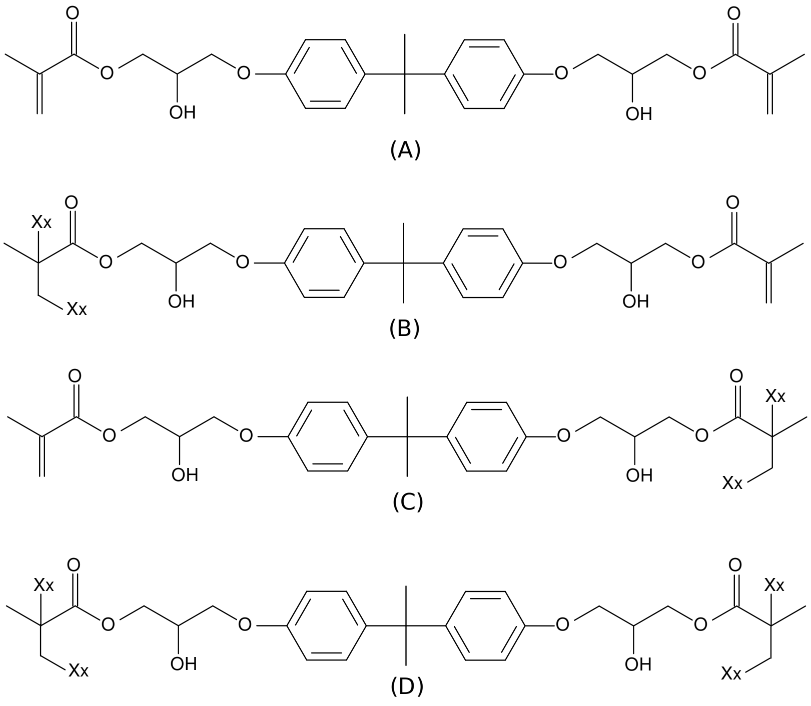


Figure 3.2: BisGMA Monomer Graph States

the vertices and edges that are not in the current state, so out-of-state vertices and edges may be ignored by algorithms.

Various properties are connected to each atom vertex, like the coordinates, atomic number, and charge. The coordinates are mostly used for displaying graphs in the software (see Chapter 7), however as part of some future work they could also be used in geometric descriptor calculation.

3.2 Polymer Graphs

Polymer graphs describe the bond patterning between monomers. Each vertex represents a monomer, and an edge represents a bond between monomer units. For simple polymers, such as linear block copolymers, polymer graphs can be constructed manually by arranging a pattern of monomers. It is more convenient to systematically generate the structure in more complicated cases. The overall design framework is equally effective whether a simple set of rules or a complicated molecular simulation is used to generate the polymer structure. The size of the generated polymer depends on what is needed to adequately represent the structure. Graph size will be examined more closely in Chapter 4. An example polymer graph is given in Figure 3.3.

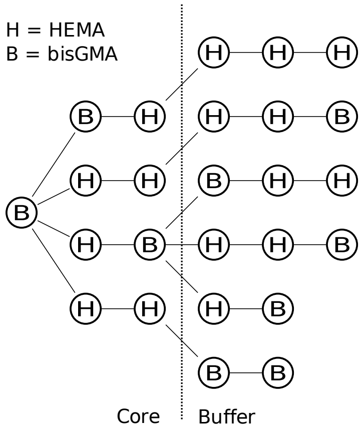


Figure 3.3: Polymer Graph Example

Simple polymers have a well-defined regular repeating pattern; however, crosslinked random copolymers form a network with no obvious repeat unit. A large representative section of polymer is generated to solve problems associated with descriptor calculation. Crosslinked polymer networks are normally treated as infinite, but the polymer graphs must be finite. Representative polymer sections will have many cut chains. The concept of a core and buffer section is used to minimize the impact of the cut chains on descriptor calculation. Calculations are carried out on the core, while some number of buffer monomers separates the core from the cut chain ends. The way the buffer section is used depends on the type of descriptor being calculated. Examples of descriptor calculation are provided in Chapter 4.

The data structure described is not dependent on the method used to generate polymer graphs. The simplest adequate method is probably best, so a simple method will be the starting point. Clearly, there is no point in using a very complicated method of generating polymer structure if a simpler method works well. A number of assumptions are made to generate the polymer structures. The exact crosslinked structure of the polymer network is not known, but the degree of conversion is measured experimentally. The degree of conversion is the fraction of carbon-carbon double bonds that react in the polymerization reaction. It is assumed that all double bonds in the methacrylate monomers have equal reactivity, and that no intramolecular reactions occur during polymerization. These assumptions work well with a limited knowledge of the reaction kinetics and crosslinking structure. Generation of a more accurate structure may be an area for study as more data is collected. The following example shows how a structure is generated for a polymer consisting of 45 wt% HEMA and 55 wt% bisGMA. The degree of conversion of carbon-carbon double bonds measured experimentally for this system is 76.93%. This example is taken from experimental data collected as part of a summer research program at the University of Missouri Kansas City (UMKC) Dental School.

First, the composition of the resin must be converted to mole fraction. The molecular weight of HEMA is 130.143 g/mol and the molecular weight of bisGMA is 512.599 g/mol. Assuming there are 100g of resin, there are $45/130.143 = 0.34577$ moles of HEMA, $55/512.599 = 0.10730$ moles of bisGMA, and $0.34577 + 0.10730 = 0.45307$ moles of resin. The mole fraction composition is 0.76317 moles of HEMA per mole of resin, and 0.23683 moles of bisGMA per mole of resin.

The second step is to calculate the amount of monomer in each state by assuming that the functional groups react with equal probability. Figures 3.1 and 3.2 show the states of the HEMA and bisGMA graphs. Based on the degree of conversion there is a 76.93% chance that a given double bond reacts. This means 23.07% of HEMA is in state A (see Figure 3.1) and 76.93% of HEMA is in state B. The portion of bisGMA existing in state A is the probability that neither functional group reacts which is 0.2307 × 0.2307 = 5.32%. The percent of bisGMA existing in state B is the probability that the first functional group reacts and the second does not, which is 0.7693 × 0.2307 = 17.75%. The portion of bisGMA in state C is 17.75% just as for B, and following the same pattern the portion of bisGMA in state D is 59.18%. The bisGMA portions add up to 100%.

The portions of the monomers in each state can be used to find the mole fraction of each monomer state in the polymer. Table 3.1 shows the composition. Refer to Figures 3.1 and 3.2 for the monomer states. The unreacted monomer portion is contained in the final polymer, however it does not become a part of the polymer network. Unreacted monomer may act as a plasticizer and affect the monomer properties, so quantifying its presence is important. To construct the polymer network graph, unreacted monomer must be removed. The monomer composition of the polymer network is also shown in Table 3.1.

Table 3.1: Polymer Composition

Monomer	Mole Frac. (total)	Mole Frac. (network)
HEMA (A)	0.176	0.000
HEMA (B)	0.587	0.723
bisGMA (A)	0.013	0.000
bisGMA (B)	0.042	0.052
bisGMA (C)	0.042	0.052
bisGMA (D)	0.140	0.173
total	1.000	1.000

Probabilities based on the composition are used to generate a polymer graph describing the connectivity of the monomers. The graph is a tree because it is assumed that no intramolecular reactions occur; however, polymer graphs are not necessarily trees. Trees are graphs that contain no cycles (rings). The assumption of no intramolecular reactions leads to the simplest method of generating polymer structure, while still providing sufficient information for the prediction of important properties. Figure 3.3 shows a small graph for this example.

3.3 Full Graphs

Full graphs are large graphs which show the detailed chemical structure of a representative section of polymer. The full graph is obtained by combining the information in the monomer and polymer graphs, and can be used to calculate most topological structure descriptors in a straightforward way. While the full graph provides a detailed chemical structure, information about the monomer connectivity provided in the polymer graph is difficult to extract. Figure 3.4 shows a small example of a full graph for poly(HEMA).

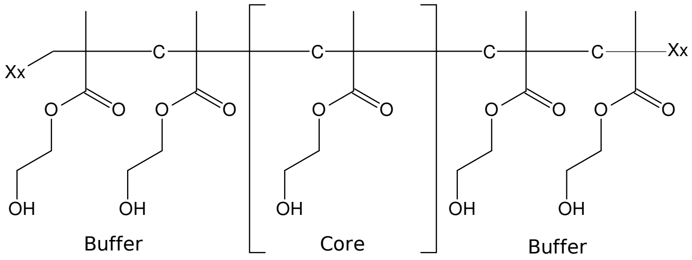


Figure 3.4: Full graph for poly(HEMA)

The simplicity of using a fixed representative section of polymer greatly accelerates implementation of methods for computation of structural descriptors needed for property prediction.

3.4 Software Implementation

This section provides an overview of how the structure graphs are implemented. Further detail is provided in Appendix C and in the source code of the software.

The basic structure of the three types of graph is the same. The different graph types can be obtained by adding some additional properties to the graph, vertices, and edges of the basic graph.

3.4.1 Vertices and Edges

The graphs consist of a set of properties for the graph, a list of vertices, and a list of edges. The vertices and edges each consist of a data structure that stores some important information. The basic graph contains a list of functional groups and their current states, an array of pointers to vertices, and an array of pointers to edges.

Vertices contain the following information:

- UID (an integer unique identifier);
- index (position of the vertex in the vertex pointer array);
- x, y, z (Cartesian coordinates);
- fGroup (the functional group it is in);
- fState (the state of the functional group it is in);
- isCore (whether vertex is in the core);
- isConnector (whether a vertex is a connector); and
- connectorLabel (a unique label for each connector).

The UID of a vertex is an integer that never changes. It is used to properly connect edges when vertex pointers change, for example, when a graph is copied or loaded from a file. The index is an integer that sequentially labels vertices starting with 0. The indexes of the vertices change when a vertex is deleted or the state of a functional group changes. A pointer to each vertex is stored in an array the index corresponds to the array index. Assignment and use of the index is detailed in Section 3.4.3. The polymer design software contains a visual graph editor, which is the primary purpose for storing the coordinates of each vertex. The variables fGroup, fState and isCore are vertex labels whose use will be described in the following sections. The variable isConnector is set to true if a vertex is a dummy vertex representing a connection to another graph, and connectorLabel is a label for connectors.

Edges contain the following information:

- UID;
- index;
- fGroup;
- fState;
- isCore;
- v1_ptr (pointer to the first endpoint vertex);
- v2_ptr (pointer to the second endpoint vertex);
- v1_uid (UID of the first endpoint vertex);
- v2_uid (UID of the second endpoint vertex);

The information contained in an edge is similar to that for a vertex. The main difference is that the pointers to the end-point vertices are included. If a directed graph is needed, edges can be considered to be directed from vertex 1 to vertex 2. The UIDs of the vertex endpoints are stored, because if the graph is copied the pointers change, and the UIDs can be used to update the pointers.

Monomer graphs can be built by adding an atomic number label to the vertices and a bond type label to the edges. Polymer graphs can be created by adding a monomer type to the vertices, and a bond type to the edges. In addition, polymer graphs need connector labels for each endpoint vertex to indicate which connector vertex in a monomer the bond attaches to. Full graphs are the same as monomer graphs, just larger.

3.4.2 Functional Groups and Graph State

The chemical structure of monomers changes when they react to form polymers. At least some of the monomers in crosslinked polymers have multiple functional groups that allow one polymer chain to be joined to another to form a network. Drawing and storing structures for every possible configuration of all of the monomers could be very tedious and confusing. For that reason, a system of functional groups and states is devised, which allows a graph to change structure.

Graphs contain functional groups. These functional groups are only used in monomer graphs and correspond to groups of atoms and bonds that may react in a polymerization reaction. Each functional group may have one or more states. The states correspond to a given monomer reaction path (see Figure 3.2 for example). The main part of the monomer which does not change is labeled as functional group zero.

The graph contains a list of all of the functional groups and their current state. If a vertex or edge is not in the current state of its functional group, it is considered to be out-of-state and given an index at the back of the array, so it can be ignored. If an edge is marked in-state but one of its endpoint vertices is out-of-state, it is also considered to be out-of-state; this avoids edges which are not connected to two vertices, meaning all bonds must connect two atoms.

3.4.3 Indexing, Adjacency Matrices, and Adjacency Lists

The index of a vertex or edge is a typical way to reference these objects in a graph theory algorithm. The polymer design software has a graph editor that allows vertices to be deleted, and the state of functional groups can change, so the index to the vertices and edges cannot remain constant. This section describes how indices are assigned and how they are used for adjacency lists and matrices.

Edges and vertices are sorted similarly in the software, so only vertex sorting is described. The graph contains an array of pointers to vertices. The vertex pointers are sorted in the array and the array index of the pointer is the vertex index. The index is stored in the vertex data structure so it can be found quickly without having to search the array.

The first step is to mark each vertex and edge as being in-state or out-of-state as described previously. The out-of-state vertices are sorted to the back. Sorting the out-of-state vertices and edges to the back allows them to be treated as if they do not exist in graph theory algorithms. It is possible that connector vertices may be treated differently in some cases, so the connector vertices are sorted to the back of their group (in-state or out-of-state). The vertices are then sorted by functional group, and finally by UID to ensure they are sorted the same each time.

Two vertices are adjacent if an edge connects them. An edge is incident to a vertex if the vertex is one of its endpoints. Adjacency matrices, adjacency lists, and incidence lists are common data structures used in graph algorithms; once the graph is indexed these structures can be made as needed. If n is the number of vertices in a graph, an adjacency matrix is a square $n \times n$ matrix. The index of each row and column correspond to the vertex indices. The matrix is usually a binary matrix where a 1 indicates that two vertices are adjacent. If the graph is undirected, the matrix is symmetrical about the diagonal so $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$. In this implementation, the edge index plus one is used to indicate a connection; this way the connecting edge is easy to find. One is added to the index because indexing starts at zero, and zero indicates no connection.

Adjacency lists are a list of adjacent vertex indices associated with each vertex. Incident lists are a list of all incident edge indices associated with each vertex.

With the graphs properly indexed and some basic data structures available descriptor calculation can be considered.

Chapter 4

Structural Descriptors

There are many numerical descriptors of chemical structure which can be used to predict the properties of polymeric systems. A few common descriptors are examined here. Connectivity (Kier and Hall, 1986; Bicerano, 2002) and shape indices (Kier, 1986, 1987) have been used successfully in QSPRs and require relatively little computational effort. Group contributions are also commonly used to predict the properties of chemicals including polymers (Van Krevelen, 1997). Identification of structural groups is also important in prediction methods aside from group contributions (Bicerano, 2002). Of course, additional descriptors are needed to describe crosslinking. Most topological descriptors calculations can be easily implemented using the polymer structure information described in Chapter 3.

A representative section of polymer is generated, so descriptors can be calculated as they would for any small molecule. A feature of this system is the use of core and buffer sections of the polymer network. Calculation of descriptors often requires information about neighboring polymer sections; the buffer provides this information. The accuracy and repeatability of the descriptor calculations for random copolymers depend on the size of the representative section. Larger sizes cause the calculations to take longer. The time required for descriptor calculation in analysis of experimental polymer data is generally insignificant, since the number of polymers for which properties are measured is generally less than 200. However, descriptor calculation time may be a concern when using optimization to find new structures. An optimization routine may evaluate thousands of polymer structures while searching for those likely to have desired properties.

The system for descriptor calculation employed in this work saves a significant amount of time in preliminary analysis of structural descriptors and property models. Many different descriptors can be evaluated without the need to deal with complex probability calculations. This allows flexibility in terms of the structural descriptors and property models to be used. The following sections provide examples of how descriptors can be calculated. Numerous other descriptors can be added easily.

4.1 Graph Algorithms

Several graph algorithms useful for descriptor calculation are implemented in the polymer design software to expedite descriptor calculation. An overview of these algorithms is provided here.

4.1.1 Subgraph Isomorphism

A subgraph isomorphism algorithm allows specific chemical substructures to be identified; it is important in group contribution methods, and in identifying functional groups in other types of descriptor calculations. Subgraph isomorphism may also be used to identify similarities between molecules.

The Ullmann (1976) subgraph isomorphism algorithm is used with some minor modification to identify chemical substructures. The result of the algorithm is a set of mappings of a smaller (or same size) graph onto a larger graph. A description of subgraph isomorphism and modifications to the algorithm is provided.

As an example, consider the graph for bisGMA (Figure 4.1), and two subgraphs (Figures 4.2 and 4.3). The numbers in gray are vertex indices. In the ester subgraph, the atoms labeled with black numbers are dummy atoms.

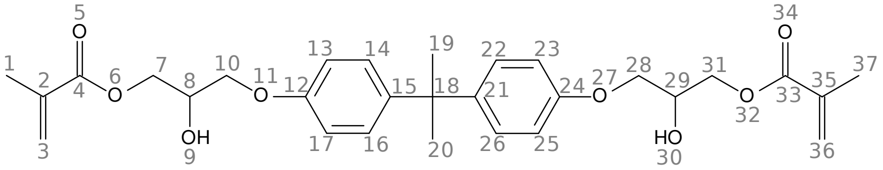


Figure 4.1: BisGMA Graph

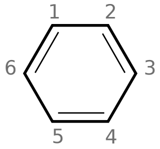


Figure 4.2: Benzene Subgraph

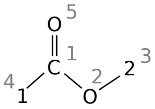


Figure 4.3: Ester Subgraph

In the original implementation of the algorithm, atom and bond types are not considered. Searching for the benzene subgraph (Figure 4.2) in bisGMA (Figure 4.1) reveals the six-membered ring can be found 24 ways in bisGMA. The subgraphs are found by mapping the first atom in the benzene ring to any one of the 6 atoms in a ring from bisGMA. The rest of the atoms can be paired up proceeding clockwise or counterclockwise. Instead of finding 24 mappings, it would be more useful just to find the two rings in bisGMA.

Usually, in a group contribution method, the molecule is broken into fragments and atoms can only be in one fragment. The first modification to the isomorphism algorithm is to allow groups to be marked as having been used when a mapping is found. Only two mappings of benzene onto bisGMA are found if reuse is disallowed.

The atom and bond type are usually important when considering molecular graphs. The next modification of the algorithm only allows atoms or bonds of the same type to be paired. With this modification, the benzene ring will be aligned so that the pattern of double and single bonds also match.

The degree of a vertex is the number of adjacent vertices. In the subgraph isomorphism algorithm, a condition for two vertices to be paired is that the degree of a vertex in the subgraph has to be less than or equal to the degree of the vertex in the larger graph. The last modification of the algorithm is to allow an option for requiring the degree to be equal. If that is done, there will be no instances of Figure 4.2 found in bisGMA. In fact, if only connected graphs are considered, Figure 4.2 will only be a subgraph of itself.

To see how requiring the degree of vertices to be equal is useful, consider the ester group in Figure 4.3. If we want to find this exact group in bisGMA, we can require degree equality, and ignore the dummy atoms after using them to calculate the vertex degrees. The ester group maps to bisGMA in two places: 1, 2, 5 to 4, 6, 5 and 1, 2, 5 to 33, 32, 34.

The isomorphism algorithm is implemented such that it can be used in all the ways described above. The result is all the mappings of the subgraph onto the larger graph, and a count of how many times a subgraph was found. The algorithm can be used in group contribution methods or in finding atoms that are part of a specific structure.

4.1.2 Path Finding

A path in a graph is a set of vertices where each vertex is adjacent to the previous one and none are repeated. An important part of calculating many types of topological indices is finding paths. For example, the shape index (${}^{2}\kappa_{\alpha}$) requires finding all paths of length two. In many cases, specialized algorithms can be used to calculate descriptors, but a general algorithm to find paths can save time when trying to test large numbers of descriptors.

The path finding algorithm is based on a breadth first search (BFS) (West, 2001). Suppose a set of connectivity indices is to be calculated up to fifth order. All of the paths up to length 5 need to be found in a molecule. To get an idea of how this algorithm works, again consider bisGMA as shown in Figure 4.1. The algorithm builds a tree by first finding all the vertices which are one edge from the original vertex, then to all the vertices that are two edges away from the original vertex, and so on. As new vertices are added to the tree, a check is made to ensure they are not already ancestors, so cycles are avoided. The tree is shown in Figure 4.4 for paths starting at vertex 15. The smaller gray numbers are the order in which the vertices are added.

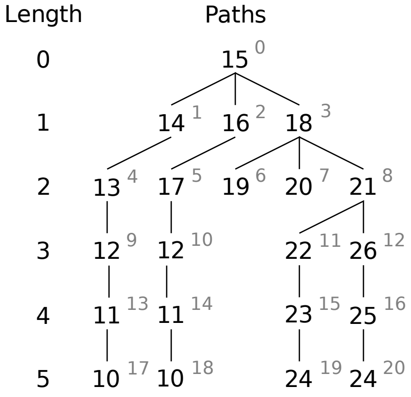


Figure 4.4: Path Algorithm Tree

To find the paths of length 5, start at the vertices on level 5 of the tree and trace back to the top of the tree. Likewise any path shorter than length 5 is also available.

The algorithm can be run in a loop over all of the vertices to find all the paths of length 5 or less. If every vertex is used as a starting point for the algorithm, all of the paths will be found twice, forward and backward; to avoid this, paths where the starting index is greater than the ending index can be ignored. If no paths of length 5 exist, the algorithm terminates at the longest paths, so level 5 would be empty.

4.1.3 Shortest Path

The path finding algorithm can be used again with minor modification to find the distance or the shortest path between two vertices. The algorithm starts from the first vertex and terminates after the level where it reaches the second vertex. The reason the algorithm does not terminate immediately upon finding the second vertex is that there may be more than one path of the same length. For example, the shortest path from 15 to 12 in Figure 4.1 has a length of 3, and there are two ways to get there: $15 \rightarrow 14 \rightarrow 13 \rightarrow 12$ and $15 \rightarrow 16 \rightarrow 17 \rightarrow 12$. The distance and the shortest paths are returned by the algorithm. The shortest path is required in some descriptor calculations. Some of the correlations in Bicerano (2002) require the shortest path across the backbone of a polymer, so finding the shortest path is useful in descriptor calculation.

4.1.4 Cycle Finding

A cycle can be made from a path by adding an edge between the first and last vertices of a path. A cycle is equivalent to a ring in a chemical structure. It is often important in descriptor calculation to know if an atom is a member of a ring.

The fundamental cycles of a graph can be found by looking at a spanning tree and co-tree of the graph (Gibbons, 1985). A spanning tree of a graph is a connected subgraph that contains every vertex, but no cycles. The co-tree consists of edges not in the spanning tree. Fundamental cycles are a set of cycles in a graph where all other cycles can be found by combining fundamental cycles. Figure 4.5 shows naphthalene. Naphthalene has two fundamental cycles 1 → 2 → 3 → 4 → 5 → 6 and 5 → 6 → 7 → 8 → 9 → 10. The larger cycle with all ten vertices can be obtained by combining the edges of the fundamental cycles and removing the common edges.

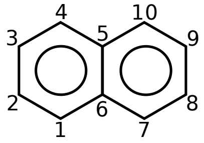


Figure 4.5: Naphthalene

To find the fundamental cycles, a spanning tree and co-tree are created using a breadth first search. Each edge in the co-tree shows the location of a fundamental cycle. The cycles can be found by finding the shortest path in the spanning tree between the two vertices that make up a co-tree edge.

4.1.5 Block Finding

Blocks of a graph are subgraphs with no cut edges or vertices (West, 2001). A connected graph is a graph were a path can be found between any two vertices in the graph. When a cut-edge or cut-vertex is removed from a graph, the graph becomes disconnected. Gibbons (1985) presents an algorithm to find the blocks of a graph.

The block finding algorithm provides a simple way to find the backbone of a polymer or reacted monomer. It is known which atoms of a monomer connect to other monomers in a polymerization reaction. By adding an extra pseudo-edge between the two atoms at each end of the backbone, the atoms in the same block as the two atoms at the ends of the backbone are also in the backbone. The side-chain groups then all have a cut edge that separate them from the backbone.

4.2 Descriptor Calculation

This section describes a few descriptors that can serve as a starting point for developing QSPRs. Numerous other descriptors can be calculated quite easily as needed.

4.2.1 Molecular Weight and Atom Type

Molecular graphs are usually written with hydrogen atoms omitted. By looking at the atomic numbers, the charge on the atoms, the number of bonds, and the types of bonds, the locations of hydrogen atoms can be inferred. Once the locations of the hydrogens have been assigned, the molecular weight can be calculated. When determining the hydrogen locations, atom hybridization and special groups are located which may be used later in descriptor calculations.

4.2.2 Connectivity Indices

Connectivity indices are based on path fragments of a molecular graph. An algorithm for finding these paths has already been presented. There are two common types of connectivity indices: simple and valence. The simple connectivity indices provide a measure of the branching in a molecule, while the valence connectivity indices relate to the electronic configuration and types of atoms present.

The first step in calculating connectivity indices is to assign δ-values to each atom. The δ value is the vertex degree for simple connectivity indices. The degree of a vertex in the number of adjacent vertices. The formula for the valence δ^v is given in Equation 4.1 (Bicerano, 2002). Z^v is the number of valence electrons around an atom, Z is the total number of electrons around an atom, and N_H is the number of hydrogens bonded to an atom. It is also possible to define other deltas to create new types of connectivity indices. The deltas in the polymer design program are stored in a lookup table, and are found using the atom hybridization and number of attached hydrogens, which were determined previously.

$$\delta^v = \frac{Z^v - N_H}{Z - Z^v - 1} \tag{4.1}$$

Once delta values are assigned to each atom, connectivity indices can be calculated. The calculation of zeroth- and first-order indices is straight-forward. The formulas are given by Equations 4.2 and 4.3 (Bicerano, 2002), where ${}^n\chi$ is the nth-order connectivity index. In Equation 4.3, i and j are the indices of the endpoint atoms of a bond.

$${}^0\chi = \sum_{i \in \text{atoms}} \frac{1}{\sqrt{\delta_i}} \tag{4.2}$$

$${}^1\chi = \sum_{i,j \in \text{bonds}} \frac{1}{\sqrt{\delta_i \delta_j}} \tag{4.3}$$

If higher order connectivity indices are needed, the path finding algorithm is useful. The general equation for connectivity indices is given Equation 4.4. Using the path finding algorithm for path lengths of the largest order connectivity index also finds all of the shorter paths, so lower order connectivity indices can also be calculated without extra computational effort.

$${}^n\chi = \sum_{k \in n-\text{length paths}} \frac{1}{\sqrt{\prod_{i \in \text{atoms in } k} \delta_i}} \tag{4.4}$$

By looking at the equations for connectivity indices, it is obvious that χ increases as the molecule size increases. Some properties are dependent on the size of a molecule, so χ may be useful. Crosslinked polymers are generally considered infinite in that they are a single large molecule (Flory, 1941), so the size independent connectivity indices may be appropriate. Therefore a set of size independent connectivities (ξ) are calculated using Equation 4.5, where N is the number of non-hydrogen atoms.

$$\xi = \frac{\chi}{N} \tag{4.5}$$

The connectivity indices are calculated on a representative section of the polymer network. Many paths extend out of the polymer section; this is where the core and buffer concepts, described in Section 3.2, are useful. The contribution of paths that start in the core and go into the buffer area can be divided. For the second order connectivity index, the formula becomes Equation 4.6, where x is a binary variable that is 1 when an atom is in the core. The buffer prevents the connectivity indices from being affected by the chain ends where the representative polymer section is cut.

$$^{2}\chi = \sum_{\text{i,j,k-paths}} \frac{x_i + x_j + x_k}{3} \frac{1}{\sqrt{\delta_i \delta_j \delta_k}} \tag{4.6}$$

4.2.2.1 Backbone and Side-chains

Sometimes it may be desirable to calculate separate descriptors for the backbone and side-chains (Bicerano, 2002). The first step is to use the backbone finding algorithm to mark the backbone and side-chain portions of the polymer.

For this project, polymers are divided into three sections. Each atom is marked as being part of a backbone, side-chain, or crosslink. The backbones are the main polymer chains. In the case of vinyl polymers, the backbones are just chains of carbon atoms. Side-chains are groups attached to the backbone that are not on a path along a backbone and do not connect two backbones. Crosslinks are like side chains, but they join two backbones together.

Each section can have its own descriptors. For connectivity indices, the calculations are handled like the core and buffer, where the path contributions are split between the backbone and side-chain. Calculations can be done other ways; for example, paths could not be counted at all unless contained completely in a particular section. Requiring paths to be completely in a side chain may better separate the structure of the side-chain from the backbone.

4.2.3 Shape Indices

Since polymers are very large molecules, shape indices are probably not useful for predicting polymer properties; however, they maybe useful in predicting monomer properties or degree of conversion, which could depend on the monomer shape. Shape indices are calculated using the definitions given by Kier (1987).

The shape index measures the shape of a molecule relative to two extreme shapes. The first shape is a straight chain of atoms. The second shape is a star molecule with a central atom having the remaining atoms bonded to it. The shape index requires counting all length 2 paths. In a linear molecule, the number of length 2 paths is a minimum and can be calculated by Equation 4.7, where A is the number of atoms. The number of length 2 paths is at a maximum in a star shaped molecule and is calculated using Equation 4.8.

$$^2P_{\text{min}} = A - 2 \tag{4.7}$$

$$^2P_{\text{max}} = \frac{(A-1)(A-2)}{2} \tag{4.8}$$

Assuming all atoms are the same size, the shape index ($^2\kappa$) can be calculated by Equation 4.9. The same algorithm that finds paths for connectivity indices can be used to find and count paths for the shape index. When all the atoms are arranged in a line

${}^2\kappa = (A-1)$ and when the molecule is a star ${}^2\kappa = 4(A-1)^{-1}$; other arrangements of atoms fall somewhere in between. When there are three atoms, there is only one way to arrange them, so star and line are the same.

$$ {}^2\kappa = \frac{2 \cdot {}^2P_{\text{max}} \cdot {}^2P_{\text{min}}}{({}^2P_i)^2} = \frac{(A-1)(A-2)^2}{({}^2P_i)^2} \tag{4.9} $$

The shape index can also be adjusted to account for atoms of different size. The shape index was first developed for molecules containing all *sp*^{3} hybridized carbon atoms. To adjust for other atoms the size is adjusted relative to *sp*^{3} carbon using equation 4.10. An overall correction factor is calculated using Equation 4.11, where r_x is the radius of atom x.

$$ \alpha_x = \frac{r_x}{r_{\text{C}sp^3}} \tag{4.10} $$

$$ \alpha = \sum_{\text{atoms}} \alpha_x \tag{4.11} $$

The adjusted shape index is given by Equation 4.12.

$$ {}^2\kappa_\alpha = \frac{(A+\alpha)(A+\alpha-2)}{({}^2P_i+\alpha)^2} \tag{4.12} $$

4.2.4 Crosslinking

The polymers studied in this project are vinyl polymers. The backbones consist of chains of carbon atoms. Each monomer contributes two carbon atoms to a backbone. The degree of crosslinking in this case can be estimated by Equation 4.13. The degree of crosslinking estimate (CD) is the number of crosslinks per number of monomers.

F_i is the mole-fraction of monomer graph state i in the polymer network. $N_{\mathrm{rv},i}$ is the number of reacted vinyl groups in monomer graph state i.

$$\mathrm{CD} = \sum_{i \in \text{monomer states}} F_i \left(N_{\mathrm{rv},i} - 1\right) \tag{4.13}$$

Other measures can also be used. If non-vinyl polymers, such as polyurethanes, are studied, a new method of estimating crosslink density will be needed.

4.3 Effect of Graph Size

Since many descriptors may be calculated from a representative section of polymer generated by a Monte Carlo method, it is important to consider the effect of the size of the section on the precision of the calculations. The larger the representative section of polymer is, the more repeatable the descriptor calculations will be. However, a larger section of polymer requires more computation time, so it is important to get an idea of how big is big enough. This may require some experimentation for the system of interest. A study for the dental polymer test case is provided in Section 8.3.

Chapter 5

Property Models

To create a predictive model, numerical descriptors of structure are related to physical or chemical properties of the polymeric systems. While phenomenological models or molecular simulation may be used to predict properties, QSPRs based on topological structural descriptors allow rapid estimation of properties directly from the chemical structure, and work well within an optimization framework for the design of novel polymer networks.

This work focuses mainly on linear regression as a means to develop property models due to the limited experimental data collected on the systems being studied. Nonlinear regression can be used to estimate parameters if there is reason to believe a property model has a particular functional form that is nonlinear with respect to the parameters, for example, $\hat{y} = b_1 \sin(b_2 x)$. The design framework implemented here does not impose any limitations on the property models.

This section provides a brief overview of the methods used to select QSPRs for this project. Standard and widely used statistical methods were used, so extensive detail is not necessary.

5.1 Multiple Linear Regression

Linear regression uses an equation of the form given by Equation 5.1, where $\hat{y}$ is the predicted value of y, x_i are the predictor variables, and β_i are the model parameters. The predictor variables come from molecular structure descriptors. Predictor variables can also be transformations of descriptors; for example, a descriptor could be squared or two descriptors could be multiplied.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k \tag{5.1}$$

Experimental data is used to estimate the model parameters by solving the optimization problem given by Equation 5.2. Here n is the number of data points, and y_i is the observed value of y at a specific set of x_i values. Equation 5.1 is used to calculate $\hat{y}$, and the bs are optimized. The problem is easily solved although the calculations are tedious and usually statistical software is used.

$$\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{5.2}$$

Several parameters are commonly calculated to assess the quality of a model. The most common is the multiple correlation coefficient (R^2) given by Equation 5.3. The value of R^2 should be close to 1 which indicates there is little error in the prediction for the data. R^2 can be interpreted as the fraction of variance about the mean explained by the regression equation (Draper and Smith, 1966). The correlation coefficient does not say anything about statistical significance. It is always possible to get an R^2 of 1 when there are as many model parameters as data points.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y}_i)^2} \tag{5.3}$$

There are two statistical tests to help determine statistical significance. The f-test can determine the significance of the whole model, and the t-test can determine significance of each parameter. Tests for significance take into account the degrees of freedom and therefore the amount of data used. In both tests, the assumption is made that the errors are normally distributed (Draper and Smith, 1966).

The f-test can be used to determine if two variances are significantly different. The null hypothesis of the f-test is that two variances are the same. Applied to linear regression, the f-test can be used to determine if a model is statistically significant (Draper and Smith, 1966). Generally a 95% probability is used to reject the null hypothesis. Software usually reports a p-value for the f-test, and a p-value of less than 0.05 is considered significant.

The $t-$test can be used to determine whether the mean of a set of numbers is significantly different than a population mean. The $t-$test applied to regression is used to determine if regression coefficients are significantly different than zero (Draper and Smith, 1966). A regression coefficient of zero would mean a particular predictor is unrelated to the response variable. Again, a $p-$value of less than 0.05 is usually considered significant.

Cross-validation can be used to determine if a model gives good predictions; this discussed further in Section 5.3.

5.2 Descriptor Selection

Far more descriptors are available than can actually be used to create a QSPR, and using too many descriptors in a model may over-fit the data leading to a model with a good R^2 that predicts poorly for new data. Therefore, a method is needed to select the best set of descriptors to use.

The only way to find the best set of descriptors is to test every combination. Testing every set of descriptors can be very time consuming. If there are 50 possible descriptors, there are $2^{50} \approx 1.13 \times 10^{15}$ possible combinations. Of course, property models would not usually contain 50 descriptors. With a more reasonable number of descriptors in the model the number of combinations to examine is much fewer. If there are up to 5 descriptors, there are a much more reasonable $\binom{50}{1}+\binom{50}{2}+\binom{50}{3}+\binom{50}{4}+\binom{50}{5} = 2,369,935$ combinations.

For this project, a software package called LEAPS is used to select descriptors (Lumley, 2004), which works with the R statistical software (R Development Core Team, 2007). LEAPS uses a branch and bound method to implicitly examine every combination of descriptors up to a certain number. LEAPS can return some number of the best models of each size. The best model can be chosen using R^2 for a given number of descriptors. The next step is to select the number of descriptors to use.

There is no firm way to select a good number of descriptors. The simplest method is to look at a plot of R^2 as a function of number of descriptors. Usually above a certain number of descriptors improvement in R^2 is minimal. The next thing to look at is the $f-$ and $t-$tests to determine if the models and descriptors are statistically significant. Cross-validation was used in this project to select the best model size.

5.3 Cross-Validation

This section describes the cross-validation procedure used in this project. Other methods of cross-validation are also possible.

The $k-$fold cross validation process is used in this project (Efron and Tibshirani, 1993). The data is split randomly into k test sets of approximately equal size. The regression is performed k times with each test set omitted in turn. In this procedure, each data point is omitted once. This method is an improvement over leave-one-out validation, because more than one point is left out at a time. In leave-one-out validation, the regression equation may change very little when only one point is left out. The error in the prediction for each point in each test set can be calculated.

The cross-validation procedure provides a measure of how well the model may predict for new data points not used in the regression. It is a good way to determine if a model over-fits the data. Large errors in the test predictions may mean the model contains too many predictors.

There are many ways to evaluate the cross-validation results. Q^2 is used for this project because it is easy to interpret (Quan, 1988). Q^2 is calculated similarly to R^2 but the residuals from the test sets are used. The test sets are data that is not used to make a model, so a measure of the predictive value of a model is obtained. A Q^2 value close to the R^2 value of a model is good, and lower Q^2 is worse. It is also possible that Q^2 could be negative if the predictions of the model are worse that just using the the mean of the model data.

PRESS is the sum of the squared prediction errors, and is calculated using Equation 5.4, where y_i is the observed response variable, $\hat{y}_{i,(k)}$ is the predicted value from the model were the kth test set is omitted, K is the set of test sets, and $|K|$ is the number of test sets.

$$\text{PRESS} = \sum_{k=1}^{|K|} \sum_{i \in K_k} \left(y_i - \hat{y}_{i,(k)} \right)^2 \tag{5.4}$$

Equation 5.5 gives the calculation for Q^2, where $\bar{y}_{(k)}$ is the mean response variable with the kth set omitted.

$$Q^2 = 1 - \frac{\text{PRESS}}{\sum_{k=1}^{|K|} \sum_{i \in K_k} (y_i - \bar{y}_{(k)})^2} \tag{5.5}$$

The test sets are selected at random, so Q^2 will be different each time it is calculated. The cross-validation procedure is repeated many (500 or 1000) times to get a more representative sample of Q^2's. The distribution of Q^2 can be used to make decisions about the proper number of descriptors to uses in a property model and whether a model is useful.

Chapter 6

Molecular Design

Molecular design is the application of optimization techniques to designing improved molecules. Section 6.1 describes the optimization problem for this project. Tabu search is the method used in this work, so Section 6.2 provides a detailed description of Tabu search and how it was implemented.

6.1 Problem Formulation

In this project, a set of desired target properties is chosen for a polymer. The properties define an optimal material for a given application. The goal is to design a new monomer that when used in the formulation of the resin, gives polymer properties matching the target properties. The objective function for the optimization problem is given by Equation 6.1, where $P_{i,\mathrm{targ}}$ is the desired value of property i and $P_{i,\mathrm{est}}$ is the value of property i estimated using the QSPRs. The scaling factor (s_i) can be used to adjust the relative importance of each property.

$$f = \sum_{i \in \mathrm{properties}} s_i \left(\frac{P_{i,\mathrm{targ}} - P_{i,\mathrm{est}}}{P_{i,\mathrm{targ}}} \right)^2 \tag{6.1}$$

The objective function value gets closer to zero as the estimated properties of the polymer approach the targets. The goal is to minimize the objective function value. The advantage of this objective function is that is easy to judge the quality of a solution, and there is a fixed lower bound on the optimum. The disadvantage is that properties cannot be minimized or maximized, but since QSPRs should not be used to extrapolate beyond the data used to make them, this should not be a problem. A linear objective function may be better for many optimization methods, but Tabu search is used here and the form of the objective function is not critical; this allows any type of QSPR to be used. In Tabu search, the objective function value of each solution is calculated to evaluate its quality, but there is no need for the problem to be linear or convex, and even black-box models can be used. However, the form of the objective function may make a difference to the performance of the Tabu search algorithm.

The molecular design problem also contains constraints. The constraints require that the monomer that is designed has a feasible molecular structure. In this project, feasible molecular structures are connected and the valency of each atom is satisfied. Other structurally constants can be imposed, for example, peroxide groups can be excluded so that more stable molecules are obtained, or symmetry could be required so that more synthesizable structures are obtained. There are also constraints that require the monomer to be of a specific type. Various other structural constraints can also be applied. The structural constraints do not need to be explicitly formulated, and are handled in the Tabu search method as explained in Section 6.2.2.

In molecular design, it is possible that some property must absolutely meet some minimal requirement. Property constraints can be added to the objective function in the form of a penalty, thus avoiding the need to explicitly add constraints to the problem formulation. Avoiding the need to consider the additional constraints may make the Tabu search formulation somewhat simpler.

6.2 Tabu Search

Tabu search is a heuristic approach to solving a variety of optimization problems, which relies on a memory of recently visited solutions. It has been applied in a number of areas including molecular design (Lin *et al.*, 2005). Tabu search cannot be guaranteed to find a global optimal solution, but it has been successful in finding good solutions to large combinatorial optimization problems (Glover, 1990a).

The limited accuracy of the QSPRs mean that the best solution in practice may not be the global optimum of the optimization problem. It is also difficult to apply deterministic algorithms for many types of molecular design problems due to non-convexities in the constraint set. Therefore, the ability of Tabu search to find good solutions to a wide variety of problems makes it well suited to molecular design.

6.2.1 General Method

This section describes the general Tabu search method for minimization problems. For a more specific example, see Sections 6.2.2 and 8.5.

A Tabu search algorithm starts with an initial solution. It is assumed here that all of the solutions generated are feasible; however, some Tabu search implementations make use of infeasible solutions. The initial solution may be specified (to search within specific area) or randomly generated. The search proceeds from one solution to the next until some termination criteria are met. The goal of the search is to find a solution with the lowest objective function value.

A set of valid moves are defined, which given a feasible starting solution, generate other nearby feasible solutions. Solutions that can be reached by a specified number of these moves are neighbors of the original solution. A subset of neighbors is chosen by applying moves. This is often done by applying some number of allowed moves at random. In some cases, the entire set of neighbors can be used. The search proceeds to the next solution by choosing the best solution from the subset of neighbors which is not Tabu. The search may move to a new solution which has a higher objective function value than the original solution, to escape a local minimum.

Selection of a new solution is subject to Tabu restrictions (Glover, 1990a). The restrictions are implemented using Tabu lists which record either recent solutions or moves. Multiple Tabu lists may be defined to record different types of recent moves, or different properties of a solution. The sizes of the Tabu lists are limited to reduce computation, and allow solutions to be revisited later with the search proceeding in a potentially different direction (de Werra and Hertz, 1989). The Tabu restrictions work by disallowing moves that would lead back to recent solutions or disallowing new solutions which are too similar to ones in the Tabu list. The best size for a Tabu list depends on the application, but it is often related to the size of the problem (Glover, 1990a)

The purpose of Tabu restrictions is to prevent the search from cycling around a local optima, and encourage searching in more diverse areas. If a local minimum is reached, the next solution chosen will have a higher objective function value. Without the Tabu restrictions, there is a good chance the solution would return directly to the local minimum (de Werra and Hertz, 1989).

Tabu restrictions can sometimes block access to solutions that may be very good. To help overcome this, aspiration criteria can be applied to allow solutions or moves that would otherwise be considered Tabu. An example of this would be to allow moves to solutions which have a lower objective function value than the best found so far (Glover, 1990a).

A set of criteria is needed to stop the search. The search can be terminated after a set number of iterations where no solution with a objective function lower that the lowest so far is found (de Werra and Hertz, 1989). In some problems with a set lower bound (as is common in molecular design when specific properties are to be matched), an additional criteria can be employed to stop when the objective function value is close enough to the bound (Glover, 1990a).

The procedure described so far uses short-term memory. In addition to short-term memory, additional levels of longer term memory can be added to accomplish intensification and diversification (Glover, 1990a). A list of good solutions may be kept these solutions may indicate especially promising areas to search more thoroughly; this is called intensification. The intensification can be accomplished by modifying the objective function to penalize solutions outside the intensification area (de Werra and Hertz, 1989). Diversification can be accomplished by repeating the search with an initial solution far from solutions already explored.

Tabu search contains a number of adjustable parameters. The parameters may be different depending the implementation, but typical parameters include the size of the Tabu lists, the size of the neighbor subset, and the size or number of moves. Other parameters may also be adjusted, such as the types of moves allowed and what types of Tabu restrictions to use. Adjustments made to parameters can make a substantial difference in the performance of the algorithm. Due to the wide range of problems and ways Tabu search can be implemented, it is difficult to find proper values for these parameters. Numerical experiments are used to find good parameter values.

6.2.2 Implementation

The Tabu search procedure is very general and can be applied in numerous ways. This section describes the implementation of Tabu search used in this project. A simple flow chart for the Tabu search procedure used is given by Figure 6.1. Additional details for a specific example can be found in Section 8.5.

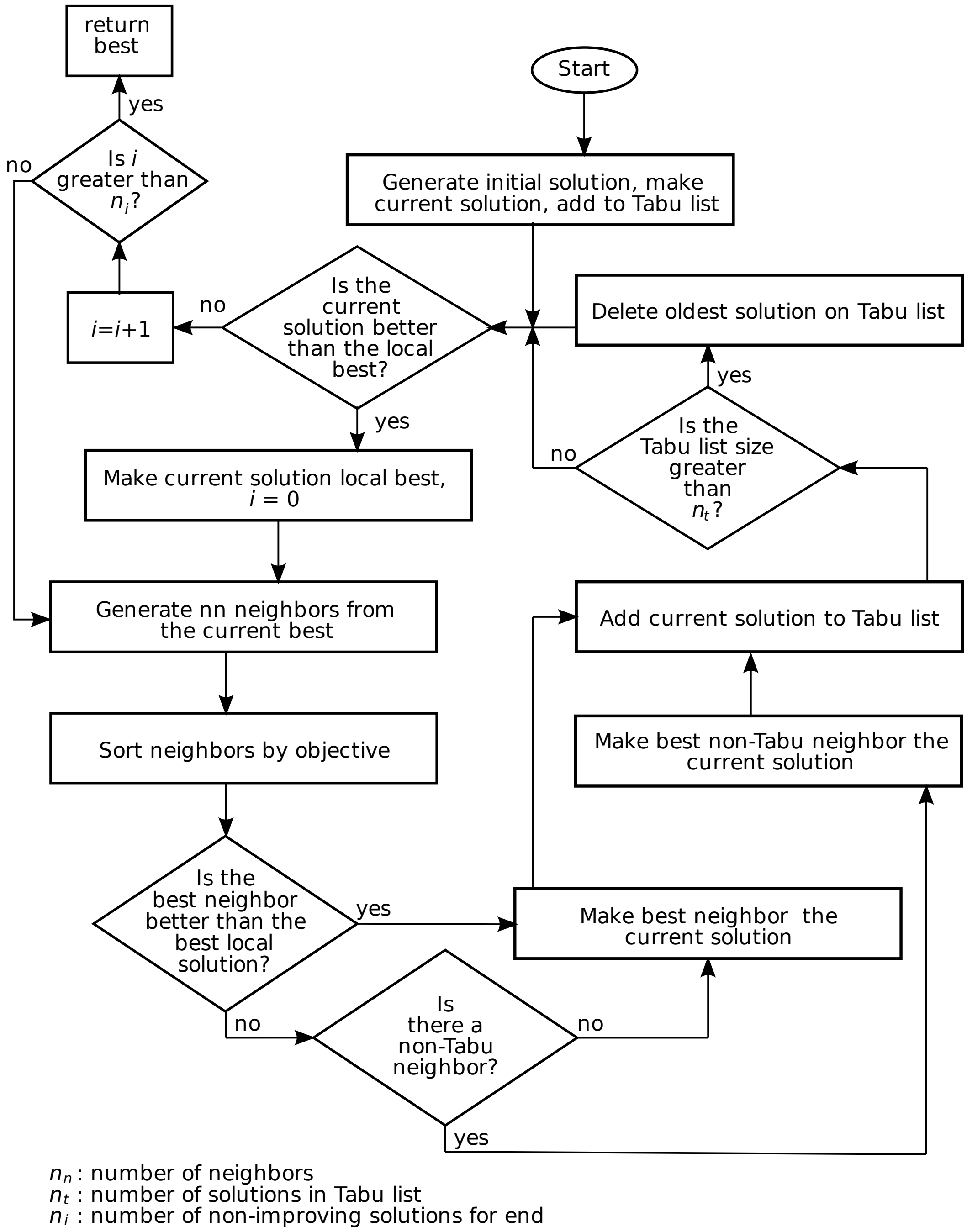

return best
yes
no
Is i greater than n_i?
Start
Generate initial solution, make current solution, add to Tabu list
i=i+1
no
Is the current solution better than the local best?
Delete oldest solution on Tabu list
yes
yes
no
Is the Tabu list size greater than n_t?
Make current solution local best, i = 0
Generate nn neighbors from the current best
Add current solution to Tabu list
Sort neighbors by objective
Make best non-Tabu neighbor the current solution
Is the best neighbor better than the best local solution?
yes
Make best neighbor the current solution
no
Is there a non-Tabu neighbor?
no
yes
n_n : number of neighbors
n_t : number of solutions in Tabu list
n_i : number of non-improving solutions for end

Figure 6.1: Tabu Search Flow Chart

The problem considered is the design of a new monomer to be used in a resin formulation to make improved dental polymers. The objective function measures the difference between the estimated properties of a solution and the desired properties. The objective function has a lower bound of zero, and properties are estimated using QSPRs.

A set of chemical groups are defined that are used to create a monomer. Solutions to the problem are graphs in which groups are represented by vertices and connections between groups are represented by edges. By representing monomers this way, the monomer structure can be represented by an oligomer of groups, and the polymer graph structure can be used in the Tabu search procedure. The monomers are modified by deleting, inserting, or replacing groups at random. These changes only yield new feasible monomers, and the maximum number of changes that can be applied in one move is specified. Some number of neighbors is generated by randomly applying the moves. The solution with the best non-Tabu objective function is used as the next solution. If all neighbor solutions are Tabu, the best neighbor is chosen; this should rarely happen. If it happens frequently, the Tabu restrictions are too strict or the moves are too small. An aspiration criteria is also applied to allow neighbors that are Tabu to be selected if they have a better objective function value than the best previously found.

The initial solutions used are existing monomers or new monomers generated randomly. A set of candidate monomers could be obtained from the best solutions of the various starting points. Often the same monomers appear repeatedly as the best solution, so all solutions beneath some threshold can be stored.

The Tabu list stores some number of recent solutions, and solutions that are too close to ones on the Tabu list cannot be selected unless they meet the aspiration criteria. The closeness of the solutions to those on the Tabu list is judged by comparing some set of descriptors. If any descriptor is sufficiently different, the solution is not Tabu.

The length of the Tabu list and the amount of difference that is sufficient are adjustable parameters.

Once a set number of non-improving iterations is reached, the best solution is returned. It is also possible to return the solutions with objective functions below some threshold, or a set of the best solutions found. The search can be repeated from a new starting point different from the solutions found to find a more diverse set of solutions, or started from a good solution with relaxed Tabu restrictions to refine the solution.

Now that the individual parts of this project have been discussed, the next chapter presents an overview of the software implementation of the molecular design method.

Chapter 7

Polymer Designer

The main product of this research is the Polymer Designer (PD) software package which implements most of the crosslinked polymer design framework. Statistics are computed using an external statistical software package; R (R Development Core Team, 2007) was used in this project. Since little previous work has been performed for CMD of crosslinked polymers, the software is meant to be a platform for numerical experimentation leading to development of useful QSPRs and optimization techniques. The goal is to make calculation of new descriptors easy to add. New ways of generating a representative section of polymer can also be evaluated. The structure storage and descriptor calculation facilities can be leveraged to quickly develop heuristic optimization techniques.

This chapter provides an overview of PD. A more complete manual is provided in Appendix A.

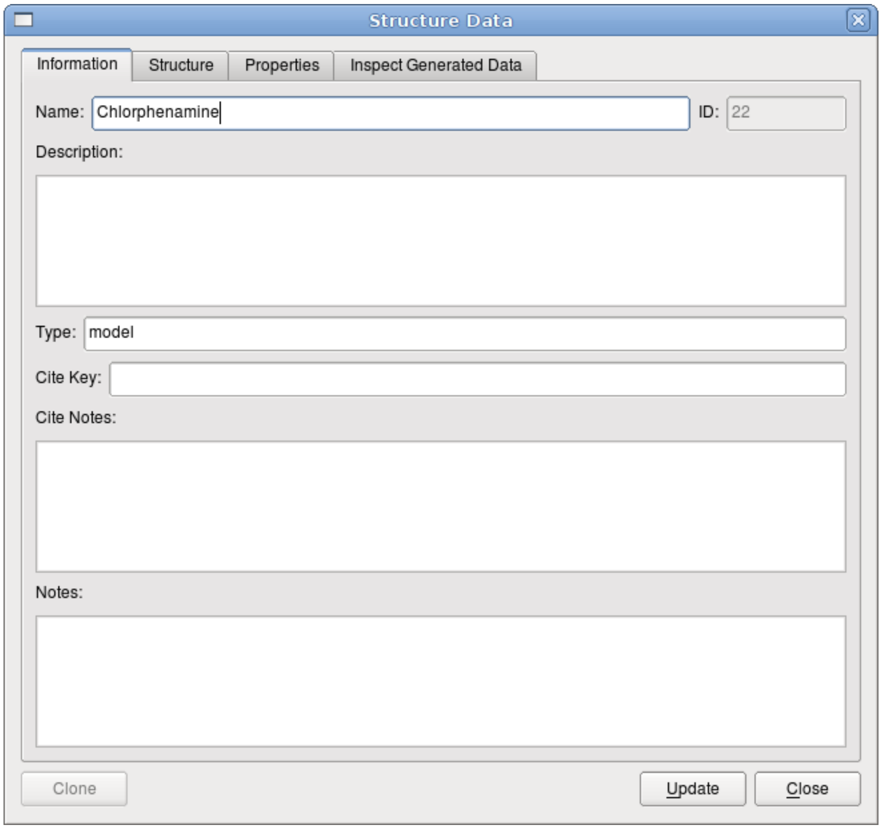
PD stores structure and property information in a database. The public domain SQLite database was used (SQLite, 2008). The QT graphical toolkit was used for the graphical user interface (Trolltech, 2008).

7.1 Structure Input

The chemical structures of monomers and polymers are stored in a database. This section describes how chemical structures may be entered and edited.

7.1.1 Monomer Structure

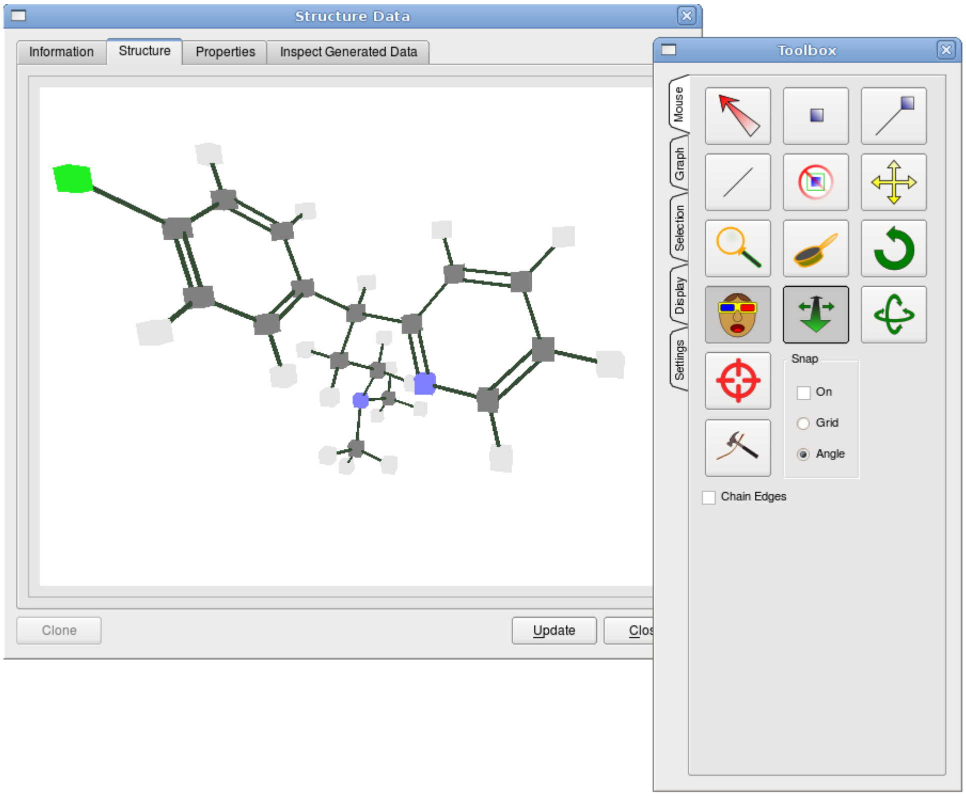
Several pieces of information can be stored with the monomer structures. This information can be used to sort the monomers and keep track of the sources of monomer data. Figure 7.1 shows the data entry form. In addition to monomers, other small molecules may also be entered. The polymer designer program can be a useful tool to study small molecules in addition to polymers.



Name: Chlorphenamine
ID: 22
Type: model

Figure 7.1: Monomer Data Entry Form

The structure of molecules can be entered by drawing them with a mouse or importing standard mol-files (MDL Information Systems, 2005). Mol-files are a common file format in computational chemistry. If an imported file contains a three-dimensional structure, it can be viewed and rotated in three dimensions. Figure 7.2 shows the molecule editor. Atoms and bonds can be drawn using the mouse, so molecule entry is very simple. This project did not make use of stereo-isomers, but if they are need that capability can be added through additional bond and atom labeling.



Snap

□ On

○ Grid

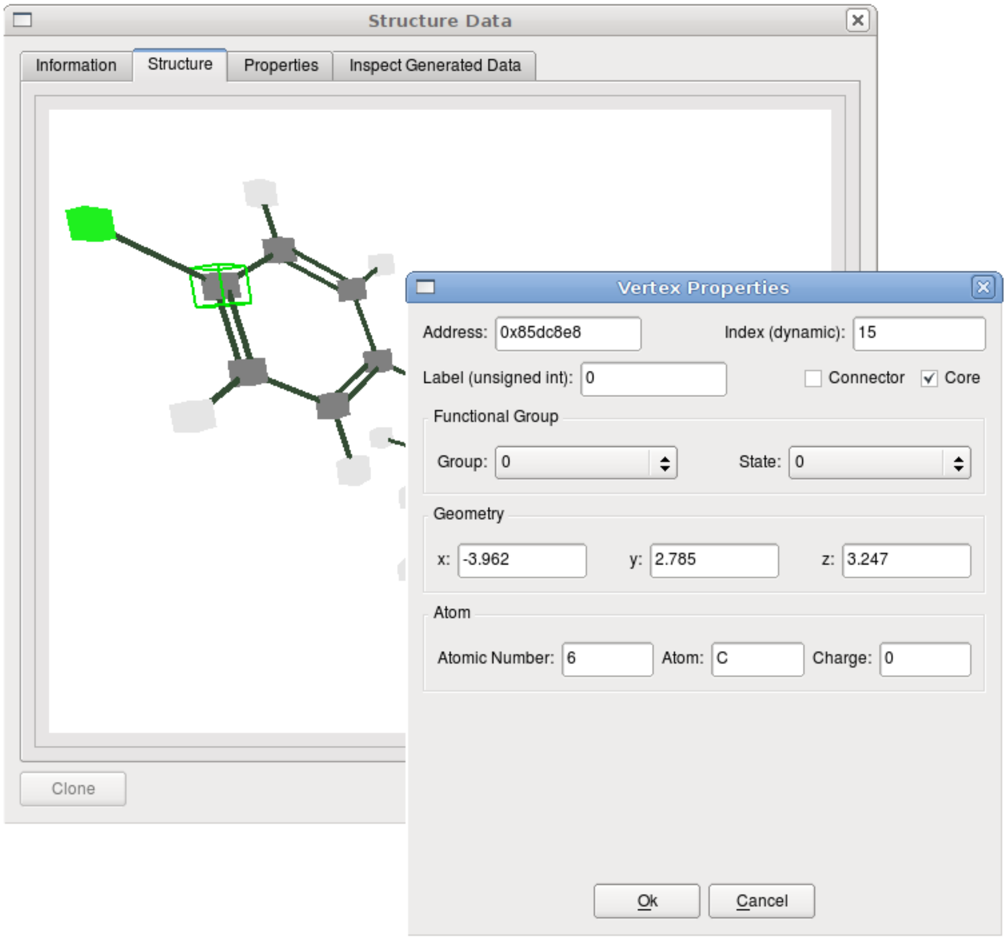
● Angle

□ Chain Edges

Figure 7.2: Graphical Molecule Graph Editing

The properties of atoms and bonds can be edited by selecting them and editing their properties in a form. Figure 7.3 shows an example of an atom being edited.

Once a molecule is saved in the database, it can be reopened and edited as needed.



Address: 0x85dc8e8 Index (dynamic): 15

Label (unsigned int): 0 □ Connector ☑ Core

Functional Group

Group: 0 State: 0

Geometry

x: -3.962 y: 2.785 z: 3.247

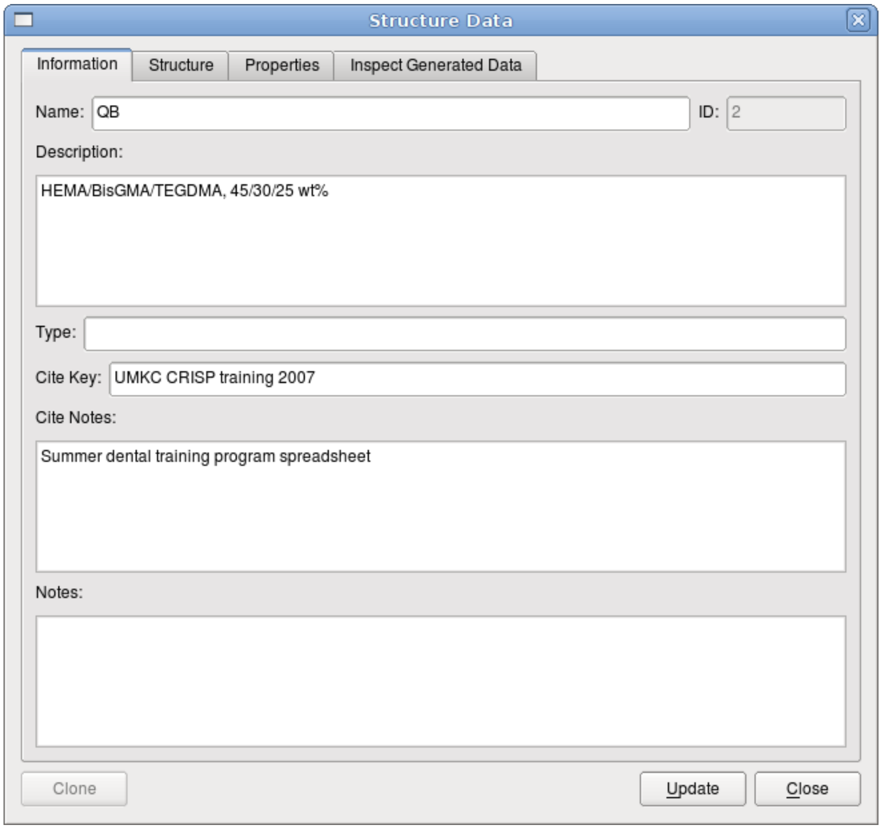
Atom

Atomic Number: 6 Atom: C Charge: 0

Figure 7.3: Molecule Graph Vertex Editing

7.1.2 Polymer Structure

Entry of polymer structures is very similar to that for monomers. The same data fields are available for polymers as shown in Figure 7.4.



Name: QB ID: 2

Description:

HEMA/BisGMA/TEGDMA, 45/30/25 wt%

Type:

Cite Key: UMKC CRISP training 2007

Cite Notes:

Summer dental training program spreadsheet

Notes:

Figure 7.4: Polymer Data Entry Form

Polymer graphs have vertices that represent monomers. Monomers can be added to a polymer by selecting them from the database of monomers. Polymer graphs for simple polymers can be drawn directly, but for more complicated crosslinked polymers, the graphs can be generated automatically based on a set assumptions about how the polymer forms. Figure 7.5 shows the polymer editing window, and Figure 7.6 shows the form to generate a polymer graph for a crosslinked polymer.

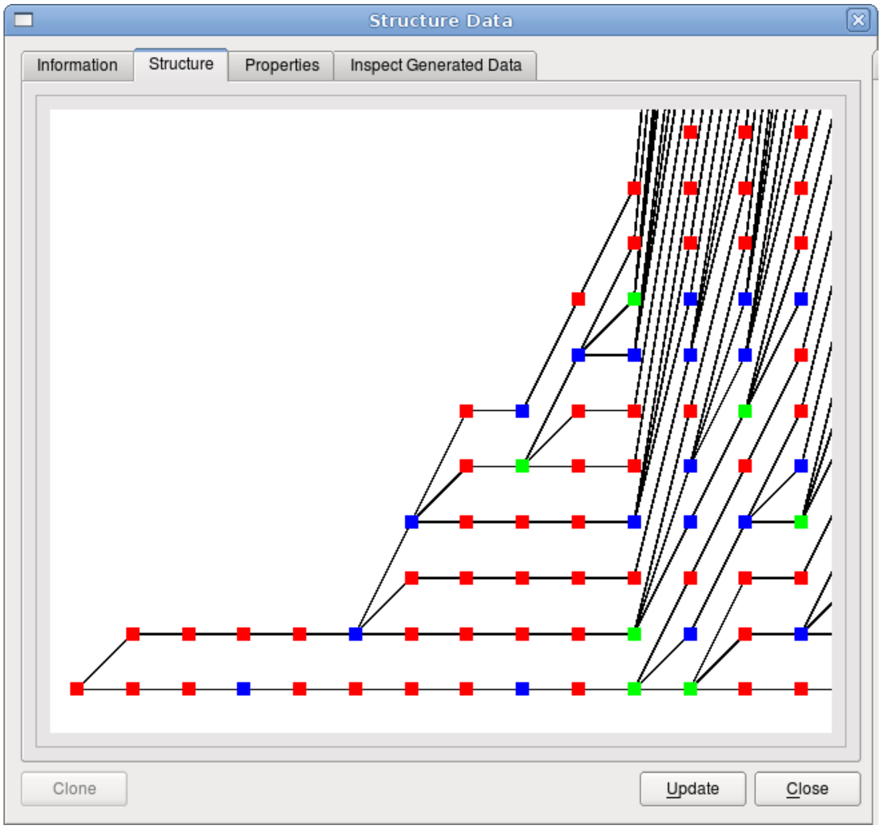
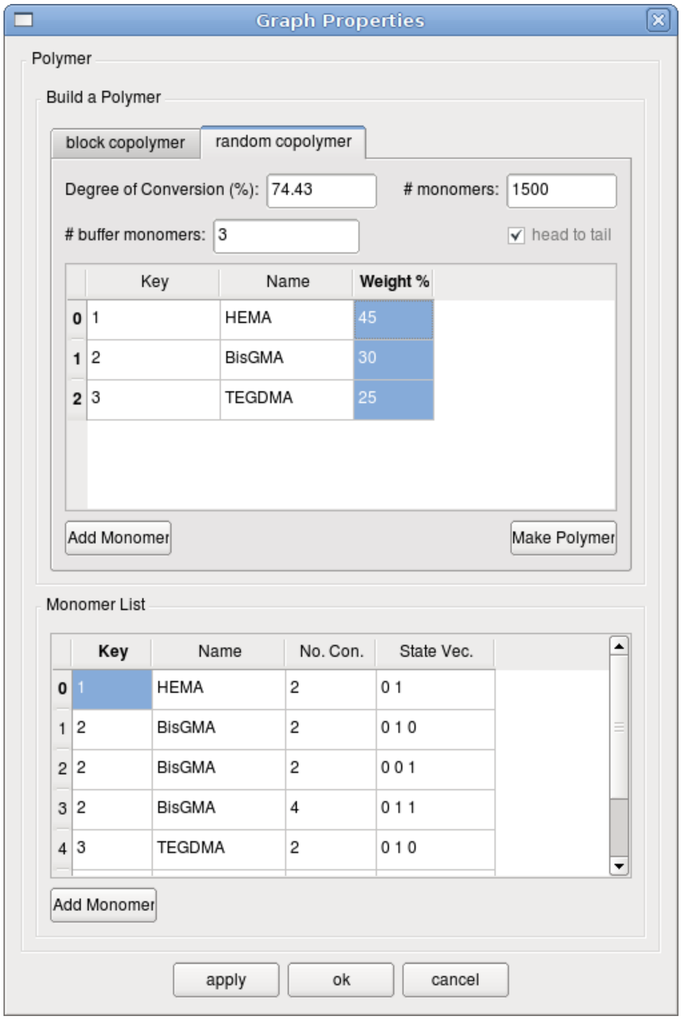


Figure 7.5: Graphical Polymer Graph Editing



Degree of Conversion (%): 74.43

monomers: 1500

buffer monomers: 3

☑ head to tail

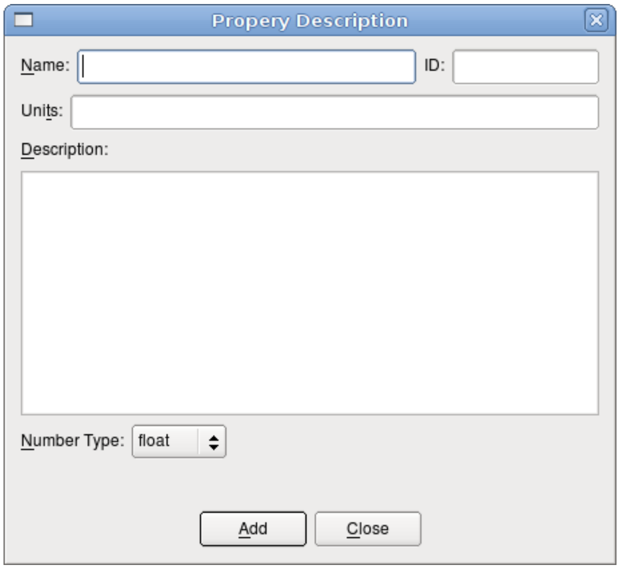
	Key	Name	Weight %
0	1	HEMA	45
1	2	BisGMA	30
2	3	TEGDMA	25

	Key	Name	No. Con.	State Vec.
0	1	HEMA	2	0 1
1	2	BisGMA	2	0 1 0
2	2	BisGMA	2	0 0 1
3	2	BisGMA	4	0 1 1
4	3	TEGDMA	2	0 1 0

Figure 7.6: Polymer Building Form

7.2 Property Input

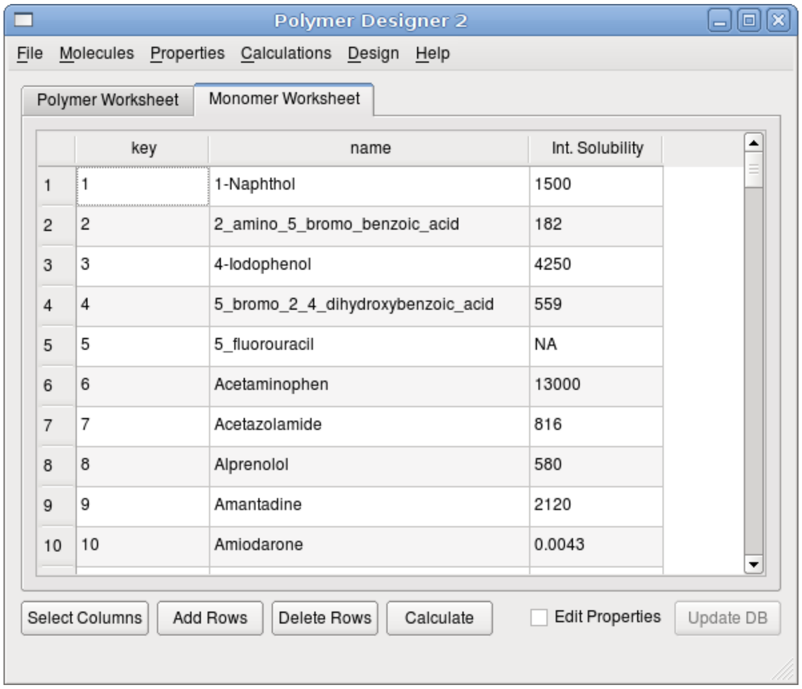
The database also stores experimental data for the polymers and monomers. The first step is to create an entry in the database for the experimental properties. The database stores units and a description of each property. The form to enter a new property is shown in Figure 7.7.



Number Type: float

Figure 7.7: New Property Entry Form

Once the property description is entered in the database, the property values for each monomer or polymer can be entered in the descriptor and property spreadsheet. First, a set of monomers or polymers and properties is selected to display on the spreadsheet. The properties for each compound can be edited on the sheet. Missing data is indicated by “NA.” Figure 7.8 shows a spreadsheet with property information.



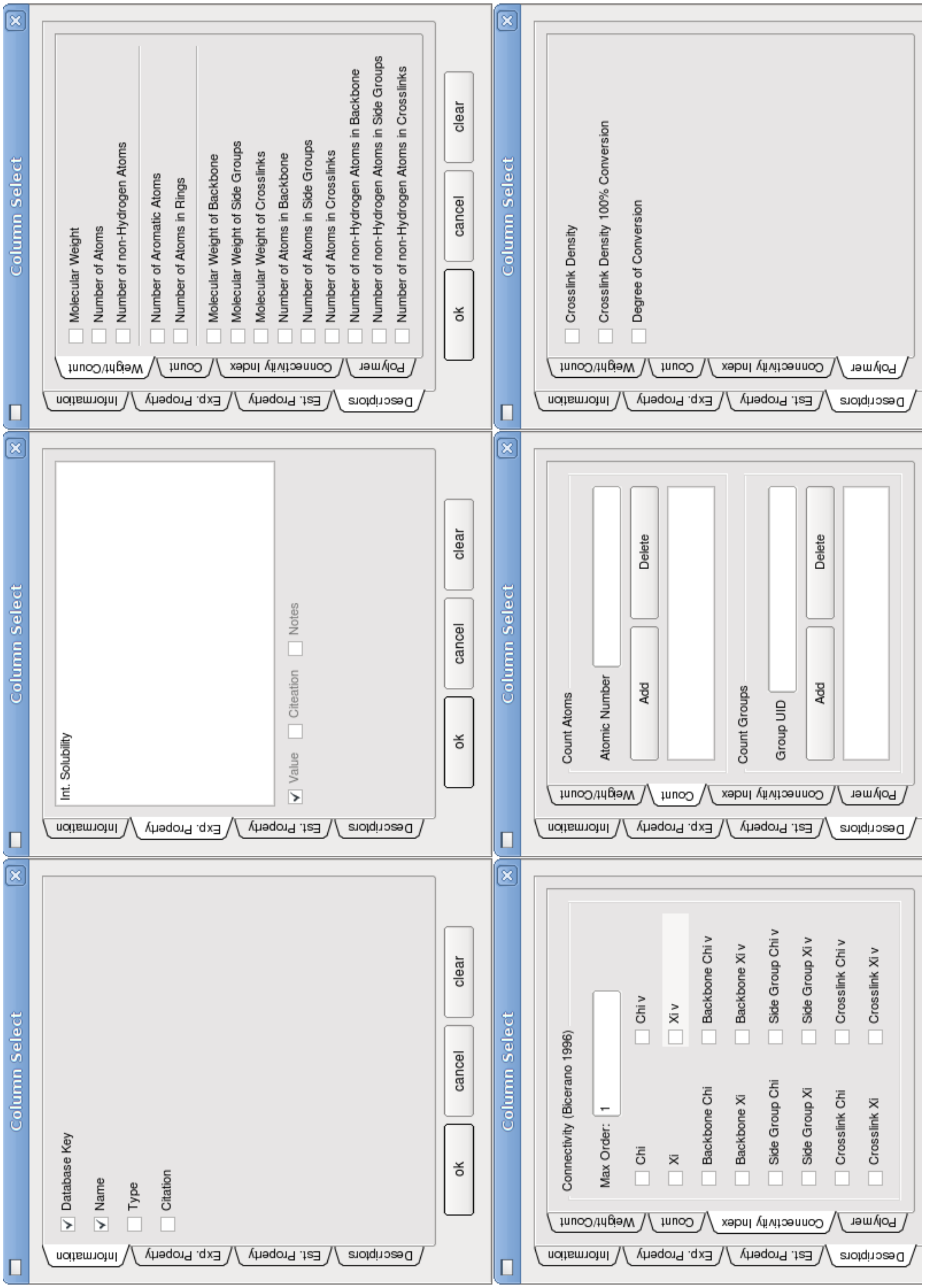
	key	name	Int. Solubility
1	1	1-Naphthol	1500
2	2	2_amino_5_bromo_benzoic_acid	182
3	3	4-Iodophenol	4250
4	4	5_bromo_2_4_dihydroxybenzoic_acid	559
5	5	5_fluorouracil	NA
6	6	Acetaminophen	13000
7	7	Acetazolamide	816
8	8	Alprenolol	580
9	9	Amantadine	2120
10	10	Amiodarone	0.0043

Figure 7.8: New Property Entry Form

7.3 Descriptor Calculation

The same spreadsheet that is used to enter and display property data is used to calculate descriptors. Using the add rows button, compounds can be added to the monomer or polymer sheet. The add columns button adds database information, properties, and descriptors to the spreadsheet. Figure 7.9 shows several tabs of the column selection form.

Once the data to display on the spreadsheet is selected, the calculate button fills in the spreadsheet from that database and by calculation. The spreadsheet can be exported as a tab-separated text file, and imported into a spreadsheet or statistics program.



☑ Database Key
☑ Name
□ Type
□ Citation

Int. Solubility

☑ Value □ Citeation □ Notes

□ Molecular Weight
□ Number of Atoms
□ Number of non-Hydrogen Atoms
□ Number of Aromatic Atoms
□ Number of Atoms in Rings
□ Molecular Weight of Backbone
□ Molecular Weight of Side Groups
□ Molecular Weight of Crosslinks
□ Number of Atoms in Backbone
□ Number of Atoms in Side Groups
□ Number of Atoms in Crosslinks
□ Number of non-Hydrogen Atoms in Backbone
□ Number of non-Hydrogen Atoms in Side Groups
□ Number of non-Hydrogen Atoms in Crosslinks

Connectivity (Bicerano 1996)

Max Order: 1

□ Chi	□ Chi v
□ Xi	□ Xi v
□ Backbone Chi	□ Backbone Chi v
□ Backbone Xi	□ Backbone Xi v
□ Side Group Chi	□ Side Group Chi v
□ Side Group Xi	□ Side Group Xi v
□ Crosslink Chi	□ Crosslink Chi v
□ Crosslink Xi	□ Crosslink Xi v

Count Atoms

Atomic Number

Count Groups

Group UID

□ Crosslink Density
□ Crosslink Density 100% Conversion
□ Degree of Conversion

Figure 7.9: Column Selection

7.4 Structure Optimization

Once QSPRs have been developed using statistical software, they can be added into PD. The QSPRs can be used as the basis for the objective function for structure optimization. The optimization routine will generally return several promising graphs representing a new monomer as well as their estimated properties and objective function values. The details of the optimization routine depend on the specific system being studied. See Chapter 8 for an example.

7.5 Modification

Much of the usefulness of Polymer Designer lies in being able to extend it. PD provides a means to store and access chemical structures; several useful graph algorithms; and a Tabu search framework. With these, new descriptors can be added, and optimization algorithms can be developed quickly. Appendix C provides more information about extending PD.

Chapter 8

Example: Restorative Dental Polymer Design

Over two-thirds of restorative dentistry involves replacement of failed restorations (Murray *et al.*, 2002). Clearly restorative dental materials with improved durability are needed. This example is used to demonstrate the methodology for the design of novel monomers to improve the durability of dentin adhesives. The dentin adhesives are used to secure dental composite materials to the dentin of a tooth. The composite contain the same polymers as the adhesive, with ceramic particles added.

Properties of a small set of polymethacrylates were measured experimentally, and QSPRs were created by relating this data to topological descriptors and formulations. The properties include tensile strength, modulus of elasticity, glass transition temperature, initial polymerization rate, and degree of conversion. The details of the experimental methods used to measure these properties are described by Ye *et al.* (2007a,b). The experimental data used here was collected as part of a summer research program at the University of Missouri Kansas City (UMKC) Dental School. Polymer samples were prepared in a consistent way. The polymers were composed of 45 wt% HEMA, 30 wt% bisGMA and 25 wt% of a third monomer. Seven monomers were used as the third monomer: bisGMA; triethyleneglycol dimethacrylate (TEGDMA); urethane dimethacrylate (UDMA); bisphenol A polyethylene glycol diether dimethacrylate (bisEMA); polyethylene glycol dimethacrylate (PEGDMA); trimethylol-propane mono allyl ether dimethacrylate (TMPEDMA); and 1,1,1-tri-[4-(methacryloxyethylaminocarbonyloxy)-phenyl]ethane (MPE). Figures 8.1 through 8.8 show the structures of these monomers. TMPEDMA and MPE are newly synthesized monomers (Park *et al.*, 2007). Resin samples were cured without water and with 11 wt% water, so 14 total samples were used. Table 8.1 provides the resin composition for each sample.

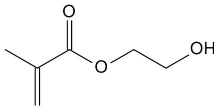


Figure 8.1: HEMA

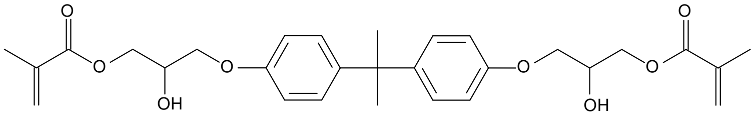


Figure 8.2: bisGMA

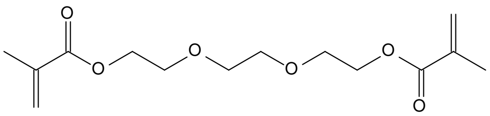


Figure 8.3: TEGDMA

The data collected so far is enough to provide a complete example of the methodology; however, additional data is need to develop more useful QSPRs. As more data is collected, better descriptors and QSPRs can be found.

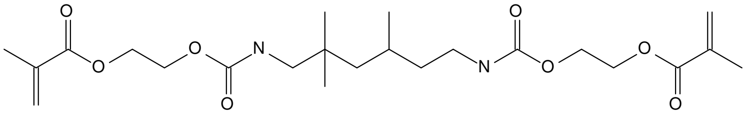


Figure 8.4: UDMA

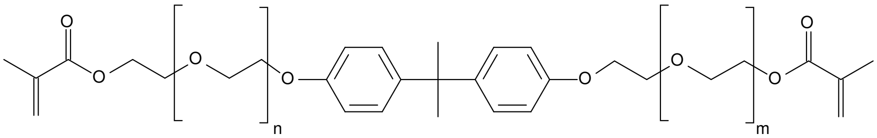


Figure 8.5: bisEMA

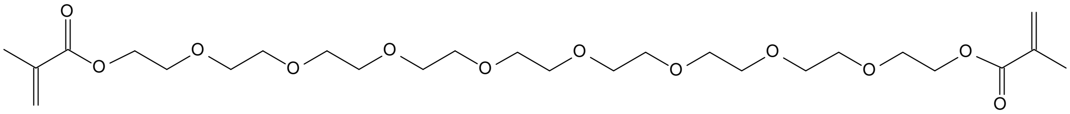


Figure 8.6: PEGDMA

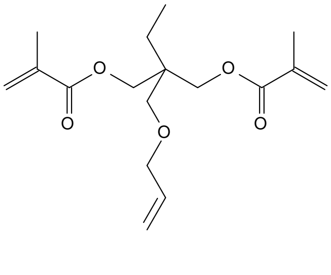


Figure 8.7: TMPEDMA

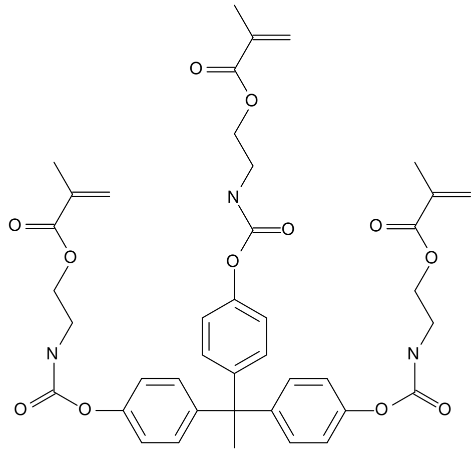


Figure 8.8: MPE

Table 8.1: Sample Compositions

Sample ID	Components	Composition at Curing (wt%)
QA	HEMA/BisGMA	45/55
QB	HEMA/BisGMA/TEGDMA	45/30/25
QC	HEMA/BisGMA/UDMA	45/30/25
QD	HEMA/BisGMA/BisEMA	45/30/25
QE	HEMA/BisGMA/PEGDMA	45/30/25
QF	HEMA/BisGMA/TMPEDMA	45/30/25
QG	HEMA/BisGMA/MPE	45/30/25
WA	QA/Water	89/11
WB	QB/Water	89/11
WC	QC/Water	89/11
WD	QD/Water	89/11
WE	QE/Water	89/11
WF	QF/Water	89/11
WG	QG/Water	89/11

8.1 Physical and Chemical Properties

Several important properties were measured for each sample. Several properties, tensile strength (σ), modulus of elasticity, elongation at break, and fracture toughness were measured with a tensile testing device. The degree of conversion (DC) of carbon-carbon double bonds and the initial polymerization rate (r_0) were measured using infrared spectroscopy. The glass transition temperature (T_g) was measured using a differential scanning calorimeter. The polymerization reaction was initiated by exposing the resin to a blue light with an intensity of 550 mW/cm^2 for 40 seconds at room temperature. The polymer samples were then stored at room temperature. The properties, aside form initial polymerization rate, were measured after more than 24, when the polymerization reaction was practically complete.

Statistically significant QSPRs were found for tensile strength, initial polymerization rate, glass transition temperature and degree of conversion. In this case, it would be best if all of the properties were as high as possible. Extrapolating use of the QSPRs, however, may result in very large errors. For that reason, properties to be maximized were set to the high end of the range measured experimentally. This is generally not a problem, because good properties values can be found in existing polymers; for example, a polymer may have a very good tensile strength but a poor initial reaction rate. Molecular design is used to find polymers with a good combination of properties. The target properties are given in Table 8.2.

Table 8.2: Target Properties

Property	**Target**
tensile strength (σ)	80 MPa
initial polymerization rate (r_0)	130 mol/l/s
glass transition temperature (T_g)	120 °C
degree of conversion (DC)	100 %

Table 8.3 provides the experimentally measured property data.

Table 8.3: Experimentally Measured Properties

Sample	σ (MPa)	r_0 (mol/l/s)	T_g (°C)	DC (%)
QA	71	96	119	69.3
QB	61	86	121	74.4
QC	73	122	109	70.9
QD	66	107	91	76.3
QE	57	104	78	82.9
QF	62	78	95	72.6
QG	72	121	126	64.0
WA	66	108	125	87.0
WB	46	103	111	87.2
WC	64	117	120	93.6
WD	51	94	94	91.5
WE	39	99	83	96.5
WF	36	82	132	89.1
WG	43	109	144	82.8

8.2 Initial Descriptor Selection

Connectivity indices have been used successfully by Bicerano (2002) in the prediction of polymer properties. Simple and valence connectivity indices up to third order were used in this study; the equations are given in Chapter 4. These values were calculated separately for the entire polymer, just the backbone, just the side groups, and for the individual monomers.

The degree of conversion was predicted from the monomer structures, assuming the curring conditions are the same as the experiments. Once the degree of conversion is predicted, it can be used in the calculation of the degree of crosslinking in the optimization routine. For the initial QSPR development, the degree of conversion is available from the experimental data. Crosslinking and degree of conversion were considered as descriptors for polymer properties. Counts of the number of atoms in rings and in aromatic rings were included as descriptors.

8.3 Graph Size and Descriptor Repeatability

The descriptors other than connectivity indices are calculated from the polymer composition and the monomer structures. Connectivity indices are calculated from a full graph of a large polymer section. The effect of the size of the polymer section on the consistency of connectivity indices is studied using numerical experiments.

Six sets of 100 polymer sections are constructed, where the core of the polymers contains 1500, 1250, 1000, 750, 500, and 250 monomers. The zeroth- to fourth-order simple connectivity indices are calculated for each polymer section to show how the size of the polymer section affects the repeatability of the calculations. Fourth order connectivity indices were not considered for QSPRs, but were included in this size study. Figure 8.9 shows histograms for the zeroth order simple connectivity index (${}^0\xi$) at each polymer sample size.

If it is assumed that the error in the descriptor calculations is normally distributed, approximately 98% of the measured values will be within ±2 standard deviations of the mean. The standard deviation then provides a good measure of the uncertainty in the measurement of the descriptors. Table 8.4 provides the standard deviation of each measurement.

Table 8.4: Standard Deviations for Different Polymer Sample Sizes

no.	standard deviation				
monomers	${}^0\xi$	${}^1\xi$	${}^2\xi$	${}^3\xi$	${}^4\xi$
1500	0.00051	0.00007	0.00042	0.00058	0.00053
1250	0.00049	0.00007	0.00041	0.00058	0.00059
1000	0.00064	0.00008	0.00055	0.00077	0.00077
750	0.00067	0.00009	0.00058	0.00083	0.00088
500	0.00074	0.00010	0.00063	0.00088	0.00094
250	0.00108	0.00015	0.00091	0.00128	0.00131

The standard deviations do not provide much information about how the measure-

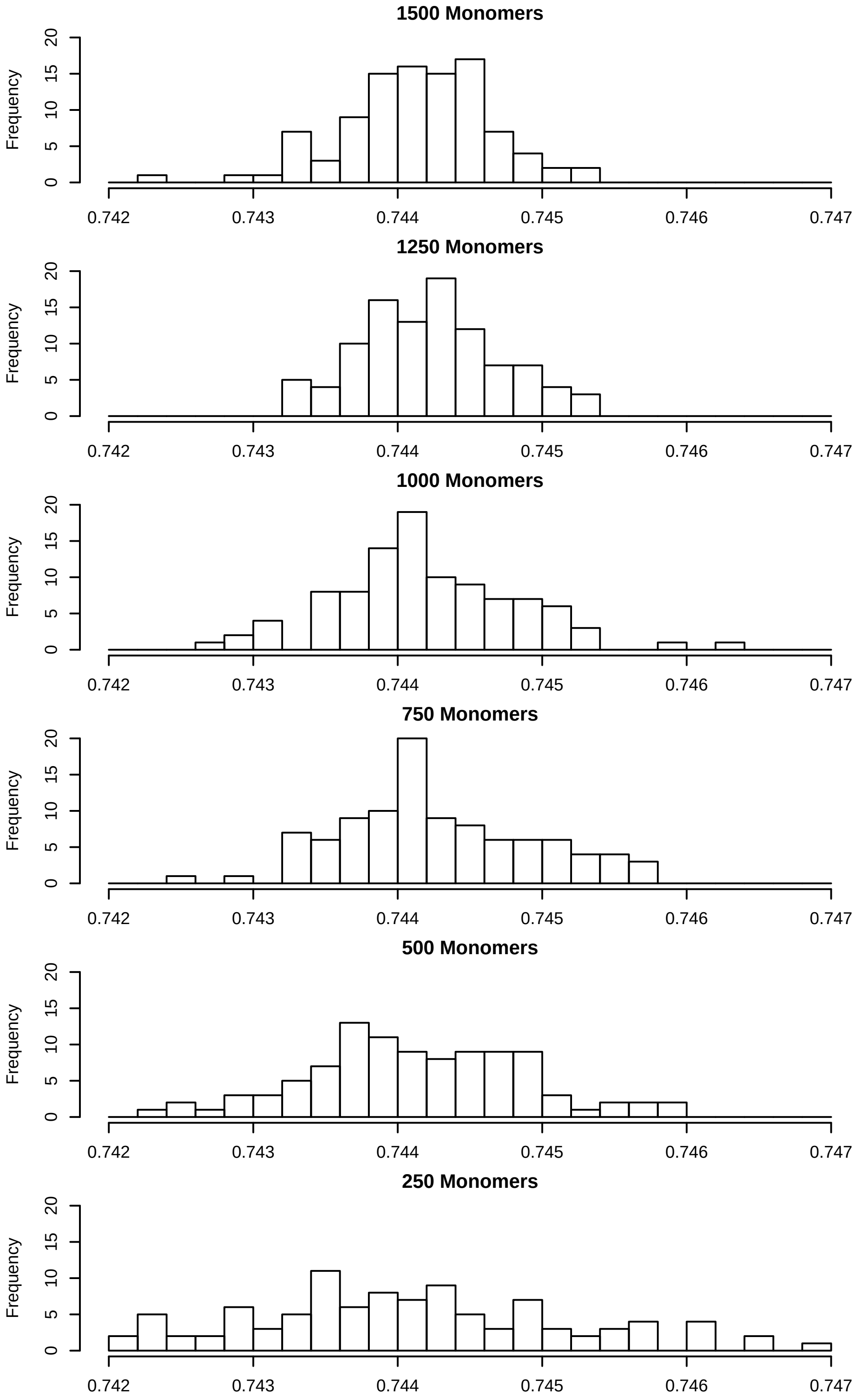


Figure 8.9: Histograms for ${}^{0}\xi$ with Different Polymer Sizes

ment error effects the predicted properties. The effect depends on the final QSPRs. The QSPRs used in this work are linear, so it is a simple matter to use the model coefficients to estimate an amount of error that could be expected. Although the descriptors used in the size study are not used in the final QSPRs, the largest coefficient for a connectivity index in a tensile strength model is less that 250. This coefficient can be used to get a rough approximation of the effect of measurement error. The effect of measurement error in ${}^{0}\xi$ would be expected to be less than 0.26 MPa for a polymer sample with 1500 monomers in the core, and less than 0.54 MPa for a polymer sample with 250 monomers in the core. Typical tensile strengths are around 60 MPa.

The polymer samples in this project used a 1500 monomer core. In general, the experimental data can be studied to determine the effect of sample size on the property estimates. Error in the property calculations should be weighed against computation time in the optimization routine.

8.4 Statistical Analysis

The polymer designer program was used to calculate descriptors for all of the polymers. The descriptors and experimental properties were exported to the R statistics program (R Development Core Team, 2007). This section describes the statistical analysis used to develop QSPRs. Since the analysis is the same for each property, the development of a QSPR for tensile strength will be followed in detail, and the results for the other properties will be presented in less detail.

In this case, the polymers are only cured at two water contents, so water content is like a binary variable. Further data will be collected at additional water contents by the Spencer research group in future work. Here water content is treated like any other continuous variable. Combining the data at different water contents may cause some problems as will be noted.

The first task is to select an initial set of structural descriptors. A subset of these descriptors will be used in the final QSPRs. Table 8.5 provides a list of descriptors considered.

Table 8.5: Initial Descriptor Set

Descriptors	symbols
Initial Resin Water Content	WC
Degree of Conversion	DC
Degree of Crosslinking	CD
Degree of Crosslinking at 100% DC	CD_{100}
Aromatic Atom Ratio	N_{aro}
Side Chain Atom Ratio	N_s
Crosslink Atom Ratio	N_c
Backbone Atom Ratio	N_b
Unreacted Monomer Connectivity Indices	${}^0\xi_m$, ${}^1\xi_m$, ${}^2\xi_m$, ${}^3\xi_m$, ${}^0\xi^v_m$, ${}^1\xi^v_m$, ${}^2\xi^v_m$, ${}^3\xi^v_m$
Polymer Connectivity Indices	${}^0\xi$, ${}^1\xi$, ${}^2\xi$, ${}^3\xi$, ${}^0\xi^v$, ${}^1\xi^v$, ${}^2\xi^v$, ${}^3\xi^v$
Backbone Connectivity Indices	${}^0\xi_b$, ${}^1\xi_b$, ${}^2\xi_b$, ${}^3\xi_b$, ${}^0\xi^v_b$, ${}^1\xi^v_b$, ${}^2\xi^v_b$, ${}^3\xi^v_b$
Side Chain Connectivity Indices	${}^0\xi_s$, ${}^1\xi_s$, ${}^2\xi_s$, ${}^3\xi_s$, ${}^0\xi^v_s$, ${}^1\xi^v_s$, ${}^2\xi^v_s$, ${}^3\xi^v_s$
Crosslink Connectivity Indices	${}^0\xi_c$, ${}^1\xi_c$, ${}^2\xi_c$, ${}^3\xi_c$, ${}^0\xi^v_c$, ${}^1\xi^v_c$, ${}^2\xi^v_c$, ${}^3\xi^v_c$

Some of the descriptors can be discarded right away. The polymer connectivity indices were calculated for the polymers as a whole, for the backbone, for side chains, and for crosslinks. The low order connectivity indices for the backbone are the same for all polymers and the higher order connectivity indices very similar. The polymers are all polymethacrylates so the backbones are just a chain of carbon atoms in similar environments. The fraction of backbone, side chain, and crosslink atoms should add up to one, so the backbone atom ratio can be dropped. Since knowing the connectivity indices of all the polymer sections almost duplicates the information in the overall polymer connectivity indices, those can be removed as well.

One may suspect that since only one monomer is varied between formulations, the monomer connectivity indices correlate to the crosslink and side chain connectivity in the polymers. That is not the case, as can be seen in Table 8.6 which shows the correlation matrix for the connectivity indices.

Table 8.6: Connectivity Index Correlation Matrix

	${}^0\xi_m$	${}^1\xi_m$	${}^2\xi_m$	${}^3\xi_m$	${}^0\xi_s$	${}^1\xi_s$	${}^2\xi_s$	${}^3\xi_s$	${}^0\xi_c$	${}^1\xi_c$	${}^2\xi_c$	${}^3\xi_c$
${}^0\xi_m$	1.00											
${}^1\xi_m$	-0.73	1.00										
${}^2\xi_m$	-0.16	-0.56	1.00									
${}^3\xi_m$	-0.45	-0.17	0.77	1.00								
${}^0\xi_s$	0.72	-0.12	-0.68	-0.81	1.00							
${}^1\xi_s$	0.08	0.61	-0.97	-0.68	0.67	1.00						
${}^2\xi_s$	-0.36	-0.37	0.96	0.80	-0.85	-0.96	1.00					
${}^3\xi_s$	-0.46	-0.21	0.85	0.91	-0.94	-0.84	0.94	1.00				
${}^0\xi_c$	-0.33	0.83	-0.79	-0.66	0.33	0.77	-0.66	-0.62	1.00			
${}^1\xi_c$	-0.42	0.88	-0.77	-0.32	0.16	0.80	-0.63	-0.42	0.83	1.00		
${}^2\xi_c$	-0.91	0.58	0.25	0.64	-0.79	-0.18	0.43	0.61	0.13	0.41	1.00	
${}^3\xi_c$	-0.88	0.44	0.42	0.75	-0.87	-0.35	0.59	0.74	-0.03	0.24	0.98	1.00

Pairwise plots of the property of interest and the descriptors can be used to find any obvious relationships. The plots are better than the correlation matrix, because polynomial or other simple mathematical relationships may be apparent from the plots; for example, the relationship between tensile strength and a descriptor could appear to be a parabola. Plots of tensile strength as a function of all the descriptors are shown in Figures 8.10 through 8.14. The labels in the diagonal of the plot matrix are the axis labels for each plot. The descriptor symbols are defined in Table 8.5.

Figures 8.10 and 8.11 seem to show a linear relationship between some of the side chain connectivity indices and the tensile strength. Figure 8.14 shows that water content lowers the tensile strength but increases degree of conversion. Generally an increased degree of conversion would be associated with higher tensile strength, but water acts as a plasticizer. Nothing in the plots seems to indicate there is a reason to use polynomial regression or transform the variables in any other way.

Figure 8.15 shows a plot of tensile strength as a function of the best single descriptor (${}^2\xi_s^v$). The line appears to fit the data reasonably well. The points WA and WC seem to have tensile strengths significantly higher than the prediction. This may be because for some reason the water in those sample did not decrease the tensile strength as

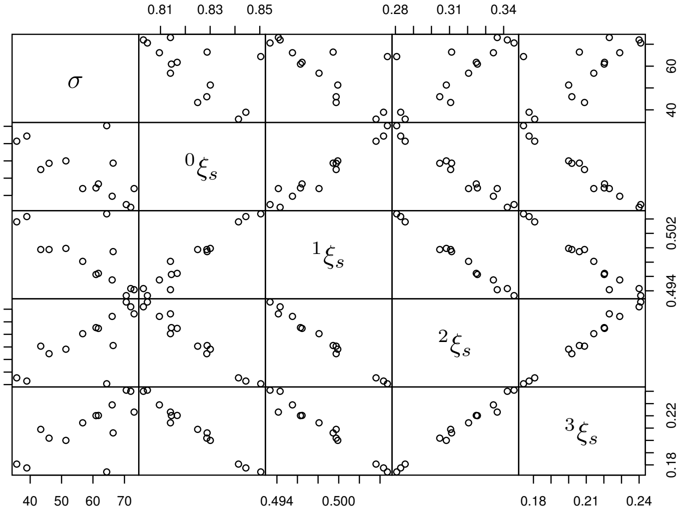


Figure 8.10: Pairwise Plots of Tensile Strength and Simple Side Chain Connectivity Indices

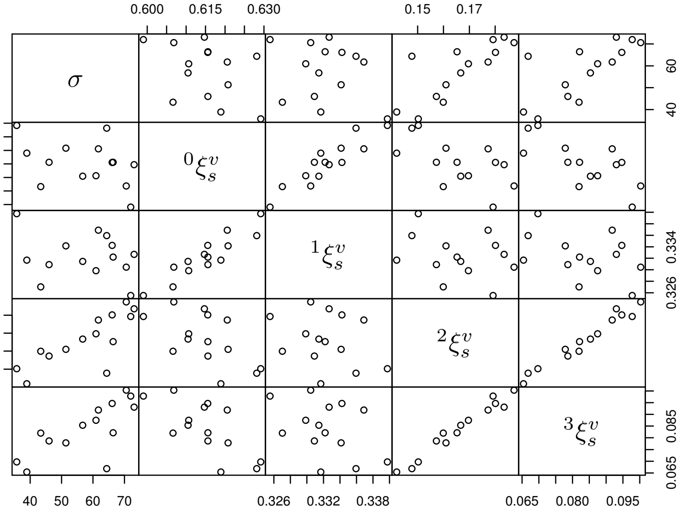


Figure 8.11: Pairwise Plots of Tensile Strength and Valence Side Chain Connectivity Indices

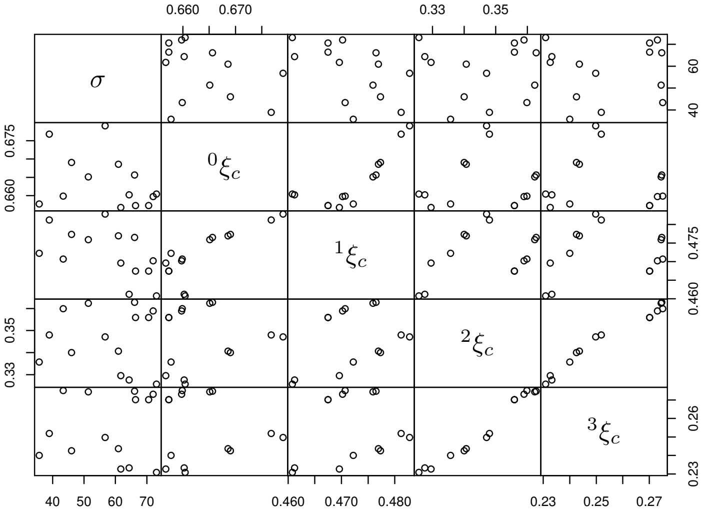


Figure 8.12: Pairwise Plots of Tensile Strength and Simple Crosslink Connectivity Indices

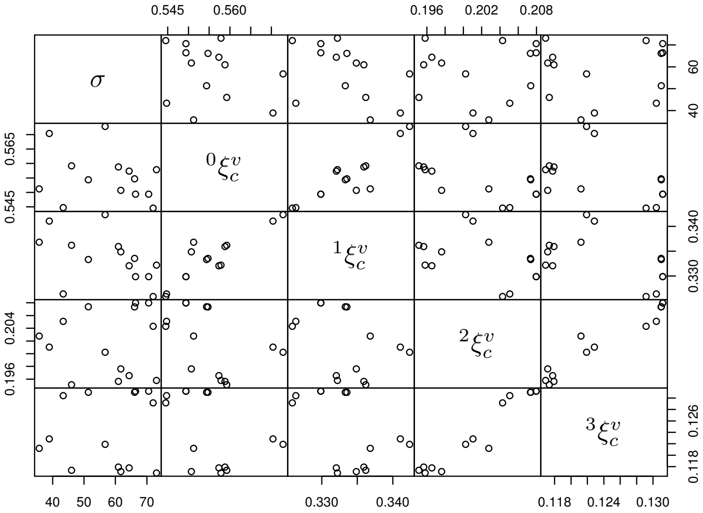


Figure 8.13: Pairwise Plots of Tensile Strength and Valence Crosslink Connectivity Indices

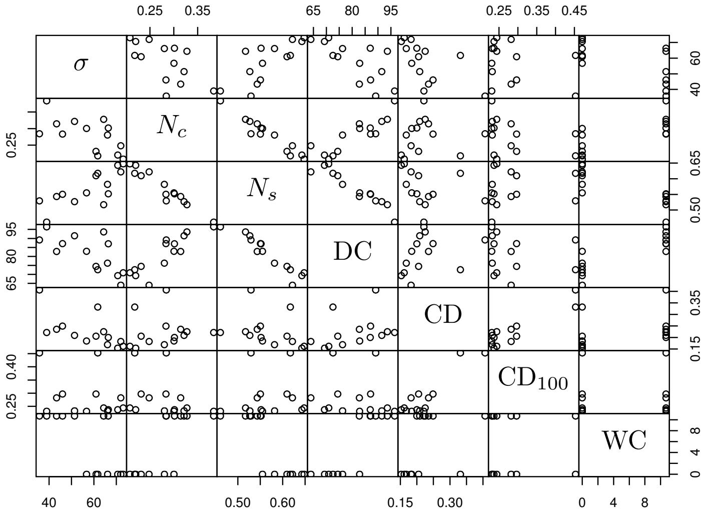


Figure 8.14: Pairwise Plots of Tensile Strength and Other Descriptors

much. The plot in Figure 8.15 also shows a problem correlating the data with and without water together. The water increases the degree of conversion and the crosslink density (both of which should increase tensile strength), but water acts as a plasticizer, decreasing tensile strength. The side chain groups in the polymer consist mostly of HEMA and bisGMA where only one functional group reacts. The addition of water shifts the balance to more HEMA side chains and lowers ${}^{2}\xi^{v}_{s}$. It is likely that the appearance of the ability of ${}^{2}\xi^{v}_{s}$ to predict tensile strength across water contents is a coincidence.

Multiple linear regression is used to make the QSPRs, but selecting the best subset of descriptors is often a difficult problem. There can be a very large number of possible combinations of descriptors, and to select the best combination, they may all need to be tried. Forward selection is often used to select descriptors, where the most significant descriptors are added to a model one at a time. Similarly, backward elimination can be used (Draper and Smith, 1966).

Forward selection and backward elimination do not necessarily provide the best result, and performing all possible regressions to find the best is not always possible.

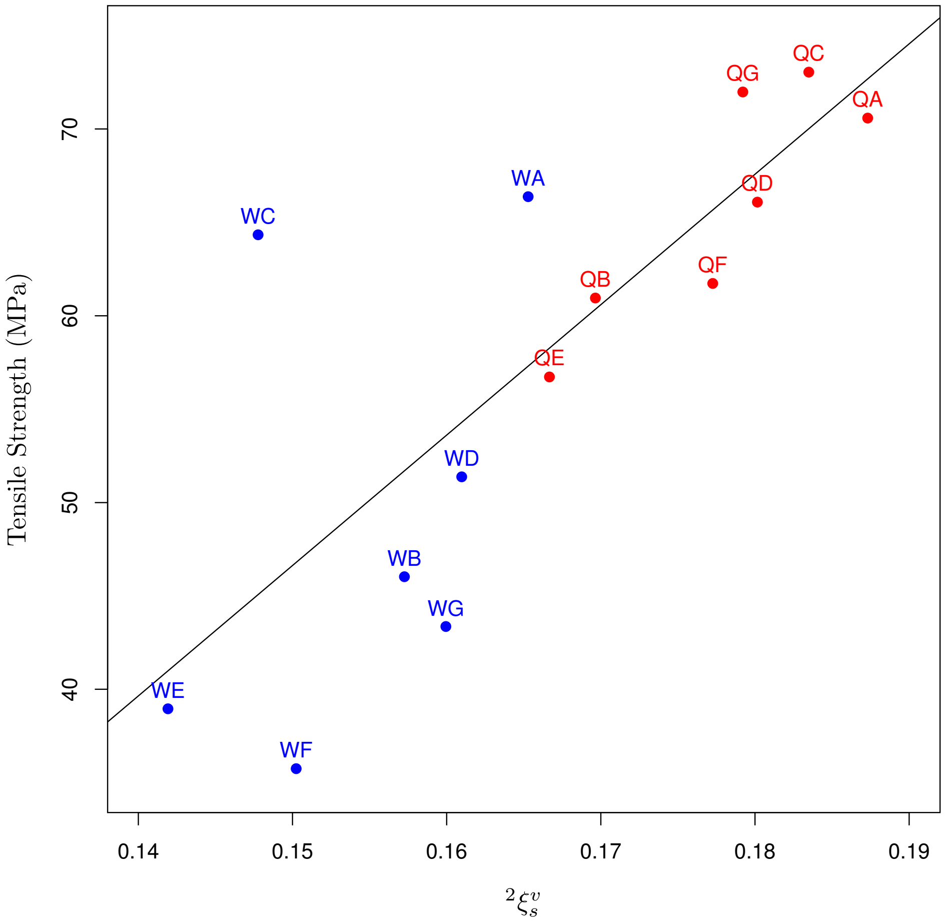


Figure 8.15: Tensile Strength as a Function of $^2\xi_s^v$

In this case, it is possible to try all combinations of descriptors up to a certain size using a branch-and-bound technique to implicitly enumerate all possibilities. A package called LEAPS is available to do this (Lumley, 2004).

There are twenty-two possible descriptors, so there are 4,194,304 possible combinations of descriptors. There are only 14 data points, so there is no way to have a 22 variable model. If only models containing up to 9 variables are considered, there are 1,097,789 possible combinations. A computer can test that many possibilities in a reasonable amount of time. Table 8.7 shows the descriptors included in the best models for tensile strength and their R^2 values.

Table 8.7: Best Combinations of Descriptors for Tensile Strength and Their R^2

No. Desc.	1	2	3	4	5	6	7	8	9	R^2 (14 points)
1	${}^2\xi_s^v$									0.616
2	CD	N_s								0.741
3	WC	CD	${}^1\xi_c$							0.872
4	CD	N_{aro}	${}^2\xi_c^v$	${}^3\xi_c^v$						0.921
5	WC	DC	CD	N_{aro}	${}^3\xi_c$					0.933
6	N_{aro}	${}^0\xi_s$	${}^3\xi_s^v$	${}^0\xi_c$	${}^0\xi_c^v$	${}^3\xi_c^v$				0.966
7	N_{aro}	${}^1\xi_s$	${}^2\xi_s$	${}^3\xi_s^v$	${}^1\xi_c$	${}^1\xi_c^v$	${}^3\xi_c^v$			0.984
8	CD_{100}	N_{aro}	N_c	${}^2\xi_s$	${}^3\xi_s$	${}^0\xi_c^v$	${}^1\xi_c^v$	${}^2\xi_c^v$		0.995
9	CD_{100}	N_{aro}	N_c	${}^0\xi_s$	${}^2\xi_s^v$	${}^3\xi_s^v$	${}^1\xi_c$	${}^0\xi_c^v$	${}^2\xi_c^v$	0.9997

It is important to check the statistical significance of the model coefficients and the model itself. Using a 95% confidence level, all of the models are statistically significant, and only the five-variable model has a parameter that is not statistically significant. It is also clear from looking at the included variables that forward selection would not have provided the best model.

To decide which model is best, it is possible to look at R^2, but R^2 always increases as more parameters are added. If a model contains too many parameters, it will overfit the data, and do a poor job of predicting properties for new polymers. K-fold cross-validation is used to get some idea of the predictive value of the models.

The cross-validation procedure starts by randomly splitting the data into three sets to be omitted in turn. The size of the sets is as close to equal as possible. Since there are 14 data points total, the sets contain 5, 5, and 4 data points. Three models are created the first with the data omitting the first set of five points, and so on. The error is calculated for the points left out, and used to calculate Q^2. Q^2 is comparable to R^2, but it is calculated from the points left out of the model. It is possible for Q^2 to be negative if the predictions are worse than just using the mean of the model data as the prediction. Since the data is split up randomly, Q^2 may be different every time it is calculated. To get a fair representation, Q^2 was calculated 500 times for each model; the results are shown in Table 8.8.

Table 8.8: Q^2 Values for 3-fold Cross-Validation

	Q^2 500 Measurements					
no. desc.	minimum	first quartile	median	mean	third quartile	maximum
7	-17.00	0.57	0.74	0.57	0.85	0.95
6	-5.39	0.67	0.78	0.67	0.84	0.94
5	-4.28	0.65	0.76	0.64	0.82	0.92
4	-5.21	0.72	0.79	0.73	0.84	0.91
3	-1.14	0.69	0.75	0.71	0.81	0.88

It is not clear from Table 8.8 which is the best model; they all seem to have a similar median. Some of the Q^2 values are very bad in the models with more parameters, which pulls down the mean. The negative Q^2's probably result when most or all of the data to be left out in one set is cured with water and in another set cured without water. This would be like making a QSPR using only data for polymer cured without water, and then trying to use it to predict for properties of polymers cured with water. Since the data set is small, the three predictor model is selected to be the best.

The large representative section of polymer is used to calculate the connectivity indices, but the other descriptors can be calculated without it. The next question is whether a suitable QSPR can be created using just the unreacted monomer connectivity indices. In that case, the calculations would be faster and the optimization routine could be more efficient. Table 8.9 provides a summary of the best models using the monomer connectivity indices.

Table 8.9: Best Tensile Strength Models Based on Monomer Connectivity

	Q^2 500 Measurements				
no. desc.	first quartile	median	mean	third quartile	R^2
6	0.34	0.55	-0.08	0.70	0.93
5	0.60	0.71	0.66	0.78	0.93
4	0.64	0.71	0.67	0.77	0.89
3	0.68	0.77	0.72	0.82	0.88

There is very little difference between the three descriptor QSPRs using the polymer or the monomer connectivity indices. Since the monomer-based QSPR takes less time to calculate and is at least as good as the polymer-based QSPR, it is clearly the best choice for the optimization routine.

Following same procedure outlined for tensile strength, QSPRs were created for initial polymerization rate (r_0), glass transition temperature (T_g), and degree of conversion (DC). Table 8.10 lists the QSPRs. No QSPRs were found for the remaining properties.

Table 8.10: QSPR Summary

QSPR	R^2
$\mathrm{DC} = -66.06 + 1.5654206 \cdot \mathrm{WC} + 479.8702353 \cdot {}^1\xi_m^v - 120.4479529 \cdot {}^3\xi_m^v$	0.97
$\sigma = 15.42 - 308.907928 \cdot \mathrm{CD} + 194.831990 \cdot \mathrm{CD}_{100} + 242.095661 \cdot {}^2\xi_m^v$	0.88
$T_g = 674.15 + 0.61089833 \cdot \mathrm{DC} - 2190.46918205 \cdot {}^1\xi_m^v + 401.65206265 \cdot {}^2\xi_m^v$	0.85
$r_0 = -528.95 - 151.20634607 \cdot \mathrm{CD}_{100} + 831.35424142 \cdot {}^0\xi_m + 0.09496638 \cdot \mathrm{MW}_m$	0.83

The data set is small, so the QSPRs developed here are probably of limited scope, but they serve to illustrate the CMD process.

8.5 Optimization

Tabu search is used in this project to find a list of candidate monomers that are predicted to have favorable properties. The resins for new polymers are formulated as in the experimental data, with the new monomer comprising 25 wt% of the resin.

The first step in the optimization process is to determine the objective function. The properties are weighted using their magnitude in the objective function. This is a reasonable starting point, but as more properties are added and more data is collected it may be useful to weight them additionally by relative importance. With the target properties, the objective function may be written as Equation 8.1. Since the solution method used is Tabu search, it is not necessary to make the function linear, and squared terms are used to avoid cancellation of errors.

$$f = \left(\frac{\sigma - 80}{80}\right)^2 + \left(\frac{r_0 - 130}{130}\right)^2 + \left(\frac{T_g - 120}{120}\right)^2 + \left(\frac{\mathrm{DC} - 100}{100}\right)^2 \tag{8.1}$$

The constraints for the optimization problem limit solutions to feasible structures for methacrylate monomers. Since crosslinking is desired, the monomers are also required to have at least two methacrylate groups. All of the constraints are handled in the procedure to build new monomers, so they do not need to be explicitly formulated.

The Tabu search algorithm takes advantage of the data structures already available in the polymer designer software. The new monomers are built from a set of groups. Changes are made by inserting, deleting or changing chain groups. Inserting and deleting chain groups only lengthens or shortens monomers; the structure is still feasible. Changing the type of chain group also results in a feasible structure. The groups are put together using a polymer graph, and the full graph becomes the monomer (see Chapter 3). Figure 8.16 shows the groups used. The groups were selected to produce monomers similar to those in the experimental data.

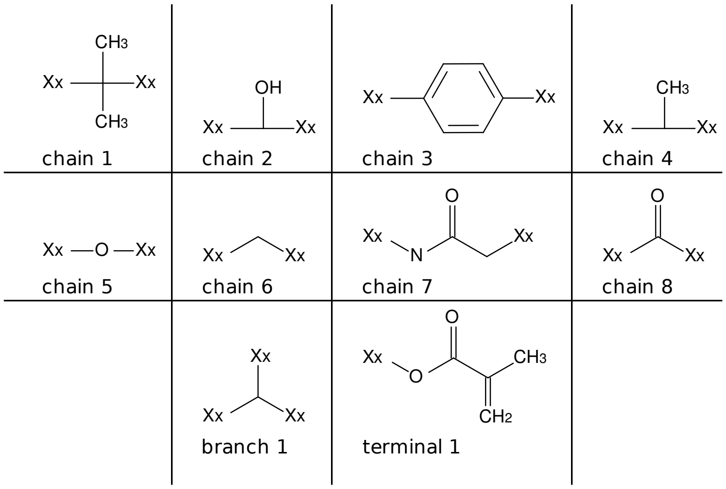


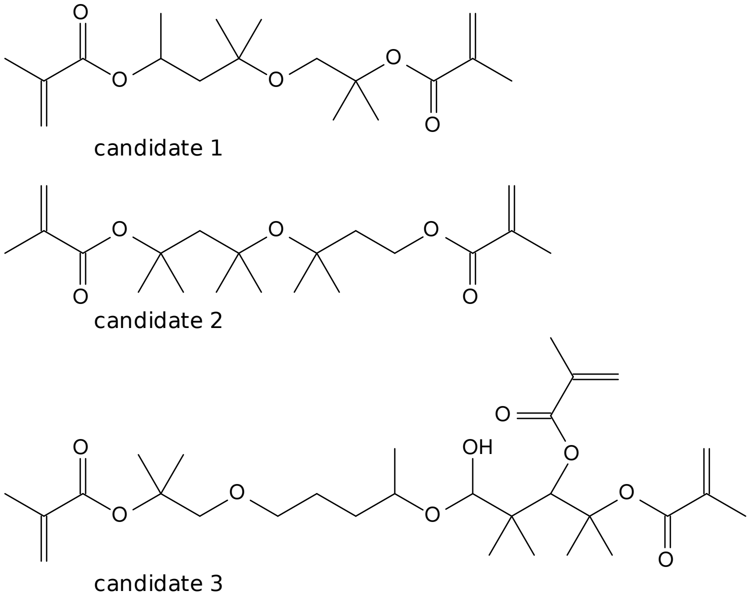
Figure 8.16: Groups Used to Build Monomers in Tabu Search

New solutions are generated by inserting, deleting, or changing chain groups. The level of branching is changed by setting the number of branch groups in the initial solution; this eliminates the possibility of large changes in the solutions when branching groups are deleted.

8.6 Results

A list of candidate structures was generated by running the Tabu search algorithm several times using different starting structures. Three representative resulting structures are shown in Figure 8.17. The first candidate monomer was obtained using bisGMA as the starting structure. The second candidate was obtained using random starting structures with two methacrylate groups. The third candidate structure was obtained using random starting structures with three methacrylate groups. Tabu search took approximately one minute to run from each new starting point. The search was terminated after 300 non-improving iterations.

The objective function values and properties of polymers made with the candidate


candidate 1

candidate 2

candidate 3

Figure 8.17: Candidate Monomer Structures

Table 8.11: Predicted Candidate Polymer Properties

Monomer	DC (%)	σ (MPa)	T_g (°C)	r_0 (mol/l $\cdot$ s)	objective
target	100.0	80	120	130	0.0000
bisGMA	69.3	71	119	96	0.0995
candidate 1	78.0	82	119	125	0.0305
candidate 2	80.4	86	115	130	0.0296
candidate 3	76.3	75	112	129	0.0426

monomer as the third monomer in the formulation are given in Table 8.11. The properties for the ideal polymer and the HEMA/bisGMA polymer are given for comparison.

All of the candidate monomers in Table 8.11 offer a significant improvement over the bisGMA control. Given the structural limitations, it seems not to be possible to obtain degrees of conversion near 100%, so the structural descriptors can provide limits on the QSPRs which are not otherwise obvious. One may expect from the linear equation for degree of conversion that values of over 100% or negative values could be predicted; those values would obviously make no physical sense.

8.7 Conclusion

Although significantly more data is needed to generate new QSPRs, this example shows that the CMD methodology can be applied to crosslinked polymers. More data must be collected and a wider variety of descriptors need to be examined. The results do look promising and suggest that the type of structure description used can be successfully applied to property prediction for crosslinked polymer networks. Tabu search also appears to be effective as an optimization method for this application.

The successful result of the demonstration is encouraging and indicates further work in this area has a high probability of success.

The next chapter provides the overall conclusions and recommendations for this work based on the design formulation implemented for polymeric dental materials.

Chapter 9

Conclusions and Recommendations

9.1 Conclusions

The dental polymer example in the previous chapter demonstrates that the new data structure for crosslinked polymers is capable of storing the necessary structural information and is efficient for the calculation of topological descriptors. The core and buffer method allows a relatively small representative section of polymer to be used for descriptor calculations without the severed polymer chain ends affecting the calculation. The QSPRs developed and cross-validated in the example strongly suggest that QSPRs can be developed for crosslinked polymers, an area in which there has been little previous research.

The suitability of Tabu search as an optimization routine for this new area of molecular design is also demonstrated. Tabu search avoids the need for rigorous mathematical formulation inherent in deterministic methods, allowing for considerable flexibility in the molecular design process. Tabu search and the formulation used in this work also help manage the combinatorial increase in problem size when designing large molecules.

The combinatorial design methodology developed for this project is shown to be a useful platform for numerical experimentation in the design of crosslinked polymers. The implementation of the crosslinked polymer data structure allows rapid calculation of numerous topological descriptors. The software can also be used for many other types of molecules, and provides much flexibility in the design methodology. The availability of the data structures and descriptor calculation functions greatly expedite development of optimization techniques for molecular design.

Finally, the example demonstrates that this methodology is an effective tool for the design of crosslinked polymeric materials such as dental materials, and is a promising area for continuing research.

9.2 Recommendations

The dental adhesive design project is ongoing, and more experimental data is currently being collected by the Spencer research group. The new data will allow for improved quantitative structure property relations and provide more details about the structure of the polymer network. Several further studies can be performed as new data is collected.

An extensive review of the polymer designer source code should be conducted to ensure that calculations are accurate, and that any omissions are well documented and produce appropriate error messages.

For dental polymers, specific target properties need to be set, and the weighting of the properties in the objective function needs to be carefully considered. A database of commercially available methacrylate monomers should then be compiled. Using available data, the properties of polymers made from all the monomers in the database can be estimated. Polymer designer can store the structures, and do the property estimate calculations. Since the monomers are readily available, properties can be measured for both good and bad monomers without the need for time consuming synthesis. This will provide a good means of testing and improving the QSPRs, while still finding improved materials.

Additional topological descriptors of molecular structure should be studied. Numerous descriptors are available in the literature, and the expanded data set will allow for a study to find which descriptors work best for predicting properties. The facilities of Polymer Designer can be used to perform the numerical experiments with different types of descriptors.

More complex QSPRs should be examined. Transformations of the descriptors can also be considered in the regression, but there is currently not sufficient data to do so. Molecular descriptors are useful for small molecules and linear polymers, but they also miss the details of a large polymer network. Large scale descriptors for polymer networks should be developed. These descriptors could be calculated on the polymer graphs with monomer vertices. The process could start using descriptors analogous to existing molecular descriptors.

Development of large scale polymer network descriptors may require a more accurate picture of the network structure. New methods should be developed to generate a more accurate representative section of polymer. As additional experimental data is collected, it should be analyzed to determine if it can provide details of the molecular structure or provide evidence to the accuracy of structures generated for the purpose of descriptor calculation.

As more experiments are performed, the data should be used to validate and update the QSPRs. Eventually, as the results of the molecular design process are synthesized and tested, the results will be fed back into the modeling process creating a loop that will lead to rapid development of improved polymers.

Nomenclature

$\mathbf{A}$	adjacency matrix
A	number of non-hydrogen atoms in a molecule
BFS	breadth first search
BisGMA	2,2-bis[4(2-hydroxy-3-methacryloyloxy-propyloxy)-phenyl]
BisEMA	bisphenol A polyethylene glycol diether dimethacrylate
CD	crosslink density
CD_{100}	crosslink density at 100% conversion
CMD	computational molecular design
DC	degree of conversion, fraction of carbon-carbon double bonds reacted
DFS	depth first search
F_i	fraction of component i
HEMA	2-hydroxyethly methacrylate
LEAPS	a software package to select the best subset of predictors
$P_{i,\text{targ}}$	target value for property i
$P_{i,\text{est}}$	estimated value for property i
MPE	1,1,1-tri-[4-(methacryloxyethylaminocarbonyloxy)-phenyl]ethane
N	number of non-hydrogen atoms in a molecule
N_{aro}	number of atoms in aromatic rings
N_c	number of atoms in crosslinks
N_H	number of hydrogen atoms attached to an atom

n_i	Tabu search parameter, number of non-improving iterations
n_n	Tabu search parameter, number of neighbors to generate
N_{rv}	number of reacted vinyl groups
N_s	number of side chain atoms
n_t	Tabu search parameter, number of solutions in the Tabu list
^2P	number of length 2 paths in a molecule
PD	the Polymer Designer software
PEGDMA	polyethylene glycol dimethacrylate
QSPR	quantitative structure property relation
R	statistical software package
r_0	initial reaction rate
R^2	multiple correlation coefficient
s_i	a scaling factor for the properties in the objective function
T_g	glass transition temperature
TEGDMA	triethyleneglycol dimethacrylate
TMPEDMA	trimethylol-propane mono allyl ether dimethacrylate
UDMA	urethane dimethacrylate
UID	unique identifier, an integer
WC	water content of resin (wt%)
wt%	weight percent
Xx	dummy atom
$\bar{y}$	average of y_i
$\hat{y}$	predicted value of y
Z	number of electrons around an atom
Z^v	number of valence electrons around an atom
δ_i	value used in the calculation of simple connectivity indices for atom i
δ_i^v	value used in the calculation of valance connectivity indices for atom i
$^2\kappa$	path index

${}^2\kappa_\alpha$	path index adjusted for atom size
σ	tensile strength
${}^n\chi$	size dependent nth-order simple connectivity index
${}^n\chi^v$	size dependent nth-order valence connectivity index
${}^n\xi$	size independent nth-order simple connectivity index
${}^n\xi^v$	size independent nth-order valence connectivity index

References

Allcock, H. R., "Rational design and synthesis of new polymeric materials," *Science*, **255**, 1106 (1992).

Bates, D. M. and D. G. Watts, *Nonlinear Regression Analysis and its Applications*, John Wiley & Sons, New York (1988).

Bernardo, M., H. Luis, M. D. Martin, B. G. Leroux, T. Rue, J. Leitão, and T. A. DeRouen, "Survival and reasons for failure of amalgam versus composite posterior restorations placed in a randomized clinical trial," *Journal of the American Dental Association*, **138**, 775 (2007).

Bicerano, J., *Prediction of Polymer Properties*, 3rd edition, Marcel Dekker, Inc. (2002).

Bicerano, J., R. L. Sammler, C. J. Carrier, and J. T. Seitz, "Correlation between glass transition temperature and chain structure for randomly crosslinked high polymers," *Journal of Polymer Science*, **34**, 2247 (1996).

Camarda, K. V. and C. D. Maranas, "Optimization in polymer design using connectivity indices," *Industrial Engineering and Chemistry Research*, **38**, 1884 (1999).

d'Anterroches, L. and R. Gani, "Group contribution based process flowsheet synthesis design and medelling," *Fluid Phase Equilibria*, **228**, 141 (2005).

de Werra, D. and A. Hertz, "Tabu Search techniques," *OR Spektrum*, **11**, 131 (1989).

Draper, N. R. and H. Smith, *Applied Regression Analysis*, John Wiley and Sons, New York (1966).

Efron, B. and R. J. Tibshirani, *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability*, Chapman and Hall, New York (1993).

Finer, Y. and J. P. Santerre, "The influence of resin chemistry on a dental composites biodegradation," *Journal of Biomedical Materials Research, Part A*, **69A**(2), 233 (2004).

Flory, P. L., "Molecular size distribution in three dimensional polymers. I. Gelation," *Journal of the American Chemical Society*, **63**, 3083 (1941).

Fredenslund, A., R. L. Jones, and J. M. Prausnitz, "Group-contribution estimation of activity coefficients in nonideal liquid mixtures," *AIChE Journal*, **21**, 1086 (1975).

Gani, R., B. Nielsen, and A. Fredenslund, "A group contribution approach to computer-aided molecular design," *AIChE Journal*, **37**(9), 1318 (1991).

Gibbons, A., *Algorithmic Graph Theory*, Cambridge University Press (1985).

Glover, F., "Artificial intelligence, heuristic frameworks and Tabu Search," *Managerial and Decision Economics*, **11**, 365 (1990a).

Glover, F., "Tabu Search: A tutorial," *Interfaces*, **20**, 74 (1990b).

González, H. E., J. Abildskov, and R. Gani, "Computer-aided framework for pure component properties and phase equilibria prediction for organic systems," *Fluid Phase Equilibria*, **261**, 199 (2007).

Joback, K. G. and R. C. Reid, "Estimation of pure-component properties from group-contributions," *Chemical Engineering Communications*, **57**, 233 (1987).

Karunanithi, A. T., L. E. K. Achenie, and R. Gani, "A new decomposition-based computer-aided molecular/mixture design methodology for the design of optimal solvents and solvent mixtures," *Industrial Engineering and Chemistry Research*, **44**, 4785 (2005).

Kier, L. B., "Distinguishing atom differences in a molecular grapth shape index," *Quantitative Structure-Activity Relations*, **5**, 7 (1986).

Kier, L. B., "Index of molecular shape from chemical graphs," *Medicinal Research Reviews*, **7**, 417 (1987).

Kier, L. B. and L. H. Hall, *Molecular Connectivity in Structure-Activity Analysis*, Research Studies Press, Letchworth, England (1986).

Lin, B., S. Chavali, K. V. Camarda, and D. C. Miller, "Computer-aided molecular design using Tabu Search," *Computers and Chemical Engineering*, **29**, 337 (2005).

Lumley, T., *The leaps Package*, URL `cran.r-project.org/doc/packages/leaps.pdf` (2004).

Maranas, C. D., "Optimal computer-aided molecular design: a polymer design case study," *Industrial Engineering and Chemistry Research*, **35**, 3403 (1996).

MDL Information Systems, *CTFile Formats*, Elsevier, Elsevier MDL, 14600 Catalina St., San Leandro, CA 94577 (2005).

Mevik, B. and R. Wehrens, "The pls package: principal component and partial least squares regression in R," *Journal of Statistical Software*, **18**(2) (2007).

Murray, P. E., L. J. Windsor, T. W. Smyth, A. A. Hafez, and C. F. Cox, "Analysis of Pulpal Reactions to Restorative Procedures, Materials, Pulp Capping, and Future Therapies," *Critical Reviews in Oral Biology and Medicine*, **13**(6), 509 (2002).

Park, J., Q. Ye, E. M. Topp, E. L. Kostoryz, Y. Wang, S. L. Kieweg, and P. Spencer, "Preparation and properties of novel dentin adhesives with esterase resistance," *Journal of Applied Polymer Science*, **107**, 3588 (2007).

Quan, N. T., “The prediction sum of squares as a general measure for regression diagnostics,” *Journal of Business & Economic Statistics*, **6**, 501 (1988).

R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, URL `http://www.R-project.org` (2007).

Randić, M., “On characterization of molecular branching,” *Journal of the American Chemical Society*, **97**(23), 6609 (1975).

Rouvray, D. H., “Predicting chemistry from topology,” *Scientific American*, **255**(3), 40 (1986).

Sahinidis, N. V., M. Tawarmalani, and M. Yu, “Design of alternative refrigerants via global optimization,” *AIChE Journal*, **49**, 1761 (2003).

SQLite, “SQLite,” http://www.sqlite.org/ (2008).

Sundaram, A. and V. Venkatasubramanian, “Parametric sensitivity and search-space characterization studies of genetic algorithms for computer-aided polymer design,” *Journal of Chemical Information and Computer Sciences*, **38**, 1177 (1998).

Todeschini, R. and V. Consonni, *Handbook of Molecular Descriptors*, Wiley-VCH, Weinheim (2000).

Trolltech, “Qt Cross-Platform Application Framework,” http://trolltech.com/products/qt (2008).

Ullmann, J. R., “An algorithm for subgraph isomorphism,” *Journal of the Association of Computing Machinery*, **23**(1), 31 (1976).

Van Krevelen, D. W., *Properties of Polymers*, 3rd edition, Elsevier (1997).

Venkatasubramanian, V., “Computer-aided molecular design using genetic algorithms,” *Computers & Chemical Engineering*, **18**, 833 (1994).

West, D. B., *Introduction to Graph Theory*, 2nd edition, Prentice Hall, Upper Saddle River (2001).

Wiener, H., "Correlation of heats of isomerization, and differences in heats of vaporization of isomers among the paraffin hydrocarbons," *Journal of the American Chemical Society*, **69**, 2636 (1947a).

Wiener, H., "Structural determination of paraffin boiling points," *Journal of the American Chemical Society*, **69**, 17 (1947b).

Ye, Q., P. Spencer, Y. Wang, and A. Misra, "Relationship of Solvent to the Photopolymerization Process, Properties, and Structure in Model Dentin Adhesives," *Journal of Biomaterials Research Part A*, **80**, 342 (2007a).

Ye, Q., Y. Wang, Willians, and P. Spencer, "Characterization of photopolymerization of dentin adhesives as a function of light source and irradiance," *Journal of Biomaterials Research Part B*, **80**, 440 (2007b).

Appendix A

Polymer Designer Manual

This appendix describes in detail how to use the Polymer Designer (PD) program. These instructions may be most helpful while running the program. Appendix C provides information about using the existing functions of PD to extend its capability.

A.1 Installing PD

PD should work under Linux, UNIX, OS X, and Windows, but it has only been tested in Linux. The QT, SQLite3, and OpenGL development files and a C++ compiler must be available to compile PD. The QT build system is used to simplify the process of compiling PD. Instructions for Linux are provided here.

In the pd2 directory type:

```
qmake -project
```

This command will make a project file that qmake will use to build a makefile. Since PD uses OpenGL and SQLite3, some small changes must be made to the project file. Edit the pd2.pro file, and add the following lines near the beginning.

```
LIBS += -lsqlite3
```

```
QT += opengl xml
```

If there may be a need to run pd2 in a a debugger, also add the following line.

```
CONFIG += debug
```

The next step is to build the makefile using the following command in the pd2 directory.

```
qmake
```

The next step is compile the project with the following command.

```
make
```

This will build an executable called “pd2.” The executable will stand alone and can be stored anywhere independent of the source files.

PD should be run from a terminal program. PD outputs some diagnostic and error messages to the standard output, and these will be lost if the program is run by clicking on it in a file manager.

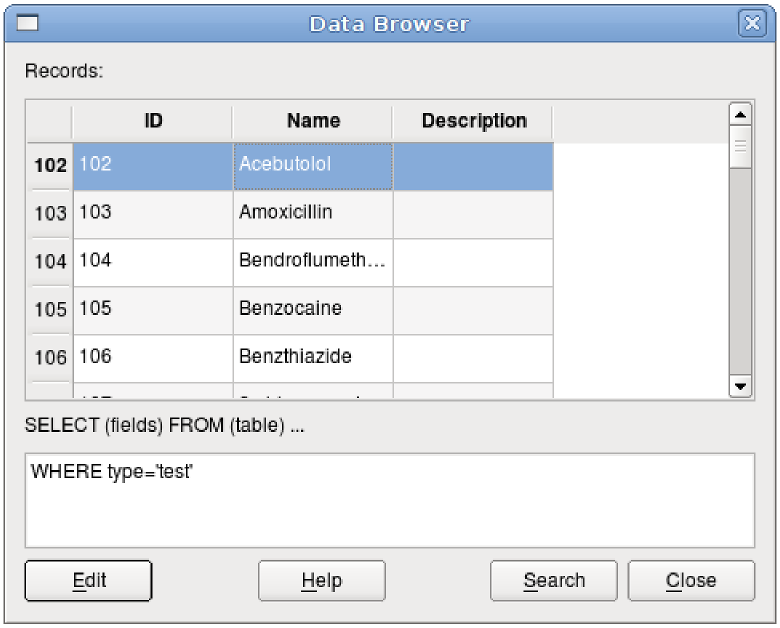
A.2 Opening a Database

The first thing to do upon running PD is to open a new or existing database. This can be done from the file menu. Databases are stored entirely in one file, so they are easy to backup. Changes made to data are committed to the database when an update button is pressed, so there is no need to save. Changes should be made with care and the database files should be backed up, as PD currently has no undo option. A different database can be opened at anytime, but only one database can be open at a time.

A.3 Adding and Editing Monomers

Monomers can refer to actual monomers used to build polymers, or the monomer facilities may also be used for molecular design studies with other small molecules.

Adding and editing monomers can be done from the "Molecules" menu. Adding or editing a monomer will bring up the "Structure Data" window. If editing monomers, the "Data Browser" window will appear fist allowing you to select from existing monomers (Figure A.1). To see a list of all monomers, just click the search button. SQL statements can also be used to search for a specific set of monomers. Fields that may be used in a query are: name, desc (description), type, cite_key, cite_notes, and notes.



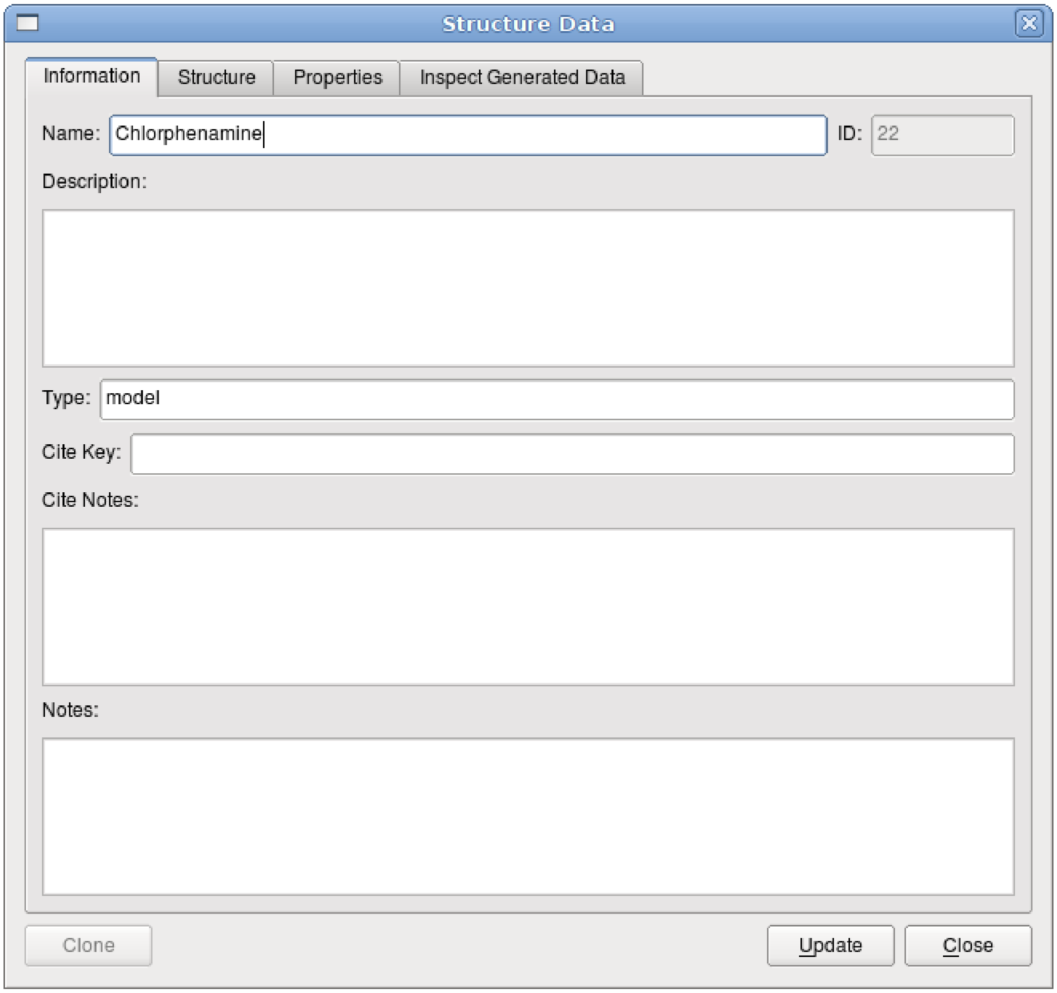
	ID	Name	Description
102	102	Acebutolol	
103	103	Amoxicillin	
104	104	Bendroflumeth...	
105	105	Benzocaine	
106	106	Benzthiazide	

WHERE type='test'

Figure A.1: Monomer Data Browser

The structure data window contains a form for basic data about a molecule under the information tab, as seen in Figure A.2.

The structure tab contains the chemical structure editor. The structure editor allows molecules to be easily viewed and edited using the mouse (Figure A.3).



Name: Chlorphenamine

ID: 22

Description:

Type: model

Cite Key:

Cite Notes:

Notes:

Figure A.2: Chemical Structure Editor

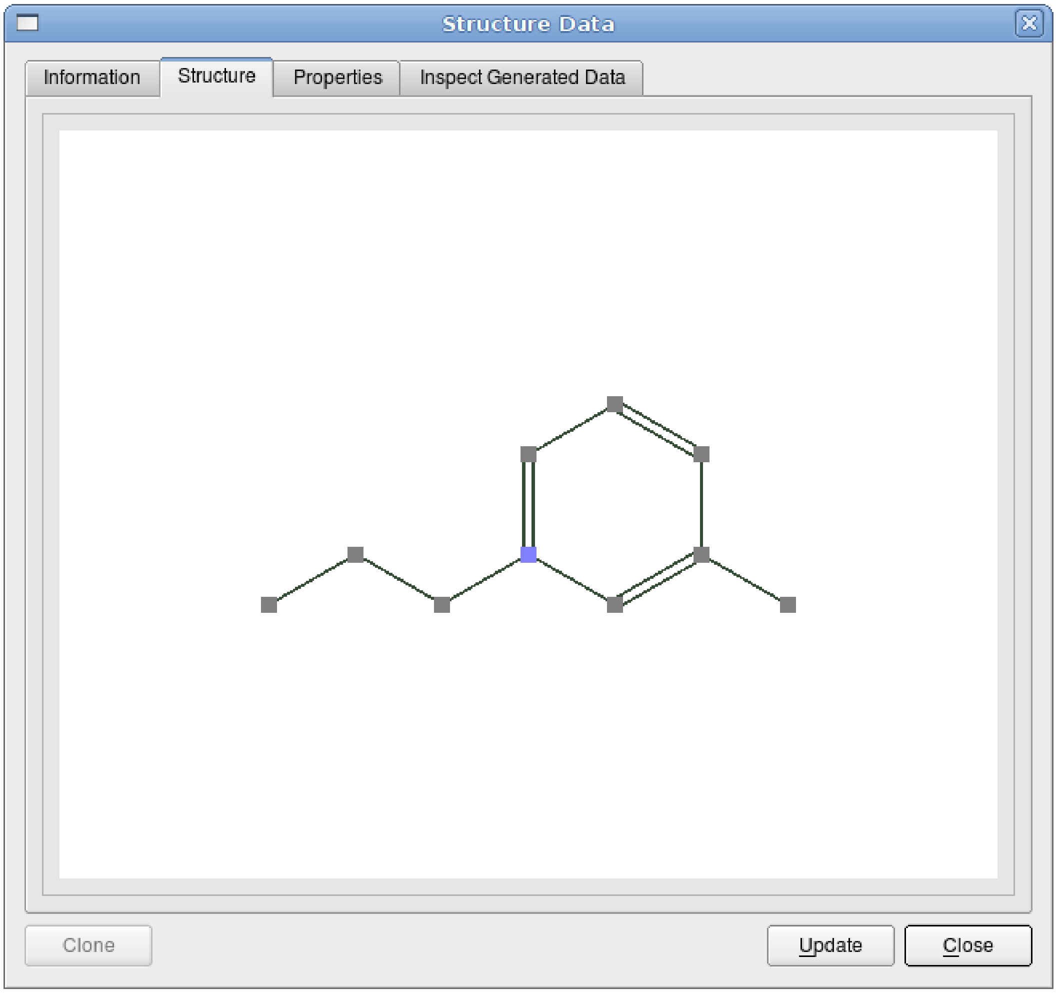
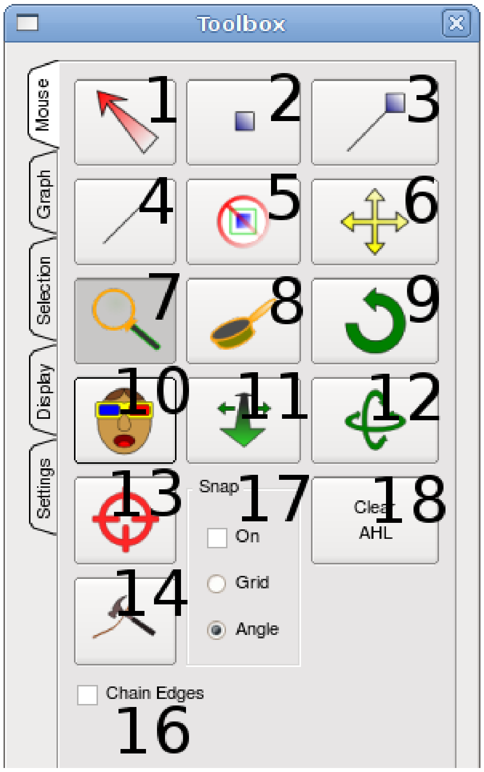


Figure A.3: Monomer Data Entry Form

Editing a molecule is accomplished using the mouse and the graph editor toolbox. The toolbox consists of several tabs. The important tools and how to use them are described here. If the toolbox is not displayed, press the ’t’ key with the graph editor selected. Figure A.4 shows the mouse tab of the toolbox.



Snap
□ On
○ Grid
● Angle

□ Chain Edges

Figure A.4: Toolbox – Mouse Tab

1. This button activates the selection tool. While using the selection tool, a vertex or edge can be selected by clicking on it. This tool can also be used to edit vertex and edge properties by double clicking on them. The vertex and edge editing dialog boxes will be described later.

2. This button activates the vertex tool. While using the vertex tool a new vertex will be created by clicking the mouse in the desired location. The vertex is added when the mouse button is pressed down. While the button is held down, the

vertex may be dragged to the desired location. Dragging with snapping enabled moves the vertex on the snap grid.

3. This button activates the vertex/edge tool. This tool draws both a vertex and an edge. First select a vertex then select the location for the new vertex. An edge will be created automatically. If, instead of clicking an empty space, an existing vertex is selected, this tool draws an edge between the existing vertices.

4. This button activates the edge tool. This tool can be used to add an edge between existing vertices.

5. This button activates the delete tool. While this tool is active, clicking a vertex or edge will delete it. If a vertex is deleted, all edges with that vertex as an endpoint will be deleted too.

6. This button activates the move mode. While this mode is active, vertices can be dragged to new positions.

7. This button activates the zoom mode. To zoom in click and drag right. To zoom out click and drag left.

8. This button activates the pan mode. To pan, click and drag.

9. This button activates the rotate about the $z-$axis mode.

10. This button activates the 3D view mode. Although editing in 3D is not implemented in PD, some molecules that are imported may have 3D coordinates.

11. This button activates the 3D pan mode.

12. This button activates the 3D rotate mode.

13. This button translates the molecule so that the center is at $(0, 0, 0)$.

14. This button activates the hammer. Edges can be bent one way by left-clicking and the other by right-clicking. This is useful if a graph has overlapping edges. Molecules generally should not have multiple edges between the same two vertices.

15. This number is missing.

16. This check box activates the chain edges mode. In vertex/edge or edge mode this allows a chain of atoms and bonds to be drawn quickly.

17. This activates snap mode. The snap to grid option forces new or moved vertices to snap to a fixed rectangular grid when they are dragged. The angle option forces new or moved vertices into configuration where an edge between the current and previously selected vertex is a one of a certain set of angles. When used with a fixed distance between the vertices, the angle snap provides a nice way to draw standard looking molecule drawings.

18. This button clears the edge and vertex highlighting that is drawn by some algorithm tests.

The graph tab of the toolbox is shown in Figure A.5. The graph tab contains some operations that apply to the whole graph. The import and export buttons allow molecules to be read and saved as mol- or mol2-files. The native XML format of PD2 can also be read and saved. The clear button erases the entire graph. The graph operations button can contain functions that do things to the graph; it does nothing for molecules. The graph information button gives some basic graph information about the graph such as, the number of vertices and the number of edges. The algorithm test button can run some of the graph algorithms on the current graph for debugging purposes.

The display tab in the toolbox allows various information about atoms and vertices to be displayed on the graph. The settings tabs allows the snap-grid, zoom, pan, and rotate settings to be edited.

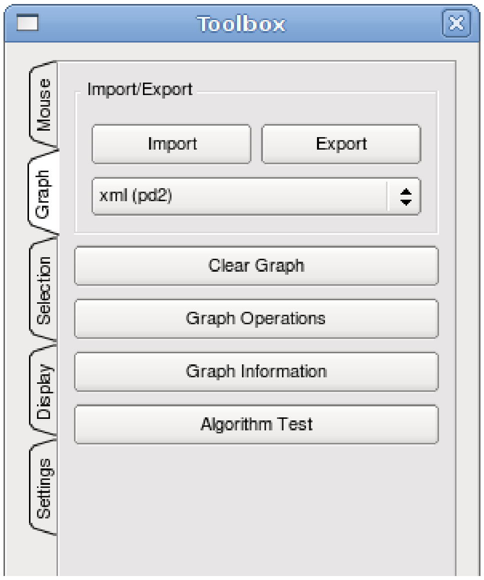


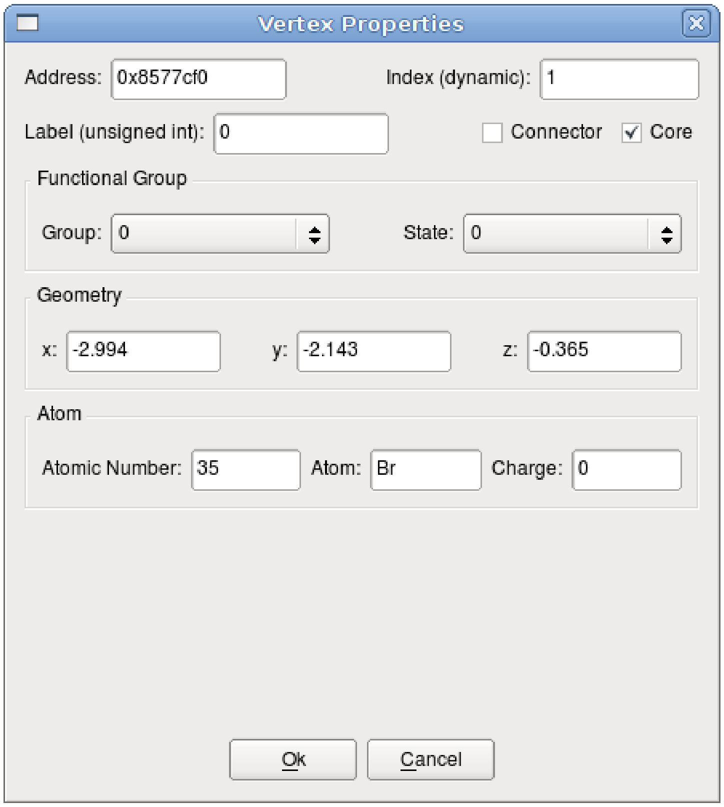
Figure A.5: Toolbox – Graph Tab

The vertex and bond editing dialog boxes are shown in Figures A.6 and A.7. These can be accessed by double-clicking an atom or bond with the selection tool. The atom type can be edited using the symbol or atomic number box; the other box will update automatically.

The default atom and bond type can be changed using the periodic table dialog box shown in Figure A.8. The periodic table can be accessed by pressing the 'p' key in the graph editor.

Once a monomer structure and data are properly entered, the insert or update button at the bottom of the structure editor will insert or update the record. The structure window retains the information the next time it is opened, which is useful when adding many similar structures.

An SDL file is another common format for storing molecular structures. An SDL file contains several mol-files. A set of molecules can be imported from an SDL file by choosing import SDL from the molecules menu.



Vertex Properties

Address: 0x8577cf0

Index (dynamic): 1

Label (unsigned int): 0

□ Connector ☑ Core

Functional Group

Group: 0

State: 0

Geometry

x: -2.994

y: -2.143

z: -0.365

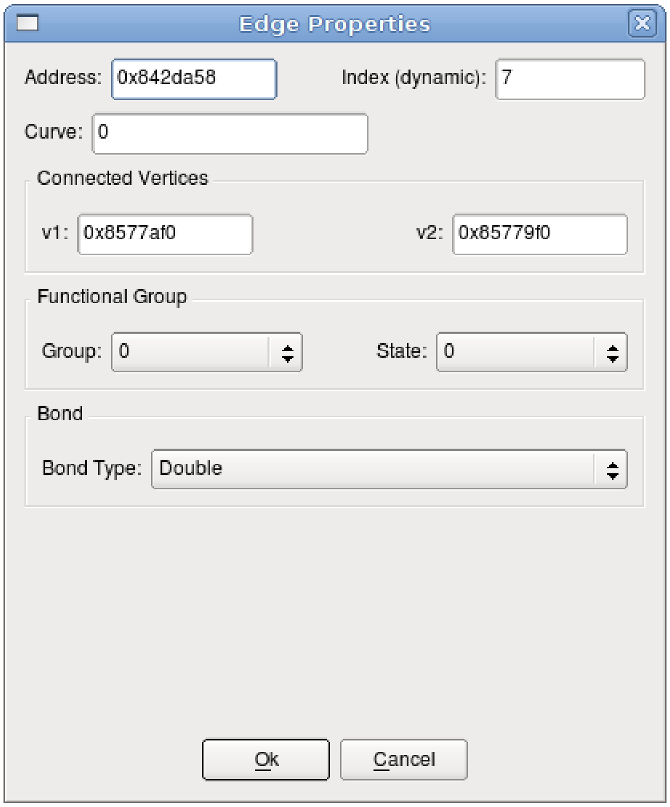
Atom

Atomic Number: 35

Atom: Br

Charge: 0

Figure A.6: Atom Properties Form



Edge Properties

Address: 0x842da58

Index (dynamic): 7

Curve: 0

Connected Vertices

v1: 0x8577af0

v2: 0x85779f0

Functional Group

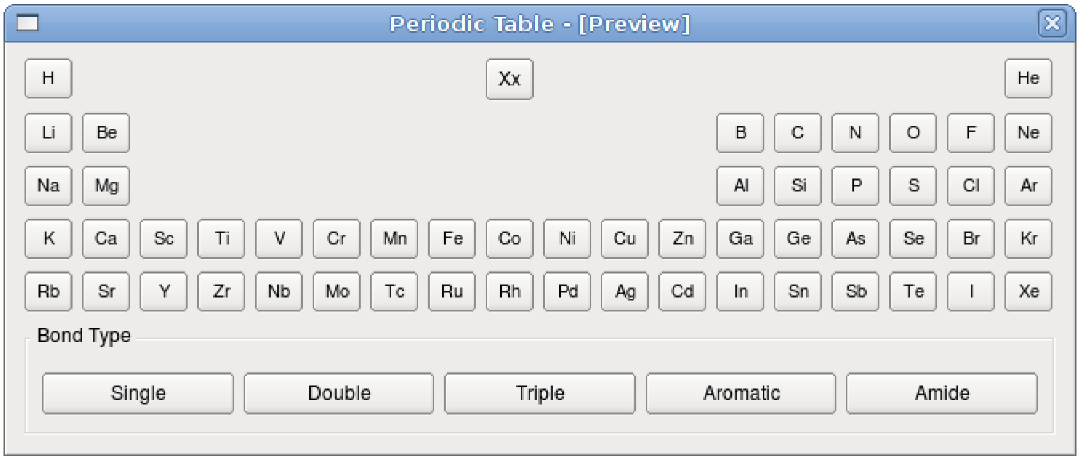
Group: 0

State: 0

Bond

Bond Type: Double

Figure A.7: Bond Properties Form



Periodic Table - [Preview]

Figure A.8: Periodic Table Dialog Box

A.4 Adding and Editing Polymers

The graph editor and molecule selector are generic and used for both polymers and monomers. This section describes only the portions unique to polymers. Section A.3 should be read first. Monomer structures are needed to build a polymer, so all required monomer structures should be stored before adding a polymer.

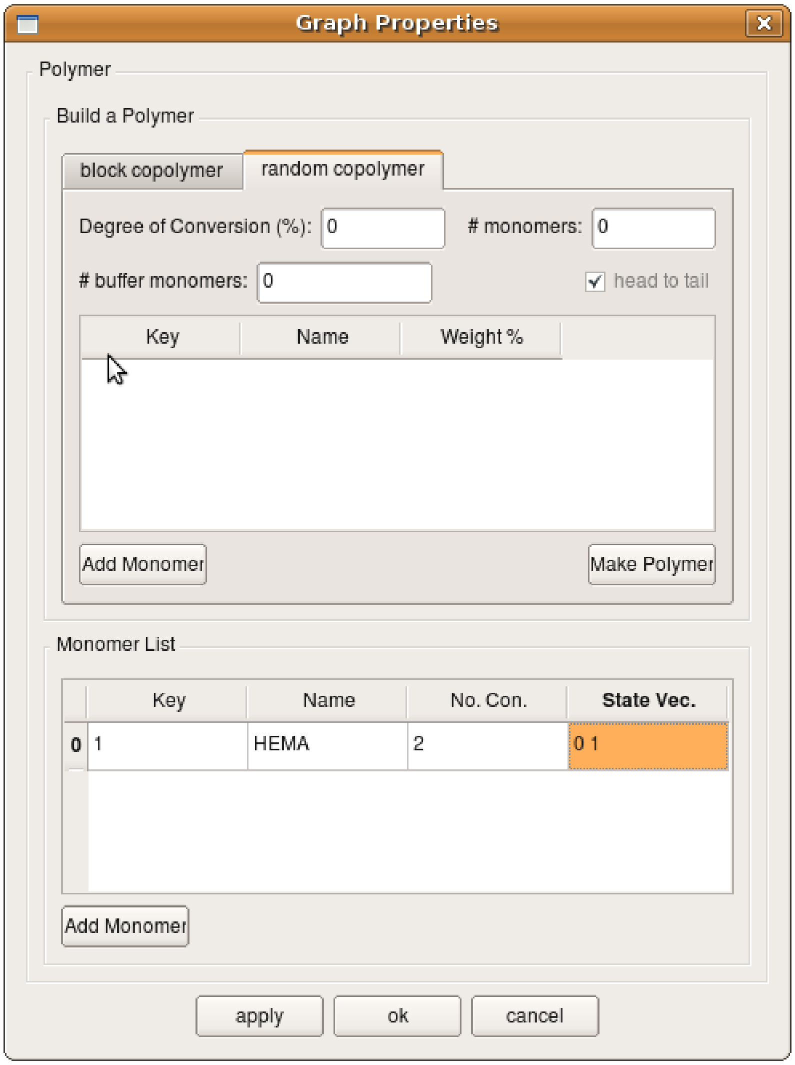
There are two ways to create a polymer: manually or automatically. First, manual creation will be discussed.

A.4.1 Manual Polymer Entry

The first step in entering a polymer structure is to select a list of monomers to use. To do this, select the the graph tab of the graph editor toolbox. Then select the graph operations button. A window like the one in Figure A.9 will appear. In the monomer list frame, select the add monomer button to add a monomer. This will allow selection from monomers in the database. The last step is to type a state vector into the newly added monomer. The state vector represents the state of each functional group.

Once all of the monomers have been added, the polymer can be drawn just like a monomer. The vertices and bonds must then be edited. Figure A.10 shows the vertex editing dialog. The monomer type and whether a vertex is in the core or buffer, can be set in this dialog.

Figure A.11 shows the bond editing dialog, which can be used to specify which connector in the monomers the bond attaches to.



Degree of Conversion (%): 0

monomers: 0

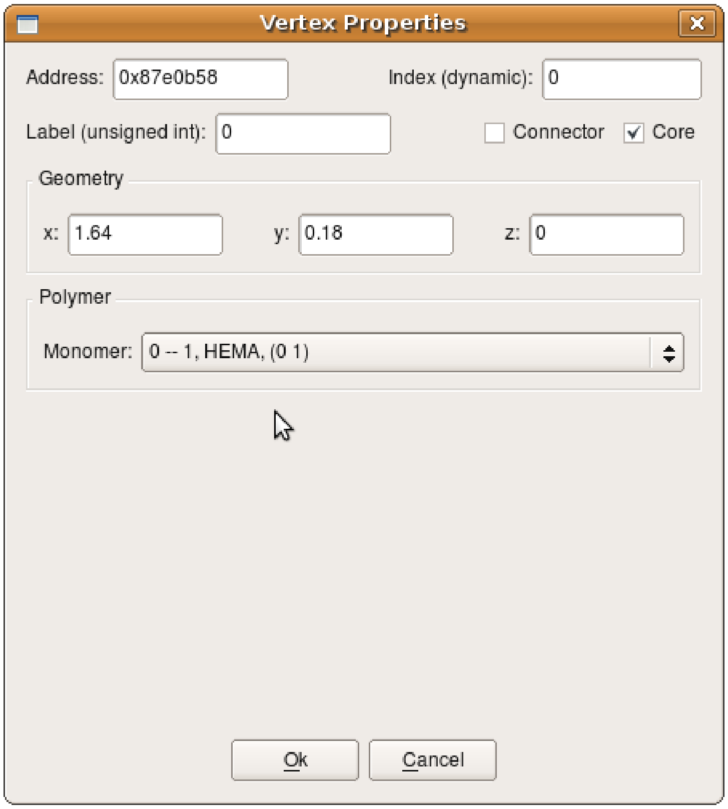
buffer monomers: 0

☑ head to tail

Key	Name	Weight %

	Key	Name	No. Con.	State Vec.
0	1	HEMA	2	0 1

Figure A.9: Including Monomers



Address: 0x87e0b58

Index (dynamic): 0

Label (unsigned int): 0

□ Connector ☑ Core

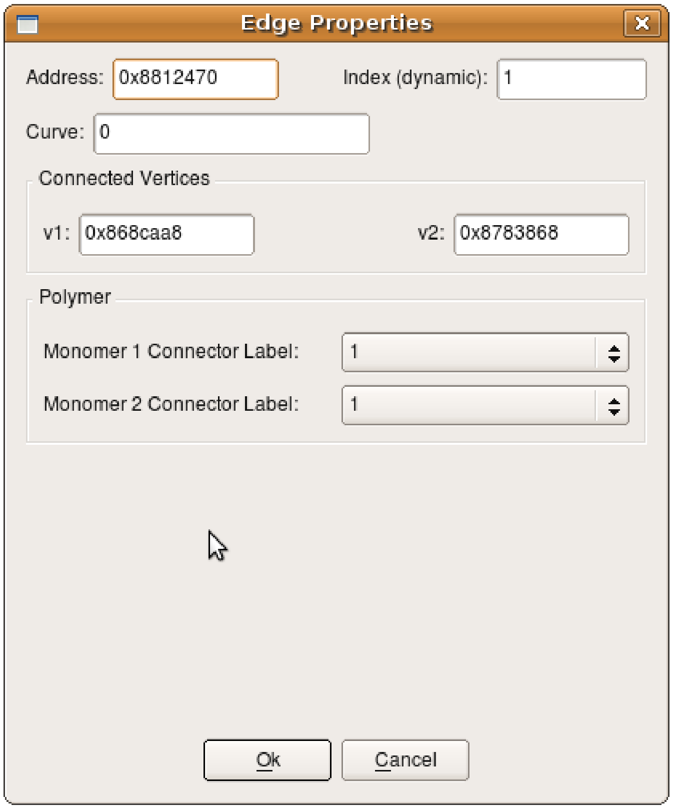
Geometry

x: 1.64 y: 0.18 z: 0

Polymer

Monomer: 0 -- 1, HEMA, (0 1)

Figure A.10: Editing a Vertex



Address: 0x8812470

Index (dynamic): 1

Curve: 0

Connected Vertices

v1: 0x868caa8

v2: 0x8783868

Polymer

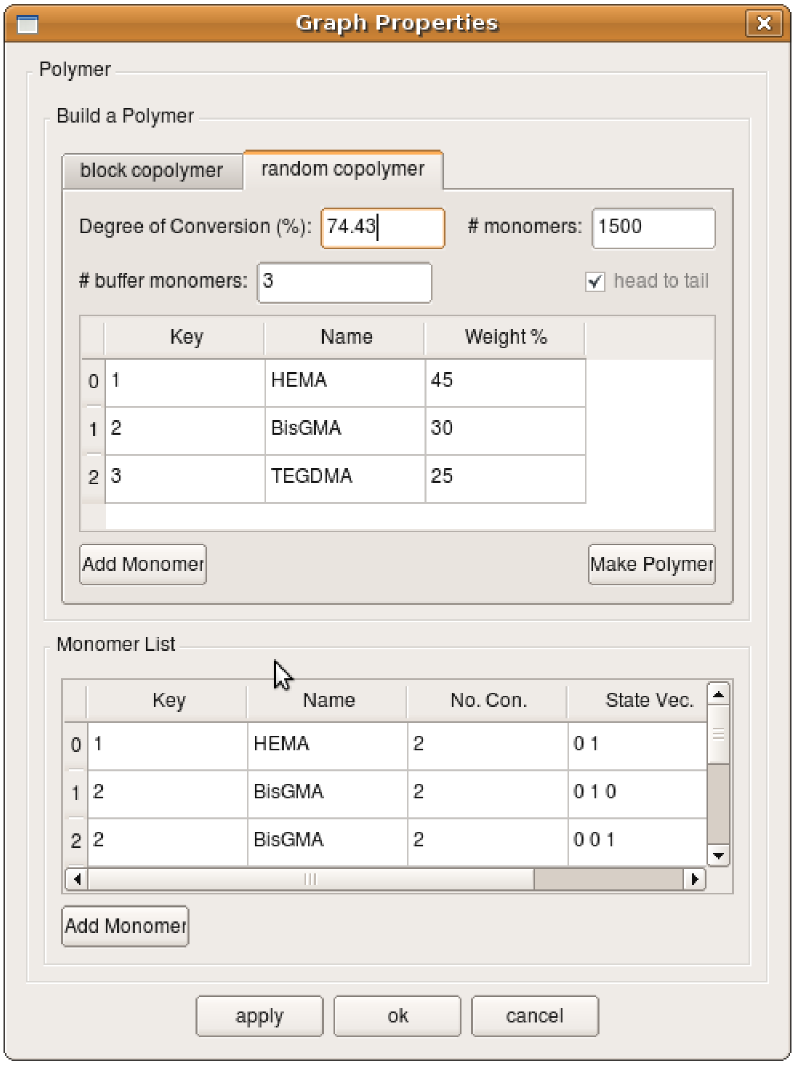
Monomer 1 Connector Label: 1

Monomer 2 Connector Label: 1

Figure A.11: Editing an Edge

A.4.2 Automatic Polymer Entry

For head-to-tail vinyl polymers, there is an automatic creation mechanism. Figure A.12 shows an example form.



Degree of Conversion (%): 74.43   # monomers: 1500

buffer monomers: 3   ☑ head to tail

	Key	Name	Weight %
0	1	HEMA	45
1	2	BisGMA	30
2	3	TEGDMA	25

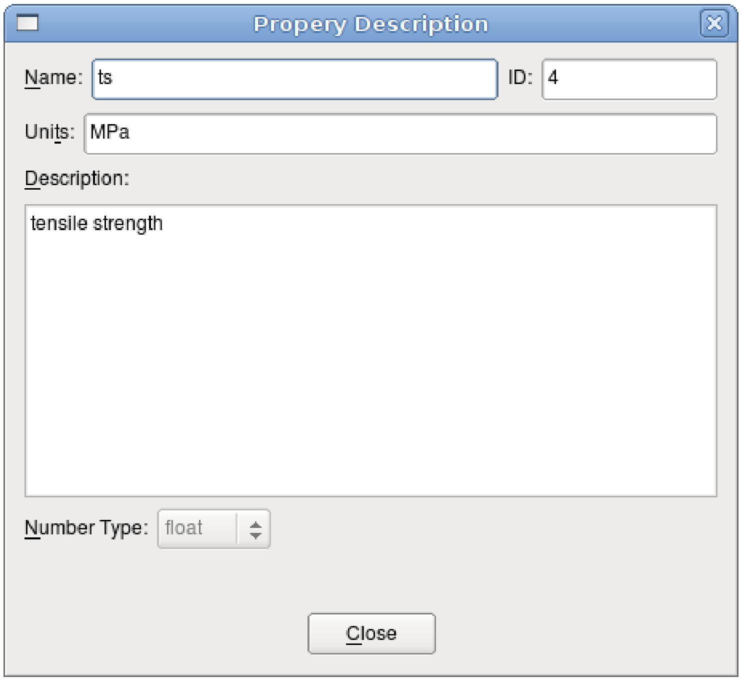
	Key	Name	No. Con.	State Vec.
0	1	HEMA	2	0 1
1	2	BisGMA	2	0 1 0
2	2	BisGMA	2	0 0 1

Figure A.12: Automatic Build Form

The monomers must again be selected. This time, use the add monomer button in the build polymer frame. You can specify a weight percent for each monomer in the resin. The number of monomers box is used to specify the number of monomers in the core of the polymer sample. The number of buffer monomers box specifies the number of monomers to be between the core and each cut chain. The degree of conversion box specifies the degree of conversion in percent. Once all the information is filled in, the make polymer button will automatically fill out the monomer list, and build a random polymer tree.

A.5 Adding and Editing Property Descriptions

Molecule properties can also be stored in the database. Property descriptions can be added and edited from the property menu. The actually property values can be edited in the spreadsheet described in the next section. A simple form is used to edit property descriptions (Figure A.13).


Propery Description

Name: ts

ID: 4

Units: MPa

Description:

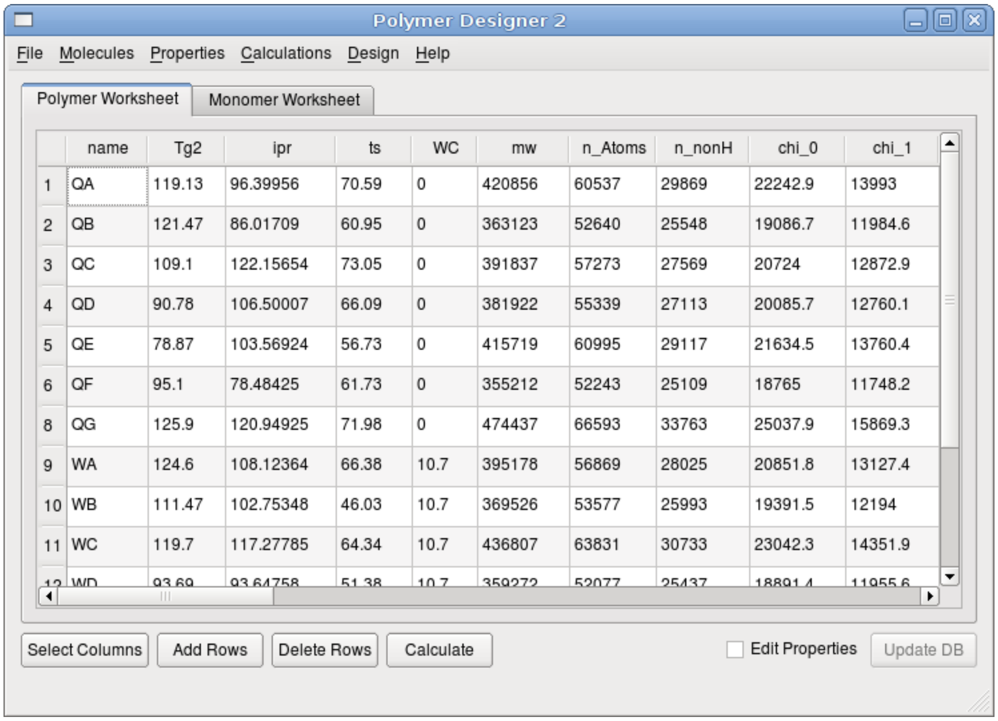
tensile strength

Number Type: float

Figure A.13: Property Description Editor

A.6 The Property/Descriptor Spreadsheet

The properly/descriptor spreadsheet allows structural descriptors to be calculated and property data to be edited. The spreadsheet can be exported as a tab-separated text file, so it can be imported into a spreadsheet or statistics program. Figure A.14 shows an example spreadsheet.



	name	Tg2	ipr	ts	WC	mw	n_Atoms	n_nonH	chi_0	chi_1
1	QA	119.13	96.39956	70.59	0	420856	60537	29869	22242.9	13993
2	QB	121.47	86.01709	60.95	0	363123	52640	25548	19086.7	11984.6
3	QC	109.1	122.15654	73.05	0	391837	57273	27569	20724	12872.9
4	QD	90.78	106.50007	66.09	0	381922	55339	27113	20085.7	12760.1
5	QE	78.87	103.56924	56.73	0	415719	60995	29117	21634.5	13760.4
6	QF	95.1	78.48425	61.73	0	355212	52243	25109	18765	11748.2
8	QG	125.9	120.94925	71.98	0	474437	66593	33763	25037.9	15869.3
9	WA	124.6	108.12364	66.38	10.7	395178	56869	28025	20851.8	13127.4
10	WB	111.47	102.75348	46.03	10.7	369526	53577	25993	19391.5	12194
11	WC	119.7	117.27785	64.34	10.7	436807	63831	30733	23042.3	14351.9
12	WD	93.69	93.64758	51.38	10.7	359272	52077	25437	18891.4	11955.6

Figure A.14: PD Spreadsheet

The spreadsheet works the same way with monomers and polymers, so the description here applies to both. First, select either the monomer or polymer tab in the main PD window. Then click the “Select Columns” button; this will bring up a window that allows data, properties, and descriptors to be selected, and displayed as columns on the spreadsheet.

Next click on the “Add Rows” button; this allows selection of molecules to add to the spreadsheet. Selection of multiple molecules is allowed using the shift and ctrl keys. Rows can be removed from the sheet by selecting the row and clicking the “Delete Rows” button.

To fill out the spreadsheet, click the “Calculate” button. The data and properties will be filled in from the database, and the structural descriptors will be calculated. If information for any cell is missing “NA” will be displayed in the cell.

To input or edit property vales, check the “Edit Properties” box. Edit the property values by selecting a property cell and typing in the new value. Non-property cells cannot be edited. Once the properties are edited edited as desired, click the “Update DB” button.

To export the spreadsheet select “Export Spreadsheet” from the file menu.

A.7 Molecular Design

PD contains functions for optimization. For the polymer design project, the molecular design routine was started from the algorithm test section of the graph editor and results were output to the terminal. The monomer structures were saved as xml files.

Appendix B

Group Library

This appendix provides a list of chemical groups available in the group library. The group library can be expanded, but the existing groups should not be changed, because some methods depend on their unique identifiers. The group library has two main purposes: identifying specific chemical structures (used in group contribution calculations) and building chemical structures (used in molecular structure optimization). In the following figures, each group is shown next to its unique integer identifier (UID). The atoms labeled with numbers are dummy atoms that represent connections to other groups, and the numbers are connection labels. The gray number next to each atom is the atom UID, which can be used if a method calls for information about specific atoms in a group. Some groups numbers are currently empty, and can be filled later.

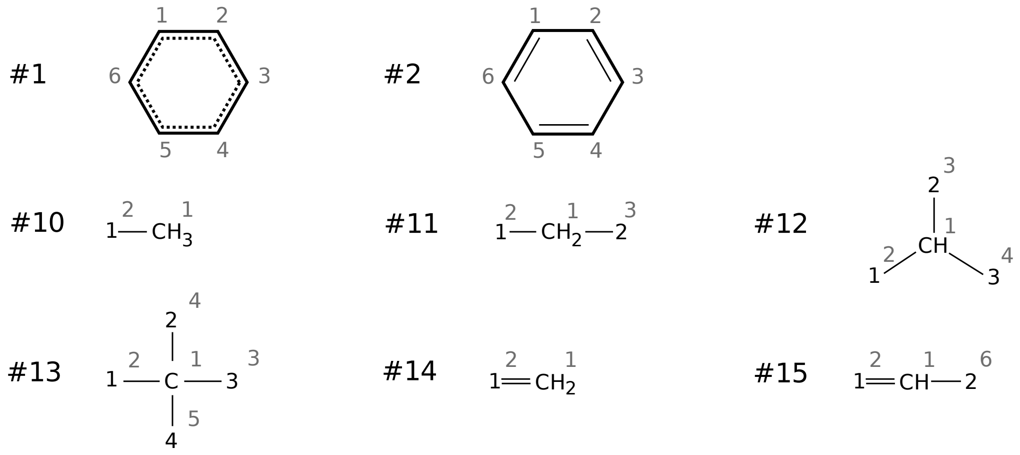


Figure B.1: Groups 0 to 15

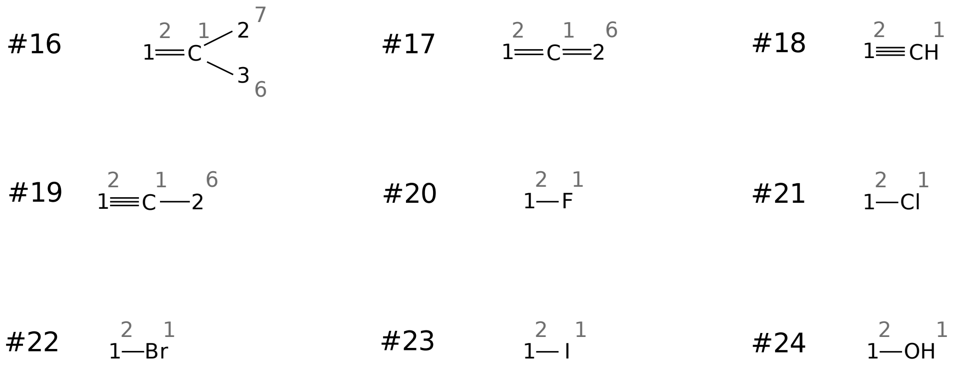


Figure B.2: Groups 16 to 24

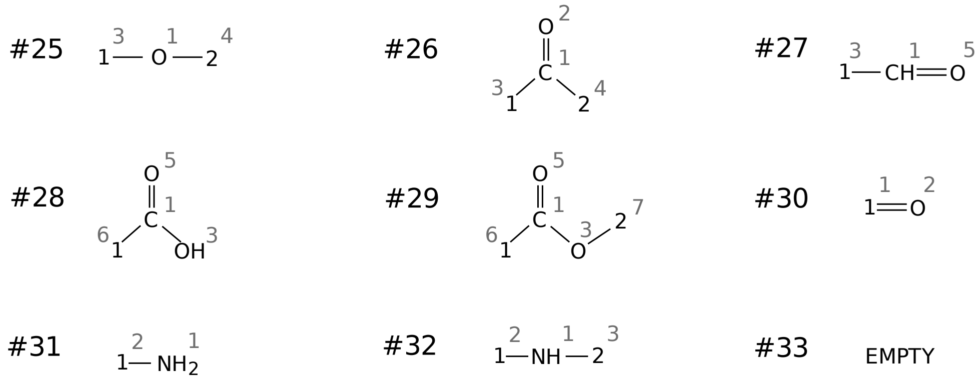


Figure B.3: Groups 25 to 33

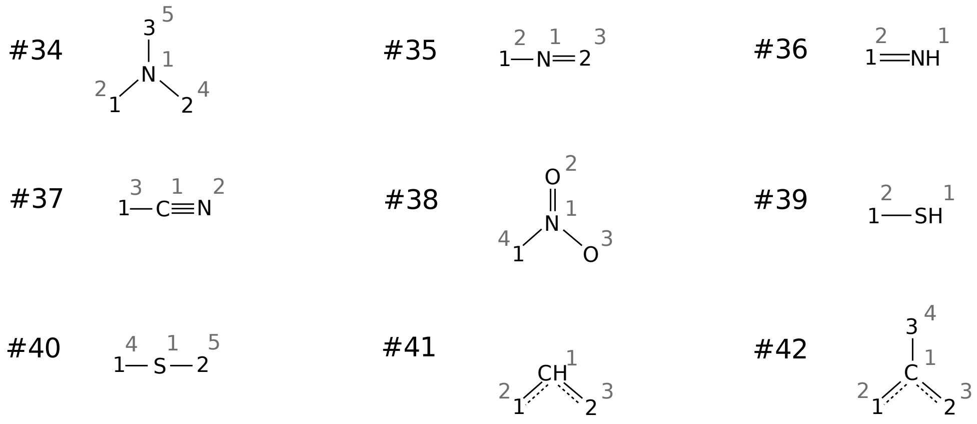


Figure B.4: Groups 34 to 42

Appendix C

Class Documentation

This appendix provides the source code of the Tabu search implementation used in the dental polymer example, and an explanation, both in text and in augmented comments. PD is written in C++. The Tabu search example uses functions of several important classes. A complete function listings can be obtained from the class header files.

In this implementation, a monomer is built from groups stored in the monomer database. The groups are assembled into a monomer using a polymer graph. The polymer graph here is used to store a monomer, since the way groups are assembled to form a monomer is similar to the way monomers assembled to form a polymer.

C.1 Tabu Search Implementation

The header file for the Tabu search implementation, `dp_tabu_01.hpp` is listed below. The functions and variables will be described in detail afterward. The header file is provided to give a complete listing of the Tabu search implementation.

```
#ifndef DP_TABU_01_HPP
```

```
#define DP_TABU_01_HPP

#include<list>
#include<vector>
#include<iostream>
#include"polymer.hpp"
#include"pd2_tabuSearch.hpp"

class dp_tabu_01{
public:
  struct sol_data_t{ //a data structure for problem solutions
    polymer mol_gen; //the monomer structure as groups
    molecule mol;    //the monomer structure as atoms
    double ts;       //estimated tensile strength
    double DC;       //estimated degree of conversion
    double Tg2;      //estimated glass transition temperature
    double ipr;      //estimated initial reaction rate
    double CD;       //crosslink density
    double CD_100;   //crosslink density at 100% DC
    double mw;       //monomer molecular weight
  };
private:
  pd2_tabuSearch ts;
  std::list<pd2_tabuSearch::solution_t*> neighbor_list;
  std::list<pd2_tabuSearch::solution_t*> tabu_list;
  std::vector<pd2_tabuSearch::solution_t> tabu_vec;
  std::vector<pd2_tabuSearch::solution_t> neighbor_vec;
  std::vector<sol_data_t> tabuData_vec;
  std::vector<sol_data_t> neighborData_vec;
  std::vector<sol_data_t> tabu_sol_data;
  std::vector<sol_data_t> neighbor_sol_data;
  pd2_tabuSearch::solution_t cur_sol;
  pd2_tabuSearch::solution_t init_sol;
  pd2_tabuSearch::solution_t best_sol;
  sol_data_t cur_sol_data;
  sol_data_t best_sol_data;
  sol_data_t init_sol_data;
  static void printSol(pd2_tabuSearch::solution_t *sol);
  static void calcObj(pd2_tabuSearch::solution_t *sol);
  static bool cmpTabu(pd2_tabuSearch::solution_t *solA,
                      pd2_tabuSearch::solution_t *solB);
  static void genNeigh(pd2_tabuSearch::solution_t *sol,
                       pd2_tabuSearch::solution_t *newSol);
  static void assignSol(pd2_tabuSearch::solution_t *from_sol,
                        pd2_tabuSearch::solution_t *to_sol);
  unsigned int nni_limit;        //number of non-improving iterations
  static double tabu_xi_close;  //scaled so 0 is no diff
                                //1 is the range of exp data
  static double tabu_mw_close;  //actual mw diff
public:
  dp_tabu_01();  //constructor
  void search(); //start tabu search
  double initialObjective();
  void setNonImproveLimit( unsigned int );
  void setTabuListSize( unsigned int );
  void setNeighborListSize( unsigned int );
  void setInitialSolution( polymer *p );
  polymer *getBestGroups();
  molecule *getBestMolecule();
};

#endif
```

The next part will describe the source code for the Tabu search implantation a small piece at a time. This example will illustrate the most commonly used and important functions in PD. It provides a good example for setting up a Tabu search algorithm for a new system.

The following code includes the header file and defines some variables, which are used in functions contained in this file. The variables `tabu_xi_close` and `tabu_mw_close` are used to set how close the connectivity indices and molecular weight of a monomer may be to a solution on the Tabu list without classifying them as tabu.

```
#include"dp_tabu_01.hpp"
#include<cmath>

double dp_tabu_01::tabu_xi_close;
double dp_tabu_01::tabu_mw_close;
```

The following function is the constructor. It sets up an instance of the Tabu search implementation. The variable `ts` is an instance of the generic Tabu search available in PD. Comments in the code provide further description.

```
dp_tabu_01::dp_tabu_01(){
  //set the list of tabu solutions
  ts.setTabuList( &tabu_list );
  //set the list of neighbor solutions
  ts.setNeighborList( &neighbor_list );
  //set the non-improving iteration limit to 0
  //this will be reset later before running the
  //algorithm
  ts.setNNILimit( 0 );
  //set the variable to store the best solution
  ts.setBestSol( &best_sol );
  //set the variable to store the initial solution
  ts.setInitSol( &init_sol );
  //set the variable to store the current solution
  ts.setCurrSol( &cur_sol );
  //set the function to compare a solution to one
  //on the tabu list, this takes a function pointer
  //argument, the function is defined below
  ts.setTabuCmp( cmpTabu );
  //set the function to calculate the objective
  //function, this takes a function pointer
  //argument, the function is defined below
  ts.setCalcObj( calcObj );
  //set the function to generate neighbors,
  //the function is defined below
  ts.setGenNeigh( genNeigh );
  //set the function to copy a solution,
  //the function is defined below
  ts.setAssign( assignSol );
  //set the function to print a solution,
  //the function is defined below
  ts.setPrintSol( printSol );
  //set the place to store data for some solutions
  //the sol member of the pd2_tabuSearch::solution_t
  //data structure is a void pointer
  cur_sol.sol = (void*)(&cur_sol_data);
  best_sol.sol = (void*)(&best_sol_data);
  init_sol.sol = (void*)(&init_sol_data);
}
```

The following function initiates the Tabu search algorithm by simply calling the search member function of `ts`.

```
void dp_tabu_01::search(){
  ts.search();
}
```

The following function calculates and returns the objective function value of the initial solution. It will be used later.

```
double dp_tabu_01::initialObjective(){
  return ts.initialObjective();
}
```

The following function is used to set the size of the the Tabu list and initialize it.

```
void dp_tabu_01::setTabuListSize( unsigned int s ){
  unsigned int i;
  tabu_list.clear();  //clears the tabu list
  tabu_vec.clear();   //clears the vector are solutions
  tabuData_vec.clear(); //clear the vector were solution data is stored
  tabuData_vec.resize(s);
  tabu_vec.resize(s);
  for(i=0; i<s; ++i){
    // assign solution pointers to the list
    tabu_list.push_back( &tabu_vec[i] );
    tabu_vec[i].sol = (void*)(&tabuData_vec[i]);
  }
}
```

The following function sets up the neighbor list. This list works the same way as the tabu list above.

```
void dp_tabu_01::setNeighborListSize( unsigned int s ){
  unsigned int i;
  neighbor_list.clear();
  neighbor_vec.clear();
  neighborData_vec.clear();
  neighborData_vec.resize(s);
  neighbor_vec.resize(s);
  for(i=0; i<s; ++i){
    neighbor_list.push_back( &neighbor_vec[i] );
    neighbor_vec[i].sol = (void*)(&neighborData_vec[i]);
  }
}
```

The function below is used to set the initial solution. A polymer graph representing the arrangement of groups to form a monomer is given as the argument.

```
void dp_tabu_01::setInitialSolution( polymer *p ){
  //set the polymer of the initial solution
  init_sol_data.mol_gen = *p;
  //make a molecule out of the polymer
  init_sol_data.mol_gen.constructMolecule();
  //get the monomer structure from the polymer
  init_sol_data.mol = *( init_sol_data.mol_gen.getConstructedMolecule() );
  //calculate molecular weight of the monomer an determine
  //atom hybridization
  init_sol_data.mol.calculateMolecularWeight();
  //calculate connectivity indices up to third order
  init_sol_data.mol.calculateBiceranoConnectivity(3);
}
```

The following two functions return the best solution structure. The first returns a polymer graph using the groups and the second returns a molecular graph.

```
polymer *dp_tabu_01::getBestGroups(){
  return &best_sol_data.mol_gen;
}

molecule *dp_tabu_01::getBestMolecule(){
  return &best_sol_data.mol;
}
```

The following function is used to print information about a solution, and writes the solution structure to an XML file that can be read by PD.

```
void dp_tabu_01::printSol(pd2_tabuSearch::solution_t *sol){
  std::cout << "Degree of conversion: " <<
    ((sol_data_t*)(sol->sol))->DC << std::endl;
  std::cout << "Crosslink density: " <<
    ((sol_data_t*)(sol->sol))->CD << std::endl;
  std::cout << "Crosslink density (DC = 100%): " <<
    ((sol_data_t*)(sol->sol))->CD_100 << std::endl;
  std::cout << "Tensile strength: " <<
    ((sol_data_t*)(sol->sol))->ts << std::endl;
  std::cout << "Glass transition temperature: " <<
    ((sol_data_t*)(sol->sol))->Tg2 << std::endl;
  std::cout << "Initial polymerization rate: " <<
    ((sol_data_t*)(sol->sol))->ipr << std::endl;
  std::cout << "Molecular weight: " <<
    ((sol_data_t*)(sol->sol))->mw << std::endl;
  //write the molecule to an xml file
  ((sol_data_t*)(sol->sol))->
    mol_gen.writeConstructedMoleculeXMLFile("tabuSol.xml");
}
```

The following function is used to calculate the objective function of a solution and estimate its properties.

```
void dp_tabu_01::calcObj(pd2_tabuSearch::solution_t *sol){
  unsigned int i, j, h, m, s, mask;
  double DC, ts, Tg2, ipr;
  double WC = 0, CD, CD_100;
  double xi_0, xiV_1, xiV_2, xiV_3;
  double mw_hema, mw_bisgma, mw_mol;
  double wt_hema, wt_bisgma, wt_mol;
  double molfrac_hema, molfrac_bisgma, molfrac_mol;
  int nvinyl_hema, nvinyl_bisgma, nvinyl_mol;
  std::vector< std::vector< std::vector< unsigned int > > > stateVecs;
  std::vector< int > nvinyl;
  double p, pnot, f, molfrac_in;
  std::vector< double > molfrac;
  std::vector< std::vector< double > > molFracState;
  unsigned int ns;

  //the indices for the state vecs are i = monomer, j = state vector,
  //h = f. group;
  //I need state vectors to do the prob calcs for CD

  //Get descriptors used in QSPRs
  xi_0 = ((sol_data_t*)(sol->sol))->mol.getBiceranoXi(0);
  xiV_1 = ((sol_data_t*)(sol->sol))->mol.getBiceranoXiV(1);
  xiV_2 = ((sol_data_t*)(sol->sol))->mol.getBiceranoXiV(2);
  xiV_3 = ((sol_data_t*)(sol->sol))->mol.getBiceranoXiV(3);
  mw_mol = ((sol_data_t*)(sol->sol))->mol.getMolecularWeight();
  DC = -66.064264794 + 1.565420561*WC + 479.870235339*xiV_1 - 120.447952881*xiV_3;

  mw_hema = 130.143;
  mw_bisgma = 512.599;
  wt_hema = 45.0;
  wt_bisgma = 30.0;
  wt_mol = 25.0;
  nvinyl_hema = 1;
  nvinyl_bisgma = 2;
  nvinyl_mol = 0;

  //this loop counts the number of vinyl groups in a monomer
  for(i=0; i < ((sol_data_t*)(sol->sol))->mol_gen.numVertices(); ++i){
    m = ((sol_data_t*)(sol->sol))->mol_gen.getMonomer(i);
    for(j=0; j < ((sol_data_t*)(sol->sol))->mol_gen.CMT1_termGroups.size(); ++j){
      if(m == ((sol_data_t*)(sol->sol))->mol_gen.CMT1_termGroups[j]){
        ++nvinyl_mol;
        break;
      }
    }//end j loop
  }//end i loop
```

```
//calculate mole fraction of each monomer
molfrac_hema = (wt_hema/mw_hema)/
               (wt_hema/mw_hema + wt_bisgma/mw_bisgma + wt_mol/mw_mol);
molfrac_bisgma = (wt_bisgma/mw_bisgma)/
                 (wt_hema/mw_hema + wt_bisgma/mw_bisgma + wt_mol/mw_mol);
molfrac_mol = (wt_mol/mw_mol)/
              (wt_hema/mw_hema + wt_bisgma/mw_bisgma + wt_mol/mw_mol);

CD_100 = (nvinyl_hema-1)*molfrac_hema +
         (nvinyl_bisgma-1)*molfrac_bisgma +
         (nvinyl_mol-1)*molfrac_mol;

//make the state vectors
// i = 0 is hema, i = 1 is bisGMA, i = 2 is third monomer
stateVecs.resize(3); // there are three monomers
stateVecs[0].resize(2); //hema has two states
stateVecs[1].resize(4);

molFracState.resize(3);
molFracState[0].resize(2);
molFracState[1].resize(4);

ns = 1;
ns <<= nvinyl_mol; // binary shift to calculate number of states in monomer
stateVecs[2].resize(ns);
molFracState[2].resize(ns);
//put in state vecs
nvinyl.resize(3);
nvinyl[0] = nvinyl_hema;
nvinyl[1] = nvinyl_bisgma;
nvinyl[2] = nvinyl_mol;
for(i=0; i<stateVecs.size(); ++i){
  // loop through monomers
  s = 0; //binary representation of a state
  //the first state vector is unreacted
  for(j=0; j<stateVecs[i].size(); ++j){
    stateVecs[i][j].resize( nvinyl[i] + 1 );
    // loop through states
    stateVecs[i][j][0] = 0;
    mask = 1;
    for(h=1; h<(unsigned int)nvinyl[i]+1; ++h){
      stateVecs[i][j][h] = (unsigned int)( (mask & s) == mask );
      mask <<= 1; //shift the 1 in mask over 1
      //std::cout << stateVecs[i][j][h];
    }
    //std::cout << std::endl;
    ++s;
  }// end j loop
}// end i loop
```

```
//probability a functional group reacts
p = DC/100.0;
//probability a functional group does not react
pnot = 1.0 - p;

// calculate mole fraction of each state
molfrac.resize(3);
molfrac[0] = molfrac_hema;
molfrac[1] = molfrac_bisgma;
molfrac[2] = molfrac_mol;
for(i=0; i<3; ++i){
  for(j=0; j<stateVecs[i].size(); ++j){
    f = 1;
    for(h=1; h<stateVecs[i][j].size(); ++h){
      if(stateVecs[i][j][h]) f *= p;
      else f *= pnot;
    }//end h loop
    molFracState[i][j] = f*molfrac[i];
  }//end j loop
}//end i loop

//Now I have the mole fraction of each state
//the unreacted states are not a part of the polymer network
//so I need new mole fractions of things that react
//remember the first state vector is unreacted
molfrac_in = 0;
for(i=0; i<3; ++i){
  for(j=1; j<stateVecs[i].size(); ++j){
    molfrac_in += molFracState[i][j];
  }
}

// OK so I know the mole fraction that is in the network
//now to calculate crosslink density
CD = 0;
for(i=0; i<3; ++i){
  for(j=1; j<stateVecs[i].size(); ++j){
    //count reacted groups
    ns = 0;
    for(h=1; h<stateVecs[i][j].size(); ++h){
      ns += stateVecs[i][j][h];
    }
    // now ns is the number of groups that react
    ns -= 1;
    // now ns is the number of cross links formed
    CD += ns*molFracState[i][j]/molfrac_in;
  }
}

//estimate the properties from QSPRs
ts = 15.42147728 - 308.90792806*CD +
      194.83199066*CD_100 + 242.09566112*xiV_2;
  Tg2 = 674.1494905637 + 0.6108983308*DC -
        2190.4691820556*xiV_1 + 401.6520626521*xiV_2;
  ipr = -528.94709587150 - 151.20634606711*CD_100 +
        831.35424142271*xi_0 + 0.09496637828*mw_mol;

  //calculate the objective function value
  sol->obj = ((120.0 - Tg2)*(120.0 - Tg2))/(120.0*120.0) +
              (ts-80.0)*(ts-80.0)/(80.0*80.0) +
              (ipr-130.0)*(ipr-130.0)/(130.0*130.0) +
              (DC-100.0)*(DC-100.0)/(100.0*100.0);

  //store the properties in the solution data structure
  ((sol_data_t*)(sol->sol))->ts = ts;
  ((sol_data_t*)(sol->sol))->Tg2 = Tg2;
  ((sol_data_t*)(sol->sol))->ipr = ipr;
  ((sol_data_t*)(sol->sol))->DC = DC;
  ((sol_data_t*)(sol->sol))->CD = CD;
  ((sol_data_t*)(sol->sol))->CD_100 = CD_100;
  ((sol_data_t*)(sol->sol))->mw = mw_mol;
}
```

The following function is used to compare two solutions to determine how close they are structurally. This is used to compare solutions to the Tabu list, and works by comparing connectivity indices and molecular weight of the two monomers. If the difference in any parameter exceeds a threshold, the structures are considered different.

```
bool dp_tabu_01::cmpTabu(pd2_tabuSearch::solution_t
                         *solA, pd2_tabuSearch::solution_t *solB){
  double xi_0_range;
  double xi_1_range;
  double xi_2_range;
  double xi_3_range;
  double xiV_0_range;
  double xiV_1_range;
  double xiV_2_range;
  double xiV_3_range;
  double A_xi_0, A_xi_1, A_xi_2, A_xi_3,
         A_xiV_0, A_xiV_1, A_xiV_2, A_xiV_3, A_mw;
  double B_xi_0, B_xi_1, B_xi_2, B_xi_3,
         B_xiV_0, B_xiV_1, B_xiV_2, B_xiV_3, B_mw;
  bool tabu;

  //these are the ranges of the connectivity
  //indices in the experimental data
  xi_0_range = 0.04534708;
  xi_1_range = 0.02072973;
  xi_2_range = 0.06434273;
  xi_3_range = 0.07711074;
  xiV_0_range = 0.06024335;
  xiV_1_range = 0.02645703;
  xiV_2_range = 0.05916363;
  xiV_3_range = 0.05039862;

  //these are the threshold values
  tabu_mw_close = 25;
  tabu_xi_close = 0.15;

  A_xi_0 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXi(0);
  A_xi_1 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXi(1);
  A_xi_2 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXi(2);
  A_xi_3 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXi(3);
  A_xiV_0 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXiV(0);
  A_xiV_1 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXiV(1);
  A_xiV_2 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXiV(2);
  A_xiV_3 = ((sol_data_t*)(solA->sol))->mol.getBiceranoXiV(3);
  A_mw = ((sol_data_t*)(solA->sol))->mol.getMolecularWeight();
  B_xi_0 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXi(0);
  B_xi_1 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXi(1);
  B_xi_2 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXi(2);
  B_xi_3 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXi(3);
  B_xiV_0 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXiV(0);
  B_xiV_1 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXiV(1);
  B_xiV_2 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXiV(2);
  B_xiV_3 = ((sol_data_t*)(solB->sol))->mol.getBiceranoXiV(3);
  B_mw = ((sol_data_t*)(solB->sol))->mol.getMolecularWeight();

  tabu = 1;
  if( (A_xi_0 - B_xi_0)*(A_xi_0 - B_xi_0)/(xi_0_range*xi_0_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xi_1 - B_xi_1)*(A_xi_1 - B_xi_1)/(xi_1_range*xi_1_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xi_2 - B_xi_2)*(A_xi_2 - B_xi_2)/(xi_2_range*xi_2_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xi_3 - B_xi_3)*(A_xi_3 - B_xi_3)/(xi_3_range*xi_3_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xiV_0 - B_xiV_0)*(A_xiV_0 - B_xiV_0)/(xiV_0_range*xiV_0_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xiV_1 - B_xiV_1)*(A_xiV_1 - B_xiV_1)/(xiV_1_range*xiV_1_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xiV_2 - B_xiV_2)*(A_xiV_2 - B_xiV_2)/(xiV_2_range*xiV_2_range) >
     tabu_xi_close ) tabu = 0;
  if( (A_xiV_3 - B_xiV_3)*(A_xiV_3 - B_xiV_3)/(xiV_3_range*xiV_3_range) >
     tabu_xi_close ) tabu = 0;
  if( fabs(A_mw - B_mw) > tabu_mw_close ) tabu = 0;
  return tabu;
}
```

The following function generates a neighbor of a solution.

```
void dp_tabu_01::genNeigh(pd2_tabuSearch::solution_t *sol,
                          pd2_tabuSearch::solution_t *newSol){
  // Generates neighbor by removing, inserting or replacing groups

  unsigned int i; // loop counter
  unsigned int op; //operation 0 = insert, 1 = delete
  unsigned int nOp; //number of operations
  unsigned int nOpMin, nOpMax; // range of nOp
  unsigned int maxVert;
  unsigned int nr, nrMax, nrMin; //number of replacements
  nOpMin = 1;
  nOpMax = 2;
  maxVert = 30;

  // select whether to insert or delete groups
  if( ((sol_data_t*)(newSol->sol))->mol_gen.numVertices() >= maxVert) op = 1;
  else if(((sol_data_t*)(newSol->sol))->mol_gen.numVertices() <= 3 ) op = 0;
  else op = pd2_tabuSearch::randomIntUniform(0,1);

  //select a number of delete or insert operations
  nOp = pd2_tabuSearch::randomIntUniform(0,nOpMax);

  //do insert or delete operations
  ((sol_data_t*)(newSol->sol))->mol_gen =
    ((sol_data_t*)(sol->sol))->mol_gen;
  for(i=0; i<nOp; ++i){
    if(op==0)
     ((sol_data_t*)(newSol->sol))->mol_gen.constructMoleculeType1_InsertGroup();
    else
     ((sol_data_t*)(newSol->sol))->mol_gen.constructMoleculeType1_DeleteGroup();
  }

  //select a number of replacements to do
  nrMax = nOpMax - nOp;
  if(nOp < nOpMin) nrMin = nOpMin - nOp;
  else nrMin = 0;
  if( nrMax == 0 ) nr = 0;
  else if( nrMax == nrMin) nr = nOpMax;
  else nr = pd2_tabuSearch::randomIntUniform(nrMin,nrMax);

  //do replacement operations
  for(i = 0; i<nr; ++i){
    ((sol_data_t*)(newSol->sol))->
      mol_gen.constructMoleculeType1_ReplaceGroup();
  }
  ((sol_data_t*)(newSol->sol))->mol_gen.constructMolecule();
  ((sol_data_t*)(newSol->sol))->mol = *( ((sol_data_t*)(newSol->sol))->
      mol_gen.getConstructedMolecule() );
  ((sol_data_t*)(newSol->sol))->mol.calculateMolecularWeight();
  ((sol_data_t*)(newSol->sol))->mol.calculateBiceranoConnectivity(3);
}
```

The following function is used to copy one solution to another.

```
void dp_tabu_01::assignSol(pd2_tabuSearch::solution_t *from_sol,
                           pd2_tabuSearch::solution_t *to_sol){
  ((sol_data_t*)(to_sol->sol))->mw = ((sol_data_t*)(from_sol->sol))->mw;
  ((sol_data_t*)(to_sol->sol))->CD = ((sol_data_t*)(from_sol->sol))->CD;
  ((sol_data_t*)(to_sol->sol))->CD_100 =
    ((sol_data_t*)(from_sol->sol))->CD_100;
  ((sol_data_t*)(to_sol->sol))->DC = ((sol_data_t*)(from_sol->sol))->DC;
  ((sol_data_t*)(to_sol->sol))->ts = ((sol_data_t*)(from_sol->sol))->ts;
  ((sol_data_t*)(to_sol->sol))->Tg2 = ((sol_data_t*)(from_sol->sol))->Tg2;
  ((sol_data_t*)(to_sol->sol))->ipr = ((sol_data_t*)(from_sol->sol))->ipr;
  ((sol_data_t*)(to_sol->sol))->mol = ((sol_data_t*)(from_sol->sol))->mol;
  ((sol_data_t*)(to_sol->sol))->mol_gen =
    ((sol_data_t*)(from_sol->sol))->mol_gen;
  to_sol->obj = from_sol->obj;
}
```

C.2 Using the Tabu Search Implementation

The following provides some code excerpts that show how the Tabu search implementation can be used. The groups are contained in the polymer graph. The initial polymer graph is created using the graphical editor. The groups are also drawn in the graphical editor.

```
dp_tabu_01 dpts;
//set the probabilities for selecting certain types of
//group when randomly constructing an initial molecule
//this is 30% terminal groups, 30% chain groups
//forty percent branching groups, and at most 2
//branching groups
g_poly->constructMoleculeType1_SetupGroups(30,30,40,2);
//sets the number of non-improving iterations
dpts.setNonImproveLimit( 200 );
//sets the size of the tabu list
dpts.setTabuListSize( 6 );
//sets the number of neighbors to generate
dpts.setNeighborListSize( 5 );
```

```
//sets the initial solution graph
dpts.setInitialSolution( g_poly );
//starts the search
dpts.search();
```

When the Tabu search algorithm completes, the result is printed to the terminal and the monomer structure is written to an XML file.