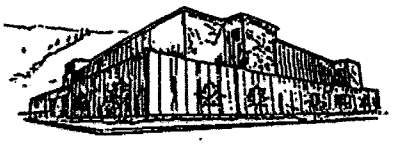
2004

A simple methodology for the production of three -dimensional models: Serotonin transporter as an example

Paul A. Wilson
The University of Montana

NOTE TO USERS



A Simple Methodology for the Production of Three-Dimensional Models: Serotonin Transporter as an Example

by
Paul A. Wilson
B.S. Chemistry, Central Washington University, 2000
B.S. Physics, Central Washington University, 2000
A.A. Clark College, 1996
A.A.S. Electronics Engineering Technology,
ITT Technical Institute, 1990

presented in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy Chemistry
The University of Montana
December 2004

Approved by:

John M. Gerdes
Committee Chair

Dean, Graduate School

12-21-04
Date

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Wilson, Paul A. Ph.D., December 2004 Chemistry

A Simple Methodology for the Production of Three-Dimensional Models: Serotonin Transporter as an Example

Committee Chair: John M. Gerdes

A conceptually simple and clearly rigorous paradigm for finding molecular similarity has been developed to elucidate three-dimensional (3-D) pharmacophore templates utilizing multi-molecule constructs without dependence upon a specific software package. A multi-molecular approach has been taken to limit induced model bias from any single molecule. The methodology involves the construction of a pharmacophore foundation from the low energy conformations identified for a select set of molecules exhibiting established potency and selectivity for the binding domain of interest. These low energy conformations are combinatorially compared to identify one set (1 set = 1 low energy conformation from each molecule) possessing the highest degree of similarity in 3-D space. This technique has been applied to produce a 3-D pharmacophore model of the serotonin selective reuptake inhibitor (SSRI) binding domain of the serotonin transporter (SERT). A set of four molecules possessing a high measure of SERT selectivity, potency and offering ample structural diversity was identified and select conformations (non-global minima) were attained. Similarity was based on distance space descriptions compared using a relative difference calculation. The relative difference equation compares the magnitudes of two measurements and produce a ratio which, in effect, describes their similarity. Data sets with measurements on both sides of zero may be overly, or not adequately, penalized with consideration to the actual magnitude between the measures. An algorithm for accomplishing a reasonable implementation of the relative difference equation when a known value does not exist, has been developed. A computational algorithm and programs have been developed that, in conjunction with novel modeling methodologies, have led to an unambiguous and descriptive 3D pharmacophore model of the SSRI binding domain at the SERT. The predictive quality of the model was demonstrated through application, by the design of a family of novel, highly potent SERT inhibitor ligands (Ki < 100 pM), exemplified by 2'-methyl-6-nitroquipazine.

Funded by tax payer dollars through grants from the NIH (NCRR COBRE P20-RR15583 and NS39814) and NSF (Montana EPSCoR, EPS-0091995).

Acknowledgments

John M. Gerdes Committee Chair: for years of guidance and attenuating my impulses to leave graduate school prematurely

My Graduate Committee: John M. Gerdes, Jennifer McNulty, Don Morton, J.B. Alexander Ross, Charles Thompson

Jennifer McNulty and Don Morton for offering interdisciplinary courses

Jennifer Parham for many excellent discussions of C pointers and parallel programming

Chris Odom for providing a great reference for the relative difference equation

JoAnn DeLuca for letting us use her SGI O2

Sharon Rosell - not only the best physics instructor ever, but a person who truly cares about you

Tyson Miller for letting me bend his ear

Bill Vetter for much advice and guidance

BMW Motorrad USA and my 1983 BMW R65 for providing an interesting, constructive, and stress relieving hobby

The Apple Student Developer email list

Gaspare Campari

and most importantly

Melodie Weller
and the Weller and Wilson families

Preface

Development of a pharmacophore model of the serotonin selective reuptake inhibitor (SSRI) binding domain at the serotonin transporter (SERT) came not from a personal interest in serotonin, but from an interest in UNIX. Instead of a traditional introduction covering the important role of SSRIs in psychiatric disorders (Vaswani), each section incorporates an introduction to the work contained within. However, at this point it may be enlightening to go though the nontraditional events that have led to the work covered in this dissertation.

The computational environment this project started in almost entirely consisted of Windows and VMS machines. However, the modeling project came with a UNIX machine. It seemed like a good trade to produce a model in exchange for the use of a SGI O2 running the IRIX flavor of UNIX. Then the situation changed and UNIX become readily available.

Four important events occurred around this time. Firstly, an interest in parallel computing was developed. Secondly, Mac OS X, a user centric operating system based upon UNIX was released. Thirdly, it became apparent scientific tools and parallel programs previously only available in the UNIX environment could now be ported to the more user friendly Mac OS. Tools which used to be limited to costly machines could now be made available on affordable machines, in essence bringing scientific research to the masses. Lastly, school is about the student studying what they are interested in. These four events led to the writing of a parallel scientific program which was presented at MacHack 2003 (appendix 4.4). This program used message passing interface (MPI) and ran on a cluster of Mac OS X machines.

What was learned from writing the parallel program and attending MacHack was used to rewrite and improve the program. The new program was a serial program, running only on one processor. One of the reasons for the new program being serial was the OS X machines previously used were no longer available for turning into a cluster. The serial program and pharmacophore modeling paradigm was presented at the 2003 annual meeting of the Society for Neuroscience.

The Society for Neuroscience (SFN) meetings are a good learning environment. The latest in scientific computing related to the neuroscience field is on display at these meetings. As well, these meetings provide a good opportunity to interact with other people who are coding algorithms. As a result, the work involving the modification of the relative difference equation, previously done on the back of a piece of paper, was formalized and presented at the 2004 annual SFN meeting.

In summary, the model was developed out of desire to use a UNIX machine. The modification of the relative difference equation was formalized into writing to gain access to the SFN learning environment. Software was written out of a desire to study something of interest and to return to the scientific community software tools that do not require expensive machines and yearly software contracts. Hopefully the work and source code presented here will be useful, especially those interested in pharmacophore modeling and are budget disadvantaged.

Vaswani, M., Linda, F.K., and Ramesh, S. Role of Selective Serotonin Reuptake Inhibitors in Psychiatric Disorders: a Comprehensive Review. *Progress in Neuro-Psychopharmacology & Biological Psychiatry*. 2002, **27**, 85-102

Table of Contents

Table of Figures

Introduction to the Methodologies for Designing and Employing a Three-Dimensional Pharmacophore Model Based on Multiple Ligands.

Overview

Employing molecular models facilitates contemporary drug design. Three types of models may be employed, including protein-, protein-ligand- and ligand-based models. This dissertation describes the design and execution of a ligand-based modeling methodology. Specifically, the development of a three-dimensional (3D) pharmacophore model of the serotonin selective reuptake inhibitor (SSRI) binding domain of the serotonin transporter (SERT) is provided. The basis of the modeling paradigm employed is founded upon the comparison of molecular descriptions structured from multiple distance space measurements and geometric attributes. In turn, this allows a quantitative examination and comparison of the 3D ligand measurements and attributes based on the use of the relative difference equation operated in a summated manner. The comparisons of all possible permutations of ligand conformations within sub-clusters can be analyzed by a summed relative difference approach. Thus, manipulations and analyses of the resultant large data can require the development of custom software. This chapter provides an overview of the modeling paradigm and the approach to drug design. The following three chapters encompass detailed accounts of the ligand-based model, the relative difference approach, and computational aspects. The specific methodology developed and employed in

this dissertation together has afforded a novel ligand-based model of the SSRI binding domain of the SERT.

Introduction

The design of new molecules targeted towards a specific binding domain is greatly aided by the use of a functional pharmacophore model. This includes the design of novel ligands to act as therapeutics, diagnostic probes or imaging agents. A design model can be based on protein structures, predicted protein-ligand interactions and/or a binding region outlined by the counterpart selective and potent ligand(s). A lock and key partnership may be used as a figurative analogy to a protein-ligand interaction. A protein-based model can be considered a description of the lock, including secondary and tertiary structural lock aspects. The protein-ligand model would define the direct structural requisites of the lock and key interaction. In other words, the protein-ligand model would predict the interior tumbler configuration. A model based upon one or more ligands could be thought of as a definition of the tumbler configuration by the analyses of an overlay of multiple keys known to fit the lock. When limited definition of the lock or protein exists ligand-based models offer an alternative and efficacious approach for development of a functional pharmacophore construct useful for drug design.

Though protein-ligand models are desirable, the lack of a predictive protein crystal structure necessitates the use of a ligand-based model. This is the

case with the protein structure of the serotonin transporter (SERT). In 2003, two twelve-transmembrane domain proteins from the major facilitator superfamily, that SERT is included within, were successfully crystallized (Abramson, Huang). However, the predicted intracellular loops in the SERT sequence are not accommodated by the published crystal structures of the *Escherichia coli* lactose permease or glycerol-3-phosphate transporters (Abramson, Huang). Thus, to build a SERT protein homology model based on these recent X-ray structures would lead to an erroneous homology construct. Since a plausible 3D SERT structure has yet to be proffered, a ligand-based model is required for SSRI SERT rationale drug design. New, highly potent and selective ligands could be therapeutic drugs or new cerebral imaging agents. These new ligands types would promote further understanding of select disease states and psychiatric disorders (Vaswani).

The primary goal of this dissertation was to produce a 3D model of the SSRI binding domain at the SERT, that would enable the design of unique and more optimal SERT specific ligands. We desired to develop a methodology for producing the SSRI SERT model that was quantitative, rigorous and based on all plausible ligand comparison permutations. The prospective size of the data sets produced from this analysis approach would require the use of custom software. Standard spreadsheet software solutions most likely would not have the capacity to deal with the amount of data encountered.

In developing the model generation methodology three primary features were addressed. The approach would: a) employ readily available software

resources (not based on a specific software package), b) be promoted by early stage objective numerical analysis to limit subjective user input and c) rely on minimal late stage subjective visual inspections and analysis. By not basing the methodology on a specific software package, the approach remains reproducible in labs with other software packages or computational resources. Beyond initial selection of ligands to be used in the exercise (the training set), removing direct user interaction with ligand conformational analysis by basing methodologies on remote, calculated investigations was thought essential for maintaining objectivity.

Our perspective has been that a model developed should provide predictive qualities, yet may not be a true representation of the biological (protein and/or ligand) motif and/or partnership event. Once a model has been developed, testing its degree of predictive quality is essential such that flawed models are avoided. Initial verification of a model can be accomplished by comparing it with ligands known to bind at the same binding domain as the model, but with varying degrees of potency. This leave-one-out verification step is addressed in Chapter Two. However, the final proof of the predictive quality comes from using the model to design novel ligands.

Ligands designed using a model in theory should possess specificity and affinity for the binding domain of interest. The fresh ligands can be used to improve the model through iterative refinement analysis. By altering structural features of the ligands, such as overall size, extensions of linking atoms (e.g. carbon), or simply adding different functional groups at periodic points about

the structure, allows for in-depth 3D mapping of the binding domain pocket. An end point to this recurrent design and refine cycle is reached when the spatial resolution of the model no longer increases. Alternatively, if the protein binding domain becomes known by X-ray determination or as a related homology model, the model mapping exercise is also accomplished.

This chapter presents a brief background of the modeling, model verification and new ligand design exercises. Chapter Two exemplifies the modeling methodology. Chapter Three discusses the use of relative difference for ligand comparison by developing a similarity score used in the modeling methodology. Chapter Four discusses the implementation of the similarity score in custom software development. The remaining portion of the dissertation consists of several appendices that present the primary data obtained during the modeling exercise. The appendices also include the source code for the software written to enable the modeling exercise.

Approaches to Modeling

The ease of developing models based on multiple ligands has greatly improved over the past few years. For example, the machine learning based programs HipHop (Accelrys, Inc.) and Gasp (Tripos, Inc.) takes multiple ligands as input and automatically produce a model. Unfortunately, these programs produce multiple models requiring further time and expense to determine which model may possess enhanced predictive qualities. By having the user

place the selected ligands into the software black box without understanding the fundamentals of pharmacophore production, this leaves the user in a quandary when interpreting the resultant pharmacophores. However, having multiple models may be helpful for defining and understanding different plausible binding scenarios. Custom, numerically oriented and manual approaches can be taken to avoid producing multiple models. In the past, SERT models have been based on the global energy minimum conformation of a single molecule (Rupp). The global energy minimum conformation need not be the bioactive conformation (Martin, Nicklaus). As well, a model based on a single molecule will only be as descriptive as the number of features presented by the single molecule itself.

A model based upon multiple molecules provides a more comprehensive description of the 3D binding domain space. The multiple molecule approach is based on comparing seemingly dissimilar molecules that possess high affinity and specificity of selected binding domain to identify the optimal superposition. In the past, this approach has based comparisons on either subjective visual inspections or root means square (RMS) similarity calculations (Mottola, Gundertofte). The approach employed here describes the generation of multi-molecule composite assemblages using distance space descriptors with comparison of those descriptors using relative difference.

The Modeling Exercise

The comparison of training set molecules (ligands) to identify the optimal superposition is actually an analysis of select low energy conformations of the molecules. In order to accomplish this, all plausible low energy conformations of each molecule must be found. Typically, low energy conformation are defined as those conformations whose energy is less than or equal to 30 kilocalories above the global energy minimum conformation of the molecule (Rupp). It is a good idea to use multiple techniques to insure full descriptive coverage of ligand conformational space. For example, multiple conformation searching techniques might include Monte Carlo and dynamics based methods. In past experience it has been found that the use of only one conformational search method usually does not yield a full set of the low energy ligand conformations.

The low energy conformations are described using distance space descriptors, such as molecular distances and angles. The same set of descriptors must exist for every molecule to allow for the inter-molecular comparisons to occur. As well, the description must be complete and robust enough to distinguish the differences between conformations of the same molecule. The methodology for model building described herein is limited by the number of common distance space descriptors used. It is possible that the molecules chosen for the model building exercise could be so dissimilar that they possess an inadequate number of common distance space descriptors. If this is the case, alternative analysis techniques such as comparing specific atoms,

pharmacophore points, or volumes may become more appropriate.

Subsequently, conformational sets consisting of one conformation of each training set molecule are formed. Then each set is combinatorially compared by analysis of the participating conformer 3D molecular descriptors, thereby providing a similarity score. This score describes the similarity of the ligands in 3D space and is based on the summed relative differences of the distance space descriptions. Since the approach is based on relative difference, the lower the similarity score the more similar the conformations are in 3D space.

Calculating similarity scores for every possible combination of the low energy conformations quickly escalates into many comparisons resulting in gigabytes of data. This data then needs to be sorted in order to find the lowest scores representing the conformations most similar in 3D space. When the data set is too large to fit into the main memory of the computer, it can still be sorted out-of-core memory. This entails a large amount of very slow reading and writing operations to files. A certain amount of speed can be gained by using a parallel file system. Additionally, run times may be improved by keeping only a subset of the data, namely, a portion of the lowest scores. Keeping a subset of the scores allows in-core sorting. Yet, the question remains as to how many of the low scores are enough to fully represent a set of functional model binding. In practice, keeping more than just the lowest score multi-ligand conformational group is useful for understanding subtle commonalities amongst the ligand conformations that were not used as scoring molecular descriptors. Hence, random visual examination of the sets of conformations with increasing

similarity scores is useful for identifying these subtle commonalities.

Model Verification

After finding an optimal superposition of ligands in 3D space, the superposition can be used as the basis for a pharmacophore model. Preliminary validation of the model can occur by employing previously unused (leave-one-out) ligands (with high affinity and selectivity for the same binding domain) and comparing them to the initial model. If one of the low energy conformations of the ligands chosen for validation matches the model, then the model can be used for an initial design exercise. If none of the low energy conformations of the leave-one-out ligands match the model then the model is deemed invalid. A non-robust model may arise as a function of distance space descriptors being too general and/or too few distance space descriptors have been employed. Further, one or more of the molecules composing the initial model may need to be replaced in order to provide a better selection of common distance space descriptors, and therefore, a more robust model.

The Drug Design Exercise

There are two ways to approach the new drug design exercise that utilizes the ligand-based pharmacophore model. This can be either as a team or as an individual effort. The more efficient of the two methods occurs when

experts in chemical synthesis, toxicology and metabolism work together with the model. The other approach is for the individual researcher to take on the design exercise alone. Potential efficiency of working in a team comes from the ease of collectively understanding synthesis details, potential toxicity issues, logP (lipophilicity) profiles and metabolism of the ligand, all of these variables may be considered in one design meeting. Conversely, existing software can aid the individual in the design of synthesis, calculation of logP values, toxicity profiles, and metabolism of the ligands. Yet, relying on several software predictions can pose a precarious situation to the individual drug designer.

New ligands designed against the model need to present points of interaction with the protein at the same 3D locals as described by the model. However, for new drug design it should be realized that it may not be necessary to present all points of interaction described by the model. The points of interaction that are important for the project can be decided by the design team. The conformation of the new ligand designed by the team or individual may require more energy than is available in the biological system to achieve the conformation necessary for protein binding. Therefore, a conformational analysis should be employed for each new ligand in order to identify if a low energy conformation(s) agrees with the model. Then the new ligand motifs afforded from the design exercise are exemplified by synthesis. Subsequently, the new drugs are evaluated for binding affinity (e.g. Ki, competitive inhibition binding constant) and selectivity at the SERT. These data provide proof of the predictive qualities of the pharmacophore model for new ligand design purposes.

Conclusion

In summary, an objective and quantitative model construction methodology based on multiple ligands has been developed for producing a 3D pharmacophore model of the SSRI binding domain of the SERT. The multi-ligand training set proffers a fairly complete description of the SSRI SERT where a select optimal superposition serves as the foundation of the model. The methodology for finding an optimal superposition of seemingly dissimilar ligands in 3D space supports the use of readily available computational resources instead of a specific software package.

The low energy conformations of ligands known to bind at the binding domain being modeled are compared with the model to provide initial validation. After initial confirmation of the predictive qualities of the model it can be used as a design template. This multidisciplinary design exercise needs to assess the ease of synthesis, potential toxicity, metabolism, and logP value of the potential new ligand. The model can be an effective tool for the design of novel ligands with high affinity and selectivity for the binding domain of interest.

References

Abramson, J., Smirnove, I., Kasho, V., Verner, G., Kaback, H.R., and Iwata, S. Structure and Meachansim of the Lactose Permease of *Escherichia coli*. *Science*, 2003, **301**, 610-615

Accelrys, Inc., San Diego, California, www.accelrys.com

Gundertofte, K., Bøgesø, K.P., and Liljefors, T. A stereoselective pharmacophore model of the serotonin re-uptake site. In: *Computer-assisted lead finding and optimization*, Waterbeemd, H., Testa, B., and Folkers, G., Eds., VHCA, Basil, and Wiley-VHC, Weinheim, 1997, pp.445-459.

Huang, Y., Lemieux, M.J., Song, J., Auer, M., Wang, D. Structure and Meachanism of the Glycerol-3-Phosphate Transporter from Eacherichia coli. *Science*, 2003, **301**, 616-620

Martin, Y.C. Pharmacophore Mapping. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 121-148.

Mottola, D.M., Laiter, S., Watts, V.J., Tropsha, A., Wyrick, S.D., Nichols, D.E., and Mailman, R.B. Conformational analysis of D1 dopamine receptor agonists: pharmacophore assessment and receptor mapping. *J. Med Chem.* 1996, **39**, 285-296.

Nicklaus, M.C., Wang, S., Driscoll, J.S., and Milne, G.W. Conformational changes of small molecules binding to proteins. *Bioorg. Med. Chem.* 1995, **3**, 411-428.

Rupp, A., Kovar, K., Beuerle, G., Ruf, C., and Folkers, G. A new pharmacophoric model for 5HT reuptake-inhibitors: differentiation of amphetamine analogues. *Pharma. Acta Helv.* 1994, **68**, 235-244.

Tripos, Inc., St. Louis, Missouri, www.tripos.com

Vaswani, M., Linda, F.K., and Ramesh, S. Role of Selective Serotonin Reuptake Inhibitors in Psychiatric Disorders: a Comprehensive Review. *Progress in Neuro-Psychopharmacology & Biological Psychiatry*. 2002, **27**, 85-102

A Simple Methodology for the Production of Three-Dimensional Models: Serotonin Transporter as Example.

Introduction

Methodologies and tools for determining molecular similarity are useful for identifying optimal superpositions of seemingly dissimilar molecules which bind at a common binding domain (Perkins, Papadopoulos). Considering molecular diversity along with the varied protein binding domain motifs, the availability of multiple methodologies and tools for finding molecular similarity allows for the development of functional models to become more efficacious. The investigation presented here describes a straight forward and numerical methodology for discerning molecular similarity amongst ligands. The easy to understand, bias limiting multi-molecule approach (Dean) has been applied to the fabrication of 3-D pharmacophore template of the serotonin selective reuptake inhibitor (SSRI) binding domain of the serotonin transporter (SERT). Previous modeling endeavors and model building software, will be covered very briefly, followed by an in-depth presentation of the approach we took to develop a 3-D pharmacophore template.

Pioneering pharmacophore development work was solely based upon the lowest energy (global minima) conformation of a single molecule (Rupp). The the global conformational energy minima conformation need not be the bioactive conformation (Martin, Nicklaus). However, molecules possessing a single low

energy conformation, or where the bioactive conformation is known, could be used as the foundation for a model. The structural features of this one molecule, or lack there of, could lend to a bias in the model (Nicklaus). Potential bias, or descriptive inadequacies, can be avoided by using multiple, equally weighted (showing no preference for one over another) molecules in the model building process.

A multi-molecule approach to model building has been previously reported (Mottola, Gundertofte). These methodologies have employed superpositioning of molecules using such force as to introduce conformational distortions, have used visual inspections to to subjectively determine conformational similarity or both. Results obtained in this manner have inherent bias, which may lead away from, instead of towards a mean result. Introduction of subjectivity into the modeling exercise will lead to results which are ambiguous. This can also lead to results which are not repeatable in other labs. By employing the same force field to calculate conformational energy and the same methodology, identical models should be reproducible at separate labs.

Advances in software (Accelrys, Tripos) have afforded users a level of abstraction from the modeling exercise. These computational tools have brought modeling to a wider audience, which in turn has enabled greater rate of model production. Unless the basis of the tools are understood the increased productivity may be in vain. Automated pharmacophore development tools usually produce multiple plausible models. It is then up to the user to pick the right one. This can either be done subjectively or through further testing of

each possible model to determine which one is right. The later can be time and resource intensive.

As a brief overview of our methodology, the approach we undertook is based on a more descriptive multi-molecule construct. Where the multiple molecules posses a high degree affinity and selectivity for the binding domain of interest. A full search for all low energy conformations of each molecule was performed and compared numerically to determine the set of conformations that are most similar. This set of most similar conformations, consisting of one conformation from each molecule, was superposed using minimal energy. The approach is covered, in-depth, below.

Each of the molecules on which the model is based is able to access a specific 3-D conformation in order to interact at the binding domain. At the binding event, each molecule must present this conformation that is functionally similar to the conformation presented by the other molecules at the same event. As long as the same dynamic response is elicited. In other words, each molecule must present a similar conformation to interact at the binding domain. If the most similar conformations of each molecule are found, then these conformations could represent the bioactive conformation of each molecule (Dean, Jin). This assumes that the molecules are binding within the same binding pocket of the binding domain.

The molecules selected (the training set) for the modeling exercise should possess high affinity, selectivity for the binding domain of interest, structural rigidity, structural diversity, and contain a common set of descriptors. A

common set of descriptors may be ambiguous at this point, but it will be discussed later in detail. Theoretically, a multi-molecule construct must contain two or more molecules. It has been found, using four to five molecules limits bias while keeping the data sets small enough to be usable. Three molecules may just provide enough information for an unbiased model, while data sets containing six or more molecules may become too large to be usable. The time and computational resources may not be available to deal with very large data sets. If more molecules are found than are needed by the training set, then one can be set aside for a later use in a simple first pass "leave one out" test of the viability of the model.

Once the training set has been selected, conformational analysis of the molecules occurs. All low energy conformations of each molecule must be found for the use in the comparison step. We defined low energy conformations as a molecular conformation resting at a local energy minima which is less than 30kcals/mol above the global energy minimum of the molecule (Rupp). It has been argued both 30kcals/mol above the global energy minimum is too high and the conformational energy of the protein bound structure could be as high as 40 kcals/mol over the global conformational energy minima in a vacuum (Nicklaus).

Conformational analysis can be accomplished by multiple methods (eg. molecular dynamics, Monte' Carlo, systematic, etc). It has been found that the use of more than one of these methods ensures better coverage of conformational space. As well, the same force field must be used in the minimization step of

the conformation search routines. Using multiple force fields will lead to the production of conformationally and energetically different, but incomparable conformations.

Similarity is based upon an objective comparison of molecular features in order to determine which conformations, between the molecules, are most alike. The comparative features can range from molecular surface areas and electrostatic fields to distance space descriptors (Jin, Greco, Crippen, Blumenthal). We elected to use distance space descriptors. As the comparative descriptors more completely describe the molecules, the uniqueness of each conformation becomes numerically more apparent. The comparison of descriptors is carried out using a combinatorial relative difference calculation. The outcome of the calculation is a similarity score or a value representing how similar a set of conformations are. The lower the score the more similar the conformations in 3-D space. The purely mathematical approach provides an objective basis as opposed to subjective decisions based upon visualization, which can be misleading.

After numerically reaching this point, a subjective check may provide initial validation of the modeling exercise. A superposition of the most similar set of conformations should produce an image which visually shows the conformations better accessing equivalent points of potential binding (better alignment in an overlay image) than a superposition of a conformational set with a median score. The set of conformations with the median score should, in turn, provide a better overlay than the most dissimilar set of conformations.

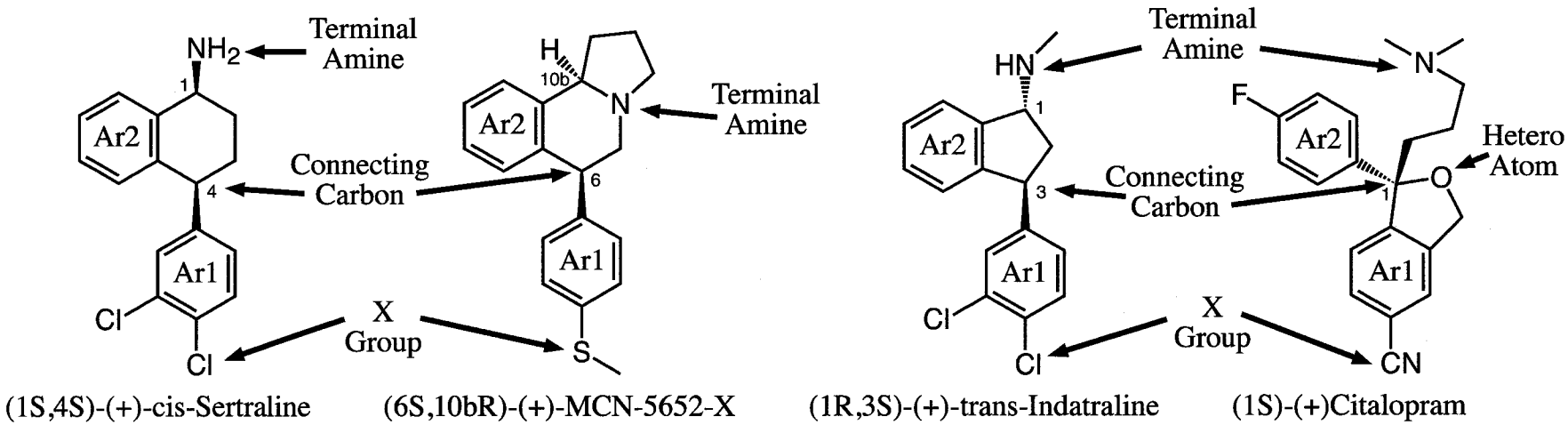
If the lower scoring overlays do not visually provide better overlap of points determined to plausibly be important during the binding event, then the modeling exercise need to be repeated with changes enacted. Changes may include using a more detailed set of descriptors, replacement of one or more of the molecules, or removal of a molecule.

As part of this proposed methodology a first pass test of the pharmacophore models validity is accomplished by means similar to a "leave one out" method. A molecule not utilized in the modeling exercise is compared with the pharmacophore for similarity. If comparison shows the leave one out molecule is similar to the model then it becomes appropriate to spend lab time and resource for further validation for the model. Even though only one molecule compared with the model is written to here, it is appropriate to compare more than one molecule with the model. The leave one out molecule can be of low or high affinity for the binding domain of interest, but should possess structural rigidity. A very flexible molecule inherently has many low energy conformations and should be avoided, if possible. The number of low energy conformations increases the possibility of finding one similar to an incorrect model. Ibogaine was selected as the molecule to be used in our "leave one out" test. Ibogaine has low affinity for the SSRI binding domain of the SERT (0.55 μM, (Baumann)) and is promiscuous at other binding domains. The structural rigidity of ibogaine extremely limits the possibilities for presenting the pharmacophore to bind at the SSRI binding domain of the SERT, making it a good candidate for an initial comparison with the 3-D SSRI SERT pharmacophore model.

Methodology

Four molecules were chosen (based on activity, selectivity and structure) for modeling the 3-D pharmacophore for the SSRI binding domain at the SERT (Figure 2.1). Conformational analysis was performed on the four molecules using both a Monte' Carlo and a molecular dynamics methodology. The Monte' Carlo method used was called "randomsearch" and is a part of Sybyl 6.4 (Tripos). The molecular dynamics method used was AESOP which was run within the Sybyl 6.4 environment (AESOP). A Sybyl molecular spreadsheet was produced for each molecule containing all the conformations found through the multiple conformational searches. All conformations were moved to their local conformational energy minima using the minimization routine with the force field set at the default settings in Sybyl 6.4. Low energy conformations were defined as conformations with energy less than or equal to 30 kcals/mol above the global energy minimum for the molecule. High energy conformations were defined as conformations with energy greater than 30 kcals/mol above the global energy minimum for the molecule. Duplicate and high energy conformations were eliminated from the Sybyl molecular spreadsheets resulting in four spreadsheets, where each one contained the low energy non-duplicate conformations of one molecule. The specific setting used in the software for the conformational searches as well as the results of these searches can be seen in Figure 2.2.

Four Molecules Chosen for Construction of the 3-Dimensional Pharmacophore of the SSRI Binding Domain at the SERT



High SERT affinity (< 3 nm) and SERT selectivity

Semi-rigid, Stereo specific

Structural Diversity (Ar1, Ar2, Ar-substituents, heteroatom, terminal amine, head group)

From left to right the Terminal Amines can be described as being up, constrained, down and flexible.

Figure 2.1 The four molecules used in model building exercise: Sertraline, MCN-5652, Indatraline, S-Citalopram. Defined location of Ar1, Ar2, X, Terminal Amine, Connecting Carbon, and the hetero atom shown.

Distance space descriptors were used in developing a comparable set of descriptors for the SERT SSRI modeling endeavor (Figure 2.3). The low energy conformations of each molecule were described using the descriptors in Figure 2.3. The descriptor measurements were obtained from and stored using Sybyl molecular spreadsheets. The four Sybyl molecular spreadsheets containing the measures were exported as text files to facilitate the comparisons, as described below.

The description of each conformation of each molecule was combinatorially compared against the descriptions of every conformation of the other molecules. The comparison was accomplished using the relative difference equation for the case when neither measurement is known to be correct. The exact implementation of the formula can be seen in Figure 2.4. A lower score indicates the conformations being compared are more similar. Descriptor comparison was carried out using a custom written program that encompasses the implementation of the relative difference equation. The program creates a text file consisting of the comparison results for each descriptor and the overall similarity score for each comparison group.

The text was loaded into the data analysis and graphing program Igor Pro (WaveMetrics). Using Igor Pro the comparison groups were sorted by their overall similarity score from lowest score to highest score. The graph of the sorted scores can be seen in Figure 2.5. The 128 lowest scoring (most similar) conformational comparison groups were various conformations of rotomers of the same conformations. The 128th lowest scoring group was the lowest

Conformational Analysis and Results

(1S,4S)-(+)-cis-Sertraline (6S,10bR)-(+)-MCN-5652-X (1R,3S)-(+)-trans-Indatraline (1S)-(+)Citalopram

randomsearch (Monte' Carlo) as implemented in Tripos Sybyl 6.4 using the default settings except check chirality was always selected and the following:

tailor random search ring bond (*angstroms*)	16	16 - 32	16 - 24	16
minimization maximum iterations	600 - 800	600 - 1,000	600 - 800	600
randomsearch maximum iterations	1,000 - 10,000	1,000 - 10,000	5,000 - 10,000	10,000 - 25,000
energy change cutoff (*kcals/mol*)	50 - 3,000	50 - 8,000	50 - 3,000	35 - 1,600
RMS threshold	0.001 - 0.002	0.000 - 0.003	0.001 - 0.002	0.002 - 0.003
convergence threshold (*kcals/mol*)	0.000	0.000	0.000	0.000
maximum hits	16	18 - 54	18	18
number of searches	5	8	8	5

Aesop (molecular dynamics) as implemented by Zeneca in the Tripos Sybyl 6.4 environment using the default settings except as follows:

temperature (*K*)	1,000 - 3,000	500 - 3,000	1,000 - 3,000	1,000 - 3,000
time (*fs*)	40,000 - 80,000	20,000 - 80,000	40,000 - 80,000	20,000 - 80,000
expression	1,000 - 10,000	1,000 - 10,000	5,000 - 10,000	10,000 - 25,000
number of searches	6	8	6	8

Totals after minimizing conformations to zero energy change using the default setting in Sybyl 6.4 and combining the resultant conformations and removing duplicate and high energy conformations

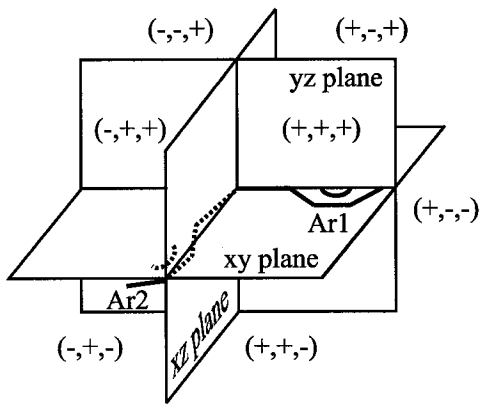
number of low energy conformations	16	12	29	233
energy range of conformations (*kcals/mol*)	1.442 - 8.085	7.397 - 13.057	7.467 - 15.832	8.853 - 19.634

Figure 2.2 Conformational analysis (software settings and results) of Sertraline, MCN-5652, Indatraline, and S-Citalopram.

Defining the Conformations in Distance Space

Definitions Made in Sybyl 6.4 to Facilitate Measurement Descriptions

Ar1 Centroid
Ar2 Centroid
Ar1 Axis
Ar2 Axis
YZ Plane - defined as the plane going through both the Ar1 Axis and Ar2 Axis
XY Plane - defined by the Ar1 Axis and 2 normals that start at the connecting carbon and are normal to the YZ Plane
XZ Plane - defined by the normals that start at the connecting carbon and are normal to the YZ Plane and the normals that start at the connecting carbon and are normal to the XY Plane
Ar1 Plane - defined by the Ar1 ring
Ar2 Plane - defined by the Ar2 ring



Descriptions Used to Describe Conformations in Distance Space

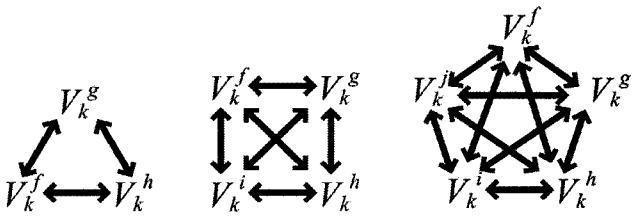
N to Ar1
N to Ar2
N to Ar1 Plane - absolute value
N to Ar2 Plane - absolute value
N to the Connecting Carbon
Ar1 Centroid to Ar2 Centroid
Ar1 to Ar2 Plane - absolute value
Ar2 to Ar1 Plane - absolute value
Angle Ar1 Cetroid to the Connecting Carbon to N
Angle Ar2 centroid to the Connecting Carbon to N
N to YZ plane - explicit value
N to XY plane - explicit value
N to XZ plane - explicit value
Plane Angle - Ar1 Plane to YZ plane - explicit value
Plane Angle - Ar2 plane to YZ plane - explicit value

Explicit disance measurments are positive or negative depending on where N is in the defined coordinate system as shown.

After measurement, plane angles were adjusted to have the angle distibution centered at 90 degrees, with all conformations having a congruent placement of zero degrees

S-Citalopram was also described with Ar1 and Ar2 reversed from what is shown in figure 2.1. This reveresed description yielded much poorer similarity scores leading to the conlcusion that the Ar1 and Ar2 definitions as shown in figure 1 is correct for the pharmacophore biulding exercise.

Figure 2.3 Definitions and measurements used to describe conformations in distance space.



n=3 n=4 n=5

n_{rdc}=3 n_{rdc}=6 n_{rdc}=10

As the number of molecules being compared increases the number of relative difference calculations increase following an arithmetic mean.

V_k^α = measure k of molecules α

$\longleftrightarrow$ = one relative difference calculation (rdc)

n = number of molecules

$$n_{\text{rdc}} = \sum_{i=1}^{n-1} i = \frac{(n\text{-}1)(n)}{2}$$

$$\text{similarity score} = \frac{1}{d_{measures} \times n_{rdc}} \sum_{1}^{d_{measures}} \sum_{i,j} \frac{\left|V_k^i - V_k^j\right|}{\left(\frac{\left|V_k^i\right| + \left|V_k^j\right|}{2}\right)}$$

where $1 \leq i < j \leq$ number of molecules

n_{rdc} = number of relative difference calculations within a conformational comparison group

$d_{measures}$ = number of distance space descriptors

V_k^α = measure k of molecule α (distances, angles, etc.)

score = the calculated similarity of one conformational comparison group

conformational comparison group = one low energy conformation of each molecule

Figure 2.4 Equation for calculating the similarity score for one conformational comparison group through the application of relative difference. The number of relative difference calculations for the SERT SSRI data set is equal to 116,760,344 (1,297,344 comparison groups x 15 compared measures x 6 combinatorial relative difference calculations per measurements).

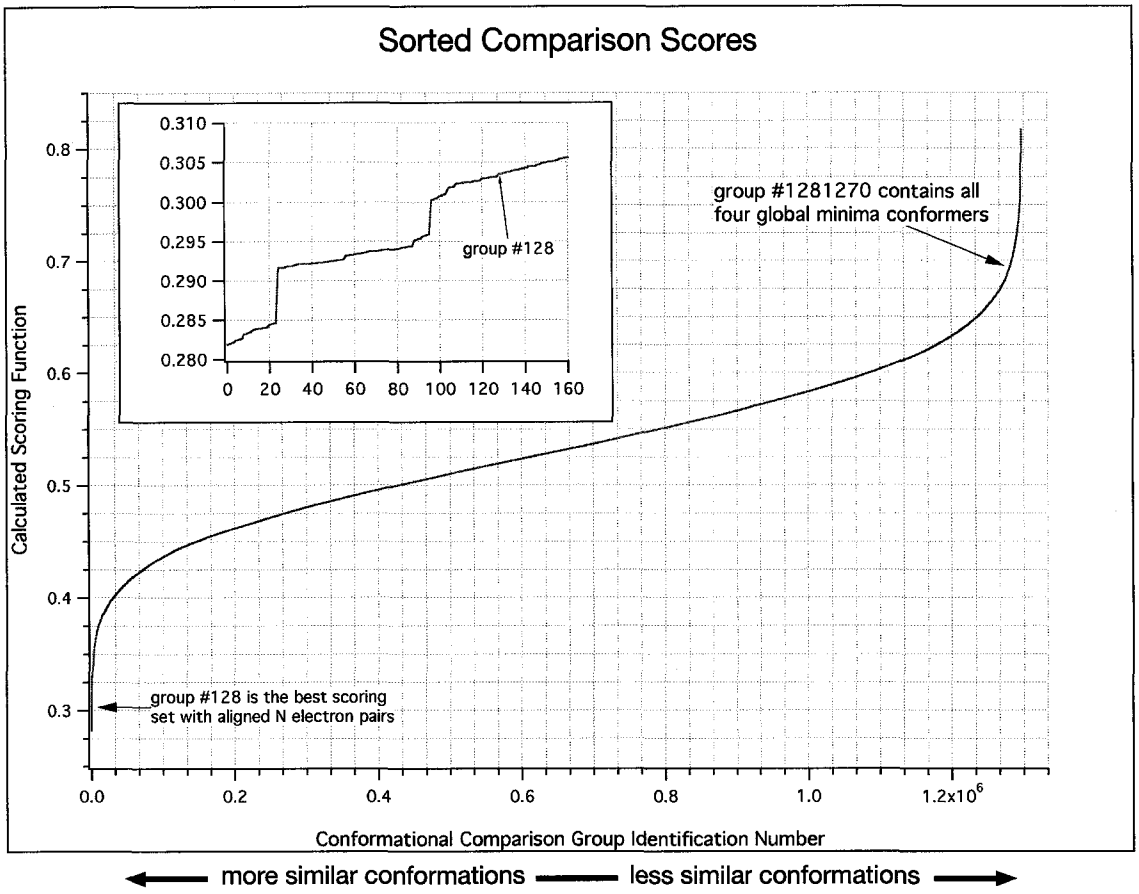
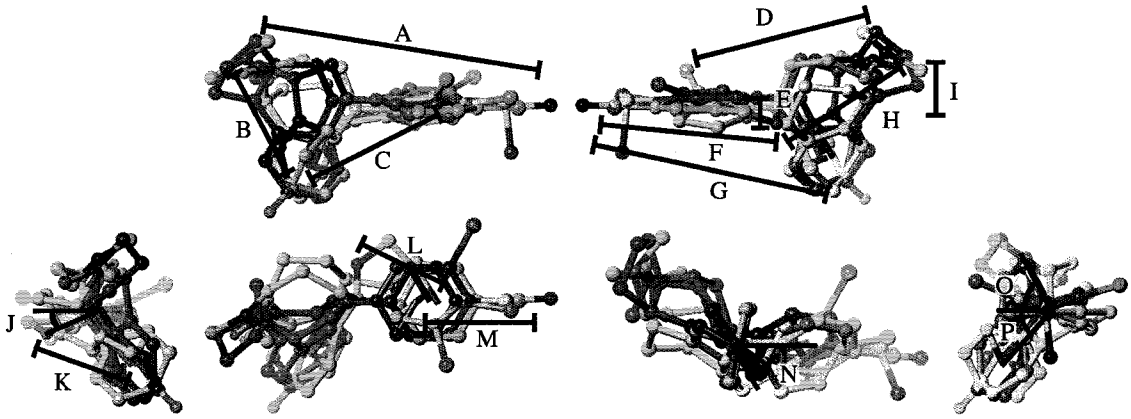


Figure 2.5 Graph of the 1,297,344 calculated similarity scores of each conformational comparison group sorted from low (most similar) to high (least similar). The inset graph shows the 160 lowest similarity scores.

scoring instance were the lone pair of electrons on the terminal amine aligned and therefore was selected as the basis for the SSRI binding domain of the SERT model (appendix 2.1).

Superposition of the 128th most similar conformational set can be seen in Figure 2.6. Superposition of the four conformations involved in a comparison group of SSRI ligands was achieved using the multifit routine in Sybyl6.4. In this routine, a spring constant was set between pairs of points on each conformation. A minimum of three pairs of points between each conformation is required. Using a large spring constant at select locations will literally force



S-Citalopram ■ Indatraline ■ MCN-5652 Sertraline

	A	B	C	D	E	F
	distance N to X *(angstroms)*	distance N to Ar2 centroid *(angstroms)*	distance Ar1 centroid to Ar2 centroid *(angstroms)*	distance N to Ar1 centroid *(angstroms)*	distance N to Ar1 Plane *(angstroms)*	distance Ar2 centroid to X *(angstroms)*
Sertraline 2.135 kcals/mol	8.737	3.720	4.822	5.949	1.352	7.611
MCN-5652 13.035 kcals/mol	8.253	3.734	4.912	5.164	1.430	7.762
Indatraline 14.532 kcals/mol	9.342	3.762	4.952	6.226	1.150	7.800
S-Citalopram 15.585 kcals/mol	11.031	4.618	4.975	7.131	1.843	8.696
average	9.341	3.959	4.915	6.118	1.444	7.967

	G	H	I	J	K	L
	distance hetero atom to X *(angstroms)*	distance N to hetero atom *(angstroms)*	distance hetero atom to Ar1 plane *(angstroms)*	torsion angle between hetero atom and Ar1 plane *(degrees)*	distance hetero atom to Ar2 centroid *(angstroms)*	distance hetero atom to Ar1 centroid *(angstroms)*
S-Citalopram 15.585 kcals/mol	7.051	5.098	0.489	16.4	3.713	3.328
average	7.051	5.098	0.489	16.4	3.713	3.328

All measurements were taken using Tripos Sybyl 6.4. The conformations have been energy minimized to zero energy change using Sybyl 6.4 default settings.

	M	N	O	P
	distance Ar1 centroid to X *(angstroms)*	torsion angle between Ar1 axis and Ar2 Plane *(degrees)*	torsion angle between N and Ar1 plane *(degrees)*	torsion angle between Ar2 axis and Ar1 plane *(degrees)*
Sertraline 2.135 kcals/mol	3.153	42.0	23.6	46.8
MCN-5652 13.035 kcals/mol	3.175	31.8	69.3	47.7
Indatraline 14.532 kcals/mol	3.153	50.9	73.2	36.3
S-Citalopram 15.585 kcals/mol	3.992	81.7	78.4	53.6
average	3.368	51.6	61.1	46.1

Figure 2.6 Superposition of Sertraline, MCN-5652, Indatraline, and S-Citalopram as the basis for the model of the SSRI binding domain at the SERT.

the conformations together, possibly raising the potential energy of each conformation well outside what is possible in the native biological system. In the superposition shown in Figure 2.6, a one calorie spring constant (the minimum allowed by the software package) was used at the minimum of three point pairs between the conformation. The point pairs used in the multifit routine were the two associated aromatic ring centroids and the terminal amine. Using both the minimum number of point pairs and spring constants introduces the least amount of energy into the system.

The low energy conformations of ibogaine were determined using the SYBYL randomsearch (Figure 2.7). Distance space descriptors (Figure 2.7) were used to compare ibogaine with the SSRI SERT pharmacophore using relative difference (Figure 2.8). The calculated score was sorted from low to high (Figure 2.9). The most similar conformation of ibogaine to the SSRI SERT model was superposed with the model (Figure 2.10). The superposition was accomplished using the one calorie spring constants methodology described above. The three points used in the superposition were the terminal amine, C8 and C9 as labeled in Figure 2.7.

Results

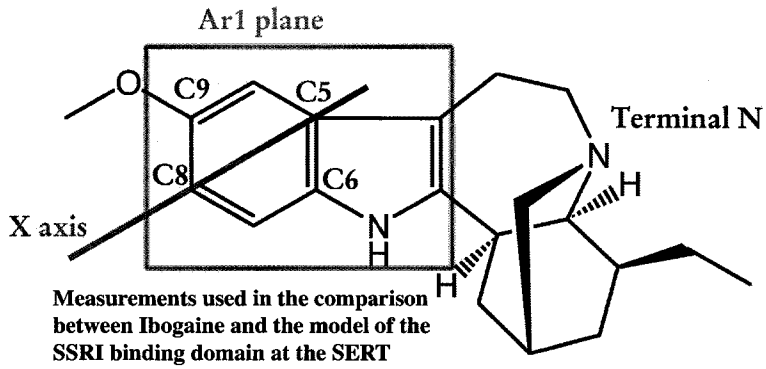
The pharmacophore for the SSRI binding domain of the SERT is based upon the 128th most similar group (Figure 2.5). The 128th most similar group is the first group in which the terminal amine lone pairs of electrons align.

Conformational Analysis of Ibogaine

SYBYL randomsearch
ring bond = 42.00 Å
minimization maximum iterations = 800
randomsearch maximum iterations = 2000
energy change cutoff = 5000K
RMS threshold = 0/003 Å
convergence threshold = 0.000
maximum hits - 32
number of searches = 1

number of low energy
conformations found = 24
energy range of
low energy conformations =
33.536 - 35.937 kcals/mol

The head group area (non-aromatic tricyclic area) is capable of two low energy conformations. Rotation of the methoxy and ethyl side chains account for 12 possible low energy conformations for each head group conformation (2 x 12 = 24).



Measurements used in the comparison between Ibogaine and the model of the SSRI binding domain at the SERT

- distance from terminal amine to C8
- distance from terminal amine to Ar1 plane
- angle between terminal amine, C8, C5
- distance along the Ar1 axis from C8 to a point which is closest to the terminal amine
- distance from the point on the Ar1 axis which is closest to the terminal amine, to the terminal amine
- distance from the point on the Ar1 axis which is closest to the terminal amine, to the point on the Ar1 plane which is closest to the terminal amine

Figure 2.7 Conformational analysis (software settings and results) of Ibogaine. Definitions and measurements used to describe Ibogaines in distance space.

$$\frac{\sum_{k=1}^{n_{measures}} \frac{\left|V_k^t - V_k^m\right|}{\left|V_k^m\right|}}{n_{measures}}$$

where m = measure from model
t = measure fron test ligand
(ibogaine, in this case)

$n_{measures}$ = number of compared measures

V_k^{α} = measure k of molecules α

Figure 2.8 Equation for calculating the relative difference similarity score comparing Ibogaine with the SSRI SERT model.

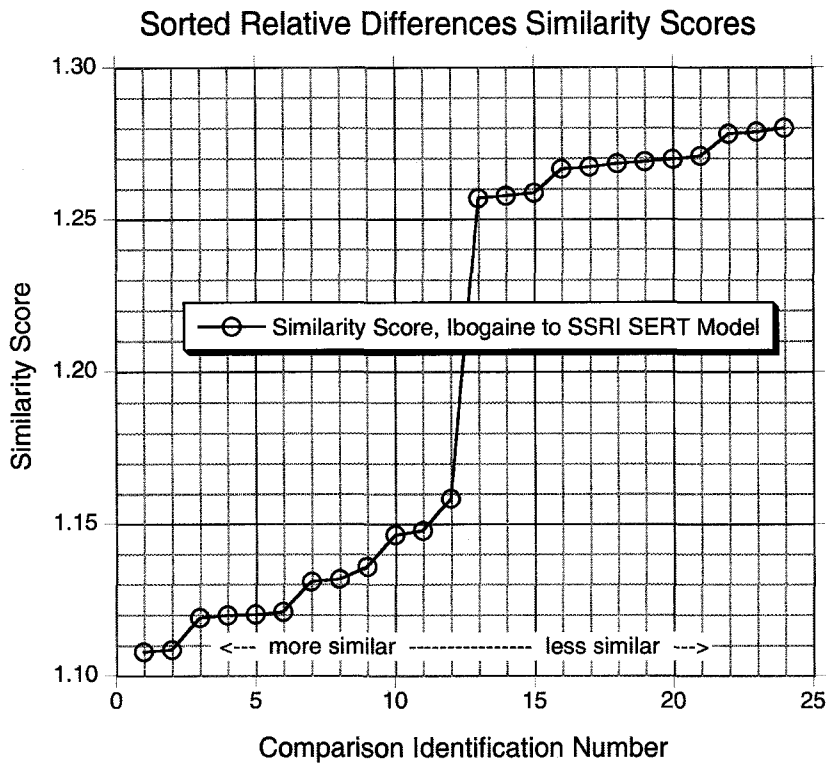


Figure 2.9 Graph of the 24 calculated relative difference similarity scores of each conformation of Ibogaine compared with the SSRI SERT model sorted from low (most similar) to high (least similar).

Terminal amine lone pair electron vectors had not been described by the descriptors. Differences between the 128th group and the 127 more similar groups are entirely accounted for by rotation of the terminal amine, X group, and the Ar1 ring. The final results of the modeling exercise, a superposition of the training set and a distance space description of the superposition are shown in Figure 2.6.

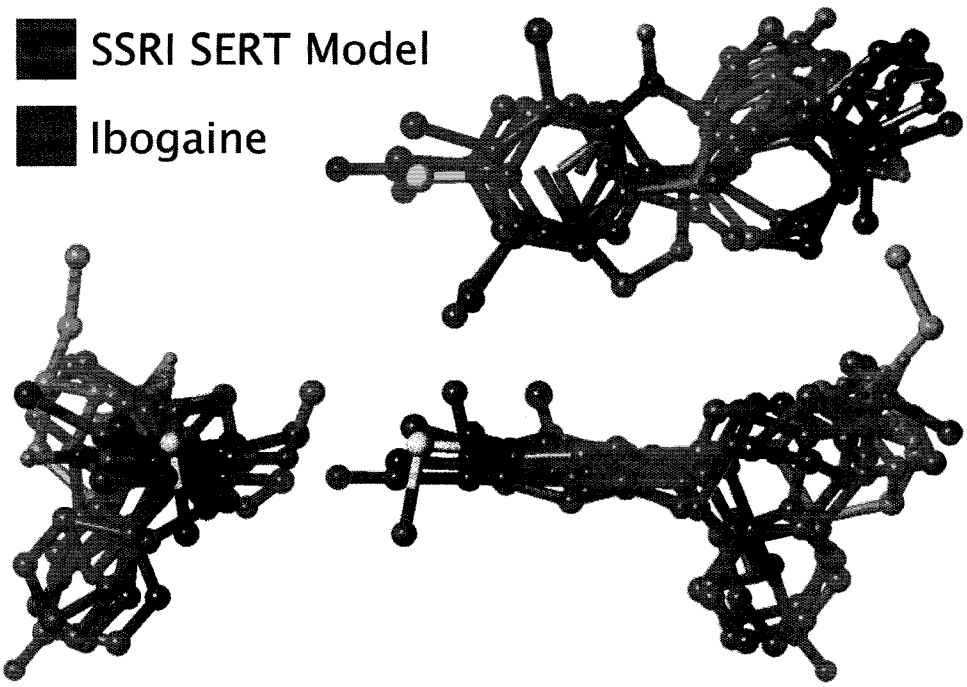


Figure 2.10 Superposition of the most similar scoreing conformation of Ibogaine to the SSRI SERT model, with the model of the SSRI binding domain at the SERT.

Discussion

Considering the shape of the graph shown in figure 2.5, a small percent of the total number of conformational comparison groups are very similar in 3D space. As well, a small percent of the total number of conformational comparison groups are very dissimilar in 3D space. The majority of similarity scores indicate there is a small consistent increase in the lack of similarity going from left to right on the graph. These properties, seem intrinsic to the type of analysis done here.

The trend in increasing similarity score should be visually apparent when a subjective viewing of a random sampling of conformational comparison groups with increasing similarity scores. This should be especially true if at least ten percent of the

data occurs between the random sampling points. Upon visual inspection, subjectively if the increasing dissimilarity is not seen the results may be classified as ambiguous. Ambiguous results warrant close inspection before continuing the modeling exercise. Experience, has shown an increase in the number of common distance space descriptors will remove the ambiguity.

The model of the SERT SSRI binding domain was developed from inhibitors which bind at the SERT, but do not transport. In theory, new molecules produced with a reduction in the number of points of interaction between ligand and transporter, addition of conformational flexibility, or a combination of both will lead to transportable substances at the SERT. This could be systematically tested to achieve understanding of the amount of flexibility and which combinations of points of interaction are necessary to prevent transport.

Ibogaine provides some initial answers. Ibogaine is a very rigid molecule, lacks a second aromatic ring and does not transport at the SERT. The ibogaine head group area occupies space not defined by the SSRI SERT model. As well, the potential of the ibogaine hetero atom to be involved in a binding event is reduced as the lone pair of electrons are involved in aromaticity. The aromaticity of the hetero atom and excessive molecular volume (Mottola) could explain the lower affinity of ibogaine to the SSRI binding domain of the SERT.

Relative difference is a basic straight forward way to compare two values. The most popular version of this equation compares an unknown value against a known value. The equation indicates how much larger or smaller the unknown value is compared to the known value. When relative difference is between two values of which neither value

may be correct, larger and smaller becomes mute. This explains taking the absolute value of the numerator (Figure 2.3), where the magnitude between two values is important and not whether one value is greater than the other. The denominator represents the average between the two magnitudes of the numerator. This is accomplished by taking the absolute value of each participant in the denominator, individually (Figure 2.3).

Conclusion

The 3-D pharmacophore template of the SSRI binding domain at the SERT resulting from this methodology has been successfully used in several design exercises. One of which is the design of the highly potent 2'-methyl-6-nitroquipazine SERT inhibitor ligand (Ki = 81 pM, (Gerdes)). The simplicity and effectiveness of this methodology allows the modeler to use any combination of a variety of readily available inexpensive software tools to develop rigorous 3-D pharmacophore models.

References

Accelrys, Inc., San Diego, California, www.accelrys.com

AESOP developed by B.B. Masek, Zeneca, Wilmington, Delaware.

Baumann, B.H., Pable, J.P., Ali, S.F., Rothman, R.B., and Mash, D.C. Noribogaine (12-hydroxyibogaine0: a biologically active metabolite of the antiaddicitive drug ibogaine. *Ann. N.Y. Acad. Sci.* 2000, **914**, 354-368.

Blumenthal, L.M. *Theory and applications of distance geometry.* Chelsea Publishing, Bronx, 1970.

Crippen, G.M. A novel approach to calculations of conformation: distance geometry. *J. Comp. Phys.* 1997, **24**, 96-107.

Dean, P.M., and Perkins, T.D.J. Calculation of three-dimensional similarity. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 199-218.

Gerdes, J.M., DeFina, S.C., Wilson, P.A., and Taylor, S.E. Serotonin transporter inhibitors: synthesis and binding potency of 2′-methyl- and 3′-methyl-6-nitroquipazine. *Bioorg. Med. Chem. Lett.* 2000, **10**, 2643-2646.

Greco, G., Novellino, E., and Martin, Y.C. 3D-QSAR methods. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 219-252.

Gundertofte, K., Bøgesø, K.P., and Liljefors, T. A stereoselective pharmacophore model of the serotonin re-uptake site. In: *Computer-assisted lead finding and optimization*, Waterbeemd, H., Testa, B., and Folkers, G., Eds., VHCA, Basil, and Wiley-VHC, Weinheim, 1997, pp.445-459.

Jin, B., and Hopfinger, A.J. A proposed common spatial pharmacophore and the corresponding active conformations of some TxA_2 receptor antagonists. *J. Chem. Inf. Comput. Sci.* 1994, **34**, 1014-1021.

Martin, Y.C. Pharmacophore Mapping. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 121-148.

Mottola, D.M., Laiter, S., Watts, V.J., Tropsha, A., Wyrick, S.D., Nichols, D.E., and Mailman, R.B. Conformational analysis of D_1 dopamine receptor agonists: pharmacophore assessment and receptor mapping. *J. Med Chem.* 1996, **39**, 285-296.

Nicklaus, M.C., Wang, S., Driscoll, J.S., and Milne, G.W. Conformational changes of small molecules binding to proteins. *Bioorg. Med. Chem.* 1995, **3**, 411-428.

Papadopoulos, M.C., and Dean, P.M. Molecular structure matching by simulated annealing. IV. Classification of atom correspondences in sets of dissimilar molecules. *J. Comput.-Aided Mol. Design* 1991, **4**, 119-133.

Perkins, T.D.J., and Dean, P.M. Molecular partial similarity using surface-volume comparisons. In: *Computer-assisted lead finding and optimization*, Waterbeemd, H., Testa, B., and Folkers, G., Eds., VHCA, Basil, and Wiley-VHC, Weinheim, 1997, pp. 421-432.

Rupp, A., Kovar, K., Beuerle, G., Ruf, C., and Folkers, G. A new pharmacophoric model for 5HT reuptake-inhibitors: differentiation of amphetamine analogues. *Pharma. Acta Helv.* 1994, **68**, 235-244.

Tripos, Inc., St. Louis, Missouri, www.tripos.com

WaveMetrics, Inc., Lake Oswego, Oregon, www.wavemetrics.com

The Nuances of Comparing Molecular Descriptors Using Relative Difference

Introduction

Comparing the low energy conformations of multiple, structurally diverse molecules active at the same binding domain, for commonalities in three-dimensional (3D) space can lead to a pharmacophore model for said binding domain. Mottola et al., briefly noted performing these comparisons to determine similarity between molecules, but lacked a detailed methodology, in their D_1 dopamine receptor pharmacophore development work. Using comparisons without describing how they were done makes it difficult to repeat the experiment, and to adapt the methodology to new work. Gundertofte used root mean square (RMS) to accomplish comparisons in developing a serotonin (5-HT) transporter (SERT) pharmacophore model. This technique involves comparing specific atoms that theoretically have identical locations in 3D space. Depending upon how structurally diverse the set of molecules being compared are, the number of identically located atoms may be very limited or non existent. It is possible to define identical points, which are not atoms, in space. However, in practice, this may prove difficult with current software packages such as Sybyl (Tripos). The methodology implemented here, to overcome the limitations of RMS, describes the molecules of interest using distance space descriptors (Blumenthal), followed by the comparison of these descriptors using relative difference. The relative difference equation used here has been modified

to produce the value two when a positive value is compared with zero or a negative number. The modified relative difference equation calculates the same values as the relative difference equation for values that are either both positive or both negative. However, the modified equation calculates the value of two when the two values used in the equation have opposite signs or one value is zero. Normally the values used in the relative difference equation are both positive. The modification provides an indicator when the values being used in the relative difference have opposite signs or one of the values is zero. After a brief overview of the pharmacophore building exercise, the modified relative difference equation and resulting similarity score equation will be explained.

In this exercise, low energy conformations have been defined as the those conformations with a conformational energy less than or equal to 30 kcals per mol over the global energy minima (Rupp). Multiple molecules with an affinity for the same binding domain will present points having the necessary properties for binding at the same locations in 3D space in order to invoke the binding event. The low energy conformations of each molecules are described using distance space descriptors. The descriptors are compared for similarity using the equation being developed below that calculates a similarity score for a set of low energy conformations (Figure 3.11). A set equals one low energy conformation from each molecule. The low energy conformation of each of the molecules that are the most similar in 3D space are representative of the bioactive, binding conformation. It is possible to have multiple sets of low energy conformations that possess equal similarity. In other words, the calculated similarity scores

are identical. In the event multiple sets are found to have equal similarity, it most likely stems from a distance space description that is unable to distinguish between conformations. This points to the necessity of using sufficiently descriptive descriptions when comparing conformations. Each conformation is unique and should be represented by the distance space description. When one set of low energy conformations is found to be the most similar in 3D space these conformations can be superposed, producing the foundation for a pharmacophore model.

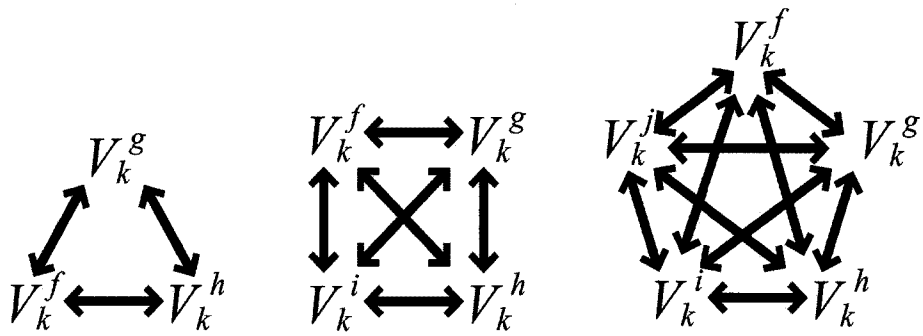
Each of the four serotonin selective reuptake inhibitors (SSRI), escitalopram, indatraline, MCN-5652, and sertraline which bind at the serotonin transporter (SERT) were described using 15 common distance space descriptors (primary data located in appendixes 3.1-3.4). The measurements that the distance space descriptors are based on were taken using the Sybyl software package. Each conformations was minimized to zero energy change using the default force field settings in Sybyl. As stated above, distance space descriptors such as distances, angles, and position vectors can be used to describe general attributes of a molecule without having to compare specific points, as in RMS comparisons. Other properties associated with the volumes and surfaces of molecules, such as regions of charge, may also be good descriptors for use in comparisons. Once every low energy conformation of each molecule is described, they can be compared.

Every combinatorial comparison set, one low energy conformation from each molecule, was compared. Within each combinatorial comparison set, each

descriptor was combinatorially compared. The total number of comparisons required in this project equals the product of the low energy conformations, number of descriptors, and the number of possible non repetitive combinatorial comparisons for each descriptor:

	12 low energy conformations of MCN-5652
x	16 low energy conformations of sertraline
x	29 low energy conformations of indatraline
x	233 low energy conformations of escitalopram
x	15 descriptors
x	6 non repetitive combinatorial comparisons for each descriptor
	(6 pairwise relative difference calculations for each descriptor)
=	116,760,960 comparisons

The number of non-repetitive combinatorial comparisons for each descriptor follows an arithmetic mean (Figure 3.1). The end result of these comparisons is an objective determination of which combinatorial conformation set is the most similar in 3D space and can serve as the basis for a pharmacophore model of the SSRI binding domain of the SERT.



n=3 n=4 n=5

n_{rdc}=3 n_{rdc}=6 n_{rdc}=10

V_k^{α} = measure k of molecules α

$\longleftrightarrow$ = one relative difference calculation (rdc)

n = number of molecules

$$n_{rdc} = \begin{matrix}\text{total number of relative}\\ \text{difference calculations per}\\ \text{distance space descriptor}\end{matrix} = \sum_{i=1}^{n-1} i = \frac{(n-1)(n)}{2}$$

Figure 3.1 The number of non-repetitive combinatorial comparisons for each distance space descriptor follows an arithmetic mean.

The Modified Relative Difference Equation

Distance space descriptors, are essentially measurements used to describe an object and can be compared using the relative difference equation (Wilson). Figure 3.2 shows the relative error equation when one value is known to be correct. The relative error and relative difference equations calculate a unit-less value. The units in the numerator and denominator cancel. The calculated value

is representative of the deviation between two measurements. The smaller this value is, the more similar the two measurements are (see the examples in Figure 3.3). Regardless of the initial magnitude of the measurements, the resultant relative differences will have similar magnitudes. This allows for these values, representative of the similarity between two measurements, to be summed without unduly weighting larger measurements.

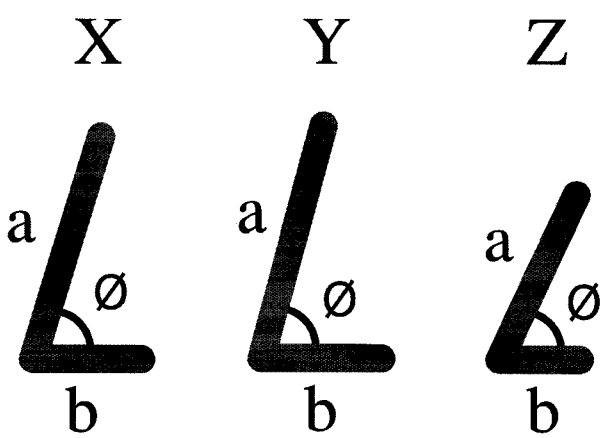
$$relative\ error = \frac{|measure - actual|}{actual}$$

Figure 3.2 The relative error equation.

Object Y is know to be the correct size. Is object X or object Z most similar to object Y?

Subjectively, in this example, one should be able to see that object X is most similar to object Y.

Objectively, this similarity between objects can be determine by using relative difference. The equation used for the special case when one object is known to be correct is referred to as relative error. The calculations using the equation shown in figure 3.2 can be seen at the bottom of this figure. Lower values indicate greater similarity. The calculations show that object X is most similar to object Y.



angle Ø = 72°	angle Ø = 75°	angle Ø = 65°
a = 58 mm	a = 60 mm	a = 45 mm
b = 27 mm	b = 30 mm	b = 20 mm

X Y comparison $\frac{|72° - 75°|}{75°} + \frac{|58mm - 60mm|}{60mm} + \frac{|27mm - 30mm|}{30mm} = 0.04 + 0.03 + 0.10 = 0.17$

Z Y comparison $\frac{|65° - 75°|}{75°} + \frac{|45mm - 60mm|}{60mm} + \frac{|20mm - 30mm|}{30mm} = 0.13 + 0.25 + 0.33 = 0.71$

Figure 3.3 Objectively, X is determined to be most similar to Y.

$$relative\ difference = \frac{\left| measure_1 - measure_2 \right|}{\left(\frac{measure_1 + measure_2}{2} \right)}$$

Figure 3.4 The relative difference equation.

In the case of developing a pharmacophore model there are no known correct values. The relative difference equation used in the case when there are no known values can be seen in Figure 3.4. The denominator in this relative difference equation is the average of the two values. In essence the relative difference equation for the case when there are no known correct values, says the correct values is halfway between the two measurements.

Normally the relative difference between a positive number and a negative number would not be calculated, because the numbers normally compared using relative difference are magnitudes, non-vector, directionless quantities. However, it would be useful in comparing distance space descriptions of molecules to be able to define a plane through the molecules, take measurements from a common point in the molecules to the plane, define which side of the plane the point is on, and have the comparison of points on opposite sides of the plane result in a high (bad) score. In the graphs, the measurements are one and x. The calculated relative difference of one and x is y. The calculated value of y corresponds to the values of the relative differences over the range of x on the graph. Looking ahead at the eventual summing of values to create an overall similarity score, unlike Figure 3.5, the summed values need to be positive.

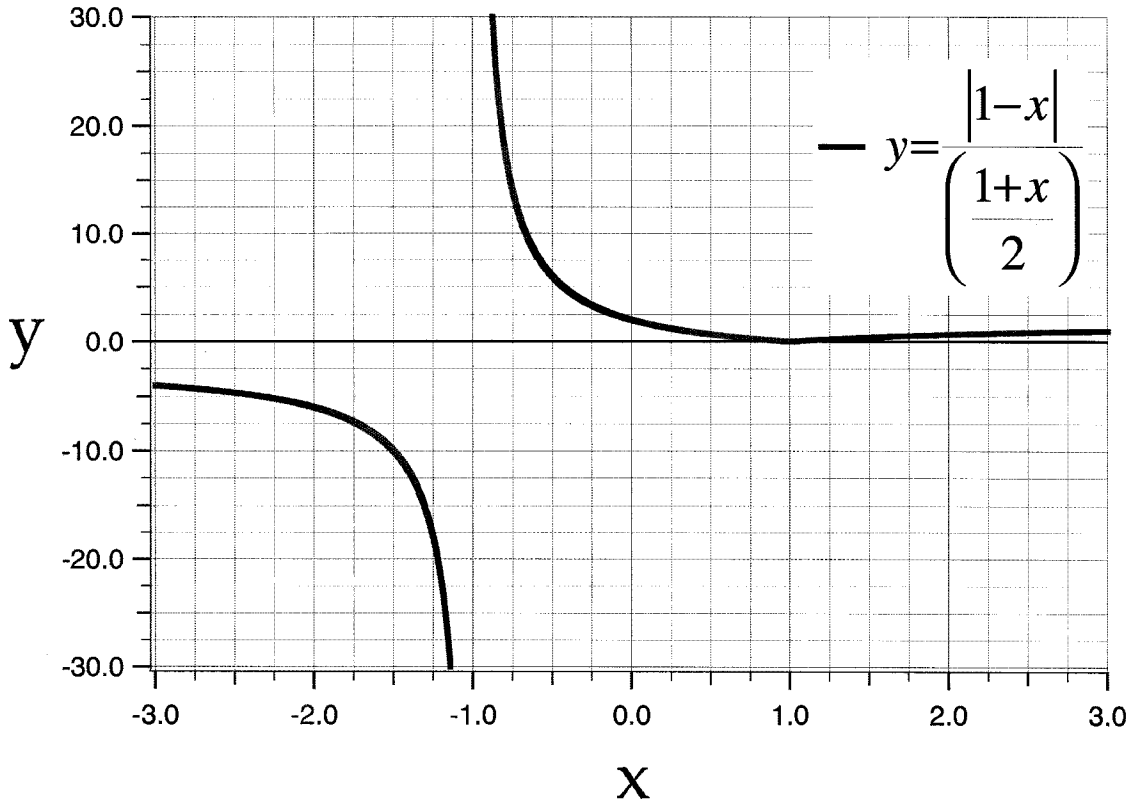


Figure 3.5 A sample graph of the relative difference equation for x in the range of -3.0 to 3.0. Figure 3.5 shows the comparison between positive and negative can equal positive, negative, or undefined values.

Taking the absolute value of the denominator (Figure 3.6) solves the problem of calculating negative y values. However, a problem still exists when comparing positive and negative values, a vertical asymptotes occurs when the denominator equals zero. While the denominator equaling zero may not have a high likelihood, the regions of high relative difference on either side of the asymptote are encountered on occasion. Figure 3.6, shows that the comparison of negative and positive values will always have worse score than when positive values are compared. It is possible for a negative and positive value to be closer together than two positive values, though the relative difference equation would not indicate this. As well, the points between -0.5 and -2.0 score much worse

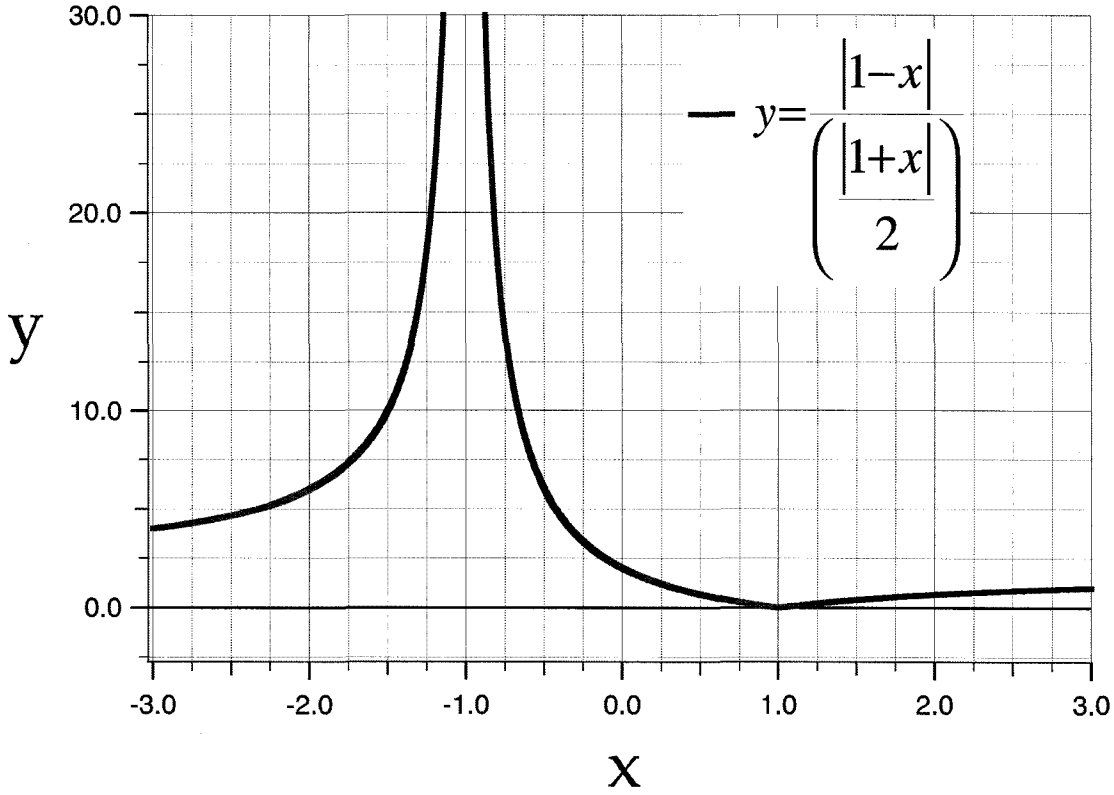


Figure 3.6 A sample graph of a modified relative difference equation that takes the absolute value of the denominator. The calculated value of the modified relative difference y for 1 and x over the range of -3.0 to 3.0, has shown that the comparison of positive and negative can equal an undefined or high value. The value y when $x \leq 0$ will always be higher than the value y when $x \geq 0$. It is indicated by the equation above that 1 and -3 or more similar than 1 and -0.75.

than values that are much further to the left.

One solution would be to assign an equal penalty, or weight, to all comparisons of a positive value with zero or a negative value. Another solution would be to take the absolute value individually for each member of the denominator (Figure 3.7) calculates the value two when one measurement is positive and the other measurement is equal to zero, or is negative (Figure 3.8). The modified relative difference equation shown in Figure 3.7 still calculates the same value as the relative difference equation for the cases when both values

are either positive and negative. Besides having one simple equation that works in the cases of interest, the penalty value two is only slightly higher, than the highest relative difference (Figure 3.4) that can be calculated for two positive values (Figure 3.9). Most importantly, this modified relative difference equation

$$modified\ relative\ difference = \frac{\left|measure_1 - measure_2\right|}{\left(\frac{\left|measure_1\right| + \left|measure_2\right|}{2}\right)}$$

Figure 3.7 The modified relative difference equation.

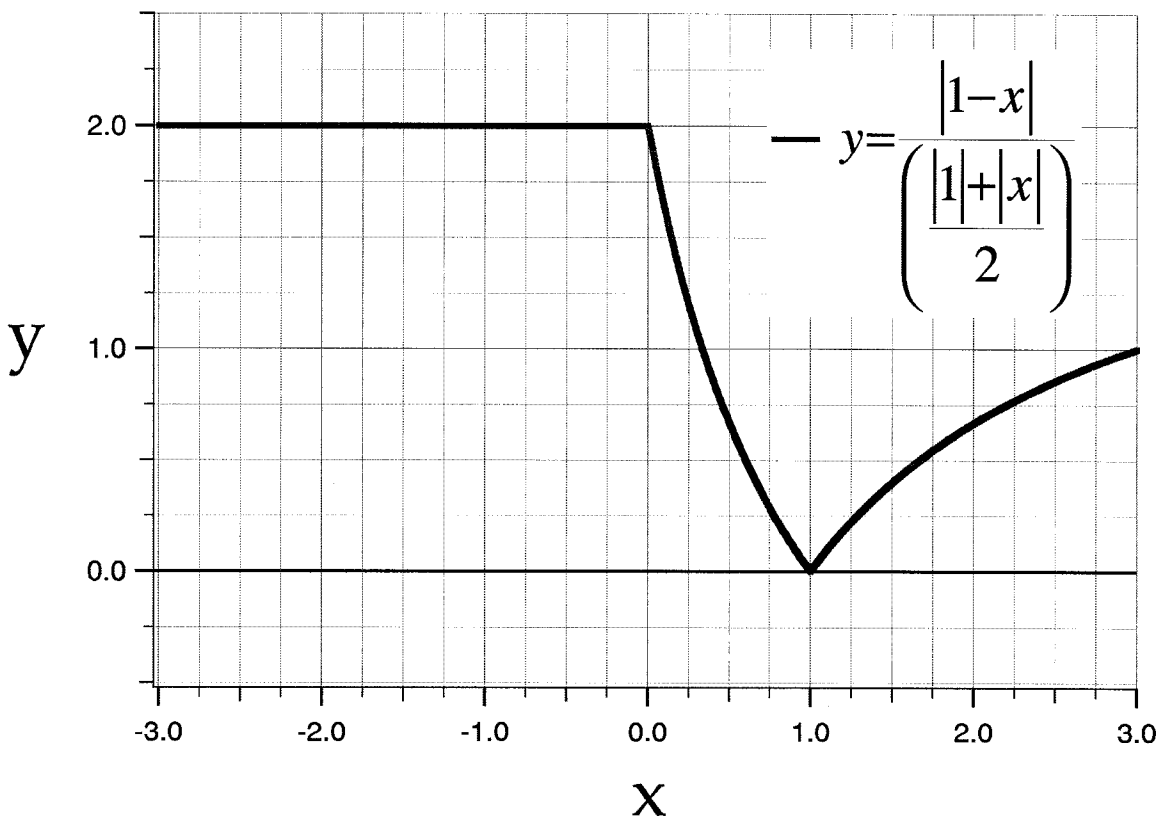


Figure 3.8 A sample graph of the modified relative difference equation. The modified relative difference y for one measurement of 1 and one measurement of x over the range of -3.0 to 3.0, is shown. When $x \leq 0$, the modified relative difference y equals 2.

$$\text{when } x>1,\ |1-x| = x-1$$

$$\lim_{x\to\infty} \frac{|1-x|}{\left(\frac{1+x}{2}\right)} = 2\left(\frac{x-1}{x+1}\right) = 2$$

Figure 3.9 The limit of the relative difference between one and x, as x approaches infinity, equals two. The relative difference between two measurements, larger than zero, is less than or equal to two.

can be used to objectively determine the similarity between objects (Figure 3.10). The quality and the quantity of distance space descriptors will have an effect on wether or not the similarity score indicates the similarity of objects.

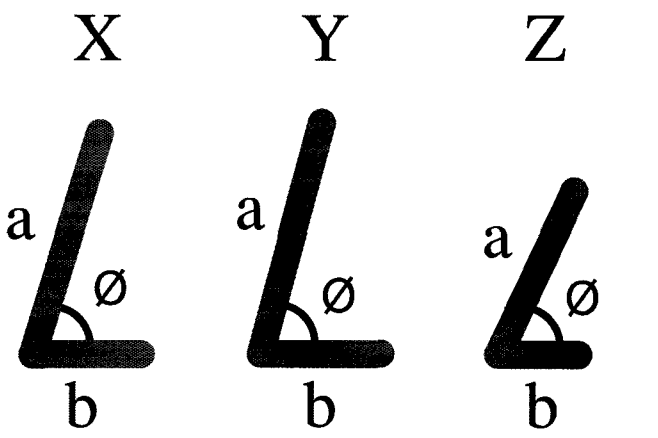
Conclusion

To summarize, a modified relative difference equation (Figure 3.7) has been created which calculates a relative difference value equal to the traditional method when comparing values that are either both positive or both negative. However, when one value is positive and one is negative, or zero, the difference score is set to the value of two. Sybyl measures distances as both positive and negative values. Whether a value is positive or negative is seemingly arbitrarily assigned by Sybyl. In this case, the absolute value of the data must be taken before the data is used in order to produce meaningful results. It must also be

Which two object are the most similar?

Subjectively, in this example, one should be able to see that object X and object Y are the most similar.

Objectively, this similarity between objects can be determine by using relative difference. The calculations using the modified relative difference equation shown in figure 3.7 can be seen at the bottom of this figure. Lower values indicate greater similarity. The calculations show that object X and object Y are the most similar.



X Y comparison $\frac{|72°-75°|}{\left(\frac{|72°|+|75°|}{2}\right)}+\frac{|58mm-60mm|}{\left(\frac{|58mm|+|60mm|}{2}\right)}+\frac{|27mm-30mm|}{\left(\frac{|27mm|+|30mm|}{2}\right)}=\frac{3°}{73.5°}+\frac{2mm}{59mm}+\frac{3mm}{28.5mm}=0.04+0.03+0.11=0.18$

X Z comparison $\frac{|72°-65°|}{\left(\frac{|72°|+|65°|}{2}\right)}+\frac{|58mm-45mm|}{\left(\frac{|58mm|+|45mm|}{2}\right)}+\frac{|27mm-20mm|}{\left(\frac{|27mm|+|20mm|}{2}\right)}=\frac{7°}{68.5°}+\frac{13mm}{51.5mm}+\frac{7mm}{23.5mm}=0.10+0.25+0.30=0.65$

Y Z comparison $\frac{|75°-65°|}{\left(\frac{|75°|+|65°|}{2}\right)}+\frac{|60mm-45mm|}{\left(\frac{|60mm|+|45mm|}{2}\right)}+\frac{|30mm-20mm|}{\left(\frac{|30mm|+|20mm|}{2}\right)}=\frac{10°}{70°}+\frac{15mm}{52.5mm}+\frac{10mm}{25mm}=0.14+0.29+0.40=0.83$

Figure 3.10 Objectively, X and Y are determined to be most the most similar objects.

determined if it would be best to move the entire data set into the positive domain by added a value to each datum. The ability to add a penalty when points are defined to be on either side of an arbitrary origin can be useful for determining similarity. The penalty could be any value. Defining the penalty as two will add to the overall similarity score (larger score, less similar) while keeping the relative differences of the other measurements from being overpowered.

The ability to calculate a score (Figure 3.11) that depicts the lack of similarity when comparing positive and negative values has proven useful in comparing low energy conformations of SSRIs for similarity (appendix 3.5). Initial results of the modeling exercise were ambiguous. Random conformational clusters with decreasing similarity were visually inspected and the calculated decreasing similarity was not visually apparent. By using the penalty with distances measured on either side of a plane defined through the molecules, the results of the similarity scores were no longer ambiguous.

$$\text{similarity score} = \frac{1}{d_{measures} \times n_{rdc}} \sum_{1}^{d_{measures}} \sum_{i,j} \frac{\left|V_k^i - V_k^j\right|}{\left(\frac{\left|V_k^i\right| + \left|V_k^j\right|}{2}\right)}$$

where $1 \leq i < j \leq$ number of molecules

n_{rdc} = number of relative difference calculations within a conformational comparison group

$d_{measures}$ = number of distance space descriptors

V_k^{α} = measure k of molecule α (distances, angles, etc.)

score = the calculated similarity of one conformational comparison group

conformational comparison group = one low energy conformation of each molecule

In this case,

number of molecules = 4

$$n_{rdc} = \binom{4}{2} = 6$$

$d_{measures} = 15$

Figure 3.11 The similarity score equation using the modified relative difference equation. The similarity score equation can be used for calculating the similarity between the low energy conformations of multiple molecules.

All functions were graphed using Igor Pro.

WaveMetrics, Inc., Lake Oswego, Oregon, www.wavemetrics.com

References

Blumenthal, L.M. *Theory and applications of distance geometry*. Chelsea Publishing, Bronx, 1970.

Gundertofte, K., Bøgesø, K.P., and Liljefors, T. A stereoselective pharmacophore model of the serotonin re-uptake site. In: *Computer-assisted lead finding and optimization*, Waterbeemd, H., Testa, B., and Folkers, G., Eds., VHCA, Basil, and Wiley-VHC, Weinheim, 1997, pp. 445-459.

Mottola, D.M., Laiter, S., Watts, V.J., Tropsha, A., Wyrick, S.D., Nichols, D.E., and Mailman, R.B. Conformational analysis of D_1 dopamine receptor agonists: pharmacophore assessment and receptor mapping. *J. Med Chem.* 1996, **39**, 285-296.

Rupp, A., Kovar, K., Beuerle, G., Ruf, C., and Folkers, G. A new pharmacophoric model for 5HT reuptake-inhibitors: differentiation of amphetamine analogues. *Pharma. Acta Helv.* 1994, **68**, 235-244.

Tripos, Inc., St. Louis, Missouri, www.tripos.com

Wilson, J.D. Physics Laboratory Experiments. D.C. Heath and Company, Lexington, 1986, pp. 9-10.

Implementation of Programs to Efficiently Calculate the Similarity Score for Large Data Sets

Introduction

The methodology for developing a pharmacophore model based upon multiple molecules described in the previous chapters is based upon the combinatorial comparison of low energy conformations. The number of comparisons required by this methodology can only be accomplished through the development of custom software. After a quick introduction to the modeling exercise, the rest this chapter will sequentially in chronological order cover the custom software developed for accomplishing the comparisons.

There can exist multiple molecules which both have a high affinity and selectivity for a single binding domain. A set of low energy conformations is one low energy conformation of each of the selected molecules that binds to this domain. The set of low energy conformations which are the most similar on three-dimensional (3D) space are representative of each molecules conformation at the time of the binding event. In this study, low energy conformations were defined as those conformations with a conformational energy of less than or equal to the global energy minima plus 30 kcals per mol (Rupp 1994, Nicklaus 1995). The binding domain of interest was the serotonin selective reuptake inhibitor (SSRI) binding domain at the serotonin transporter (SERT). The superposition of the most similar low energy conformations of select SSRIs

provided a good basis for a 3D model of the SSRI binding domain at the SERT.

The combinatorial comparison of all low energy conformations between the molecules, that is necessitated by this approach to developing a 3D pharmacophore model, can produce data sets that are too large to be analyzed using typical solutions which are limited by computer memory. Typical solutions effected by this problem include spreadsheet programs such as Excel (Microsoft), and, for Tripos users, solutions which use the Sybyl Programming Language (SPL) (Tripos). The combinatorial comparison of all low energy conformations compares each low energy conformation with all other low energy conformations of the other molecules. A conformational comparison group can be defined as consisting of one low energy conformation of each molecule. In the work presented here, a conformational comparison group consists of one low energy conformation from each of the following SSRI's: escitalopram, indatraline, MCN-5652, and sertraline. The total number of conformational comparison groups is equal to the product of the number of low energy conformations of each molecule. For the SSRI SERT data the product of 233 low energy conformations of escitalopram, 29 low energy conformations of indatraline, 12 low energy conformations of MCN-5652 and 16 low energy conformations of sertraline equals 1,297,344 conformational comparison groups.

To determine similarity, every low energy conformation of each molecule is described using multiple distance space descriptors (distances and angles). These distance space descriptors are then compared to determine similarity. If the distance space descriptors are adequate then every low energy conformation

will have a distinct description. Within each conformational comparison groups a set of combinatorial comparisons must take place for each one of the distance space descriptor. These comparisons are accomplished using relative difference. The equation in Figure 4.1 shows how the comparisons take place and the similarity score is generated for each conformational comparison group. The lower the similarity score the more similar the conformational comparison group is in 3D space.

It is easy to see how the ensuing data set could outgrow the capabilities of a prepackaged solution, such as Excel, and lead to the need for a custom solution. The desire to extend the methodology for developing 3D

$$\text{similarity score} = \frac{1}{d_{measures} \times n_{rdc}} \sum_{1}^{d_{measures}} \sum_{i,j} \frac{\left|V_k^i - V_k^j\right|}{\left(\frac{\left|V_k^i\right| + \left|V_k^j\right|}{2}\right)}$$

where $1 \leq i < j \leq$ number of molecules

n_{rdc} = number of relative difference calculations within a conformational comparison group

$d_{measures}$ = number of distance space descriptors

V_k^{α} = measure k of molecule α (distances, angles, etc.)

score = the calculated similarity of one conformational comparison group

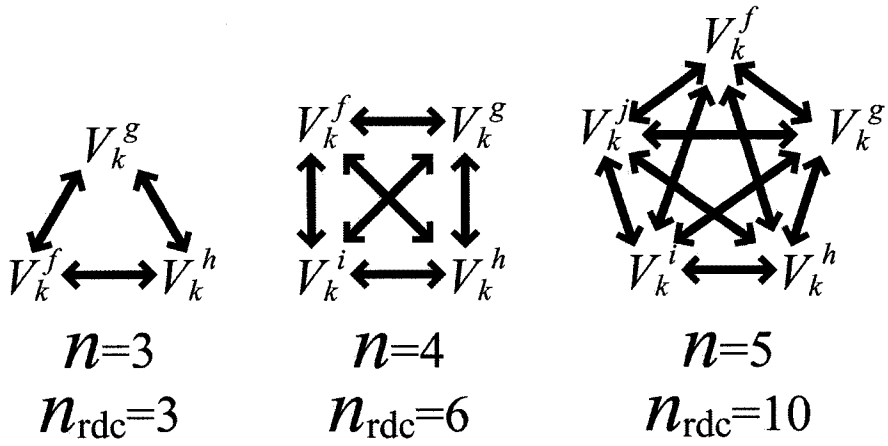
conformational comparison group = one low energy conformation of each molecule

Figure 4.1 The modified relative difference equation used in the programs.

pharmacophore models beyond the SSRI binding domain at the SERT, led to the need for the custom solution to be generalized in order to work in other cases. For instance, the more generalized solution would be able to handle changes in the number of descriptors, molecules and low energy conformations. After discussing the algorithm for calculating similarity scores the path to the current solution, from SPL to Perl to the C programming language, will be covered.

Development and Implementation of Similarity Score Calculating Programs

Referencing the equation shown in Figure 4.1, the relative difference is calculated for every distance space descriptor between each low energy conformation in a conformational comparison group. The number of relative difference calculations per descriptor increases following an arithmetic series (Figure 4.2). As the number of molecules increases the number relative difference calculation significantly increases. The algorithm calculates all of the combinatorial relative differences for a descriptor, sums the relative differences and divides this sum by the number of combinatorial relative difference calculations. This value can be thought of as an intermediate score and is produced for each distance space descriptor. Each one of the combinatorial relative differences, intermediate scores, is summed and divided by the number of distance space descriptors to produce the similarity score for a conformational comparison group.



V_k^{α} = measure k of molecules α

$\longleftrightarrow$ = one relative difference calculation (rdc)

n = number of molecules

$$n_{rdc} = \begin{matrix}\text{total number of relative} \\ \text{difference calculations per} \\ \text{distance space descriptor}\end{matrix} = \sum_{i=1}^{n-1} i = \frac{(n\text{-}1)(n)}{2}$$

Figure 4.2 The number of non-repetitive combinatorial comparisons for each distance space descriptor follows an arithmetic mean.

In the implemented algorithm, after the similarity score has been calculated for a conformational comparison group, the last molecule in the set of molecules being compared is incremented to the next low energy conformation. The similarity score is calculated for this new conformational comparison group and the last molecule, again, increments to the next low energy conformation. Once the last low energy conformation of the last molecule is reached, the last molecule is reset to its first low energy conformation and the second to the

last molecule increments to its next low energy conformation. It is easiest to think of the action of a mechanical odometer to understand this process, which repeats until a similarity score has been calculated for all combinatorial of conformational comparison groups.

In the case of a four molecule comparison, as was done here, after the first similarity score is calculated the fourth molecule is incremented to the next low energy conformation and the next similarity score proceeds to be calculated. When the final low energy conformation of the fourth molecule is reached during this increment and calculate process, the fourth molecule resets back to its first low energy conformation and the third molecule increments to its next low energy conformation. After the ensuing similarity score calculation the fourth molecule increments again to its next low energy conformation. This process repeats until all four molecules have reached there last low energy conformation.

The original work for this project, searching conformational space and developing molecular spreadsheets which contained the low energy conformations and the distance space descriptors, was done in Sybyl. It follows that the first implementation of the algorithm described above would be in SPL. One benefit of SPL is, it can work directly with the Sybyl molecular spreadsheet files. The SPL program (appendix 4.1) read the data from four Sybyl molecular spreadsheets, performed the calculations, and placed the results into a fifth molecule spreadsheet. The program was tested and worked on small data sets. Due to Sybyl molecular spreadsheets having a large footprint in memory, and with the SSRI SERT data set consisting of 1,297,344 conformational comparison

groups, after running for hours the scratch disk on the SGI server would fill to capacity and the computer would stop running before the program completed. The other drawback with the SPL program was even though it handled various numbers of descriptors and low energy conformations, it was written to only work with four Sybyl molecular spreadsheets. Explicitly, it could only calculate a similarity score for four molecules, no more or no less.

A Perl script was written which dealt with the memory problem but still only worked with four molecules (appendix 4.2). The Perl script reads data from comma delimited text files, which required the Sybyl molecular spreadsheets to be exported as comma delimited text files. Fortunately, Sybyl provides a mechanism for exporting molecular spreadsheets as comma delimited text files. Each row in the exported text files contains the row name from the Sybyl molecular spreadsheet and is unique for each low energy conformation. The label is followed in comma delimited form by all of the column data for that row.

The Perl script reads a row from each of the four data files into memory, performs the calculations, and write the results into a new file. This row by row approach uses very little memory but results in constant file reading and writing. From a performance stand point, reading and writing to files is always one of the slowest routines on a computer.

The comma delimited text file produced by the Perl script contains the four labels followed by the results of the combinatorial relative difference calculation for each descriptor and lastly the similarity score. The text file produced by the Perl script from SSRI SERT data set contained 1,297,344 rows,

where each row representing one possible conformational comparison group. Each row contained four labels, 15 intermediate scores and one similarity score. On a 2003, 867 MHz G4 12 inch Apple Powerbook, the Perl script takes 13.5 minutes to run and produces a file approximately 510 megabyte (MB) in size. Within this 510 MB comma delimited text file, the nearly 1.3 million rows are not in any particular order with regards to the similarity scores.

A lower numeric similarity score is representative of a conformational group which is more similar in 3D space. Therefore, it would be helpful to sort the conformational groups in ascending order according to similarity score. Two ways of sorting a data set of this size include, 1) using the UNIX sort command, and 2) using Igor Pro (Wavemetrics). The advantage Igor Pro has is it will produce a tabular view and a graph of the sorted data.

Though, neither the SPL program or Perl script sort the data, the Perl script has an advantage in that it would run to completion. Similar to the SPL program, the Perl script could readily deal with a change in the number of descriptors and low energy conformations (the number of rows and columns in the data files) read from the data files. However, the Perl script still could not deal with a differing number of input files. Though, at one point, the script was physically modified to read three data files and perform a three molecule comparison (appendix 4.3).

The desire to produce a platform independent program which would calculate a similarity score for any number of molecules and and have an improvement in performance led to the development of the C program currently

being used in our lab. Preliminary work was done using a different algorithm for storing calculated data, accepted any number of data files for comparison, dynamically allocated memory at run time, and was written in both serial and parallel forms. This preliminary set of programs, which can be read about in appendix 4.4, was written to explore dynamic memory allocation and parallel computing ideas using the C programming language, message passing interface (MPI), and OpenMP.

This history led to the following list of requirements for the current program:

1) the capability to use any number (3 or more) data files for input
2) dynamically allocate memory at run time,
3) calculate all combinatorial similarity scores,
4) sort all similarity scores in ascending order,
5) if necessary sort similarity score out of core,
6) potential for easy cross platform implementation,
7) potential for easy implementation of multiprocessing, and
8) better performance than previously achieved through Perl scripting.

The potential for cross platform compatibility and multiprocessing were the two reasons the C programming language was chosen as the language of choice for this program. The C language provides performance, cross platform

compatibility through the gcc compiler, and support within the MPI community. Fortran would have been another good choice due to its inherent calculation performance and the provisions in MPI implementations for Fortran. The lack of free compilers, such as gcc, for Fortran 90 or newer across multiple platforms was the reason Fortran was not chosen.

In the C program, extensive use of C pointers were required in order to deal with dynamic memory calculations, allow for faster sorting and allow for more logical data and storage indexing. The program dynamically allocates memory, at run time, for storing the information read from the input data files. Storing the data required for the relative difference calculations in memory, instead of reading it from a file as needed, decreases data access time improving the performance of the program. It is not always possible to store the calculated scores (the output data) in memory. There may be much more output data than available memory. The user is given control over how much of the output data will be stored in memory. The program then allocates the appropriate amount of memory. This is handled in three different ways in the three C programs which were written. Before going into the differences between the three programs, it is important to talk about the implementation of the similarity score calculations (Figure 4.1), which is the same in all three programs.

Calculation of the similarity score is accomplished identically in all three C programs. The similarity score for one conformational comparison group is accomplished with three nested loops. The inner two most loops cycle through the combinatorial relative difference calculations for one distance

space descriptor. The outer loop cycles though the descriptors and calculates the intermediate scores. These three loops are are nested inside a fourth loop which controls the order the low energy conformations are combined to create conformational comparison groups.

In essence, the forth loop combinatorially builds the combinatorial comparison groups. The first time through the loop, the first low energy conformation of each molecule is used to make up the conformational comparison group. The second time through the loop the last molecule is incremented to its next low energy conformation creating the next conformational comparison group. In this case, last refers to the last command line argument when the program was invoked. For example, invoking the program at the command line in the following manner escitalopram is the last molecule (forth molecule) and MCN-5652 is the first molecule: "./compsort ./ MCN-5652.txt ./sertraline.txt ./indatraline.txt ./escitalopram.txt" As explained before, the the forth loop continues repeating, the last molecule continues to iterate through its low energy conformations until its last low energy conformation is reached. At this point the last molecule resets back to its first low energy conformations and the second to the last molecule (indatraline), increments to its next low energy conformation. This process of incrementing and reseting continues until the similarity score for the conformational comparison group containing the last low energy conformation of each molecule is calculated. The process of incrementing though all combinatorial combinations of low energy conformations between the molecules is analogous

to the way a mechanical odometer works. Besides the calculation of the similarity score, concatenation of the labels of each conformational comparison group, placement of the concatenated label, intermediate scores and similarity score into an array takes place in the forth loop.

Algorithm analysis for four nested loops indicates the asymptotic maximum upper bound (O) is equal to the maximum loop iterations raised to the power of four. However, the number of iterations of the three inner loops will always remain very small in comparison with the number of iterations of the fourth loop. This means O will actually be less than four. Changing to an out of core program, where the data is not all held in main memory, the file reading and writing becomes the overshadowing slow step at runtime.

Chronologically, the first program written, compsortall (appendix 4.5), meets all of the requirements listed above. Plus, in ascending order of the similarity scores writes the labels, intermediate scores and similarity scores for all conformational comparison groups to a comma delimited text file. The program prompts the user for a number of similarity scores to calculate before sorting, allowing for the calculations to take place in main memory (in core). The program calculates the similarity scores in sets of this size until all scores have been calculated.

More specifically, three arrays of equal size are created. The size of these arrays is determined from the number of similarity scores to calculate provided by the user. Array one stores the labels, intermediate scores, and similarity scores that have just been calculated. Array two holds labels, intermediate

scores, and similarity score read from a temporary file. Array three holds the same information before it is written to a temporary file. Inside the fifth loop, the labels, intermediate scores, and similarity scores are placed in array one. When array one fills up it is sorted in ascending order of the similarity score using the quicksort algorithm (Baase, Weiss). The first time this happens, all of the information contained in array one is written to a temporary file. The second time this happens the information in the temporary file is read into array two. Array two and array one are merge sorted (Baase, Weiss) into array three. When array three fills up it is written to a second temporary file. Array three will fill twice. The first time array three is written to the second temporary file, the file will be created and written to from its beginning. The second write to the file will be appended to the end of the file. The third time array one fills up, the second temporary file will read into array two in two sets, and first temporary file will be re-created and written to in three sets. This process of calculating, quicksorting, file reading, merge sorting, and file writing continues until the last set of similarity score calculations is reached. This final set of calculations will be smaller than array one, but the same process will occur, with the exception that array three will write to the output file. The name of the output file is provided by the user at runtime. The final result, is a text file containing the concatenated labels, intermediates scores, and similarity scores, in ascending order, for all conformational comparison groups.

The implementation of the merge sort, quicksort, and file reading and writing in the program compsortall have been optimized for better performance.

The merge sort described above is a special case where both data sets being merged are already sorted and is the fastest form of the merge sort. The pivot for the quicksort routine comes from a median-of-three (Weiss) routine instead of just using the first element of the array. Using the median-of-three, the first, middle, and last elements are sampled and the median value is chosen as the pivot. Using the median value increases the chances the value of the chosen pivot is in the middle of the data set. This is important for the performance of quicksort. Experimentally, in this application, the C function *fscanf* was found to be faster than the UNIX *read* routine. In the code *read* is still being used for reading the input data files, and *fscanf* is being used to read the temporary files. The input data files a relatively small and *read* provides adequate performance in this situation. The C function *fprintf* is used to write to the temporary files and output file. This function provides a convenient way to format the text in these files.

Even with optimization, the file reading and writing necessary to produce a file containing all conformational comparisons groups sorted in ascending order, may be too time consuming for data sets larger than the SSRI SERT data set. This can be especially true during early experimentation when descriptions may not yet be developed adequately and many trials of similarity score calculation and sorting are likely to occur. To provide a less time consuming alternative, the program compsort (appendix 4.6) was written. The program compsort is very similar to compsortall in that it calculates and sorts the similarity scores for all conformational comparison groups. It differs in that

it only keeps, and subsequently writes to a file, a portion of the lowest (best) scoring conformational comparison groups.

Similar to the compsortall program the compsort program prompts the user for the number of similarity score to calculate in main memory before sorting. The program also prompt the user for the number of similarity scores to keep and write to the output file. This program only has one array which holds both the calculated scores and sorted scores. The array is initialized at the necessary size to hold both the number of similarity scores calculated in a set plus the number of scores to keep. The sorted scores which will be eventually saved are kept at the beginning of the array and the set of calculated scores take up the rest of the array. The section of the array holding the scores to be saved is initialized with a large number, 1,000,000 specifically. After the first set of similarity score calculations is complete the whole array is sorted using quicksort and the median-of-three pivot described above. The quicksort routine moves the large numbers, initialized at the beginning of the array, to the end of the array and low similarity scores to the beginning of the array. During the subsequent calculation and sort cycles the lowest scores end up at the beginning of the array. After the similarity scores have been calculated, and sorted for all conformational comparison groups, the program writes the user determined number of lowest similarity scores to keep from the beginning of the array to the output file.

The last version of the program, comp (appendix 4.7), just calculates the similarity scores for all conformational comparison groups in sets of size

determined by the user. Once a set of similarity scores have been calculated they are written to an output file. In practice, due to all similarity scores being written to the output file instead of just a subset, the program takes longer to run than compsort. The comp program can be good for small data sets where the similarity scores will be imported into a spreadsheet program, such as Igor Pro, for sorting and graphing. As well the comp program serves as basis for those people who want to write their own sorting routines or use the UNIX *sort* command.

Conclusion

In conclusion, SPL, Perl and C were used to write a total of six programs to accomplish the task of comparing low energy conformations to determine the conformational comparison group, most similar in 3D space. SPL proved inadequate for this task and Perl proved to have limited performance. C provided the best performance and lends itself to cross platform compatibility and future multiprocessing work. Figure 4.3 shows the overall runtimes for the Perl script and three C programs when calculating the similarity scores for the SSRI SERT data set.

The serial C program can be made parallel much the same was as was done in appendix 4.4. The outer loop of the four nested loops can be split among multiple processes. Little interprocess communication would need to occur. Initially each process would need a copy of the data to work on and which

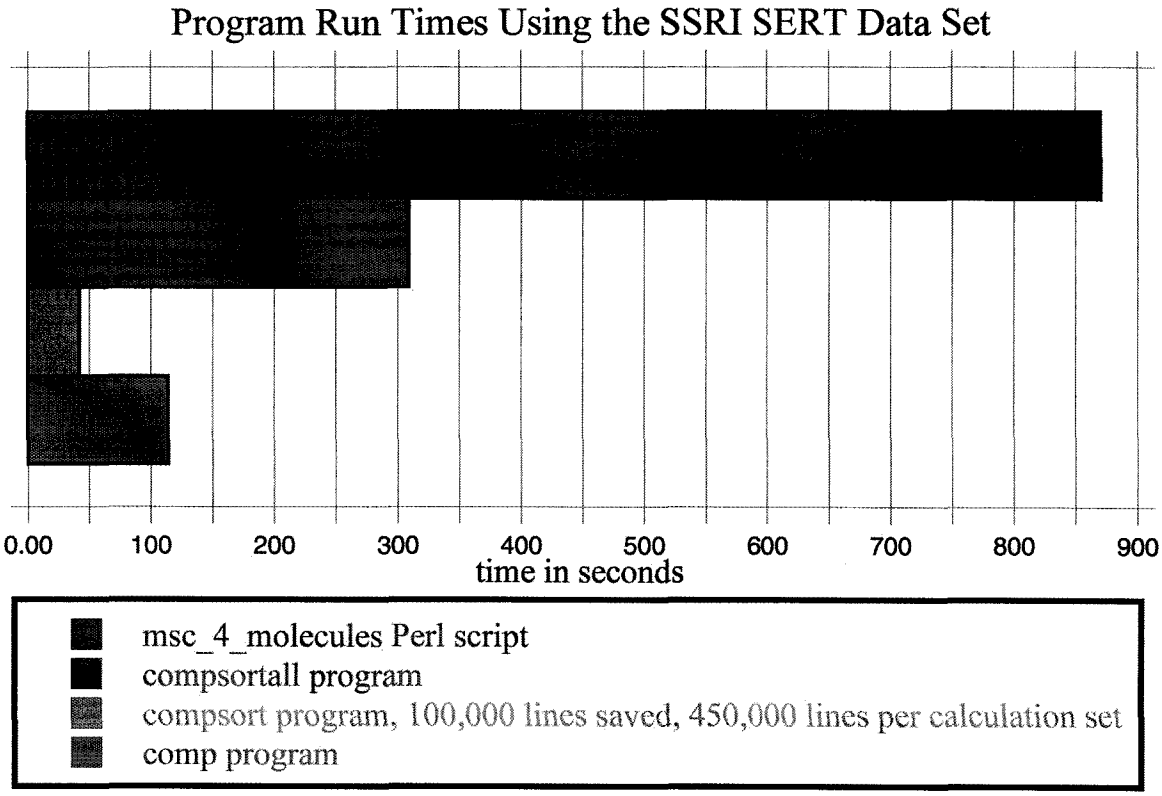


Figure 4.3 The Perl script and C program run times using the SSRI SERT data set. The programs were all run on the same 2003, Apple 867 MHz G4 PowerBook with 640 MB of RAM.

iterations of the outer loop to accomplish. The processes would then only need to communicate once more to send the calculated similarity score to process zero. The final communication would have to broken up into many small messages in the case MPI due to the limited size of messages. Unbuffered messages or MPI-2 may solve this issue.

In Mac OS X a C variable of type double is eight bytes and a variable of type char is one byte, the size of the similarity score calculation, read and write arrays used in the compsortall program can be calculated. On a 867 MHz G4 12 inch Apple Powerbook with 640 MB of RAM running Mac OS 10.3.5, for the

SSRI SERT data set virtual memory will start to be used when the arrays reach about 550,000 lines in size. When this happens performance degrades. The time required for reading and writing/appending the temporary files are shown in Figure 4.4, along with the over all run times for the SSRI SERT data at varying array sizes. One thing that is clear from this graph is the impact that one less out of core sorting cycle at 650,000 lines per array will have on run time, even though virtual memory is being used at this point. The file containing all of the concatenated labels, intermediate scores, and similarity scores for the SSRI SERT data set sorted in ascending order is 453.6 MB in size. It is clear calculating similarity scores for every conformational comparison group, even for data sets much larger than the SSRI SERT data set, is practical. However, if all similarity scores are to be sorted and saved in to a file in a timely manner, both large and fast, possible parallel, file systems will be required.

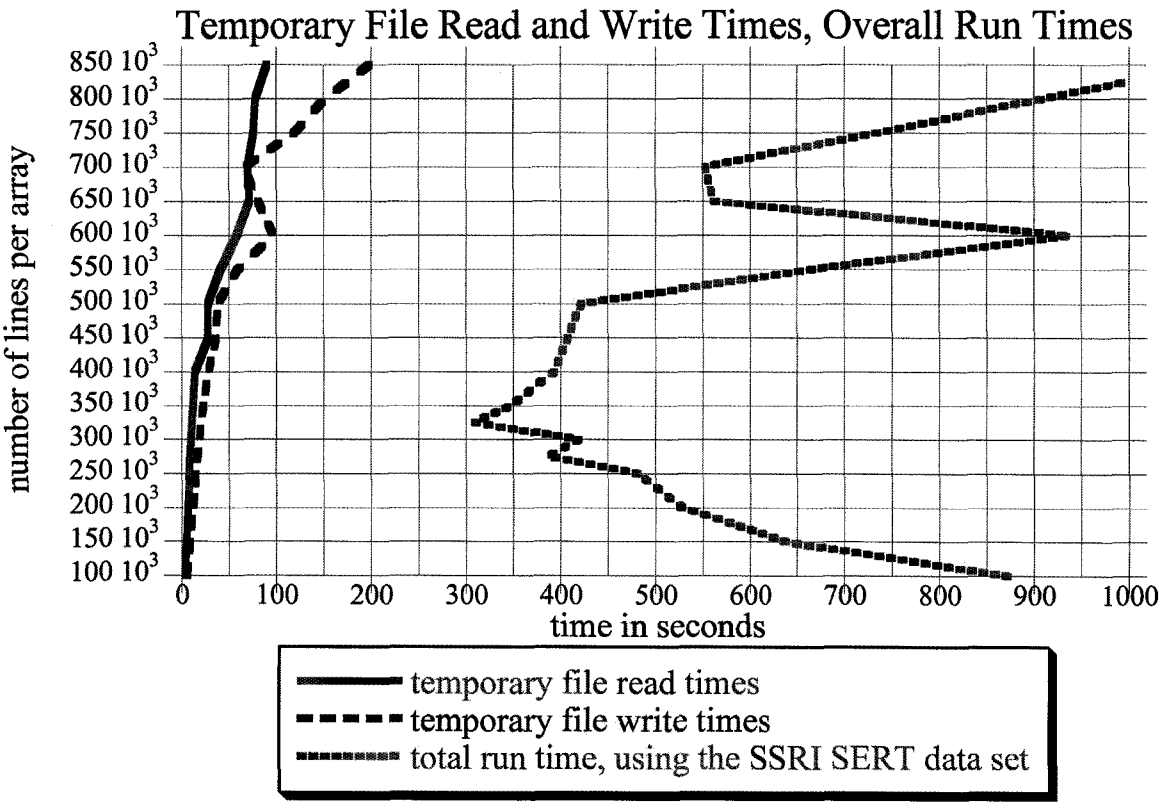


Figure 4.4 The read, write, and overall run times for the compsortall at various array sizes. The data is good for looking at general trends, but the exact number will vary from run to run depending on memory and cache loading in Mac OS X. This points to why a balance between theory and experiment is desirable. The program was run on the a 2003, Apple 867 MHz G4 PowerBook with 640 MB of RAM.

References

Baase, S., Van Gelder, A. *Computer Algorithms Introduction to Design and Analysis.* Addison-Wesley, Menlo Park, CA, 2000

Igor Pro, WaveMetrics, Inc., Lake Oswego, Oregon, www.wavemetrics.com

Microsoft Corporation, Redmond, Washington, www.microsoft.com

Nicklaus, M.C., Wang, S., Driscoll, J.S., and Milne, G.W. Conformational changes of small molecules binding to proteins. *Bioorg. Med. Chem.* 1995, **3**, 411-428.

Rupp, A., Kovar, K., Beuerle, G., Ruf, C., and Folkers, G. A new pharmacophoric model for 5HT reuptake-inhibitors: differentiation of amphetamine analogues. *Pharma. Acta Helv.* 1994, **68**, 235-244.

Tripos, Inc., St. Louis, Missouri, www.tripos.com

Weiss, M.A. *Data Structures and Algorithm Analysis in C.* Menlo Park, CA, 1997.

	distance between Ar1 centroid and Ar2 centroid *(angstroms)*	distance from Ar1 centroid to Ar2 plane *(angstroms)*	angle between Ar1 centroid, Ar2 centroid and the inner Ar2 axial C *(degrees)*	distance from Ar1 to the axial X group on the Ar1 axis *(angstroms)*	distance from Ar1 centroid to hetero atom *(angstroms)*	distance from Ar1 centroid to N *(angstroms)*
Sertraline 2.135 kcals/mol	4.822	1.924	35.900	3.153		5.949
MCN-5652 13.035 kcals/mol	4.912	1.430	33.660	3.175		5.164
Indatraline 14.532 kcals/mol	4.952	1.886	29.350	3.153		6.226
S-Citalopram 15.585 kcals/mol	4.975	2.462	30.680	3.992	3.328	7.131
average	4.915	1.925	32.398	3.368	3.328	6.117

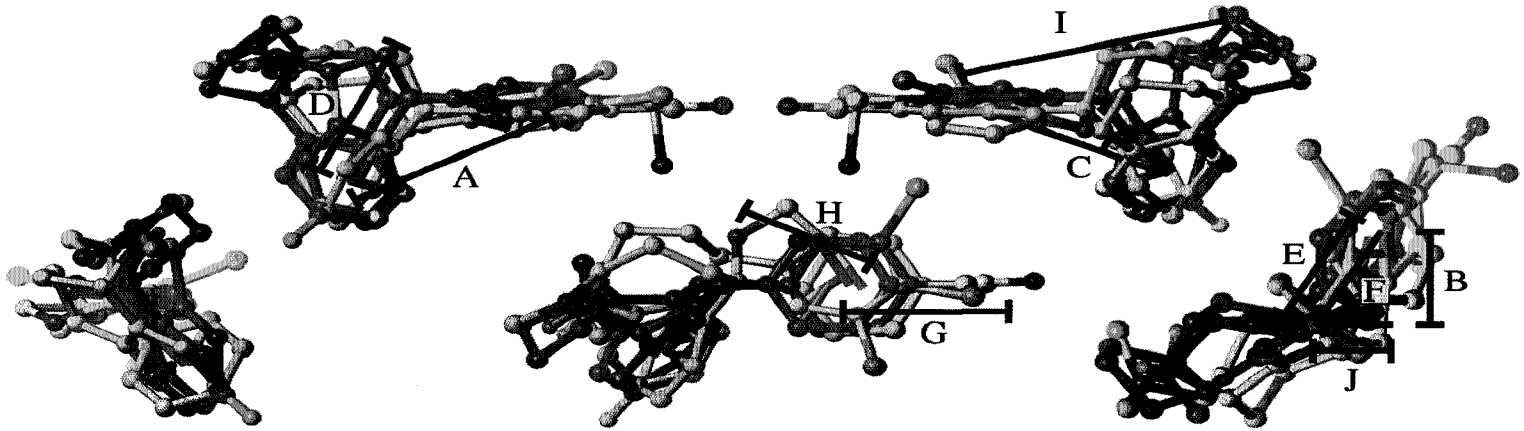
	distance between Ar1 centroid and Ar2 centroid *(angstroms)*	distance from Ar2 centroid to Ar1 plane *(angstroms)*	angle between Ar2 centroid, Ar1 centroid and the inner Ar1 axial C *(degrees)*	distance from Ar2 to the axial X group on the Ar1 axis *(angstroms)*	distance from Ar2 centroid to hetero atom *(angstroms)*	distance from Ar2 centroid to N *(angstroms)*
Sertraline 2.135 kcals/mol	4.822	2.042	35.390	7.611		3.720
MCN-5652 13.035 kcals/mol	4.912	1.985	33.110	7.762		3.734
Indatraline 14.532 kcals/mol	4.952	1.556	31.970	7.800		3.762
S-Citalopram 15.585 kcals/mol	4.975	1.905	28.420	8.696	3.713	4.618
average	4.915	1.872	32.222	7.967	3.713	3.959

	distance between N and the Ar1 centroid *(angstroms)*	distance from N to the Ar1 plane *(angstroms)*	angle between N, Ar1 centroid and the inner Ar1 axial C *(degrees)*	distance from N to the axial X group on the Ar1 axis *(angstroms)*	distance from N to the to hetero atom *(angstroms)*	distance from Ar2 centroid to N *(angstroms)*
Sertraline 2.135 kcals/mol	5.949	1.352	34.590	8.737		3.720
MCN-5652 13.035 kcals/mol	5.164	1.430	17.230	8.253		3.734
Indatraline 14.532 kcals/mol	6.226	1.150	11.360	9.342		3.762
S-Citalopram 15.585 kcals/mol	7.131	1.843	15.270	11.031	5.098	4.618
average	6.117	1.444	19.613	9.341	5.098	3.959

	distance between Ar1 centroid and hetero atom *(angstroms)*	distance from hetero atom to Ar1 plane *(angstroms)*	angle between hetero atom, Ar1 centroid and the inner Ar1 axial C *(degrees)*	distance from hetero atom to the axial x group on the Ar1 axis *(angstroms)*	distance from Ar2 centroid to hetero atom *(angstroms)*	distance from hetero atom to N *(angstroms)*
Sertraline 2.135 kcals/mol						
MCN-5652 13.035 kcals/mol						
Indatraline 14.532 kcals/mol						
S-Citalopram 15.585 kcals/mol	3.328	0.489	31.490	7.051	3.713	5.098
average	3.328	0.489	31.490	7.051	3.713	5.098

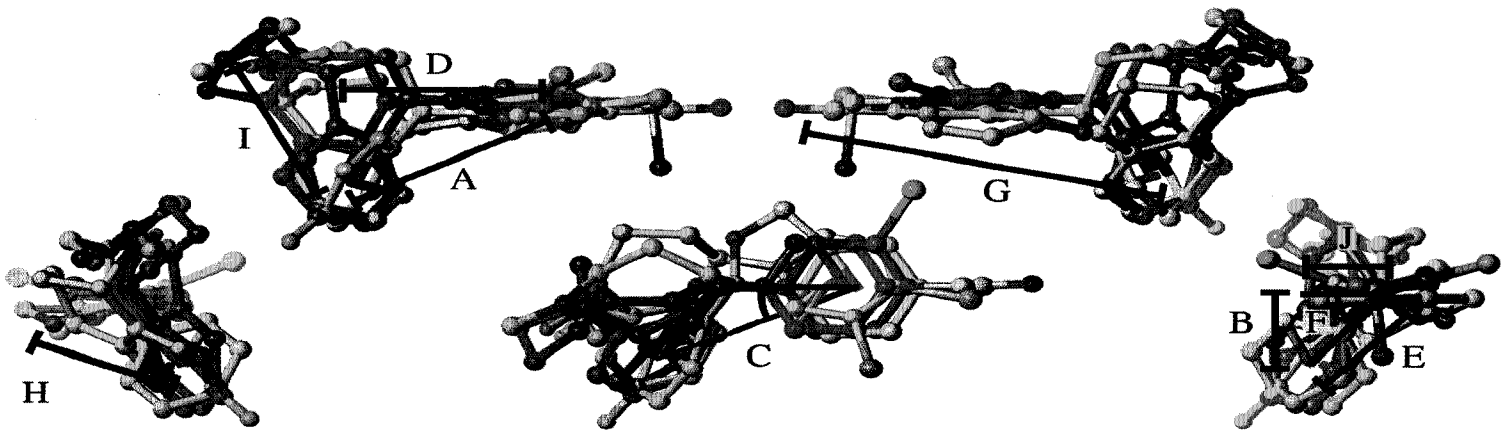
	torsion angle Ar2 plane to Ar1 (front side outer C, Ar2 outer axial C, Ar2 inner axial C, Ar1 centroid) *(degrees)*	torsion angle Ar1 plane to Ar2 (front side outer C, Ar1 outer axial C, Ar1 inner axial C, Ar2 centroid) *(degrees)*	torsion angle Ar1 plane to N (front side outer C, Ar1 outer axial C, Ar1 inner axial C, N) *(degrees)*	torsion angle Ar1 plane to hetero (front side outer C, Ar1 outer axial C, Ar1 inner axial C, hetero atom) *(degrees)*
Sertraline 2.135 kcals/mol	42.000	46.800	23.600	
MCN-5652 13.035 kcals/mol	31.800	47.700	69.300	
Indatraline 14.532 kcals/mol	50.900	36.300	73.200	
S-Citalopram 15.585 kcals/mol	81.700	53.600	78.400	16.400
average	51.600	46.100	61.125	16.400

Appendix 2.1 Measurements of the three-dimensional (3D) pharmacophore model of the serotonin selective reuptake inhibitor (SSRI) binding domain at the serotonin transporter (SERT).



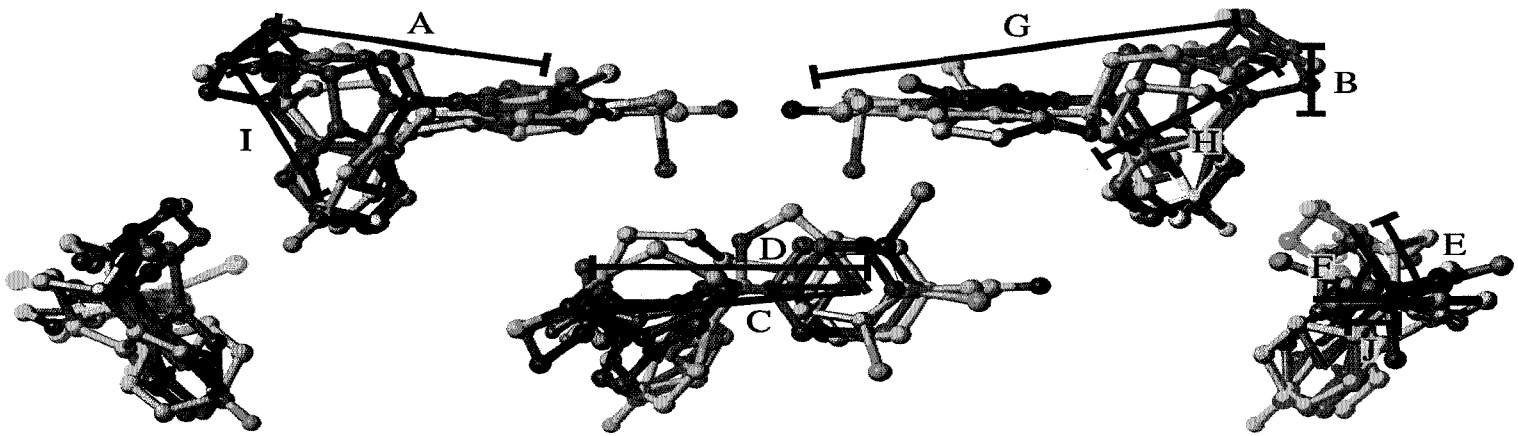
	A distance between Ar1 centroid and Ar2 centroid *(angstroms)*	B distance from Ar1 centroid to Ar2 plane *(angstroms)*	C angle between Ar1 centroid, Ar2 centroid and the inner Ar2 axial C *(degrees)*	$D = A \times \cos(C)$ distance along the Ar2 axis from the Ar2 centroid to a point which is normal to the Ar1 centroid *(angstroms)*	$E = \sqrt{E^2 - A^2}$ distance from the point on on the Ar2 axis which is normal to the Ar1 centroid, to the Ar1 centroid *(angstroms)*	$F = \arccos\left(\frac{B}{E}\right)$ angle from the point on the Ar2 plane which is normal to the Ar1 centroid, a point on the Ar2 axis which is normal to the Ar1 centroid, and the Ar1 centroid *(degrees)*
Sertraline 2.135 kcals/mol	4.822	1.924	35.900	3.906	2.827	42.880
MCN-5652 13.035 kcals/mol	4.912	1.430	33.660	4.088	2.723	31.685
Indatraline 14.532 kcals/mol	4.952	1.886	29.350	4.316	2.427	50.989
S-Citalopram 15.585 kcals/mol	4.975	2.462	30.680	4.279	2.538	75.902
average	4.915	1.925	32.398	4.147	2.629	50.364

	G distance from Ar1 centroid to the axial X group on the Ar1 axis *(angstroms)*	H distance from Ar1 centroid to hetero atom *(angstroms)*	I distance from Ar1 centroid to N *(angstroms)*	$J = \sqrt{F^2 - B^2}$ distance from the point on the Ar2 axis which is normal to the Ar1 centroid to the point on the Ar2 plane which is normal to the Ar1 centroid *(angstroms)*	F same angle as above via SYBYL torsion measurement (front side outer C, Ar2 outer axial C, Ar2 inner axial C, Ar1 centroid) *(degrees)*
Sertraline 2.135 kcals/mol	3.153		5.949	2.072	42.000
MCN-5652 13.035 kcals/mol	3.175		5.164	2.317	31.800
Indatraline 14.532 kcals/mol	3.153		6.226	1.528	50.900
S-Citalopram 15.585 kcals/mol	3.992	3.328	7.131	0.618	81.700
average	3.368	3.328	6.117	1.634	51.600



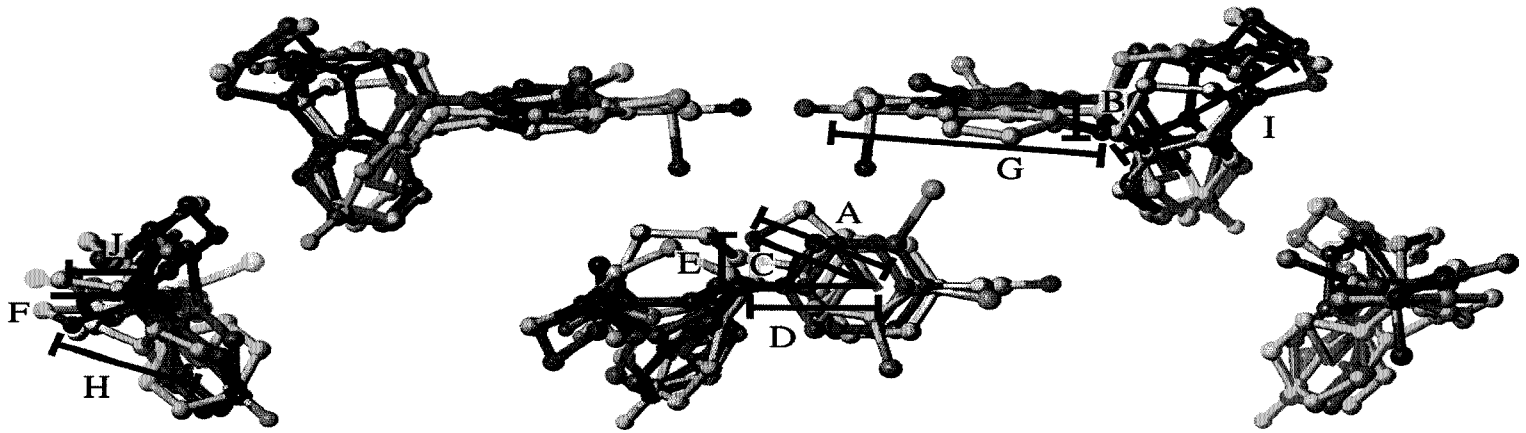
	A distance between Ar1 centroid and Ar2 centroid *(angstroms)*	B distance from Ar2 centroid to Ar1 plane *(angstroms)*	C angle between Ar2 centroid, Ar1 centroid and the inner Ar1 axial C *(degrees)*	$D = A \times \cos(C)$ distance along the Ar1 axis from the Ar1 centroid to a point which is normal to the Ar2 centroid *(angstroms)*	$E = \sqrt{E^2 - A^2}$ distance from the point on on the Ar1 axis which is normal to the Ar2 centroid, to the Ar2 centroid *(angstroms)*	$F = \arccos\left(\frac{B}{E}\right)$ angle from the point on the Ar1 plane which is normal to the Ar2 centroid, a point on the Ar1 axis which is normal to the Ar2 centroid, and the Ar2 centroid *(degrees)*
Sertraline 2.135 kcals/mol	4.822	2.042	35.390	3.931	2.793	46.988
MCN-5652 13.035 kcals/mol	4.912	1.985	33.110	4.114	2.683	47.714
Indatraline 14.532 kcals/mol	4.952	1.556	31.970	4.201	2.622	36.402
S-Citalopram 15.585 kcals/mol	4.975	1.905	28.420	4.375	2.368	53.568
average	4.915	1.872	32.222	4.155	2.616	46.168

	G distance from Ar2 centroid to the axial X group on the Ar1 axis *(angstroms)*	H distance from Ar2 centroid to hetero atom *(angstroms)*	I distance from Ar2 centroid to N *(angstroms)*	$J = \sqrt{F^2 - B^2}$ distance from the point on the Ar1 axis which is normal to the Ar2 centroid to the point on the Ar1 plane which is normal to the Ar2 centroid *(angstroms)*	F same angle as above via SYBYL torsion measurement (front side outer C, Ar1 outer axial C, Ar1 inner axial C, Ar2 centroid) *(degrees)*
Sertraline 2.135 kcals/mol	7.611		3.720	1.905	46.800
MCN-5652 13.035 kcals/mol	7.762		3.734	1.805	47.700
Indatraline 14.532 kcals/mol	7.800		3.762	2.110	36.300
S-Citalopram 15.585 kcals/mol	8.696	3.713	4.618	1.406	53.600
average	7.967	3.713	3.959	1.807	46.100



	A distance between N and the Ar1 centroid *(angstroms)*	B distance from N to the Ar1 plane *(angstroms)*	C angle between N, Ar1 centroid and the inner Ar1 axial C *(degrees)*	$D = A \times \cos(C)$ distance along the Ar1 axis from the Ar1 centroid to a point which is normal to N *(angstroms)*	$E = \sqrt{E^2 - A^2}$ distance from the point on on the Ar1 axis which is normal to N, to N *(angstroms)*	$F = \arccos\left(\frac{B}{E}\right)$ angle from the point on the Ar1 plane which is normal to N, a point on the Ar1 axis which is normal to N, and N *(degrees)*
Sertraline 2.135 kcals/mol	5.949	1.352	34.590	4.897	3.377	23.599
MCN-5652 13.035 kcals/mol	5.164	1.430	17.230	4.932	1.530	69.208
Indatraline 14.532 kcals/mol	6.226	1.150	11.360	6.104	1.226	69.675
S-Citalopram 15.585 kcals/mol	7.131	1.843	15.270	6.879	1.878	78.909
average	6.117	1.444	19.613	5.703	2.003	60.348

	G distance from N to the axial X group on the Ar1 axis *(angstroms)*	H distance from N to the to hetero atom *(angstroms)*	I distance from Ar2 centroid to N *(angstroms)*	$J = \sqrt{F^2 - B^2}$ distance from the point on the Ar1 axis which is normal to N to the point on the Ar1 plane which is normal to N *(angstroms)*	E same angle as above via SYBYL torsion measurement (front side outer C, Ar1 outer axial C, Ar1 inner axial C, N) *(degrees)*
Sertraline 2.135 kcals/mol	8.737		3.720	3.095	23.600
MCN-5652 13.035 kcals/mol	8.253		3.734	0.543	69.300
Indatraline 14.532 kcals/mol	9.342		3.762	0.426	73.200
S-Citalopram 15.585 kcals/mol	11.031	5.098	4.618	0.361	78.400
average	9.341	5.098	3.959	1.106	61.125



	A distance between Ar1 centroid and hetero atom *(angstroms)*	B distance from hetero atom to Ar1 plane *(angstroms)*	C angle between hetero atom, Ar1 centroid and the inner Ar1 axial C *(degrees)*	$D = A \times \cos(C)$ distance along the Ar1 axis from the Ar1 centroid to a point which is normal to the hetero atom *(angstroms)*	$E = \sqrt{E^2 - A^2}$ distance from the point on on the Ar1 axis which is normal to the hetero atom, to the hetero atom *(angstroms)*	$F = \arccos\left(\frac{B}{E}\right)$ angle from the point on the Ar1 plane which is normal to the hetero atom, a point on the Ar1 axis which is normal to the hetero atom, and the hetero atom *(degrees)*
Sertraline 2.135 kcals/mol						
MCN-5652 13.035 kcals/mol						
Indatraline 14.532 kcals/mol						
S-Citalopram 15.585 kcals/mol	3.328	0.489	31.490	2.838	1.738	16.338
average	3.328	0.489	31.490	2.838	1.738	16.338

	G hetero atom to the axial X group on the Ar1 axis *(angstroms)*	H distance from Ar2 centroid to hetero atom *(angstroms)*	I distance from hetero atom to N *(angstroms)*	$J = \sqrt{F^2 - B^2}$ distance from the point on the Ar2 axis which is normal to the hetero atom, to the point on the Ar1 plane which is normal to the hetero atom *(angstroms)*	F same angle as above via SYBYL torsion measurement (front side outer C, Ar1 outer axial C, Ar1 inner axial C, hetero atom) *(degrees)*
Sertraline 2.135 kcals/mol					
MCN-5652 13.035 kcals/mol					
Indatraline 14.532 kcals/mol					
S-Citalopram 15.585 kcals/mol	7.051	5.098	3.713	1.668	16.400
average	7.051	5.098	3.713	1.668	16.400

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
1	MCN-5652_7.397_kcals_per_mol	4.057713	3.820207	2.59942	0.18698	2.496496	4.922337	2.696829	0.212015	96.031944	88.977898	2.50173	0.044578	-0.2201	172.221451	179.248505
2	MCN-5652_7.554_kcals_per_mol	4.09275	3.819848	2.54573	0.16686	2.496892	4.924443	2.685277	0.34132	97.113083	88.956314	2.48753	0.078268	-0.28314	170.469406	177.020264
3	MCN-5652_8.497_kcals_per_mol	4.187012	3.79071	2.59122	0.146598	2.494437	4.946166	2.525844	0.183903	100.198753	88.146706	2.46442	0.110494	-0.40475	172.126526	165.981903
4	MCN-5652_8.630_kcals_per_mol	4.210892	3.789369	2.55312	0.160453	2.494617	4.947843	2.504032	0.266209	100.964157	88.10453	2.4509	0.133507	-0.44818	171.03656	164.422607
5	MCN-5652_8.855_kcals_per_mol	5.13862	3.844918	1.42688	0.121778	2.477792	4.8755	1.510074	2.04049	142.646774	90.158264	1.30248	0.78175	-1.966295	132.800201	120.815102
6	MCN-5652_8.876_kcals_per_mol	5.137488	3.84488	1.43624	0.121074	2.477935	4.877818	1.512538	2.030834	142.55835	90.154022	1.30538	0.78398	-1.963914	132.939148	120.899399
7	MCN-5652_9.539_kcals_per_mol	4.054133	3.6917	1.62043	0.46448	5.252925	4.762656	2.67873	2.021795	97.911736	86.578354	2.37535	0.026567	-0.339882	133.553757	161.605835
8	MCN-5652_9.547_kcals_per_mol	4.060848	3.691659	1.60095	0.46805	5.260593	4.757907	2.67659	2.045943	98.110817	86.560333	2.37479	0.025581	-0.348683	133.034821	161.307541
9	MCN-5652_9.773_kcals_per_mol	5.133485	3.811006	1.37517	0.15872	2.463346	4.847583	1.756129	2.01816	143.320801	89.333023	1.29762	0.7102	-1.975027	133.997177	127.1437
10	MCN-5652_9.792_kcals_per_mol	5.132393	3.810932	1.38491	0.15942	2.463509	4.850017	1.758674	2.007723	143.231415	89.327316	1.3006	0.7125	-1.972685	134.156876	127.234001
11	MCN-5652_13.035_kcals_per_mol	5.163851	3.733799	1.429698	0.721143	2.504837	4.912115	1.43014	1.98522	142.438751	86.572899	1.46068	0.6349	-1.967453	135.067596	114.438301
12	MCN-5652_13.057_kcals_per_mol	5.162751	3.733734	1.4382	0.721147	2.50492	4.914307	1.432145	1.976706	142.3582	86.568954	1.46332	0.63677	-1.96526	135.189041	114.500404

Appendix 3.1 The distance space descriptors of MCN-5652 used in the combinatorial comparison.

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
1	Sertraline_1.442_kcals_per_mol	5.697295	3.702194	2.303795	0.946996	4.283523	4.852257	2.628142	1.25135	102.380226	58.175797	0.55594	-1.07274	-3.657294	152.724304	162.476349
2	Sertraline_1.483_kcals_per_mol	5.671344	3.67857	1.41379	1.393	3.738482	4.820113	1.9945	2.028513	115.688438	65.574364	0.88106	-0.82952	-3.63541	133.449951	135.640137
3	Sertraline_1.528_kcals_per_mol	5.706602	3.702525	2.31642	0.945721	4.284617	4.856043	2.621969	1.243326	102.60186	58.16613	0.547761	-1.077942	-3.64123	152.886414	162.286102
4	Sertraline_1.545_kcals_per_mol	5.674929	3.678742	1.4079	1.39263	3.738701	4.823016	1.98967	2.031701	115.796555	65.575272	0.878126	-0.83167	-3.64823	133.245804	135.549881
5	Sertraline_2.094_kcals_per_mol	5.99305	3.678331	1.293066	1.261502	3.924105	4.814296	1.984101	2.03142	120.9123	62.886681	0.87609	-0.83466	-3.637251	133.781204	133.685501
6	Sertraline_2.135_kcals_per_mol	5.948879	3.719596	1.35208	1.315996	3.894259	4.82183	1.923922	2.04169	120.321045	64.164642	0.88478	-0.82553	-3.636168	133.186798	133.039307
7	Sertraline_2.158_kcals_per_mol	5.997147	3.678498	1.28714	1.260899	3.924496	4.817301	1.97895	2.034604	121.038414	62.88483	0.87353	-0.836559	-3.64727	133.575806	133.586105
8	Sertraline_2.198_kcals_per_mol	5.952915	3.719803	1.34632	1.31554	3.894608	4.82484	1.918478	2.044821	120.445656	64.164268	0.882133	-0.827411	-3.648239	132.979828	132.935104
9	Sertraline_2.472_kcals_per_mol	5.626016	3.771204	2.40154	1.093764	4.26152	4.866656	2.537645	1.23547	101.003517	59.884949	0.55621	-1.0726	-3.656622	152.701508	161.292496
10	Sertraline_2.557_kcals_per_mol	5.63638	3.77168	2.41381	1.091089	4.262994	4.870162	2.532162	1.227974	101.241631	59.872448	0.54849	-1.077601	-3.64185	152.851105	161.103027
11	Sertraline_3.253_kcals_per_mol	5.416879	3.727677	2.358448	1.380772	4.192945	4.908484	2.35484	1.18379	97.135284	60.047222	0.54503	-1.07824	-3.655385	153.274704	161.133301
12	Sertraline_3.332_kcals_per_mol	5.426314	3.72809	2.37372	1.379349	4.194463	4.911926	2.349082	1.174296	97.341743	60.033096	0.537183	-1.083517	-3.641724	153.461075	160.965103
13	Sertraline_7.949_kcals_per_mol	7.195394	3.808184	0.23745	0.357329	4.302158	4.91502	1.4048	1.990754	165.538834	60.048553	0.85257	-0.85654	-3.66016	135.137558	113.506668
14	Sertraline_8.012_kcals_per_mol	7.196291	3.808244	0.24088	0.356189	4.302127	4.917623	1.39876	1.994213	165.641739	60.050243	0.858357	-0.851802	-3.638384	135.137024	113.411552
15	Sertraline_8.022_kcals_per_mol	7.196042	3.860692	0.335965	0.239596	4.312263	4.912727	1.3575	2.02163	164.394073	60.94413	0.87126	-0.83757	-3.658765	133.867996	113.084831
16	Sertraline_8.085_kcals_per_mol	7.196961	3.860728	0.33862	0.237966	4.31218	4.915359	1.35055	2.025586	164.498138	60.946125	0.877626	-0.832026	-3.639353	133.570602	112.97332

Appendix 3.2 The distance space descriptors of sertraline used in the combinatorial comparison.

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
1	Indatraline_7.467_kcals_per_mol	6.086811	3.717476	0.98418	1.413273	3.229259	4.949376	1.93165	1.536079	161.108643	74.468376	0.98053	0.484896	-0.98053	146.744049	135.813004
2	Indatraline_7.525_kcals_per_mol	6.086793	3.717408	0.989384	1.413802	3.229395	4.952251	1.92887	1.5367	161.081375	74.464897	0.97918	0.49156	-0.97918	146.629898	135.793488
3	Indatraline_7.922_kcals_per_mol	6.207501	3.726457	1.12749	1.308398	3.379548	4.944895	1.90084	1.574568	158.226624	72.345192	1.25415	0.333373	-1.25415	145.471786	136.711945
4	Indatraline_7.978_kcals_per_mol	6.207664	3.726384	1.132067	1.308835	3.379635	4.947837	1.89765	1.57499	158.225937	72.342529	1.25235	0.340509	-1.25235	145.362106	136.682877
5	Indatraline_8.133_kcals_per_mol	5.784319	3.823945	2.88486	0.927184	3.76892	4.889596	2.69692	0.757729	118.442123	68.588432	3.27003	-0.81413	-3.27003	165.810928	173.974106
6	Indatraline_8.197_kcals_per_mol	5.783612	3.823889	2.903292	0.926834	3.769078	4.893402	2.69371	0.73039	118.41581	68.584862	3.2715	-0.8107	-3.2715	166.415894	174.030106
7	Indatraline_8.241_kcals_per_mol	6.053363	3.680354	0.90295	1.406618	3.183814	4.96759	1.81436	1.536148	162.504898	74.330795	0.84539	0.531695	-0.84539	146.286209	133.477264
8	Indatraline_8.298_kcals_per_mol	6.053227	3.680251	0.908317	1.407157	3.183887	4.970438	1.81158	1.53644	162.468399	74.327545	0.84389	0.538341	-0.84389	146.176498	133.456726
9	Indatraline_8.478_kcals_per_mol	5.778759	3.823142	2.94175	0.834519	3.798552	4.895892	2.69086	0.710039	117.365814	68.079506	3.31574	-0.88946	-3.31574	167.057907	173.828598
10	Indatraline_8.540_kcals_per_mol	5.778113	3.823193	2.95814	0.834233	3.798631	4.899573	2.68772	0.68612	117.344284	68.079384	3.31708	-0.88609	-3.31708	167.580795	173.882706
11	Indatraline_8.854_kcals_per_mol	5.751822	3.853444	2.94952	0.867962	3.758017	4.901834	2.65362	0.708582	117.67691	69.376686	3.29554	-0.77153	-3.29554	166.892807	173.477798
12	Indatraline_8.919_kcals_per_mol	5.751588	3.853467	2.964552	0.868008	3.758063	4.905498	2.65037	0.68433	117.670128	69.376488	3.29634	-0.76752	-3.29634	167.431702	173.513397
13	Indatraline_11.348_kcals_per_mol	6.011423	3.682737	0.678964	1.359331	3.102983	5.03983	1.15365	2.54836	165.675247	75.580841	0.01684	0.744855	-0.01684	79.349586	116.99469
14	Indatraline_13.572_kcals_per_mol	6.010073	3.701306	0.88838	1.436621	3.140397	4.96255	1.8668	1.524366	162.464355	75.506943	0.80052	0.58167	-2.98986	146.852844	133.976669
15	Indatraline_13.630_kcals_per_mol	6.009866	3.701221	0.89385	1.437174	3.140457	4.965391	1.86406	1.524995	162.420547	75.504227	0.79907	0.58821	-2.989037	146.735474	133.956985
16	Indatraline_13.702_kcals_per_mol	5.793983	3.782612	2.88637	0.86223	3.78539	4.899821	2.665006	0.724111	118.273087	67.421944	3.27157	-0.900934	-1.76274	13.31006	173.531235
17	Indatraline_13.764_kcals_per_mol	5.793467	3.782688	2.903863	0.86222	3.785462	4.903484	2.662002	0.69864	118.255859	67.422554	3.27284	0.897243	-1.76213	167.252594	173.585709

Appendix 3.3 The distance space descriptors of indatraline used in the combinatorial comparison.

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
18	Indatraline_14.164_kcals_per_mol	6.117846	3.664682	0.92809	1.392571	3.251082	4.963384	1.83137	1.542825	162.227859	72.896393	0.9383	0.43797	-3.09122	146.156235	134.23938
19	Indatraline_14.184_kcals_per_mol	6.130847	3.648163	0.93183	1.38921	3.265626	4.958974	1.8451	1.55134	162.059158	72.29763	0.97361	0.392886	-3.10192	145.945007	134.749573
20	Indatraline_14.221_kcals_per_mol	6.117845	3.664597	0.933353	1.39307	3.251199	4.966274	1.82853	1.54295	162.203308	72.892906	0.936852	0.4448	-3.09079	146.052063	134.218491
21	Indatraline_14.241_kcals_per_mol	6.130921	3.648073	0.936941	1.389683	3.265768	4.961845	1.84224	1.55161	162.040802	72.293709	0.97217	0.399666	-3.10164	145.837799	134.727997
22	Indatraline_14.476_kcals_per_mol	6.226276	3.762174	1.14545	1.304931	3.397526	4.94945	1.88938	1.556208	158.306747	72.875755	1.24369	0.3773	-3.150889	145.978638	136.120636
23	Indatraline_14.532_kcals_per_mol	6.226388	3.762107	1.1503	1.3054	3.397626	4.9524	1.88632	1.556438	158.299835	72.872971	1.24206	0.38446	-3.150736	145.873795	136.095322
24	Indatraline_14.967_kcals_per_mol	5.766745	3.861947	2.99848	0.79974	3.809037	4.898981	2.676859	0.689147	116.656868	68.752357	3.35348	-0.860546	-1.67847	167.453186	173.666214
25	Indatraline_15.029_kcals_per_mol	5.766123	3.86207	3.01386	0.799352	3.809141	4.902615	2.67368	0.666212	116.635651	68.753403	3.35479	-0.857208	-1.677538	167.95401	173.715515
26	Indatraline_15.680_kcals_per_mol	5.737864	3.872137	3.02319	0.881976	3.772985	4.900225	2.669	0.642878	116.76725	69.58181	3.3321	-0.776567	-1.671806	168.165756	174.576263
27	Indatraline_15.745_kcals_per_mol	5.737631	3.871969	3.037987	0.88133	3.773087	4.903924	2.665244	0.6179	116.759094	69.576492	3.33294	-0.773138	-1.67169	168.718201	174.603439
28	Indatraline_15.768_kcals_per_mol	5.724724	3.831525	3.07244	0.751646	3.791121	4.91977	2.59409	0.562073	115.810005	68.413025	3.35508	-0.87531	-1.62137	169.947723	173.862564
29	Indatraline_15.832_kcals_per_mol	5.72489	3.831431	3.08572	0.75193	3.791138	4.923189	2.59085	0.53913	115.817276	68.410652	3.35552	0.87175	-1.62212	170.457001	173.892197

										angle	angle					
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	explicit	explicit
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	angle	angle
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	Ar1 plane to	Ar2 plane to
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	YZ plane	YZ plane
1	S-Citalopram_8.853_kcals_per_mol	3.67128	6.029668	2.911011	0.666201	3.176697	4.859788	1.821014	2.68086	74.237083	160.042252	-1.20798	2.82326	0.916822	86.404572	45.5163
2	S-Citalopram_9.263_kcals_per_mol	3.51516	6.32309	2.733964	0.643029	3.623144	4.806541	2.16051	2.78593	64.077255	148.28096	1.69754	2.77546	1.5016	88.944313	216.796402
3	S-Citalopram_9.270_kcals_per_mol	2.969067	6.190492	2.733179	0.581698	3.287195	4.781022	2.151451	2.83541	56.957394	166.285553	0.060717	2.76131	1.769955	87.984909	217.366501
4	S-Citalopram_9.493_kcals_per_mol	7.736428	6.091556	2.046491	4.722579	5.101118	4.821425	2.19525	2.79738	149.250702	94.611092	1.15754	2.1613	-4.451498	87.916107	214.509598
5	S-Citalopram_9.546_kcals_per_mol	6.128332	7.684354	4.634368	2.90889	5.110119	4.84676	1.961773	2.7519	95.899811	143.762756	1.71854	4.73995	-0.620314	87.292412	221.768402
6	S-Citalopram_9.566_kcals_per_mol	5.327249	5.831194	3.049858	0.81549	3.464435	4.866247	1.802135	2.70363	113.453369	130.792709	-1.208935	2.98919	-1.334331	87.227562	45.95554
7	S-Citalopram_9.613_kcals_per_mol	6.621847	5.536495	1.424929	4.309587	4.437746	4.81956	2.24031	2.80089	127.728477	95.018135	2.94888	1.559804	-2.84996	88.493057	212.803894
8	S-Citalopram_9.655_kcals_per_mol	5.815384	6.375084	2.67933	3.676115	4.423545	4.849961	1.96888	2.74115	103.193314	118.442909	3.12303	2.82572	-1.153446	87.591698	221.505905
9	S-Citalopram_9.830_kcals_per_mol	6.408161	6.458917	3.266955	3.674347	4.423198	4.820396	2.20584	2.79798	120.692932	121.174095	1.54334	-0.37734	-2.346988	87.967934	214.291901
10	S-Citalopram_9.872_kcals_per_mol	6.231034	3.521811	0.6851	3.110569	3.649953	4.740547	2.65777	2.84525	143.621536	63.451935	1.86895	-0.69689	-3.01478	88.046867	200.869202
11	S-Citalopram_9.884_kcals_per_mol	6.512244	6.332406	3.130657	3.693465	4.43156	4.847899	1.9439	2.75245	124.033661	116.873657	1.52451	3.23694	-2.563437	87.238441	222.250504
12	S-Citalopram_9.885_kcals_per_mol	5.600403	5.79424	2.84903	1.93861	3.522907	4.81081	2.34096	2.8119	120.938858	126.891846	-0.93966	2.90158	-1.79401	89.995132	208.568802
13	S-Citalopram_9.920_kcals_per_mol	5.772329	3.718035	0.52129	2.750113	3.030013	4.910169	2.53349	2.00769	153.709213	76.82975	-1.403175	0.480459	-2.67034	130.469803	176.650406
14	S-Citalopram_10.113_kcals_per_mol	5.233475	7.440292	4.592022	0.947193	4.590252	4.874766	1.78855	2.68209	85.528694	161.302185	-0.139685	4.5756	0.36587	87.791519	46.2388
15	S-Citalopram_10.124_kcals_per_mol	7.370233	5.582248	1.702269	3.57101	4.615999	4.812906	2.27257	2.80997	156.367294	92.460037	-0.707161	1.76235	-4.217071	88.97831	211.063904
16	S-Citalopram_10.164_kcals_per_mol	7.750619	6.020913	1.895809	4.76974	5.10498	4.821994	2.189415	2.79646	149.857712	92.900055	1.284221	2.01712	-4.48667	87.816772	214.776703
17	S-Citalopram_10.173_kcals_per_mol	7.71894	6.181511	2.218518	4.667063	5.101921	4.818953	2.20463	2.79978	148.198883	96.708397	1.03396	2.329277	-4.39965	87.933929	214.284302
18	S-Citalopram_10.226_kcals_per_mol	6.207811	7.67901	4.67659	2.91977	5.112393	4.844372	1.975225	2.755	97.759705	143.385498	1.56085	4.77585	-0.779508	87.314758	221.421295
19	S-Citalopram_10.244_kcals_per_mol	5.614021	5.421623	1.292641	2.875949	3.411166	4.951975	2.56363	1.94188	125.418037	116.572975	-1.068362	2.656964	-1.89898	131.818695	182.847
20	S-Citalopram_10.245_kcals_per_mol	6.073746	7.686506	4.586885	2.917866	5.113157	4.847612	1.95366	2.75129	94.543747	143.720703	1.864177	4.6991	-0.50589	87.27803	221.988693

Appendix 3.4 The distance space descriptors of escitalopram used in the combinatorial comparison.

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
21	S-Citalopram_10.375_kcals_per_mol	7.700635	6.067554	2.057102	4.65489	5.058	4.818998	2.208518	2.79997	149.637817	94.930603	1.05023	2.16775	-4.42752	87.984581	214.146103
22	S-Citalopram_10.379_kcals_per_mol	7.713125	6.012269	1.946963	4.684509	5.060709	4.820753	2.1985	2.79778	150.22084	93.590599	1.136064	2.061709	-4.45857	87.887589	214.502808
23	S-Citalopram_10.404_kcals_per_mol	6.089415	7.65746	4.632396	2.822062	5.066425	4.844637	1.9709	2.7555	95.890198	144.546249	1.60684	4.73505	-0.611287	87.269592	221.540604
24	S-Citalopram_10.409_kcals_per_mol	5.349728	7.43084	4.61515	1.037194	4.60497	4.871751	1.8077	2.6908	87.896103	159.08429	-0.19302	4.60013	0.176272	87.764587	45.7593
25	S-Citalopram_10.429_kcals_per_mol	4.924042	7.507266	4.427255	1.00749	4.619393	4.818191	2.105075	2.75565	78.144669	165.34967	0.706356	4.45966	0.907971	87.91938	218.181595
26	S-Citalopram_10.434_kcals_per_mol	6.048257	7.660216	4.604825	2.82437	5.06784	4.846367	1.958935	2.75364	94.883293	144.616699	1.70298	4.7115	-0.526304	87.25721	221.858093
27	S-Citalopram_10.543_kcals_per_mol	7.345959	5.768628	1.989226	3.60192	4.644158	4.810195	2.290275	2.81364	152.66156	96.333458	-0.70102	2.05338	-4.11584	89.156113	310.569092
28	S-Citalopram_10.603_kcals_per_mol	6.303288	3.37792	0.742362	2.888142	3.57283	4.923568	2.64122	1.95328	153.330536	61.41375	1.4099	-0.29833	-3.24394	132.199005	184.384201
29	S-Citalopram_10.636_kcals_per_mol	4.915548	7.516567	4.422876	1.08198	4.638282	4.829172	2.085622	2.73265	77.626831	164.203949	0.8357	4.44636	0.95506	88.231796	218.612305
30	S-Citalopram_10.640_kcals_per_mol	5.903018	4.328012	0.757864	2.004621	3.25879	4.836014	2.4092	2.77497	146.385834	88.229866	-1.72577	0.89716	-2.65409	93.349579	202.026703
31	S-Citalopram_10.661_kcals_per_mol	6.831855	5.761461	1.66913	4.51722	4.613357	4.823572	2.200497	2.79697	129.399521	96.851334	2.85555	1.82172	-3.061354	88.115921	214.127304
32	S-Citalopram_10.679_kcals_per_mol	6.787777	5.711567	1.614851	4.47739	4.574956	4.823462	2.201959	2.79698	129.108917	96.439331	2.87306	1.766292	-3.019309	88.143112	214.083603
33	S-Citalopram_10.718_kcals_per_mol	6.010917	6.631873	2.991349	3.802778	4.62248	4.851039	1.9531	2.74134	103.739616	120.821968	3.07355	3.14359	-1.241238	87.470589	221.944794
34	S-Citalopram_10.719_kcals_per_mol	7.359798	5.615926	1.765431	3.58933	4.614882	4.812679	2.268184	2.81052	155.696289	93.278015	-0.64748	1.82652	-4.19706	88.892509	211.329605
35	S-Citalopram_10.732_kcals_per_mol	5.943298	6.591911	2.935977	3.750659	4.582255	4.850987	1.9595	2.74106	102.892563	120.731483	3.10167	3.08489	-1.166386	87.54319	221.742706
36	S-Citalopram_10.761_kcals_per_mol	7.509819	4.583016	0.07834	4.08476	4.660927	4.810377	2.235606	2.80089	166.058014	70.002335	0.871694	0.01997	-4.56508	88.076012	212.734207
37	S-Citalopram_10.907_kcals_per_mol	6.13253	2.940504	0.50518	2.71132	3.263299	4.86902	2.67182	2.03957	168.587387	56.152103	-0.134409	-0.568782	-3.208778	130.561005	179.2173
38	S-Citalopram_10.991_kcals_per_mol	6.583346	6.577964	3.334468	3.825002	4.613795	4.841763	1.96877	2.76165	120.711708	119.349281	1.73748	3.45779	-2.451022	87.167702	221.601898
39	S-Citalopram_11.006_kcals_per_mol	6.567338	6.530877	3.298951	3.79422	4.576425	4.843346	1.960777	2.75982	121.32769	118.947327	1.6876	3.41919	-2.471644	87.171227	221.803802
40	S-Citalopram_11.009_kcals_per_mol	6.649724	6.532744	3.229493	3.955	4.602333	4.812642	2.247076	2.80555	123.239258	118.229309	1.692696	3.34279	-2.617981	88.16581	213.093399
41	S-Citalopram_11.010_kcals_per_mol	6.587308	6.514064	3.246217	3.87628	4.55395	4.814907	2.236105	2.80329	122.649147	119.047348	1.63611	3.358516	-2.549852	88.106003	213.412201
42	S-Citalopram_11.023_kcals_per_mol	6.297149	3.028336	1.05388	2.752625	3.482617	4.729653	2.67977	2.84352	160.573715	55.448986	0.389249	1.01112	-3.301369	91.384827	201.315598
43	S-Citalopram_11.102_kcals_per_mol	5.224814	7.42953	4.574342	1.026695	4.58007	4.871855	1.79424	2.68961	85.527954	161.233032	-0.02597	4.56586	0.357809	87.703049	46.118568
44	S-Citalopram_11.111_kcals_per_mol	6.944791	5.789494	0.265172	5.03378	5.13282	4.862968	1.670693	2.78229	116.793129	87.192329	4.31981	0.83322	-2.509616	83.439438	48.954689
45	S-Citalopram_11.124_kcals_per_mol	5.746517	6.280715	3.509317	1.398089	3.908643	4.854609	1.89265	2.72214	114.35965	132.28923	-0.91772	3.47224	-1.58293	87.397621	223.640396

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
46	S-Citalopram_11.125_kcals_per_mol	6.042092	6.081628	3.117161	2.27646	3.911161	4.812812	2.26518	2.80698	124.237503	124.341599	-0.79984	3.15345	-2.19164	88.667	211.825104
47	S-Citalopram_11.156_kcals_per_mol	7.306728	5.196371	1.060584	4.500884	4.522201	4.955112	2.55063	1.93883	159.075455	85.313393	-0.350356	1.704	-4.185516	131.925598	183.638199
48	S-Citalopram_11.208_kcals_per_mol	7.365758	5.573311	1.68911	3.65471	4.605937	4.808537	2.299119	2.81334	156.786987	92.448677	-0.5944	1.75259	-4.2257	89.069221	210.477402
49	S-Citalopram_11.260_kcals_per_mol	4.966774	7.48874	4.436712	1.06662	4.605865	4.818724	2.104591	2.75236	79.306808	164.729263	0.685117	4.46752	0.815395	87.884491	218.184097
50	S-Citalopram_11.289_kcals_per_mol	7.644684	6.047111	2.934421	4.888201	5.098363	4.945837	2.54589	1.98202	144.498779	93.477943	1.62247	2.46516	-4.156917	130.861603	180.033401
51	S-Citalopram_11.296_kcals_per_mol	6.443331	5.656911	3.577506	3.199812	4.44082	4.963596	2.51661	1.95374	121.34584	97.732666	3.17795	1.97035	-2.367302	131.3647	182.668304
52	S-Citalopram_11.331_kcals_per_mol	6.213522	3.132647	1.13048	2.936912	3.589836	4.759466	2.52225	2.83247	146.09758	56.261715	1.528893	-1.03559	-3.042594	88.612732	207.537003
53	S-Citalopram_11.398_kcals_per_mol	7.382246	5.453105	1.489947	3.50805	4.594362	4.807016	2.281137	2.81165	159.033051	89.88633	-0.72618	1.54327	-4.27629	88.544228	211.505295
54	S-Citalopram_11.411_kcals_per_mol	7.514437	4.629946	0.017998	4.16083	4.67068	4.819448	2.222587	2.79426	165.365585	70.800751	0.92608	0.106711	-4.56265	88.640182	211.998901
55	S-Citalopram_11.411B_kcals_per_mol	5.694211	6.233182	2.055097	3.8234	4.369406	4.865041	1.649294	2.77971	101.116501	115.613564	3.33135	2.52069	-1.014047	83.215561	49.660568
56	S-Citalopram_11.450_kcals_per_mol	6.276934	6.477898	3.827063	3.469977	4.426013	4.95015	2.54757	1.96842	116.425148	121.379662	1.67195	3.612523	-1.95335	131.120193	181.442993
57	S-Citalopram_11.452_kcals_per_mol	6.758376	5.743695	1.78056	4.159604	4.457839	4.869797	1.61982	2.77331	132.087387	99.81562	2.27378	2.14376	-3.11247	83.24617	50.308998
58	S-Citalopram_11.473_kcals_per_mol	7.575408	4.925171	0.34547	4.512525	4.749558	4.78169	2.5194	2.81835	163.085663	75.546295	1.07582	0.384279	-4.59405	88.733871	203.253098
59	S-Citalopram_11.489_kcals_per_mol	5.10444	7.460761	4.558051	0.85569	4.584248	4.871685	1.776339	2.68691	82.75444	164.051575	-0.05068	4.54731	0.581561	87.676277	46.6092
60	S-Citalopram_11.506_kcals_per_mol	4.917309	7.501167	4.436247	0.8928	4.600819	4.809992	2.118074	2.76481	78.328865	166.982178	0.53402	4.46991	0.89383	87.526901	217.9039
61	S-Citalopram_11.556_kcals_per_mol	7.29513	5.342762	1.197198	4.535147	4.545444	4.953709	2.55548	1.94448	156.231293	88.178741	-0.387548	1.91607	-4.116388	131.779099	183.436707
62	S-Citalopram_11.771_kcals_per_mol	7.297261	5.270324	1.168872	4.52156	4.530617	4.955998	2.551587	1.94045	157.568176	86.816589	-0.320792	1.820912	-4.14789	131.854294	176.353699
63	S-Citalopram_11.794_kcals_per_mol	7.022367	5.745288	0.233478	5.062118	5.137936	4.863354	1.66595	2.7827	119.003181	86.126862	4.22848	0.799188	-2.68337	83.373161	49.114201
64	S-Citalopram_11.837_kcals_per_mol	7.489782	4.627618	0.042937	4.07484	4.634347	4.812217	2.226434	2.80114	166.928757	71.344971	0.787404	0.13606	-4.55268	88.309738	212.515198
65	S-Citalopram_11.845_kcals_per_mol	7.466413	4.509067	0.10695	3.94437	4.602426	4.812578	2.19787	2.80138	168.207718	69.440453	0.69394	0.0134	-4.539047	88.123207	213.378693
66	S-Citalopram_11.867_kcals_per_mol	6.881627	5.852278	0.333088	5.00041	5.135271	4.860571	1.699064	2.7821	114.857292	88.532478	4.39292	0.89953	-2.357741	83.518387	48.26973
67	S-Citalopram_11.976_kcals_per_mol	7.654096	5.984981	2.921732	4.85308	5.101922	4.946103	2.551281	1.98006	144.791687	91.985779	1.75705	2.332642	-4.181	130.915802	181.254196
68	S-Citalopram_11.991_kcals_per_mol	7.631953	6.134649	2.96852	4.923759	5.100459	4.944689	2.54764	1.98512	143.74556	95.465515	1.4805	2.631922	-4.112292	130.802521	181.097794
69	S-Citalopram_12.007_kcals_per_mol	6.925265	5.711552	0.182585	4.994686	5.092368	4.863656	1.66168	2.78169	117.334763	86.245636	4.2735	0.75561	-2.532149	83.30822	49.302299
70	S-Citalopram_12.036_kcals_per_mol	6.82006	4.346968	1.45897	4.1635	4.695468	4.841375	1.799737	2.80862	125.937775	64.697754	3.50113	-0.97747	-2.89321	83.430771	224.883896

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
71	S-Citalopram_12.058_kcals_per_mol	6.29883	6.22961	4.122199	3.95744	5.128085	4.931818	1.34641	1.97685	99.669998	97.109039	4.87222	1.220433	-0.94243	130.895004	115.784599
72	S-Citalopram_12.062_kcals_per_mol	6.887144	5.740716	0.219817	4.97543	5.090031	4.862641	1.677907	2.78044	116.254608	86.92971	4.31425	0.78621	-2.446757	83.414932	48.863461
73	S-Citalopram_12.071_kcals_per_mol	6.041539	3.421366	0.926139	2.948426	3.680804	5.130327	0.25743	2.15019	132.549088	60.883488	2.5395	-0.438024	-2.578696	121.982803	76.296402
74	S-Citalopram_12.085_kcals_per_mol	6.728584	4.301741	0.25141	3.918985	4.162107	4.762608	2.37772	2.84614	143.883316	72.190559	2.25006	-0.13585	-3.455131	88.889709	209.055695
75	S-Citalopram_12.092_kcals_per_mol	7.457646	4.87164	1.128331	4.519943	4.618295	4.939952	2.51313	1.96653	165.599503	76.651718	0.49839	1.03514	-4.467854	131.398605	181.873703
76	S-Citalopram_12.109_kcals_per_mol	6.027313	3.265519	0.62592	2.950076	3.583662	4.940699	1.1639	2.53391	135.984543	59.018753	2.189532	-0.68326	-2.686877	87.820831	61.641041
77	S-Citalopram_12.152_kcals_per_mol	7.301576	5.202076	1.134315	4.50914	4.516711	4.948862	2.564147	1.94524	159.131042	85.543175	-0.24467	1.707066	-4.184988	131.845108	183.205795
78	S-Citalopram_12.158_kcals_per_mol	7.613601	6.010723	2.854866	4.87253	5.053648	4.943943	2.547481	1.98512	145.149597	93.536972	1.51059	2.45575	-4.150735	130.806305	180.873901
79	S-Citalopram_12.166_kcals_per_mol	7.621631	5.966874	2.845506	4.85195	5.056316	4.944695	2.55191	1.9823	145.426361	92.476814	1.69082	2.366191	-4.17085	130.875793	181.070206
80	S-Citalopram_12.193_kcals_per_mol	6.4756	5.419167	3.275182	1.618294	4.345537	4.932067	1.35949	1.98412	125.641243	93.856834	3.09705	1.62146	-2.568967	130.862198	116.0392
81	S-Citalopram_12.228_kcals_per_mol	6.259583	3.277705	1.15928	2.972112	3.679689	4.868899	1.60523	2.76806	143.636093	58.069492	1.828738	-0.81041	-3.041765	81.40506	52.529999
82	S-Citalopram_12.256_kcals_per_mol	7.470888	4.8497	1.198464	4.517549	4.632031	4.951267	2.49315	1.95429	164.852798	75.875587	0.63963	1.00436	-4.474907	131.554504	182.661194
83	S-Citalopram_12.308_kcals_per_mol	6.691462	4.122433	0.11641	3.329281	3.981718	4.851617	1.70533	2.78489	155.855804	72.110069	1.362796	0.17159	-3.65409	82.133842	48.65926
84	S-Citalopram_12.336_kcals_per_mol	7.249367	5.411089	0.987831	4.7212	4.765625	4.816979	2.26697	2.80335	140.832352	85.69651	2.56441	1.1101	-3.81126	88.73671	211.880997
85	S-Citalopram_12.338_kcals_per_mol	5.557433	6.555893	4.444422	2.653023	4.462522	4.966082	1.177632	1.92876	95.712044	122.816086	3.30377	2.98334	-0.44297	131.623795	111.800102
86	S-Citalopram_12.373_kcals_per_mol	6.836229	5.893644	1.723951	4.40467	4.636616	4.874558	1.551065	2.77556	128.589493	99.682167	2.678941	2.150044	-3.034719	83.035988	51.8918
87	S-Citalopram_12.386_kcals_per_mol	6.671645	5.860311	3.706837	3.460735	4.6261	4.957063	2.53142	1.96488	123.265007	98.819176	3.12542	2.19257	-2.59002	131.156097	182.133392
88	S-Citalopram_12.389_kcals_per_mol	6.629848	5.775598	3.63694	3.383711	4.580764	4.956737	2.52911	1.96446	123.286163	97.69397	3.15322	2.07551	-2.569696	131.171204	181.993103
89	S-Citalopram_12.405_kcals_per_mol	7.325719	5.099924	0.95978	4.474749	4.514707	4.944092	2.57818	1.94435	161.524414	83.312057	-0.301472	1.52524	-4.248486	131.9599	183.382202
90	S-Citalopram_12.550_kcals_per_mol	5.863019	7.097195	3.545526	3.461691	4.839209	4.854194	1.95569	2.72763	95.221893	129.900421	3.01774	3.67392	-0.577585	87.678497	221.868393
91	S-Citalopram_12.585_kcals_per_mol	7.002985	6.587263	3.33889	3.88607	4.750747	4.819647	2.20307	2.79846	130.967514	115.743629	0.7653	3.43006	-3.173235	87.93663	214.392807
92	S-Citalopram_12.626_kcals_per_mol	6.455626	6.527915	3.894082	3.631449	4.553877	4.946218	2.54748	1.97465	118.383842	119.179642	1.77141	3.61028	-2.15192	130.974396	180.939301
93	S-Citalopram_12.632_kcals_per_mol	6.521235	6.546663	3.932776	3.67636	4.605115	4.943018	2.549717	1.9818	118.974342	118.283829	1.84609	3.59497	-2.221146	130.837494	180.727402
94	S-Citalopram_12.649_kcals_per_mol	5.35919	6.578666	4.099564	0.981466	4.416777	4.762471	2.288404	2.42734	91.881851	125.180176	3.28334	2.90256	-0.230834	118.076797	144.695892
95	S-Citalopram_12.650_kcals_per_mol	6.014739	7.810945	5.065375	0.53977	5.132688	4.76028	2.31632	2.42812	92.844696	148.870575	1.70694	4.831067	-0.2619	118.112701	146.020996

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
96	S-Citalopram_12.660_kcals_per_mol	6.108328	6.376241	2.246164	4.139738	4.607378	4.864134	1.73466	2.75865	106.545349	113.431061	3.37083	2.65824	-1.481474	83.897491	47.597698
97	S-Citalopram_12.675_kcals_per_mol	6.058819	6.354263	2.2581	4.08816	4.568108	4.865093	1.73822	2.7561	106.14859	113.832489	3.33849	2.65702	-1.437642	84.018982	47.483089
98	S-Citalopram_12.701_kcals_per_mol	6.594846	6.999796	4.055818	3.25369	4.771293	4.848111	1.94066	2.74884	116.472298	128.692307	0.892415	4.126	-2.18511	87.324432	222.369904
99	S-Citalopram_12.715_kcals_per_mol	6.827592	4.42574	1.3551	4.21426	4.717646	4.848284	1.790875	2.80482	125.450333	65.925095	3.56875	-0.89283	-2.87509	83.881126	224.123596
100	S-Citalopram_12.760_kcals_per_mol	6.221455	6.328655	4.202806	3.98312	5.135745	4.92614	1.36656	1.99338	97.619957	99.317528	4.87736	1.350503	-0.76295	130.535095	116.304199
101	S-Citalopram_12.771_kcals_per_mol	6.179153	4.514678	1.68422	4.20657	4.566637	4.892572	1.113743	2.78706	109.501717	70.122414	4.0827	-0.95375	-1.67213	80.726189	60.59407
102	S-Citalopram_12.831_kcals_per_mol	6.385833	6.155388	4.063092	3.944752	5.129249	4.940022	1.300031	1.96476	101.801071	95.337845	4.85367	1.14037	-1.128782	131.115005	121.671303
103	S-Citalopram_12.883_kcals_per_mol	5.990409	4.784033	2.066938	4.54459	4.639699	5.073948	0.37489	2.02117	102.698875	74.398338	4.40004	-0.82536	-1.12497	128.6931	93.092003
104	S-Citalopram_12.905_kcals_per_mol	6.994477	4.613972	1.15533	4.43246	4.820052	4.846573	1.771411	2.79924	127.957932	68.063812	3.55622	-0.65765	-3.10906	83.308029	45.687389
105	S-Citalopram_12.938_kcals_per_mol	7.433849	4.917511	1.176782	4.53471	4.596422	4.93972	2.49967	1.9688	164.76738	77.907631	0.46369	1.130307	-4.43179	131.332001	181.443802
106	S-Citalopram_12.949_kcals_per_mol	6.221987	6.201175	4.070663	3.9597	5.086903	4.929045	1.35828	1.98776	98.685799	97.296806	4.85184	1.18951	-0.85232	130.650696	116.114304
107	S-Citalopram_13.009_kcals_per_mol	6.270695	6.148624	4.043615	3.99691	5.084308	4.940077	1.3034	1.96665	99.931564	96.10598	4.84548	1.12837	-0.95753	131.040802	114.7966
108	S-Citalopram_13.021_kcals_per_mol	3.658969	6.22629	3.14095	1.781297	3.752375	4.694618	2.433452	2.64175	65.373825	135.937378	2.37355	2.4449	1.47816	112.6567	150.603806
109	S-Citalopram_13.035_kcals_per_mol	5.580179	2.834556	1.92168	2.704866	3.507227	4.889656	1.26543	2.69996	120.438705	51.226368	2.46341	-1.51475	-1.88474	80.38517	60.954201
110	S-Citalopram_13.077_kcals_per_mol	6.077218	4.751171	2.054069	4.541817	4.654022	5.085279	0.314442	2.01898	104.64064	73.488571	4.37441	-0.82266	-1.28107	128.837601	86.5522
111	S-Citalopram_13.093_kcals_per_mol	6.366212	6.487455	3.862217	1.417526	4.456531	4.758352	2.31714	2.44024	118.53743	120.875221	2.05697	3.2842	-2.170615	117.833	146.067398
112	S-Citalopram_13.202_kcals_per_mol	7.438579	4.885306	0.990017	4.53019	4.586788	4.929261	2.493092	1.98396	166.73201	77.391579	0.25422	1.06286	-4.456875	131.054001	179.940399
113	S-Citalopram_13.207_kcals_per_mol	6.769694	4.41507	1.37687	4.18309	4.690831	4.848493	1.783454	2.80457	124.320007	66.136223	3.60218	-0.89602	-2.78539	83.620193	224.613998
114	S-Citalopram_13.249_kcals_per_mol	5.845204	6.622628	4.531893	2.35036	4.637832	4.920745	1.42626	2.00345	99.155701	119.718071	3.48075	2.97204	-0.758061	130.016403	118.117203
115	S-Citalopram_13.279_kcals_per_mol	6.110469	7.803648	5.037932	0.776289	5.138828	4.759972	2.32365	2.42618	94.963768	148.11734	1.59602	4.86272	-0.448308	118.275497	146.393402
116	S-Citalopram_13.300_kcals_per_mol	6.739715	4.25535	1.55874	4.043442	4.650074	4.847327	1.80661	2.80801	124.590408	63.586742	3.5068	-1.09073	-2.77379	83.668388	223.796906
117	S-Citalopram_13.306_kcals_per_mol	6.686216	4.165984	1.62139	3.94644	4.604447	4.847966	1.810334	2.8091	124.21225	62.517643	3.46389	-1.1767	-2.71903	83.963257	223.258499
118	S-Citalopram_13.374_kcals_per_mol	6.209652	4.516154	1.59781	4.21332	4.560147	4.891496	1.142016	2.78266	110.533623	70.256638	4.053859	-0.90733	-1.748092	81.181282	59.834702
119	S-Citalopram_13.384_kcals_per_mol	6.243495	4.869205	0.15616	4.480792	4.643909	4.956393	1.01244	2.51213	109.31971	76.14122	4.23986	-0.307714	-1.711811	90.456619	117.427101
120	S-Citalopram_13.399_kcals_per_mol	5.942158	7.815693	5.08313	0.35439	5.131108	4.758521	2.316236	2.43517	91.1987	149.226105	1.80935	4.79614	-0.119734	117.929497	145.974304

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
121	S-Citalopram_13.433_kcals_per_mol	7.166744	4.883199	0.23379	4.482763	4.611162	4.813421	2.18604	2.8006	144.103592	77.056396	2.45104	0.38623	-3.841833	87.906387	214.180695
122	S-Citalopram_13.530_kcals_per_mol	5.998984	7.782405	5.01049	0.6222	5.095322	4.759266	2.318845	2.4298	93.242622	149.334946	1.60276	4.826156	-0.29268	118.147697	146.106995
123	S-Citalopram_13.532_kcals_per_mol	6.328946	4.824431	0.16367	4.488198	4.656109	4.956639	1.02575	2.51225	111.431122	75.004173	4.18423	-0.336207	-1.873495	90.717934	61.946201
124	S-Citalopram_13.542_kcals_per_mol	4.715559	6.387212	3.806	3.74379	4.730013	4.85228	1.97994	1.94924	71.890015	110.287163	4.2949	1.29799	1.40039	130.573593	134.1492
125	S-Citalopram_13.577_kcals_per_mol	5.40344	6.401368	3.818061	4.0154	4.425255	5.046546	0.424732	2.4224	92.68071	118.958313	3.4864	2.66758	-0.321174	115.117203	74.173813
126	S-Citalopram_13.590_kcals_per_mol	5.889493	7.674368	5.056283	2.974912	5.129911	5.04106	0.47375	2.44772	89.916389	141.868805	2.108175	4.670245	-0.056366	113.052498	73.294563
127	S-Citalopram_13.619_kcals_per_mol	6.593572	5.768174	3.765035	2.906748	4.598807	5.037197	0.774214	1.90099	121.403641	97.125305	3.355022	2.036259	-2.414821	132.387665	101.384598
128	S-Citalopram_13.621_kcals_per_mol	5.938636	7.783247	5.020007	0.481773	5.087762	4.758193	2.31336	2.43759	91.991837	149.81665	1.6602	4.803115	-0.1845	117.880997	145.423996
129	S-Citalopram_13.676_kcals_per_mol	6.562425	5.926339	3.945557	2.988993	4.643187	5.045282	0.739161	1.89359	119.03009	100.0951	3.38354	2.2526	-2.267453	132.322296	100.700203
130	S-Citalopram_13.703_kcals_per_mol	6.533696	5.779537	1.795056	3.78172	4.089733	4.895619	2.659736	2.09377	137.383148	109.159615	-0.55735	2.799775	-2.951946	129.335342	184.250793
131	S-Citalopram_13.746_kcals_per_mol	6.255787	4.810531	1.26902	4.372736	4.673665	4.88882	1.24998	2.76493	108.925606	74.463058	4.25412	-0.605236	-1.681833	81.740227	57.659901
132	S-Citalopram_13.770_kcals_per_mol	6.820146	6.15567	2.913503	3.11411	4.397594	4.805963	2.24542	2.80788	137.089294	112.417091	-0.50662	2.95634	-3.22351	87.822968	213.185196
133	S-Citalopram_13.812_kcals_per_mol	5.950595	4.826107	2.13554	4.56239	4.625239	5.071091	0.40137	2.0219	101.996712	75.519531	4.41334	-0.76187	-1.06414	128.890503	93.280067
134	S-Citalopram_13.866_kcals_per_mol	6.861252	4.520342	0.3373	4.076147	4.354347	4.929908	1.48135	2.6188	140.930405	73.694038	2.51188	0.340241	-3.48984	88.550758	51.562099
135	S-Citalopram_13.878_kcals_per_mol	6.914415	4.69758	0.630769	4.12719	4.39029	4.930212	1.507215	2.6107	141.796799	76.853882	2.42905	0.6224	-3.555256	88.537178	51.24984
136	S-Citalopram_13.880_kcals_per_mol	4.851023	7.610651	4.481732	0.474349	4.729997	4.710312	2.389769	2.58044	74.756554	163.482071	1.14748	4.41586	1.22831	114.104301	148.691406
137	S-Citalopram_13.900_kcals_per_mol	5.509242	6.679039	3.954319	0.90464	4.088691	4.873209	1.760196	2.68044	102.785774	142.910294	-0.97309	3.894993	-0.862265	92.405121	47.018799
138	S-Citalopram_13.924_kcals_per_mol	5.451443	6.72574	4.242028	0.84675	4.521892	4.760205	2.31047	2.437	92.000786	126.996521	3.25309	3.09018	-0.24126	117.710197	145.641693
139	S-Citalopram_13.951_kcals_per_mol	6.220877	4.586583	1.53867	4.25385	4.574466	4.889758	1.175989	2.78204	110.474861	71.4729	4.08585	-0.802644	-1.753739	80.622726	59.612782
140	S-Citalopram_13.965_kcals_per_mol	4.88152	5.446744	0.359198	3.48285	4.003055	4.876349	1.363457	2.76693	88.379028	102.196274	3.79863	0.99733	-0.052951	81.116051	56.22823
141	S-Citalopram_13.975_kcals_per_mol	5.517201	6.793462	4.312157	0.79632	4.578132	4.759634	2.309588	2.43942	92.428169	127.535866	3.24865	3.17449	-0.275822	117.559601	145.613297
142	S-Citalopram_13.983_kcals_per_mol	6.436864	6.742755	4.13329	1.28926	4.634889	4.760136	2.293464	2.43154	115.638802	123.882385	2.119623	3.56034	-2.046687	118.321327	144.997604
143	S-Citalopram_13.997_kcals_per_mol	6.447595	6.851274	3.988836	2.54708	4.566571	4.836956	1.918597	2.76003	117.6922	130.364471	-0.00698	4.02659	-2.146208	86.333427	223.276901
144	S-Citalopram_14.002_kcals_per_mol	6.91871	4.438868	0.03954	3.85769	4.219462	4.843071	1.806555	2.80047	151.707809	74.227936	1.74763	0.26958	-3.796506	83.284569	224.652802
145	S-Citalopram_14.004_kcals_per_mol	6.166113	4.564879	1.67726	4.21392	4.570492	4.888887	1.17077	2.78862	109.020554	71.093048	4.111325	-0.87587	-1.643007	79.8162	60.171791

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
146	S-Citalopram_14.040_kcals_per_mol	4.541509	7.007318	3.888718	1.37759	4.191405	4.861312	1.897774	2.65794	77.235947	157.513626	1.27452	3.87069	0.8934	90.698013	223.817703
147	S-Citalopram_14.066_kcals_per_mol	5.87522	4.784394	2.025095	4.53523	4.62148	5.051806	0.5232	2.026	100.162056	74.70887	4.41	-0.92946	-0.91695	129.144699	94.962769
148	S-Citalopram_14.093_kcals_per_mol	4.831441	7.6034	4.529954	0.79135	4.755871	4.713413	2.329	2.5934	73.89859	160.20517	1.42694	4.33951	1.29136	113.483902	145.572205
149	S-Citalopram_14.146_kcals_per_mol	7.1467	5.60378	3.246299	4.093516	4.817234	4.979821	2.50248	1.9326	134.098175	88.891724	2.901	1.75578	-3.403561	131.687607	184.776794
150	S-Citalopram_14.162_kcals_per_mol	6.165826	6.366575	1.838729	4.357697	4.816657	4.872244	1.74634	2.73162	103.114296	107.831764	3.99223	2.2257	-1.279866	84.796059	47.1493
151	S-Citalopram_14.230_kcals_per_mol	5.74817	2.892974	0.146297	2.7604	3.539439	5.146401	0.605618	2.20905	125.583954	51.964428	2.48813	-1.107226	-2.171844	118.068497	72.511047
152	S-Citalopram_14.244_kcals_per_mol	6.92525	6.556086	3.443008	4.290711	4.757364	4.943465	2.56139	1.98758	127.853218	114.355072	1.02266	3.65518	-2.885734	130.7948	181.860703
153	S-Citalopram_14.256_kcals_per_mol	6.245617	4.8922	0.13061	4.49928	4.635457	4.954503	0.994431	2.52534	109.597298	76.768318	4.22929	-0.25243	-1.72776	90.165848	62.827709
154	S-Citalopram_14.282_kcals_per_mol	5.989824	7.664607	5.0491	2.916135	5.137467	5.043545	0.4741	2.44625	92.054314	141.055649	1.97681	4.73039	-0.246014	114.391701	73.134697
155	S-Citalopram_14.304_kcals_per_mol	4.487834	6.671957	4.076967	1.672836	4.137147	4.852235	2.008464	1.91361	77.097733	139.775024	2.55213	3.11759	0.94672	131.270493	135.318405
156	S-Citalopram_14.426_kcals_per_mol	4.422421	6.162088	3.415454	3.85535	4.56895	4.85441	1.96734	1.96463	68.49987	108.041397	4.13425	0.96139	1.5989	129.934998	133.780106
157	S-Citalopram_14.451_kcals_per_mol	7.211241	5.583378	0.90669	4.698041	4.846249	4.869791	1.61368	2.77287	135.417084	88.07119	2.87097	1.35354	-3.594244	82.956917	50.513302
158	S-Citalopram_14.457_kcals_per_mol	4.703095	6.485029	3.941477	3.49932	4.743675	4.847755	2.07746	1.89895	71.40609	112.714432	4.23137	1.487998	1.460415	131.018005	137.895203
159	S-Citalopram_14.469_kcals_per_mol	4.833162	7.142706	3.815765	1.79432	4.462231	4.830517	2.10217	2.7043	78.916924	148.402832	2.06087	3.84283	0.77939	88.712563	218.313507
160	S-Citalopram_14.498_kcals_per_mol	5.862763	7.646698	5.011508	2.884	5.089701	5.041741	0.475432	2.4502	90.101814	142.39743	2.00102	4.67354	-0.071595	114.127899	73.169258
161	S-Citalopram_14.510_kcals_per_mol	6.922374	4.330392	1.450856	3.89072	4.145681	4.969351	2.59774	1.8793	156.89592	72.960258	1.38955	0.63985	-3.841828	133.390396	191.326904
162	S-Citalopram_14.568_kcals_per_mol	6.128926	4.881261	0.25011	4.465164	4.626158	4.956389	0.931	2.5248	106.605156	76.687393	-4.29779	0.375577	-1.496468	89.727638	64.1026
163	S-Citalopram_14.587_kcals_per_mol	4.956507	3.196209	2.15734	2.726547	3.441467	4.863193	1.13487	2.79225	102.286331	59.472672	2.92244	-1.502	-0.82898	78.030548	63.281399
164	S-Citalopram_14.651_kcals_per_mol	4.818657	6.550873	4.026316	3.55397	4.80703	4.847785	2.03492	1.93763	72.725952	112.944191	4.30082	1.57614	-1.36669	130.539307	136.1436
165	S-Citalopram_14.770_kcals_per_mol	6.001868	7.452925	3.970478	2.23502	4.734702	4.73877	2.421594	2.4831	101.012932	150.669785	-0.453376	4.65362	-0.841189	116.4767	151.164993
166	S-Citalopram_14.828_kcals_per_mol	4.91483	7.588011	4.462726	0.22048	4.697536	4.709526	2.42156	2.57863	76.671005	164.575455	1.0171	4.45524	-1.07324	65.798302	150.428207
167	S-Citalopram_14.845_kcals_per_mol	5.977912	4.874672	2.257366	4.58359	4.606832	5.189302	0.118265	1.95307	102.871796	76.89743	4.39897	-0.450401	-1.156927	127.2799	79.124519
168	S-Citalopram_14.941_kcals_per_mol	4.861857	7.568986	4.312187	0.06002	4.65051	4.714713	2.507848	2.54139	76.357452	168.265411	0.70521	4.46395	1.10387	115.723198	156.050598
169	S-Citalopram_15.001_kcals_per_mol	5.444537	6.564736	4.290547	3.046929	4.426671	5.180628	2.26962	2.11988	93.227394	123.932701	2.840615	3.345199	-0.344565	121.748482	210.212494
170	S-Citalopram_15.063_kcals_per_mol	6.40096	6.519641	4.01557	3.42538	4.635763	5.04581	0.495159	2.45579	114.294922	116.615829	2.266161	3.504747	-1.988532	114.654732	72.432098

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
171	S-Citalopram_15.126_kcals_per_mol	5.896742	5.041533	2.495658	4.66926	4.629876	5.189209	0.08142	1.91723	100.263351	80.042976	4.48781	-0.29907	-0.94935	128.078094	80.153198
172	S-Citalopram_15.157_kcals_per_mol	6.782777	5.715854	1.676183	4.0486	4.248036	4.921146	2.629882	2.02225	142.113434	103.558167	-0.522562	2.65579	-3.29467	130.646606	185.165207
173	S-Citalopram_15.237_kcals_per_mol	4.33996	6.747425	3.934718	0.97889	4.193488	4.727224	2.53199	2.4339	72.967293	140.743256	2.51002	3.11732	1.18189	118.302803	158.503998
174	S-Citalopram_15.263_kcals_per_mol	7.155494	4.970819	0.039583	4.50446	4.650797	4.84992	1.796919	2.80939	141.330429	78.236885	2.628195	0.48676	-3.7522	82.780373	244.870895
175	S-Citalopram_15.277_kcals_per_mol	5.429535	6.589427	4.470526	4.22158	4.772616	5.040645	0.79395	1.86855	86.43898	114.914192	4.25273	2.16875	0.27226	132.238602	102.493301
176	S-Citalopram_15.280_kcals_per_mol	6.315082	4.26144	0.113326	3.916901	4.123413	4.943807	1.45909	2.4824	126.72773	71.934746	3.119	0.05133	-2.60636	89.71019	53.4174
177	S-Citalopram_15.285_kcals_per_mol	5.670229	7.359921	3.680012	1.91603	4.577411	4.745663	2.346482	2.44557	96.126236	154.828476	-0.685475	4.52602	-0.416152	118.022003	147.376495
178	S-Citalopram_15.349_kcals_per_mol	5.268024	7.194711	4.60178	1.22739	4.747445	4.761655	2.26856	2.42612	83.426674	136.990738	2.99993	3.62149	0.4775	118.142403	143.772003
179	S-Citalopram_15.392_kcals_per_mol	6.018812	7.452884	4.021456	2.2654	4.740363	4.741119	2.450346	2.48279	101.318787	150.334259	-0.37582	4.660425	-0.869084	116.289391	152.884598
180	S-Citalopram_15.433_kcals_per_mol	6.22988	4.265713	1.526217	3.96003	4.148894	5.152352	0.756492	2.25607	122.767555	71.554466	3.3464	0.03814	-2.374392	118.231201	65.753532
181	S-Citalopram_15.459_kcals_per_mol	6.154568	7.449764	4.024103	2.50338	4.782028	4.742888	2.493352	2.48238	103.879684	147.735657	-0.35634	4.658446	-1.087168	63.638191	155.669006
182	S-Citalopram_15.460_kcals_per_mol	6.828699	5.412234	3.300652	4.17905	4.800148	5.117133	0.22556	1.96033	122.873337	84.996376	3.812	1.176508	-2.66938	130.322998	85.336777
183	S-Citalopram_15.484_kcals_per_mol	6.368377	6.111452	2.841362	3.12235	4.078134	4.744709	2.46447	2.47545	131.227478	119.817673	0.17358	3.089812	-2.66565	116.775803	154.159103
184	S-Citalopram_15.555_kcals_per_mol	6.254859	6.012955	4.076548	2.35877	4.601536	4.913922	2.590777	1.90502	110.945267	103.277695	3.72711	2.07037	-1.69647	133.043304	181.418198
185	S-Citalopram_15.567_kcals_per_mol	6.377226	6.064562	2.744937	3.204	4.055403	4.740639	2.47448	2.4957	132.49295	118.943512	0.03913	3.02384	-2.713669	115.881401	154.466705
186	S-Citalopram_15.576_kcals_per_mol	6.28805	6.080044	4.130323	2.411091	4.63698	4.912863	2.59042	1.91265	110.980171	104.207481	3.71197	2.16433	-1.710623	132.872498	181.454895
187	S-Citalopram_15.585_kcals_per_mol	7.131011	4.618261	1.843394	4.038799	4.446034	4.974563	2.46244	1.90501	151.166046	74.137856	1.90191	0.72611	-3.936758	132.539398	174.244797
188	S-Citalopram_15.593_kcals_per_mol	4.504199	6.748565	4.151427	1.95297	4.247329	4.852704	1.96804	1.93241	75.575562	138.383499	2.71919	3.08103	1.07306	131.046204	133.801193
189	S-Citalopram_15.595_kcals_per_mol	4.367346	6.682403	4.131801	0.318247	4.660706	4.985142	2.561503	1.52085	65.820648	121.040016	3.8573	1.76971	1.92866	139.691605	188.008606
190	S-Citalopram_15.763_kcals_per_mol	5.975768	4.914479	2.367785	4.59647	4.599517	5.186879	0.081805	1.93466	102.985542	77.862297	4.401041	-0.35007	-1.159996	127.589363	80.3265
191	S-Citalopram_16.029_kcals_per_mol	4.969188	6.852631	4.47034	2.698933	4.635128	4.866753	1.845425	1.99263	78.895927	127.647789	3.574534	2.807114	0.86918	129.670654	129.856903
192	S-Citalopram_16.058_kcals_per_mol	6.577971	7.170398	4.38362	1.970545	4.854043	4.752605	2.35837	2.45598	113.850288	131.84491	1.43326	4.19262	-1.972821	117.282501	148.055206
193	S-Citalopram_16.061_kcals_per_mol	5.588441	7.221912	3.884796	0.58885	4.565235	5.042899	0.485849	2.45553	94.223068	146.835403	-0.44375	4.543697	-0.315269	114.318581	72.941002
194	S-Citalopram_16.085_kcals_per_mol	5.990248	4.801491	2.139657	4.528291	4.588524	5.186867	0.12108	1.9663	103.605766	75.66806	4.354038	-0.50788	-1.214425	126.537498	79.714378
195	S-Citalopram_16.123_kcals_per_mol	5.436732	5.573564	1.574544	2.7155	3.363748	4.782751	2.627538	2.44765	120.370369	123.604958	-1.5251	2.561	-1.62378	117.519798	169.926697

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
196	S-Citalopram_16.135_kcals_per_mol	4.9994	6.842805	4.47399	2.82198	4.666904	4.871491	1.81843	1.99004	78.983482	126.228203	3.67441	2.72978	0.86582	129.601105	129.111496
197	S-Citalopram_16.185_kcals_per_mol	6.6458	5.951672	3.765682	4.30062	4.640366	5.172508	2.308777	2.09187	121.112823	100.580162	2.70841	2.79232	-2.486728	122.915703	208.591507
198	S-Citalopram_16.217_kcals_per_mol	4.414973	6.668324	4.171132	0.262195	4.66118	4.982629	2.563645	1.49656	66.777916	120.594727	3.8918	1.76919	1.86244	140.060806	197.937195
199	S-Citalopram_16.286_kcals_per_mol	5.304221	7.08678	4.437953	3.775716	4.769753	5.049509	0.41546	2.40766	83.711304	131.651749	3.266505	3.428628	0.42269	115.548897	74.273743
200	S-Citalopram_16.331_kcals_per_mol	3.739131	6.12818	3.508342	0.882985	3.76664	5.153482	2.36201	1.39603	66.575333	130.630356	2.78004	2.0232	1.54407	137.549103	204.946899
201	S-Citalopram_16.531_kcals_per_mol	6.305679	4.371194	1.819225	4.08595	4.359756	5.110598	0.02112	2.09251	119.03154	70.437538	3.71766	-0.3839	-2.20889	126.309601	80.378166
202	S-Citalopram_16.532_kcals_per_mol	5.165296	5.011348	3.334236	1.065796	3.985298	4.967947	2.57206	1.56296	95.815643	91.253136	3.91833	0.41245	-0.44264	139.721405	187.327301
203	S-Citalopram_16.626_kcals_per_mol	6.450945	5.90265	2.45373	3.69848	4.041929	4.781873	2.603518	2.4178	136.197006	114.115677	-0.069354	2.84183	-2.886741	118.879402	167.762604
204	S-Citalopram_16.652_kcals_per_mol	4.701764	6.641087	4.346329	0.058286	4.769341	4.977299	2.566509	1.47283	70.864349	116.672272	4.18222	1.6582	1.58542	140.706299	187.645706
205	S-Citalopram_16.660_kcals_per_mol	4.27288	6.223992	3.88601	1.42878	3.988402	5.14703	2.391443	1.47779	74.561722	126.338753	3.11599	2.236201	1.094572	135.867401	205.094803
206	S-Citalopram_16.799_kcals_per_mol	6.335804	4.373991	0.01219	4.105323	4.288407	4.961657	1.14955	2.48359	122.080772	71.648979	3.46612	-0.225342	-2.422187	90.897781	58.8088
207	S-Citalopram_16.808_kcals_per_mol	4.751679	7.173917	4.350724	0.748321	4.526806	4.722316	2.501115	2.50575	76.148895	146.153793	2.35278	3.70578	1.02754	116.731003	155.938095
208	S-Citalopram_16.811_kcals_per_mol	6.448724	5.857688	2.37458	3.72854	4.020493	4.783736	2.617391	2.41785	137.004608	113.311852	-0.118941	2.78658	-2.909468	119.008301	169.365295
209	S-Citalopram_16.815_kcals_per_mol	5.517049	5.751163	3.890072	2.12849	4.710601	5.156244	2.366054	1.44475	89.35482	94.268074	4.64382	0.579115	0.007681	136.371399	205.096603
210	S-Citalopram_16.817_kcals_per_mol	6.488681	6.890963	3.628996	2.75715	4.561327	4.744516	2.359791	2.45555	119.336754	131.736115	0.18694	3.99953	-2.202614	117.731796	148.117905
211	S-Citalopram_16.859_kcals_per_mol	4.666145	3.734403	0.843261	3.13916	3.508785	5.193251	0.259	1.95505	92.668282	70.023399	3.21752	-1.22882	-0.2775	125.290901	78.03968
212	S-Citalopram_16.870_kcals_per_mol	3.924755	4.473753	2.037653	3.692084	3.524501	5.11634	0.897825	1.40649	74.501892	86.871727	3.27712	-0.90159	0.922718	140.087799	104.528702
213	S-Citalopram_16.956_kcals_per_mol	4.457418	6.682086	4.213261	0.193628	4.6653	4.979903	2.566531	1.50741	67.57412	120.926422	3.89564	1.83336	1.80207	140.091293	187.9021
214	S-Citalopram_17.038_kcals_per_mol	4.25831	6.661273	4.033535	0.42164	4.611446	4.985337	2.56324	1.54953	64.40274	121.82074	3.759982	1.761148	2.00693	138.954514	187.777695
215	S-Citalopram_17.068_kcals_per_mol	6.484793	6.955051	3.712587	2.689676	4.597774	4.74331	2.35307	2.46223	118.147423	132.905838	0.213421	4.0743	-2.13758	117.51619	147.696198
216	S-Citalopram_17.254_kcals_per_mol	6.076308	5.2487	0.1101	4.28827	4.54084	4.863742	1.578919	2.74318	107.247887	86.232933	4.192354	0.364846	-1.520414	83.751602	50.324902
217	S-Citalopram_17.283_kcals_per_mol	6.240521	6.219627	4.306985	1.98855	4.863059	4.924794	2.576183	1.91136	104.105644	102.646301	4.2831	1.88964	-1.253239	132.672195	179.985199
218	S-Citalopram_17.298_kcals_per_mol	4.283357	6.29513	3.924935	1.317596	4.105715	5.143974	2.39611	1.48467	72.887039	124.950172	3.282499	2.129697	1.23633	135.779922	204.602402
219	S-Citalopram_17.302_kcals_per_mol	5.39322	6.065724	4.184618	2.08005	4.769916	5.161364	2.349855	1.39272	85.44738	100.64045	4.63208	0.976349	0.362492	136.891907	205.413605
220	S-Citalopram_17.325_kcals_per_mol	5.841961	5.449676	0.085225	4.18993	4.497002	4.866697	1.541409	2.75257	101.958344	91.725342	4.23148	0.66785	-1.112404	82.531754	51.526531

										angle	angle				explicit	explicit
				distance	distance	distance		distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane
		(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)	(angstroms)	(angstroms)	(angstroms)	(degrees)	(degrees)
221	S-Citalopram_17.359_kcals_per_mol	4.152347	6.690853	3.476263	1.46245	3.846461	5.118779	0.205422	2.119	74.450523	158.865967	0.92165	3.59745	1.05307	123.920502	77.829178
222	S-Citalopram_17.396_kcals_per_mol	5.493875	5.882125	4.051983	2.16806	4.736525	5.159367	2.354795	1.40889	88.336891	96.856224	4.64564	0.7698	0.10798	136.805695	205.329102
223	S-Citalopram_17.414_kcals_per_mol	3.166068	5.50514	2.695144	1.13077	3.671313	4.952781	2.58811	1.57742	56.114967	111.950043	2.98964	0.57482	2.02267	140.279907	187.806595
224	S-Citalopram_17.670_kcals_per_mol	5.30831	6.740975	4.512121	2.492773	4.706613	5.159101	2.37616	1.90273	84.709389	121.088684	3.721544	2.830245	0.36616	127.164513	204.806793
225	S-Citalopram_17.724_kcals_per_mol	5.507049	5.525394	3.652398	2.03293	4.628986	5.148624	2.392365	1.45563	90.718582	90.625717	4.58094	0.30206	-0.116143	136.296005	24.850071
226	S-Citalopram_17.824_kcals_per_mol	5.536049	5.684962	3.857501	2.17668	4.675781	5.15523	2.376812	1.43809	90.472397	93.408165	4.60937	0.55748	-0.083974	136.348999	205.280197
227	S-Citalopram_17.862_kcals_per_mol	6.417928	6.720751	3.800414	1.787227	4.586622	5.040639	0.55548	2.4797	116.103996	124.603935	0.37305	4.09788	-2.032756	113.462097	71.387001
228	S-Citalopram_18.209_kcals_per_mol	2.771981	6.5439	2.733016	1.353754	3.752679	5.068839	0.23009	2.32758	47.127995	154.324921	1.27059	2.461288	2.52305	120.056183	78.164833
229	S-Citalopram_18.601_kcals_per_mol	4.906754	5.424735	2.832499	4.2225	4.188679	5.208512	0.268514	1.93595	85.286674	97.367813	4.121518	0.519843	0.2371	126.363289	76.007103
230	S-Citalopram_18.860_kcals_per_mol	5.274194	6.178856	4.249069	1.060496	4.596223	4.991771	2.55074	1.65662	86.226746	107.752495	4.3139	1.53507	0.28328	136.4328	187.884293
231	S-Citalopram_19.102_kcals_per_mol	5.291896	6.173758	4.25404	1.021491	4.604072	4.984659	2.5593	1.6557	86.483559	107.423981	4.32884	1.51869	0.26288	136.564301	187.265198
232	S-Citalopram_19.181_kcals_per_mol	5.39026	5.792012	3.942619	1.49707	4.402606	4.994636	2.538375	1.70964	92.698029	102.018677	4.19643	1.26519	-0.245983	135.8255	189.617294
233	S-Citalopram_19.634_kcals_per_mol	5.02013	6.297687	4.312664	1.940054	4.513255	5.141773	2.40512	1.46266	81.891075	113.078537	4.06007	1.84147	0.64485	136.277298	204.4543

										score	score				score	score	
				score	score	score		score	score	angle	angle	score	score	score	explicit	explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
0	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153594	0.0771478	0.124988	0.754554	0.271219	0.0164284	0.297398	0.14425	0.152638	0.149399	0.265786	1.1663	0.349832	0.0475503	0.257602	0.281912
1	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.1537	0.0771508	0.125963	0.754552	0.271203	0.016354	0.296726	0.143544	0.152732	0.149376	0.266668	1.16627	0.350343	0.0477003	0.257338	0.281975
2	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153481	0.07712	0.125702	0.754487	0.271233	0.0161165	0.296925	0.145011	0.152127	0.149402	0.267214	1.1663	0.350332	0.0483274	0.257729	0.2821
3	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153587	0.077123	0.126678	0.754485	0.271217	0.0160421	0.296253	0.144305	0.152222	0.149379	0.268097	1.16627	0.350844	0.0484774	0.257465	0.282163
4	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153591	0.0771508	0.127078	0.754604	0.271224	0.0163291	0.297129	0.144323	0.152659	0.149392	0.265563	1.17217	0.349823	0.0479089	0.257634	0.282438
5	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153697	0.0771537	0.128053	0.754602	0.271208	0.0162547	0.296458	0.143617	0.152754	0.14937	0.266445	1.17214	0.350335	0.0480589	0.25737	0.282501
6	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153478	0.077123	0.127793	0.754537	0.271238	0.0160172	0.296656	0.145084	0.152149	0.149395	0.266992	1.17217	0.350324	0.0486859	0.257761	0.282627
7	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153584	0.0771259	0.128767	0.754535	0.271222	0.0159428	0.295985	0.144378	0.152244	0.149373	0.267874	1.17214	0.350836	0.0488359	0.257497	0.282689
8	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161514	0.0826817	0.117494	0.765364	0.264719	0.0166064	0.303372	0.141033	0.17194	0.138632	0.267795	1.1663	0.3498	0.0467129	0.254454	0.283228
9	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161619	0.0826846	0.11847	0.765362	0.264703	0.016532	0.302702	0.140328	0.172035	0.138609	0.268677	1.16627	0.350312	0.0468629	0.25419	0.28329

Appendix 3.5 The best scoring, most similar, 251 conformational comparison groups of the 1,297,344 combinatorial comparisons.

				score	score	score		score	score	score	score			score	score	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
10	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161409	0.0826584	0.118197	0.765314	0.264728	0.0163054	0.302972	0.141814	0.171482	0.138625	0.269384	1.1663	0.350332	0.047329	0.254562	0.283427
11	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161515	0.0826613	0.119173	0.765312	0.264712	0.016231	0.302302	0.141108	0.171577	0.138602	0.270266	1.16627	0.350844	0.047479	0.254298	0.28349
12	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161511	0.0826847	0.119581	0.765414	0.264724	0.0165071	0.303103	0.141106	0.171962	0.138625	0.267572	1.17217	0.349792	0.0470715	0.254485	0.283754
13	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161616	0.0826876	0.120557	0.765412	0.264708	0.0164327	0.302433	0.140401	0.172057	0.138603	0.268454	1.17214	0.350304	0.0472215	0.254221	0.283816
14	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152362	0.0827141	0.132434	0.75769	0.272425	0.0172101	0.302509	0.141745	0.150221	0.159347	0.270489	1.1663	0.349877	0.0463005	0.256816	0.283896
15	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161406	0.0826613	0.120284	0.765364	0.264733	0.0162061	0.302704	0.141886	0.171504	0.138619	0.269161	1.17217	0.350324	0.0476876	0.254594	0.283953
16	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152467	0.082717	0.133408	0.757688	0.272409	0.0171357	0.301839	0.141039	0.150316	0.159324	0.271371	1.16627	0.350388	0.0464505	0.256551	0.283958
17	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161512	0.0826643	0.12126	0.765362	0.264717	0.0161317	0.302033	0.141181	0.171599	0.138596	0.270043	1.17214	0.350835	0.0478376	0.25433	0.284016
18	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152248	0.0826916	0.133198	0.757755	0.272441	0.0168982	0.30208	0.142523	0.149707	0.159361	0.271882	1.1663	0.350292	0.0465561	0.256937	0.284058
19	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152353	0.0826945	0.134171	0.757753	0.272425	0.0168238	0.301409	0.141817	0.149801	0.159339	0.272763	1.16627	0.350804	0.0467061	0.256672	0.28412
20	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152359	0.0827171	0.134526	0.75762	0.27243	0.0171108	0.302241	0.141818	0.150243	0.15934	0.270266	1.17217	0.349868	0.0466591	0.256847	0.284414
21	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152464	0.08272	0.1355	0.757618	0.272414	0.0170364	0.30157	0.141112	0.150337	0.159318	0.271148	1.17214	0.35038	0.0468091	0.256583	0.284476
22	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152245	0.0826946	0.13529	0.757685	0.272446	0.0167989	0.301811	0.142596	0.149729	0.159355	0.271658	1.17217	0.350284	0.0469147	0.256968	0.284576
23	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152351	0.0826975	0.136263	0.757683	0.27243	0.0167245	0.301141	0.14189	0.149823	0.159332	0.27254	1.17214	0.350795	0.0470647	0.256704	0.284638

				score	score	score		score	score	score angle	score angle	score	score	score	score explicit	score explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
24	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150678	0.0839467	0.226934	0.764245	0.278571	0.0168946	0.302526	0.148539	0.164721	0.149353	0.311367	1.11993	0.353145	0.0481599	0.255235	0.291616
25	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158601	0.0857837	0.219222	0.756106	0.272091	0.0170726	0.308495	0.145324	0.184	0.138586	0.313395	1.11993	0.353113	0.0473224	0.255599	0.291643
26	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150783	0.0839437	0.227877	0.764243	0.278555	0.0168202	0.301854	0.147834	0.164815	0.149331	0.312237	1.11991	0.353657	0.0483099	0.254971	0.291676
27	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158707	0.0857807	0.220166	0.756104	0.272075	0.0169982	0.307824	0.144619	0.184094	0.138564	0.314265	1.11991	0.353626	0.0474724	0.255335	0.291703
28	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150565	0.0839374	0.227668	0.764294	0.278585	0.0165827	0.302054	0.1493	0.164211	0.149356	0.312809	1.11993	0.353645	0.0489369	0.255363	0.291815
29	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158497	0.085776	0.219947	0.756145	0.272101	0.0167716	0.308096	0.146104	0.183542	0.138579	0.315	1.11993	0.353644	0.0479386	0.255486	0.291837
30	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15067	0.0839345	0.228611	0.764292	0.278569	0.0165083	0.301381	0.148595	0.164305	0.149334	0.313679	1.11991	0.354157	0.0490869	0.255098	0.291875
31	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158602	0.085773	0.220891	0.756143	0.272085	0.0166972	0.307425	0.145399	0.183637	0.138557	0.315869	1.11991	0.354157	0.0480886	0.255221	0.291897
32	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147716	0.0789854	0.247563	0.770016	0.284278	0.0168649	0.29936	0.1543	0.165384	0.154915	0.368638	1.02804	0.358859	0.0504893	0.254902	0.29202
33	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147822	0.0789824	0.248497	0.770014	0.284262	0.0167906	0.298688	0.153596	0.165478	0.154893	0.369496	1.02803	0.359373	0.0506394	0.254638	0.29208
34	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150678	0.0839351	0.229642	0.764178	0.278577	0.0167976	0.302271	0.148579	0.164796	0.149345	0.311131	1.12491	0.353121	0.0485157	0.255262	0.292116
35	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155643	0.0828686	0.239789	0.761883	0.27782	0.017043	0.305333	0.151087	0.184661	0.144157	0.369264	1.02804	0.358828	0.049652	0.255916	0.292132
36	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158601	0.0857722	0.221923	0.756141	0.272097	0.0169756	0.308241	0.145364	0.184074	0.138578	0.313159	1.12491	0.353089	0.0476782	0.255574	0.292145
37	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147603	0.0789762	0.248303	0.770065	0.284292	0.016553	0.298887	0.15506	0.164874	0.154918	0.369083	1.02804	0.359357	0.0512663	0.255029	0.292154

				score	score	score			score		score	score								score	score		
		score	score	distance	distance	distance	score		distance		distance	angle	angle							explicit	explicit		
		distance	distance	absolute value	absolute value	N to	distance	score	absolute value	score	absolute value	connecting C	connecting C							Ar1 plane to	Ar2 plane to	similarity	
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	distance	to N	to N		N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score				
38	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150783	0.0839321	0.230584	0.764176	0.278561	0.0167232	0.301599	0.147874	0.16489	0.149323	0.312001	1.1249	0.353633	0.0486657	0.254998	0.292176							
39	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155748	0.0828657	0.240724	0.761881	0.277804	0.0169686	0.304662	0.150383	0.184756	0.144134	0.370122	1.02803	0.359342	0.049802	0.255652	0.292192							
40	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158707	0.0857692	0.222866	0.756139	0.272081	0.0169012	0.30757	0.144659	0.184169	0.138556	0.314029	1.1249	0.353602	0.0478283	0.255309	0.292206							
41	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147709	0.0789732	0.249237	0.770062	0.284277	0.0164787	0.298215	0.154356	0.164969	0.154895	0.369942	1.02803	0.359871	0.0514163	0.254765	0.292213							
42	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155538	0.0828453	0.240519	0.761922	0.277829	0.016742	0.304933	0.151866	0.184204	0.14415	0.369759	1.02804	0.359357	0.0502681	0.255803	0.292251							
43	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155644	0.0828424	0.241454	0.76192	0.277813	0.0166676	0.304262	0.151162	0.184299	0.144127	0.370617	1.02803	0.359871	0.0504181	0.255538	0.292311							
44	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150565	0.0839259	0.230376	0.764227	0.278591	0.0164857	0.301799	0.14934	0.164286	0.149348	0.312573	1.12491	0.353621	0.0492927	0.255389	0.292316							
45	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158497	0.0857644	0.222648	0.756092	0.272107	0.0166746	0.307841	0.146144	0.183617	0.138572	0.314764	1.12491	0.35362	0.0482944	0.25546	0.292334							
46	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15067	0.0839229	0.231318	0.764225	0.278575	0.0164113	0.301127	0.148635	0.16438	0.149326	0.313443	1.1249	0.354133	0.0494427	0.255125	0.292375							
47	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158602	0.0857615	0.223591	0.75609	0.272091	0.0166002	0.30717	0.145439	0.183712	0.138549	0.315634	1.1249	0.354133	0.0484444	0.255196	0.292394							
48	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147722	0.0789739	0.250476	0.769946	0.284281	0.0167696	0.299118	0.154503	0.165517	0.154921	0.36778	1.03176	0.358812	0.0508883	0.254927	0.292426							
49	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147827	0.078971	0.251409	0.769943	0.284266	0.0166952	0.298446	0.153799	0.165612	0.154899	0.368639	1.03175	0.359326	0.0510383	0.254663	0.292486							
50	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155649	0.0828648	0.242692	0.761813	0.277823	0.0169476	0.30509	0.15129	0.184795	0.144163	0.368406	1.03176	0.358781	0.0500509	0.255892	0.292534							
51	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147609	0.0789647	0.251216	0.769994	0.284296	0.0164577	0.298645	0.155263	0.165008	0.154924	0.368225	1.03176	0.359311	0.0516652	0.255055	0.292559							

		score distance N to Ar1	score distance N to Ar2	score distance absolute value N to Ar1 plane	score distance absolute value N to Ar2 plane	score distance N to connecting C	score distance Ar1 to Ar2	score distance absolute value Ar1 to Ar2 plane	score distance absolute value Ar2 to Ar1 plane	score angle Ar1 to connecting C to N	score angle Ar2 to connecting C to N	score explicit distance N to YZ plane	score explicit distance N to XY plane	score explicit distance N to XZ plane	score explicit angle Ar1 plane to YZ plane	score explicit angle Ar2 plane to YZ plane	similarity score
52	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155754	0.0828619	0.243626	0.76181	0.277807	0.0168732	0.30442	0.150586	0.184889	0.14414	0.369265	1.03175	0.359295	0.0502009	0.255628	0.292594
53	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.147714	0.0789617	0.252149	0.769992	0.28428	0.0163833	0.297973	0.154559	0.165102	0.154901	0.369084	1.03175	0.359824	0.0518153	0.25479	0.292619
54	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155544	0.0828415	0.243423	0.761851	0.277832	0.0166466	0.304691	0.152069	0.184337	0.144156	0.368901	1.03176	0.35931	0.050667	0.255779	0.292654
55	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15565	0.0828386	0.244357	0.761849	0.277816	0.0165722	0.30402	0.151365	0.184432	0.144133	0.369759	1.03175	0.359824	0.050817	0.255514	0.292713
56	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158955	0.0880379	0.217388	0.756445	0.271348	0.0169239	0.30727	0.142565	0.183504	0.139955	0.307701	1.1541	0.352511	0.0465897	0.254983	0.293219
57	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151032	0.0862008	0.225095	0.76379	0.277829	0.0167459	0.301299	0.145781	0.164224	0.150724	0.305674	1.1541	0.352542	0.0474272	0.255882	0.293223
58	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.159061	0.0880349	0.218333	0.756443	0.271332	0.0168495	0.306599	0.14186	0.183598	0.139932	0.308573	1.15408	0.353023	0.0467397	0.254718	0.293278
59	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151138	0.0861978	0.226038	0.763788	0.277814	0.0166715	0.300627	0.145076	0.164318	0.150701	0.306546	1.15408	0.353055	0.0475772	0.255618	0.293283
60	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158851	0.0880302	0.218112	0.756395	0.271357	0.0166229	0.30687	0.143345	0.183046	0.139948	0.309305	1.1541	0.353042	0.0472059	0.254869	0.293407
61	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150919	0.0861916	0.225828	0.763838	0.277844	0.016434	0.300827	0.146542	0.163714	0.150726	0.307116	1.1541	0.353042	0.0482042	0.256009	0.293423
62	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158956	0.0880272	0.219057	0.756393	0.271341	0.0165485	0.306199	0.14264	0.183141	0.139926	0.310177	1.15408	0.353554	0.0473559	0.254605	0.293467
63	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151025	0.0861886	0.226771	0.763836	0.277828	0.0163596	0.300155	0.145837	0.163808	0.150704	0.307988	1.15408	0.353554	0.0483542	0.255745	0.293482
64	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149445	0.0857945	0.234565	0.770065	0.279773	0.0176763	0.307634	0.146035	0.162307	0.159301	0.316116	1.11993	0.35319	0.0469101	0.254449	0.293546
65	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149551	0.0857915	0.235507	0.770063	0.279757	0.0176019	0.306962	0.14533	0.162401	0.159279	0.316986	1.11991	0.353702	0.0470601	0.254184	0.293606

				score	score	score		score	score	score angle	score angle	score	score	score	score explicit	score explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
66	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149331	0.085787	0.235346	0.77013	0.279788	0.0173643	0.307204	0.146813	0.161793	0.159316	0.317523	1.11993	0.353604	0.0471656	0.25457	0.293711
67	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158953	0.0880256	0.220003	0.756494	0.271355	0.0168275	0.307015	0.142651	0.18356	0.139946	0.307472	1.1595	0.352495	0.0469565	0.254957	0.293747
68	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15103	0.0861885	0.227716	0.763726	0.277837	0.0166495	0.301044	0.145867	0.16428	0.150715	0.305444	1.1595	0.352527	0.0477939	0.255909	0.293748
69	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149437	0.085784	0.236287	0.770128	0.279772	0.0172899	0.306533	0.146108	0.161888	0.159293	0.318392	1.11991	0.354117	0.0473156	0.254305	0.293771
70	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.159059	0.0880226	0.220946	0.756492	0.271339	0.0167531	0.306344	0.141946	0.183654	0.139923	0.308344	1.15947	0.353008	0.0471065	0.254692	0.293807
71	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151136	0.0861855	0.228659	0.763724	0.277821	0.0165751	0.300372	0.145162	0.164374	0.150692	0.306317	1.15947	0.353039	0.0479439	0.255645	0.293808
72	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146484	0.0829011	0.255251	0.775826	0.285476	0.0176467	0.30447	0.151797	0.16297	0.164854	0.370103	1.02804	0.358904	0.0492397	0.254116	0.293872
73	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146589	0.0828981	0.256184	0.775824	0.28546	0.0175723	0.3038	0.151094	0.163065	0.164831	0.370961	1.02803	0.359418	0.0493897	0.253851	0.293931
74	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158849	0.0880178	0.220727	0.756445	0.271364	0.0165265	0.306616	0.143431	0.183102	0.139939	0.309076	1.1595	0.353026	0.0475726	0.254843	0.293935
75	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150917	0.0861793	0.22845	0.763775	0.277851	0.0163376	0.300572	0.146628	0.16377	0.150717	0.306886	1.1595	0.353027	0.048571	0.256036	0.293948
76	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.14637	0.0828785	0.256038	0.77589	0.285491	0.0173347	0.304041	0.152575	0.162457	0.164869	0.370536	1.02804	0.359317	0.0494952	0.254237	0.293971
77	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158954	0.0880149	0.221671	0.756443	0.271348	0.0164521	0.305945	0.142726	0.183197	0.139917	0.309948	1.15947	0.353538	0.0477226	0.254579	0.293995
78	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151023	0.0861763	0.229393	0.763772	0.277835	0.0162632	0.2999	0.145923	0.163865	0.150695	0.307758	1.15947	0.353539	0.048721	0.255772	0.294007
79	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146475	0.0828756	0.25697	0.775888	0.285476	0.0172603	0.30337	0.151871	0.162551	0.164846	0.371393	1.02803	0.359831	0.0496453	0.253972	0.294031

		score distance N to Ar1	score distance N to Ar2	score distance absolute value N to Ar1 plane	score distance absolute value N to Ar2 plane	score distance N to connecting C	score distance Ar1 to Ar2	score distance absolute value Ar1 to Ar2 plane	score distance absolute value Ar2 to Ar1 plane	score angle Ar1 to connecting C to N	score angle Ar2 to connecting C to N	score explicit distance N to YZ plane	score explicit distance N to XY plane	score explicit distance N to XZ plane	score explicit angle Ar1 plane to YZ plane	score explicit angle Ar2 plane to YZ plane	similarity score
80	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149445	0.0857829	0.237281	0.769996	0.279779	0.0175793	0.307379	0.146075	0.162382	0.159293	0.31588	1.12491	0.353166	0.0472659	0.254475	0.294046
81	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149551	0.08578	0.238221	0.769996	0.279763	0.0175049	0.306708	0.14537	0.162476	0.159271	0.31675	1.1249	0.353678	0.0474159	0.254211	0.294106
82	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149331	0.0857754	0.233062	0.770063	0.279794	0.0172674	0.30695	0.146853	0.161868	0.159308	0.317287	1.12491	0.353581	0.0475215	0.254596	0.294211
83	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149437	0.0857725	0.239002	0.770061	0.279778	0.017193	0.306279	0.146148	0.161963	0.159285	0.318156	1.1249	0.354093	0.0476715	0.254332	0.294271
84	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146489	0.0828973	0.258173	0.775755	0.285479	0.0175513	0.304228	0.152	0.163104	0.16486	0.369245	1.03176	0.358857	0.0496387	0.254141	0.294278
85	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146595	0.0828943	0.259104	0.775753	0.285463	0.0174769	0.303557	0.151297	0.163198	0.164837	0.370103	1.03175	0.359371	0.0497887	0.253876	0.294338
86	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146375	0.0828747	0.258959	0.77582	0.285495	0.0172394	0.303799	0.152778	0.16259	0.164874	0.369678	1.03176	0.359271	0.0498942	0.254262	0.294378
87	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.146481	0.0828718	0.25989	0.775818	0.285479	0.017165	0.303128	0.152074	0.162685	0.164852	0.370535	1.03175	0.359784	0.0500442	0.253997	0.294437
88	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.1498	0.0880487	0.232721	0.76961	0.279032	0.0175276	0.306408	0.143277	0.16181	0.160673	0.310421	1.1541	0.352587	0.0461773	0.255096	0.295153
89	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149905	0.0880457	0.233663	0.769608	0.279016	0.0174532	0.305737	0.142571	0.161904	0.160651	0.311293	1.15408	0.353099	0.0463273	0.254831	0.295212
90	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149686	0.0880412	0.239501	0.769675	0.279047	0.0172157	0.305979	0.144055	0.161296	0.160688	0.311827	1.1541	0.353002	0.0464329	0.255216	0.295318
91	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.241_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149791	0.0880382	0.234443	0.769673	0.279032	0.0171413	0.305308	0.143349	0.161391	0.160665	0.312699	1.15408	0.353514	0.0465829	0.254952	0.295377
92	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149798	0.0880363	0.23535	0.769547	0.279039	0.0174312	0.306153	0.143362	0.161866	0.160664	0.310191	1.1595	0.352572	0.0465441	0.255123	0.295678
93	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.184_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149903	0.0880334	0.236291	0.769545	0.279023	0.0173568	0.305482	0.142657	0.16196	0.160642	0.311063	1.15947	0.353084	0.0466941	0.254858	0.295737

				score	score	score		score		score		score		score		score		score	score						
		score	score	distance	distance	distance		distance		distance		angle		angle		explicit		explicit	explicit						
		distance	distance	absolute value	absolute value	N to	score	absolute value	score	absolute value	score	connecting C	score	connecting C	score	distance	score	distance	distance	score					
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar1 to Ar2	Ar2 to Ar1 plane	Ar1 to Ar2	to N	Ar1 to	to N	Ar2 to	N to YZ plane	N to XY plane	N to XZ plane	Ar1 plane to	Ar2 plane to	similarity					
																			YZ plane	YZ plane	score					
94	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.184_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.149684	0.0880289	0.236131	0.769611	0.279055	0.0171192	0.305724	0.14414	0.161353	0.160679	0.311597	1.1595	0.352987	0.0467997	0.255244	0.295843									
95	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.184_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.149789	0.0880259	0.237071	0.769609	0.279039	0.0170448	0.305053	0.143435	0.161447	0.160656	0.312469	1.15947	0.353499	0.0469497	0.254979	0.295902									
96	MCN-5652_9.773_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.156508	0.0780574	0.11857	1.16188	0.279192	0.0186322	0.197395	0.146947	0.151606	0.164872	0.23041	1.19967	0.348067	0.0462221	0.20598	0.300267									
97	MCN-5652_9.792_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.156613	0.0780541	0.119738	1.1612	0.279161	0.0185486	0.19668	0.146098	0.15171	0.16484	0.230779	1.2007	0.348612	0.046421	0.20563	0.300319									
98	MCN-5652_9.773_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.156395	0.0780296	0.119286	1.16184	0.279206	0.0183203	0.196913	0.147708	0.151096	0.164874	0.231851	1.19967	0.348569	0.0469992	0.206106	0.300458									
99	MCN-5652_9.792_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.1565	0.0780263	0.120453	1.16116	0.279175	0.0182367	0.196198	0.146859	0.1512	0.164843	0.23222	1.2007	0.349113	0.047198	0.205756	0.300509									
100	MCN-5652_9.773_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.156505	0.0780544	0.120663	1.16196	0.279197	0.0185329	0.197131	0.14702	0.151628	0.164865	0.230186	1.20545	0.348059	0.0465807	0.206011	0.300789									
101	MCN-5652_9.792_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.15661	0.0780512	0.12183	1.16128	0.279166	0.0184493	0.196417	0.146171	0.151732	0.164834	0.230555	1.20647	0.348604	0.0467795	0.205661	0.300841									
102	MCN-5652_9.773_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.156392	0.0780266	0.121378	1.16192	0.279211	0.018221	0.196649	0.147781	0.151118	0.164868	0.231627	1.20545	0.34856	0.0473577	0.206138	0.30098									
103	MCN-5652_9.792_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.156497	0.0780234	0.122545	1.16125	0.27918	0.0181374	0.195935	0.146932	0.151222	0.164837	0.231995	1.20647	0.349105	0.0475566	0.205788	0.301032									
104	MCN-5652_9.792_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.164535	0.0835876	0.119121	1.16699	0.272683	0.0187266	0.202783	0.142882	0.17101	0.15409	0.232805	1.2007	0.348581	0.0455835	0.202499	0.301771									
105	MCN-5652_9.773_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.16443	0.0835909	0.120306	1.16765	0.272715	0.0188102	0.203497	0.143731	0.170905	0.154121	0.232436	1.19967	0.348036	0.0453847	0.202849	0.301876									
106	MCN-5652_9.792_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.164431	0.0835643	0.118432	1.16696	0.272693	0.0184256	0.202374	0.143662	0.170552	0.154083	0.234407	1.2007	0.349113	0.0461997	0.202607	0.30188									
107	MCN-5652_9.773_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol S-Citalopram_14.510_kcals_per_mol	0.164326	0.0835676	0.119617	1.16763	0.272724	0.0185092	0.203088	0.144511	0.170447	0.154114	0.234039	1.19967	0.348568	0.0460008	0.202957	0.301984									

					score	score	score			score	score	score	score	score	score		score	score		score	score		score	score						
		score	score	score	distance	distance	distance	score	score	distance	distance	distance	angle	angle	score	score	explicit	explicit	score	explicit	explicit	score	explicit	explicit						
		distance	distance	distance	absolute value	absolute value	N to	distance	distance	absolute value	absolute value	absolute value	Ar1 to	Ar2 to	explicit	explicit	distance	distance	explicit	distance	distance	explicit	Ar1 plane to	Ar2 plane to						
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score													
108	MCN-5652_9.792_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.164532	0.0835847	0.12121	1.16707	0.272688	0.0186273	0.20252	0.142954	0.171031	0.154084	0.23258	1.20647	0.348572	0.0459421	0.202531	0.302293														
109	MCN-5652_9.773_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.164427	0.0835879	0.122396	1.16773	0.27272	0.018711	0.203233	0.143803	0.170926	0.154115	0.232212	1.20545	0.348027	0.0457432	0.202881	0.302398														
110	MCN-5652_9.792_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.164428	0.0835613	0.120522	1.16704	0.272698	0.0183263	0.20211	0.143735	0.170573	0.154077	0.234183	1.20647	0.349105	0.0465582	0.202639	0.302402														
111	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153095	0.076212	0.132886	0.754188	0.27211	0.0162748	0.296405	0.138404	0.152406	0.150612	0.264383	1.20368	0.622	0.0457965	0.258335	0.302452														
112	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153201	0.0762091	0.13386	0.754186	0.272095	0.0162004	0.295734	0.137697	0.1525	0.150589	0.265266	1.20363	0.622159	0.0459465	0.25807	0.30249														
113	MCN-5652_9.773_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.164323	0.0835646	0.121707	1.16771	0.272729	0.01841	0.202824	0.144584	0.170468	0.154108	0.233815	1.20545	0.34856	0.0463593	0.202989	0.302507														
114	MCN-5652_9.773_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155275	0.0836233	0.126025	1.16875	0.280394	0.019414	0.202614	0.144442	0.14919	0.174803	0.235154	1.19967	0.348112	0.0449723	0.205197	0.302509														
115	MCN-5652_9.792_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15538	0.08362	0.127191	1.16806	0.280362	0.0193303	0.201901	0.143593	0.149294	0.174772	0.235522	1.2007	0.348857	0.0451711	0.204847	0.30256														
116	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152983	0.0761842	0.133602	0.754121	0.272125	0.0159629	0.295932	0.139165	0.151896	0.150614	0.265811	1.20368	0.622371	0.0465736	0.258462	0.302632														
117	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153088	0.0761813	0.134575	0.754119	0.272109	0.0158885	0.295261	0.138458	0.15199	0.150592	0.266694	1.20363	0.62253	0.0467236	0.258198	0.302669														
118	MCN-5652_9.773_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155161	0.0836008	0.12679	1.16885	0.28041	0.019102	0.202175	0.14522	0.148676	0.174818	0.236559	1.19967	0.348528	0.0452279	0.205318	0.302674														
119	MCN-5652_9.792_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155266	0.0835975	0.127956	1.16817	0.280378	0.0190184	0.201461	0.144371	0.14878	0.174786	0.236927	1.2007	0.349073	0.0454267	0.204968	0.302725														
120	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153091	0.0762088	0.134887	0.754235	0.272115	0.0161757	0.296126	0.138536	0.152408	0.150605	0.264139	1.21007	0.621412	0.046173	0.258371	0.30297														
121	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153196	0.0762058	0.13586	0.754233	0.272099	0.0161013	0.295455	0.137829	0.152502	0.150583	0.265021	1.21002	0.621571	0.046323	0.258107	0.303007														

			score	score	score	score				score	score	score	score									score	score		
			distance	score	distance	distance		score	score	distance	distance	angle	angle	score	score	score	score					explicit	explicit		
			distance	distance	absolute value	N to		distance	absolute value	absolute value	absolute value	connecting C	Ar2 to	explicit	explicit	explicit	explicit					angle	angle		
			N to Ar1	N to Ar2	N to Ar1 plane	connecting C	N to Ar2 plane	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	distance	distance	distance	distance					Ar1 plane to	Ar2 plane to	similarity	
																						YZ plane	YZ plane	score	
122	MCN-5652_9.773_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155272	0.0836203	0.12812	1.16871	0.280399	0.0193147	0.202351	0.144515	0.149212	0.174797	0.23493	1.20545	0.348104	0.0453308	0.205229	0.303023								
123	MCN-5652_9.792_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155377	0.0836171	0.129286	1.16802	0.280367	0.019231	0.201637	0.143666	0.149316	0.174765	0.235298	1.20647	0.348649	0.0455297	0.204879	0.303074								
124	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.152978	0.076181	0.135602	0.754168	0.272129	0.0158638	0.295654	0.139297	0.151898	0.150608	0.265567	1.21007	0.621784	0.0469501	0.258498	0.30315								
125	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153083	0.076178	0.136575	0.754166	0.272113	0.0157894	0.294982	0.138591	0.151992	0.150586	0.266449	1.21002	0.621943	0.0471	0.258234	0.303187								
126	MCN-5652_9.773_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155158	0.0835978	0.128885	1.16882	0.280415	0.0190027	0.201912	0.145293	0.148698	0.174811	0.236334	1.20545	0.34852	0.0455864	0.205349	0.303188								
127	MCN-5652_9.792_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.155264	0.0835946	0.13005	1.16813	0.280383	0.0189191	0.201198	0.144444	0.148802	0.17478	0.236702	1.20647	0.349065	0.0457853	0.204999	0.303239								
128	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168274	0.109068	0.242832	0.767098	0.305196	0.0169525	0.271695	0.141968	0.146539	0.152072	0.400168	1.20678	0.361549	0.0502389	0.212385	0.303521								
129	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168379	0.109071	0.243823	0.767096	0.30518	0.0168782	0.27102	0.141262	0.146633	0.15205	0.400454	1.20675	0.362059	0.0503889	0.212119	0.303544								
130	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168161	0.10904	0.24353	0.76703	0.30521	0.0166406	0.271226	0.142729	0.146028	0.152075	0.401555	1.20678	0.36205	0.0504976	0.212514	0.303671								
131	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168266	0.109043	0.244521	0.767028	0.305194	0.0165662	0.270551	0.142023	0.146123	0.152053	0.40184	1.20675	0.36256	0.0506475	0.212248	0.303694								
132	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161016	0.081746	0.125381	0.764998	0.265612	0.0164528	0.30238	0.135185	0.171709	0.139843	0.266391	1.20368	0.621976	0.044959	0.255187	0.303768								
133	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161121	0.081743	0.126355	0.764996	0.265597	0.0163784	0.30171	0.134479	0.171804	0.139821	0.267274	1.20363	0.622135	0.045109	0.254922	0.303805								
134	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.160911	0.0817227	0.126085	0.764948	0.265622	0.0161518	0.30198	0.135966	0.171251	0.139836	0.26798	1.20368	0.622371	0.0455752	0.255295	0.303958								
135	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161017	0.0817197	0.127059	0.764946	0.265606	0.0160774	0.30131	0.13526	0.171346	0.139814	0.268862	1.20363	0.62253	0.0457252	0.255031	0.303996								

										score	score				score	score	
				score	score	score		score	score	angle	angle	score	score	score	explicit	explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
136	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168271	0.109071	0.244893	0.767147	0.305201	0.0168532	0.271424	0.142041	0.146561	0.152066	0.399954	1.21254	0.361541	0.0505975	0.212416	0.304039
137	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168376	0.109074	0.245884	0.767145	0.305185	0.0167789	0.27075	0.141335	0.146655	0.152044	0.40024	1.21251	0.362051	0.0507474	0.21215	0.304062
138	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168158	0.109043	0.245592	0.767079	0.305215	0.0165413	0.270955	0.142802	0.14605	0.152069	0.401341	1.21254	0.362042	0.0508561	0.212545	0.304189
139	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.168263	0.109046	0.246582	0.767077	0.305199	0.016467	0.27028	0.142096	0.146144	0.152047	0.401627	1.21251	0.362552	0.0510061	0.212279	0.304212
140	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161011	0.0817427	0.127378	0.765045	0.265617	0.0163537	0.302101	0.135318	0.171711	0.139837	0.266147	1.21007	0.621388	0.0453355	0.255223	0.304285
141	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161117	0.0817398	0.128352	0.765042	0.265601	0.0162793	0.301431	0.134611	0.171806	0.139814	0.267029	1.21002	0.621547	0.0454855	0.254958	0.304322
142	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.160907	0.0817194	0.128082	0.764995	0.265626	0.0160527	0.301701	0.136098	0.171253	0.13983	0.267735	1.21007	0.621784	0.0459517	0.255331	0.304476
143	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151863	0.0817784	0.140341	0.7582	0.273316	0.0170565	0.301517	0.135897	0.149989	0.160561	0.269084	1.20368	0.622033	0.0445465	0.257548	0.304494
144	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161012	0.0817165	0.129056	0.764993	0.26561	0.0159783	0.301031	0.135392	0.171348	0.139808	0.268617	1.21002	0.621943	0.0461016	0.255067	0.304513
145	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151968	0.0817755	0.141313	0.758197	0.2733	0.0169821	0.300847	0.135191	0.150084	0.160539	0.269966	1.20363	0.622192	0.0446965	0.257284	0.304531
146	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151749	0.0817559	0.141105	0.758264	0.273332	0.0167446	0.301087	0.136676	0.149475	0.160576	0.270476	1.20368	0.622342	0.0448022	0.257669	0.304649
147	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_7.978_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151855	0.0817529	0.142077	0.758262	0.273316	0.0166702	0.300417	0.135969	0.14957	0.160553	0.271358	1.20363	0.622501	0.0449522	0.257405	0.304686
148	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176171	0.114591	0.235485	0.778039	0.298668	0.0171305	0.277629	0.138751	0.165873	0.141309	0.402118	1.20678	0.361517	0.0499106	0.209192	0.304877
149	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176276	0.114594	0.236477	0.778037	0.298652	0.0170562	0.276956	0.138045	0.165968	0.141287	0.402403	1.20675	0.362028	0.0500606	0.208926	0.304901

				score	score	score		score	score	score angle	score angle	score	score	score	score explicit	score explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
150	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151859	0.0817752	0.142344	0.758135	0.273321	0.0169574	0.301238	0.13603	0.149991	0.160555	0.26884	1.21007	0.621445	0.044923	0.257585	0.305004
151	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151964	0.0817722	0.143315	0.758133	0.273305	0.016883	0.300568	0.135323	0.150086	0.160533	0.269722	1.21002	0.621604	0.045073	0.25732	0.305041
152	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176067	0.114568	0.236176	0.777989	0.298677	0.0168296	0.277232	0.139532	0.165414	0.141302	0.40366	1.20678	0.36205	0.0501653	0.209302	0.305049
153	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176172	0.114571	0.237167	0.777987	0.298661	0.0167552	0.276558	0.138826	0.165509	0.14128	0.403945	1.20675	0.36256	0.0503152	0.209036	0.305073
154	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.151745	0.0817526	0.143108	0.7582	0.273336	0.0166455	0.300809	0.136808	0.149477	0.16057	0.270231	1.21007	0.621754	0.0451787	0.257705	0.305159
155	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_7.922_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15185	0.0817497	0.144079	0.758197	0.27332	0.0165711	0.300139	0.136102	0.149572	0.160547	0.271113	1.21002	0.621913	0.0453286	0.257441	0.305196
156	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176168	0.114594	0.237544	0.778088	0.298673	0.0170313	0.277359	0.138824	0.165895	0.141303	0.401904	1.21254	0.361509	0.0502692	0.209223	0.305395
157	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176273	0.114597	0.238535	0.778086	0.298657	0.0169569	0.276685	0.138118	0.16599	0.14128	0.40219	1.21251	0.362019	0.0504192	0.208957	0.305418
158	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.167044	0.114623	0.250093	0.770134	0.306406	0.0177343	0.276772	0.139463	0.144119	0.162017	0.404732	1.20678	0.361594	0.0494982	0.211587	0.305506
159	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.167149	0.114626	0.251082	0.770132	0.306391	0.0176599	0.276098	0.138757	0.144213	0.161994	0.405018	1.20675	0.362104	0.0496482	0.211321	0.30553
160	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176064	0.114571	0.238235	0.778038	0.298682	0.0167303	0.276961	0.139604	0.165436	0.141296	0.403446	1.21254	0.362041	0.0505238	0.209333	0.305567
161	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.176169	0.114574	0.239226	0.778036	0.298666	0.0166559	0.276288	0.138899	0.165531	0.141273	0.403731	1.21251	0.362551	0.0506738	0.209067	0.30559
162	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.166931	0.114601	0.250835	0.770198	0.306422	0.0174223	0.276345	0.140241	0.143604	0.162031	0.406084	1.20678	0.36201	0.0497538	0.21171	0.305664
163	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.167036	0.114604	0.251824	0.770196	0.306406	0.0173479	0.275671	0.139535	0.143698	0.162009	0.406369	1.20675	0.36252	0.0499038	0.211443	0.305688

										score	score				score	score	
				score	score	score		score	score	angle	angle	score	score	score	explicit	explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
164	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.167041	0.114626	0.252156	0.770063	0.306411	0.017635	0.276501	0.139536	0.14414	0.16201	0.404518	1.21254	0.361586	0.0498568	0.211618	0.306016
165	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.167146	0.114629	0.253145	0.770061	0.306395	0.0175606	0.275828	0.13883	0.144235	0.161988	0.404804	1.21251	0.362096	0.0500068	0.211352	0.306039
166	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.166928	0.114604	0.252899	0.770127	0.306427	0.017323	0.276075	0.140314	0.143626	0.162025	0.405869	1.21254	0.362001	0.0501124	0.211741	0.306174
167	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_15.585_kcals_per_mol	0.167033	0.114607	0.253887	0.770125	0.306411	0.0172486	0.275401	0.139608	0.14372	0.162002	0.406154	1.21251	0.362512	0.0502624	0.211475	0.306197
168	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.10705	0.054451	0.346642	0.645373	0.226799	0.0137471	0.30532	0.137754	0.148874	0.191792	0.268078	1.57159	0.305678	0.0515236	0.239822	0.307633
169	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.106936	0.0544417	0.345994	0.645312	0.226843	0.0134352	0.304847	0.138516	0.148363	0.191793	0.269505	1.57189	0.307281	0.0517823	0.239949	0.307793
170	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.107155	0.0544481	0.349494	0.645371	0.226783	0.0136727	0.30465	0.137048	0.148969	0.19177	0.268961	1.57205	0.306198	0.0516736	0.239556	0.307854
171	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.107047	0.0544599	0.347365	0.64543	0.226804	0.0134492	0.305053	0.137827	0.148896	0.191798	0.267856	1.57453	0.30567	0.0518821	0.239853	0.307861
172	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.107042	0.0544388	0.348846	0.64531	0.226827	0.0133608	0.304176	0.13781	0.148458	0.191771	0.270388	1.57235	0.307801	0.0519323	0.239684	0.308013
173	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.106933	0.0544506	0.346718	0.645369	0.226848	0.0131373	0.304579	0.138588	0.148385	0.191799	0.269283	1.57483	0.307273	0.0521408	0.239981	0.308021
174	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.107152	0.054457	0.350217	0.645427	0.226788	0.0133748	0.304382	0.137121	0.14899	0.191776	0.268738	1.57499	0.30619	0.0520321	0.239588	0.308082
175	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.107039	0.0544477	0.349569	0.645366	0.226832	0.0130629	0.303908	0.137882	0.14848	0.191777	0.270165	1.57528	0.307793	0.0522908	0.239716	0.308241
176	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105812	0.056314	0.339815	0.649254	0.230547	0.0145289	0.310444	0.135248	0.146455	0.195051	0.272776	1.57302	0.305822	0.0507829	0.23903	0.308327
177	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105697	0.0563065	0.339106	0.649328	0.230596	0.0142169	0.310014	0.136027	0.145941	0.195056	0.274167	1.57331	0.307153	0.0510385	0.239152	0.308474

		score distance N to Ar1	score distance N to Ar2	score distance absolute value N to Ar1 plane	score distance absolute value N to Ar2 plane	score distance N to connecting C	score distance absolute value Ar1 to Ar2	score distance absolute value Ar1 to Ar2 plane	score distance absolute value Ar2 to Ar1 plane	score angle Ar1 to connecting C to N	score angle Ar2 to connecting C to N	score explicit distance N to YZ plane	score explicit distance N to XY plane	score explicit distance N to XZ plane	score explicit angle Ar1 plane to YZ plane	score explicit angle Ar2 plane to YZ plane	similarity score
178	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105917	0.0563111	0.342665	0.649252	0.230531	0.0144545	0.309775	0.134542	0.14655	0.195029	0.273658	1.57348	0.306342	0.0509329	0.238765	0.308547
179	MCN-5652_13.035_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105809	0.0563229	0.340541	0.649191	0.230552	0.0142309	0.310176	0.135321	0.146477	0.195057	0.272554	1.57596	0.305814	0.0511415	0.239062	0.308547
180	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105803	0.0563035	0.341956	0.649325	0.23058	0.0141425	0.309344	0.135321	0.146035	0.195033	0.275049	1.57377	0.307673	0.0511885	0.238887	0.308694
181	MCN-5652_13.035_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105694	0.0563154	0.339832	0.649264	0.230601	0.013919	0.309746	0.1361	0.145962	0.195062	0.273944	1.57625	0.307144	0.0513971	0.239183	0.308694
182	MCN-5652_13.057_kcals_per_mol_Sertraline_2.094_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.105914	0.05632	0.343391	0.649189	0.230536	0.0141565	0.309507	0.134615	0.146572	0.195035	0.273435	1.57642	0.306334	0.0512915	0.238797	0.308767
183	MCN-5652_13.057_kcals_per_mol_Sertraline_2.158_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.1058	0.0563124	0.342681	0.649262	0.230585	0.0138446	0.309077	0.135394	0.146057	0.19504	0.274826	1.57671	0.307664	0.0515471	0.238918	0.308914
184	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.115017	0.0563031	0.353344	0.655205	0.206711	0.0139251	0.311309	0.134537	0.168197	0.188254	0.270085	1.57222	0.305577	0.0511953	0.236654	0.309236
185	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.114912	0.0562953	0.352722	0.65516	0.20674	0.0136241	0.310908	0.135317	0.167738	0.188252	0.271672	1.57256	0.30728	0.05145	0.236763	0.309426
186	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.115122	0.0563002	0.356197	0.655203	0.206695	0.0138507	0.31064	0.133831	0.168292	0.188232	0.270967	1.57268	0.306097	0.0513453	0.236389	0.309456
187	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.115014	0.056312	0.354065	0.655262	0.206716	0.0136272	0.311041	0.13461	0.168218	0.18826	0.269862	1.57516	0.305569	0.0515539	0.236685	0.309463
188	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.532_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.115017	0.0562924	0.355575	0.655158	0.206724	0.0135497	0.310239	0.134611	0.167833	0.188229	0.272554	1.57302	0.3078	0.0516	0.236498	0.309646
189	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.114909	0.0563042	0.353443	0.655216	0.206745	0.0133262	0.31064	0.13539	0.16776	0.188258	0.27145	1.57549	0.307272	0.0518085	0.236795	0.309654
190	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.115119	0.0563091	0.356917	0.655259	0.2067	0.0135528	0.310372	0.133903	0.168313	0.188238	0.270744	1.57562	0.306089	0.0517038	0.23642	0.309684
191	MCN-5652_9.773_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153593	0.0873731	0.220714	1.16581	0.286561	0.0190983	0.20244	0.151233	0.16369	0.164826	0.276651	1.15391	0.351378	0.0468316	0.203599	0.309847

				score	score	score		score	score	score	score				score	score	
		score	score	distance	distance	distance	score	distance	distance	angle	angle	score	score	score	explicit	explicit	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	connecting C to N	connecting C to N	N to YZ plane	N to XY plane	N to XZ plane	Ar1 plane to YZ plane	Ar2 plane to YZ plane	similarity score
192	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.476_kcals_per_mol_S-Citalopram_10.603_kcals_per_mol	0.115014	0.0563013	0.356296	0.655214	0.206729	0.0132518	0.309971	0.134684	0.167855	0.188236	0.272332	1.57595	0.307792	0.0519585	0.23653	0.309874
193	MCN-5652_9.792_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153698	0.0873698	0.221848	1.16515	0.28653	0.0190147	0.201725	0.150385	0.163795	0.164795	0.27701	1.15495	0.351924	0.0470305	0.203249	0.309899
194	MCN-5652_9.773_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15348	0.0873638	0.221449	1.16589	0.286576	0.0187864	0.201958	0.151994	0.163181	0.164829	0.278106	1.15391	0.351878	0.0476086	0.203725	0.310049
195	MCN-5652_9.792_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161625	0.0892065	0.221013	1.15199	0.280073	0.0191927	0.207823	0.14717	0.18307	0.154045	0.279055	1.15495	0.351892	0.046193	0.203631	0.310062
196	MCN-5652_9.792_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153585	0.0873606	0.222582	1.16523	0.286544	0.0187028	0.201243	0.151146	0.163285	0.164798	0.278465	1.15495	0.352424	0.0478075	0.203375	0.3101
197	MCN-5652_9.792_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.16152	0.0891988	0.220346	1.15205	0.280083	0.0188917	0.207414	0.14795	0.182613	0.154038	0.280674	1.15495	0.352423	0.0468092	0.203517	0.310166
198	MCN-5652_9.773_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.16152	0.0892097	0.222233	1.15264	0.280105	0.0192763	0.208537	0.148018	0.182966	0.154076	0.278697	1.15391	0.351346	0.0459941	0.203981	0.310167
199	MCN-5652_9.773_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150633	0.0824122	0.2414	1.16882	0.292281	0.0190687	0.199323	0.15699	0.164354	0.170404	0.334599	1.0627	0.357088	0.0491608	0.203264	0.310167
200	MCN-5652_9.792_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150738	0.082409	0.242523	1.16817	0.292249	0.0189851	0.198609	0.156144	0.164458	0.170372	0.334946	1.06378	0.357635	0.0493597	0.202914	0.310219
201	MCN-5652_9.773_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_14.221_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.161415	0.089202	0.221566	1.1527	0.280114	0.0189753	0.208128	0.148798	0.182508	0.154069	0.280316	1.15391	0.351878	0.0466103	0.203867	0.31027
202	MCN-5652_9.773_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.15052	0.082403	0.24214	1.1689	0.292295	0.0187568	0.198841	0.157751	0.163844	0.170406	0.335057	1.0627	0.357586	0.0499378	0.203391	0.310302
203	MCN-5652_9.773_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153593	0.0873615	0.22343	1.16578	0.286567	0.0190014	0.202189	0.151273	0.163765	0.164818	0.27641	1.15884	0.351354	0.0471874	0.203626	0.310346
204	MCN-5652_9.792_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_13.630_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.150625	0.0823997	0.243263	1.16825	0.292263	0.0186732	0.198127	0.156904	0.163948	0.170375	0.335404	1.06378	0.358134	0.0501367	0.20304	0.310354
205	MCN-5652_9.792_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_14.164_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.153698	0.0873582	0.224562	1.16512	0.286536	0.0189177	0.201474	0.150425	0.16387	0.164787	0.276769	1.15988	0.3519	0.0473862	0.203276	0.310397

				score	score	score		score	score	score	score				score	score	
		score	score	distance	distance	distance	score	distance	distance	angle Ar1 to	angle Ar2 to	score explicit	score explicit	score explicit	explicit angle	explicit angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
206	MCN-5652_9.792_kcals_per_mol_ Sertraline_1.483_kcals_per_mol_ Indatraline_13.630_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.158668	0.0862918	0.241626	1.15501	0.285813	0.0191631	0.20471	0.15293	0.183732	0.15963	0.33559	1.06378	0.357604	0.0485223	0.203946	0.310468
207	MCN-5652_9.792_kcals_per_mol_ Sertraline_1.545_kcals_per_mol_ Indatraline_13.630_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.158563	0.0862685	0.240965	1.15507	0.285823	0.0188621	0.204301	0.153709	0.183275	0.159624	0.336098	1.06378	0.358133	0.0491384	0.203832	0.310496
208	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.198_kcals_per_mol_ Indatraline_14.164_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.15348	0.0873523	0.224165	1.16586	0.286581	0.0186894	0.201707	0.152034	0.163256	0.164821	0.277865	1.15884	0.351854	0.0479644	0.203752	0.310548
209	MCN-5652_9.792_kcals_per_mol_ Sertraline_1.483_kcals_per_mol_ Indatraline_14.164_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.161625	0.0891949	0.22372	1.15206	0.280079	0.0190957	0.207572	0.14721	0.183145	0.154037	0.278814	1.15988	0.351868	0.0465488	0.203606	0.310564
210	MCN-5652_9.773_kcals_per_mol_ Sertraline_1.483_kcals_per_mol_ Indatraline_13.630_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.158562	0.0862951	0.242856	1.15565	0.285845	0.0192467	0.205424	0.153777	0.183628	0.159662	0.335243	1.0627	0.357056	0.0483234	0.204296	0.310571
211	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.135_kcals_per_mol_ Indatraline_13.572_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.150639	0.0824007	0.244321	1.16879	0.292284	0.0189734	0.199085	0.157193	0.164487	0.17041	0.333732	1.06641	0.357041	0.0495597	0.203289	0.310574
212	MCN-5652_9.792_kcals_per_mol_ Sertraline_2.198_kcals_per_mol_ Indatraline_14.164_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.153585	0.087349	0.225296	1.16519	0.28655	0.0186058	0.200993	0.151186	0.16336	0.16479	0.278224	1.15988	0.3524	0.0481633	0.203402	0.310599
213	MCN-5652_9.773_kcals_per_mol_ Sertraline_1.545_kcals_per_mol_ Indatraline_13.630_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.158458	0.0862718	0.242195	1.15572	0.285854	0.0189457	0.205015	0.154556	0.18317	0.159655	0.335751	1.0627	0.357586	0.0489395	0.204182	0.3106
214	MCN-5652_9.792_kcals_per_mol_ Sertraline_2.135_kcals_per_mol_ Indatraline_13.572_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.150744	0.0823975	0.245443	1.16813	0.292252	0.0188897	0.19837	0.156346	0.164592	0.170378	0.33408	1.06748	0.357588	0.0497586	0.202939	0.310626
215	MCN-5652_9.792_kcals_per_mol_ Sertraline_1.545_kcals_per_mol_ Indatraline_14.164_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.16152	0.0891872	0.223053	1.15203	0.280089	0.0187947	0.207164	0.14799	0.182688	0.15403	0.280433	1.15988	0.3524	0.0471649	0.203492	0.310661
216	MCN-5652_9.773_kcals_per_mol_ Sertraline_1.483_kcals_per_mol_ Indatraline_14.164_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.16152	0.0891982	0.22494	1.15271	0.280111	0.0191794	0.208287	0.148058	0.18304	0.154068	0.278456	1.15884	0.351322	0.0463499	0.203956	0.310669
217	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.198_kcals_per_mol_ Indatraline_13.572_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.150526	0.0823915	0.245062	1.16887	0.292298	0.0186614	0.198603	0.157953	0.163978	0.170412	0.33419	1.06641	0.35754	0.0503367	0.203416	0.31071
218	MCN-5652_9.792_kcals_per_mol_ Sertraline_2.198_kcals_per_mol_ Indatraline_13.572_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.150631	0.0823883	0.246184	1.16821	0.292266	0.0185778	0.197889	0.157107	0.164082	0.170381	0.334537	1.06748	0.358087	0.0505356	0.203066	0.310761
219	MCN-5652_9.773_kcals_per_mol_ Sertraline_1.545_kcals_per_mol_ Indatraline_14.164_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.161415	0.0891905	0.224274	1.15268	0.28012	0.0188784	0.207878	0.148838	0.182583	0.154061	0.280074	1.15884	0.351854	0.0469661	0.203841	0.310766

					score	score	score			score	score	score		score			score			score	score						
		score	score	score	distance	distance	distance	score	score	distance	distance	angle	score	angle	score		score	score		explicit	explicit						
		distance	distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	absolute value	Ar1 to	angle	Ar2 to	explicit		explicit	explicit		angle	angle						
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	Ar2 to Ar1 plane	connecting C	to N	connecting C	distance	N to YZ plane	distance	distance	N to XZ plane	Ar1 plane to	Ar1 plane to	Ar2 plane to	similarity				
220	MCN-5652_9.792_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158673	0.086288	0.244536	1.15497	0.285817	0.0190677	0.204472	0.153133	0.183866	0.159636	0.334723	1.06748	0.357557	0.0489212	0.203922	0.310871											
221	MCN-5652_9.792_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158569	0.0862647	0.243876	1.15504	0.285826	0.0187667	0.204063	0.153912	0.183408	0.15963	0.335231	1.06748	0.358087	0.0495373	0.203808	0.3109											
222	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.157757	0.0821457	0.191202	0.758857	0.273209	0.0168014	0.299667	0.147323	0.180566	0.141845	0.306585	1.0874	0.717583	0.0492928	0.254076	0.310941											
223	MCN-5652_13.035_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149832	0.0768078	0.198838	0.766994	0.279684	0.0164234	0.294547	0.150538	0.16128	0.152608	0.304558	1.0874	0.717603	0.0501302	0.257224	0.310965											
224	MCN-5652_9.773_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158568	0.0862913	0.245767	1.15562	0.285848	0.0191514	0.205186	0.153979	0.183761	0.159668	0.334376	1.06641	0.35701	0.0487223	0.204272	0.310975											
225	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.157862	0.0821427	0.192157	0.758855	0.273193	0.016527	0.298998	0.148619	0.18066	0.141823	0.307457	1.08739	0.717753	0.0494429	0.253812	0.31098											
226	MCN-5652_13.057_kcals_per_mol_Sertraline_2.135_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149938	0.0768049	0.199793	0.766992	0.279668	0.016349	0.293877	0.149833	0.161374	0.152585	0.30543	1.08739	0.717774	0.0502802	0.25696	0.311003											
227	MCN-5652_9.773_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_13.572_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.158464	0.086268	0.245107	1.15568	0.285858	0.0188504	0.204777	0.154759	0.183304	0.159661	0.334884	1.06641	0.357539	0.0493384	0.204158	0.311004											
228	MCN-5652_13.035_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.157653	0.0821224	0.19192	0.758896	0.273218	0.0163004	0.299267	0.148103	0.180108	0.141839	0.308189	1.0874	0.717923	0.0499089	0.254184	0.311135											
229	MCN-5652_13.057_kcals_per_mol_Sertraline_1.545_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.157758	0.0821194	0.192874	0.758894	0.273202	0.016226	0.298597	0.147398	0.180203	0.141816	0.309061	1.08739	0.718093	0.050059	0.25392	0.311174											
230	MCN-5652_13.035_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149719	0.0767986	0.199565	0.767043	0.279698	0.0161115	0.295019	0.151298	0.16077	0.15261	0.306	1.0874	0.717923	0.0509072	0.257351	0.311214											
231	MCN-5652_13.057_kcals_per_mol_Sertraline_2.198_kcals_per_mol_Indatraline_7.525_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.149825	0.0767956	0.200519	0.767041	0.279683	0.0160371	0.294348	0.150594	0.160864	0.152588	0.306872	1.08739	0.718094	0.0510572	0.257087	0.311253											
232	MCN-5652_13.035_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.467_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.157757	0.0821426	0.19375	0.758788	0.273216	0.0165046	0.299427	0.147522	0.180649	0.141853	0.30637	1.09187	0.717076	0.0496811	0.2541	0.311381											
233	MCN-5652_13.057_kcals_per_mol_Sertraline_1.483_kcals_per_mol_Indatraline_7.467_kcals_per_mol_S-Citalopram_14.510_kcals_per_mol	0.157863	0.0821397	0.194704	0.758786	0.2732	0.0164303	0.298758	0.146818	0.180744	0.141831	0.307243	1.09186	0.717247	0.0498312	0.253836	0.311419											

				score	score	score		score	score	score	score			score	score	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
234	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.135_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.153947	0.089627	0.21887	1.16557	0.285818	0.0189497	0.201231	0.148478	0.163194	0.166192	0.270831	1.18765	0.350776	0.046099	0.204249	0.311433
235	MCN-5652_13.035_kcals_per_ mol_Sertraline_2.135_kcals_per_ mol_Indatraline_7.467_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.149833	0.0767987	0.201392	0.766925	0.279691	0.0163266	0.294787	0.150737	0.161364	0.152615	0.304343	1.09187	0.717097	0.0505185	0.257249	0.311436
236	MCN-5652_13.057_kcals_per_ mol_Sertraline_2.135_kcals_per_ mol_Indatraline_7.467_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.149938	0.0767957	0.202346	0.766923	0.279675	0.0162522	0.294117	0.150032	0.161458	0.152593	0.305216	1.09186	0.717267	0.0506685	0.256984	0.311475
237	MCN-5652_9.792_kcals_per_mol_ Sertraline_2.135_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.154053	0.0896237	0.220004	1.16491	0.285787	0.018866	0.200517	0.147629	0.163298	0.166161	0.271192	1.18869	0.351321	0.0462978	0.203899	0.311483
238	MCN-5652_13.035_kcals_per_ mol_Sertraline_1.545_kcals_per_ mol_Indatraline_7.467_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.157653	0.0821193	0.194468	0.758827	0.273225	0.0162037	0.299027	0.148302	0.180192	0.141846	0.307974	1.09187	0.717417	0.0502972	0.254209	0.311575
239	MCN-5652_13.057_kcals_per_ mol_Sertraline_1.545_kcals_per_ mol_Indatraline_7.467_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.157758	0.0821164	0.195422	0.758825	0.273209	0.0161293	0.298357	0.147597	0.180287	0.141824	0.308847	1.09186	0.717587	0.0504472	0.253945	0.311614
240	MCN-5652_9.792_kcals_per_mol_ Sertraline_1.483_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.161979	0.0914604	0.219175	1.15254	0.279328	0.019044	0.206616	0.144413	0.182574	0.155409	0.273236	1.18869	0.35129	0.0454604	0.203018	0.311616
241	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.198_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.153834	0.0896177	0.219604	1.16565	0.285833	0.0186378	0.20075	0.149239	0.162684	0.166195	0.272285	1.18765	0.351276	0.046876	0.204376	0.311634
242	MCN-5652_9.792_kcals_per_mol_ Sertraline_2.198_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.15394	0.0896145	0.220738	1.16499	0.285801	0.0185541	0.200035	0.14839	0.162788	0.166164	0.272646	1.18869	0.351822	0.0470749	0.204026	0.311685
243	MCN-5652_13.035_kcals_per_ mol_Sertraline_2.198_kcals_per_ mol_Indatraline_7.467_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.14972	0.0767895	0.20212	0.766974	0.279705	0.0160147	0.295259	0.151497	0.160854	0.152618	0.305785	1.09187	0.717417	0.0512955	0.257376	0.311686
244	MCN-5652_9.792_kcals_per_mol_ Sertraline_1.545_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.161874	0.0914527	0.218508	1.15252	0.279337	0.018743	0.206207	0.145193	0.182117	0.155402	0.274854	1.18869	0.351821	0.0460765	0.202904	0.311713
245	MCN-5652_9.773_kcals_per_mol_ Sertraline_1.483_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.161873	0.0914637	0.220394	1.15319	0.27936	0.0191277	0.20733	0.145261	0.18247	0.155441	0.272876	1.18765	0.350744	0.0452615	0.203368	0.311721
246	MCN-5652_13.057_kcals_per_ mol_Sertraline_2.198_kcals_per_ mol_Indatraline_7.467_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.149825	0.0767865	0.203073	0.766971	0.27969	0.0159403	0.294588	0.150793	0.160948	0.152596	0.306658	1.09186	0.717588	0.0514455	0.257112	0.311725
247	MCN-5652_9.773_kcals_per_mol_ Sertraline_1.545_kcals_per_mol_ Indatraline_14.241_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.161769	0.091456	0.219726	1.15317	0.279369	0.0188267	0.206921	0.146042	0.182012	0.155434	0.274493	1.18765	0.351276	0.0458777	0.203254	0.311818

										score	score				score	score	
				score	score	score		score	score	angle	angle	score	score	score	explicit	explicit	
		score	score	distance	distance	distance	score	distance	distance	Ar1 to	Ar2 to	explicit	explicit	explicit	angle	angle	
		distance	distance	absolute value	absolute value	N to	distance	absolute value	absolute value	connecting C	connecting C	distance	distance	distance	Ar1 plane to	Ar2 plane to	similarity
		N to Ar1	N to Ar2	N to Ar1 plane	N to Ar2 plane	connecting C	Ar1 to Ar2	Ar1 to Ar2 plane	Ar2 to Ar1 plane	to N	to N	N to YZ plane	N to XY plane	N to XZ plane	YZ plane	YZ plane	score
248	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.135_kcals_per_mol_ Indatraline_14.184_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.153945	0.0896146	0.221499	1.16554	0.285826	0.0188532	0.20098	0.148563	0.16325	0.166183	0.270596	1.19297	0.35076	0.0464657	0.204277	0.311955
249	MCN-5652_9.792_kcals_per_mol_ Sertraline_2.135_kcals_per_mol_ Indatraline_14.184_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.154051	0.0896114	0.222632	1.16488	0.285794	0.0187696	0.200266	0.147715	0.163354	0.166152	0.270957	1.194	0.351306	0.0466646	0.203927	0.312005
250	MCN-5652_9.773_kcals_per_mol_ Sertraline_2.094_kcals_per_mol_ Indatraline_14.221_kcals_per_mol_ S-Citalopram_14.510_kcals_per_mol	0.15236	0.0892205	0.228354	1.17537	0.287759	0.01988	0.207655	0.148729	0.161277	0.174758	0.281442	1.15391	0.351423	0.0455818	0.202817	0.312035

Appendix 4.1 The Sybyl Programming Language (SPL) four molecule comparison program. Calculates and saves the similarity scores for four molecule comparisons. Only works for small data sets.

```
# Program:  msc_4_molecules.spl
# author:  Paul A. Wilson
# email:  pwilson@cobremail.itrc.umt.edu
# date of latest version:  August 22, 2004
# date of origination: late 1990s
#
#  Sybyl Programming Language - run from within the Sybyl program by Tripos
#
#  Change the path and name of table files below in the code as appropriate.
#  In practice this program uses all available memory including scratch
#  space, until the system crashes making this program unusable in most cases.
#

# molecular similarity comparison

UIMS DEFINE MACRO msc SybylBasic


SETVAR TA_PRECISION 9


#  Change the path and name of these table files as appropriate

SETVAR tbl1_file /home/pwilson/sybyl/5HT3/sertraline/sertraline_low_energy_conformations.tbl
SETVAR tbl2_file /home/pwilson/sybyl/5HT3/MCN-5652/mcn_5652_low_energy_conformations.tbl
SETVAR tbl3_file /home/pwilson/sybyl/5HT3/indatraline/indatraline_low_energy_conformations.tbl
SETVAR tbl4_file /home/pwilson/sybyl/5HT3/s-citalopram/s_citalopram_low_energy_conformations.tbl

SETVAR tbl1 sertraline_low_energy_conformations
SETVAR tbl2 mcn_5652_low_energy_conformations
SETVAR tbl3 indatraline_low_energy_conformations
SETVAR tbl4 s_citalopram_low_energy_conformations

SETVAR column_number 0

SETVAR row_tbl1 1
SETVAR row_tbl2 1
SETVAR row_tbl3 1
SETVAR row_tbl4 1

SETVAR cell_value_tbl1
SETVAR cell_value_tbl2
SETVAR cell_value_tbl3
SETVAR cell_value_tbl4


TABLE NEW
TABLE RENAME UMSS_1 comparisons
TABLE COLUMN_APPEND STRING COL_1
```

```
TABLE COLUMN_APPEND STRING COL_2
TABLE COLUMN_APPEND STRING COL_3
TABLE COLUMN_APPEND STRING COL_4


SETVAR comparisons_row 0
SETVAR comparison_column_prefix COL_
SETVAR column_number 0
SETVAR comparisons_column_number 4
SETVAR comparisons_column_name %cat($comparison_column_prefix $column_number)


TABLE OPEN $tbl1_file
TABLE OPEN $tbl2_file
TABLE OPEN $tbl3_file
TABLE OPEN $tbl4_file


SETVAR points_to_row_name_tbl1 0
SETVAR points_to_row_name_tbl2 0
SETVAR points_to_row_name_tbl3 0
SETVAR points_to_row_name_tbl4 0


SETVAR point_to_column_in_comparisons 1


# the column loop

TABLE DEFAULT $tbl1


FOR current_column_in_spreadsheet IN %table((*) column number)

SETVAR comparisons_row 0

SETVAR comparisons_column_number %MATH($comparisons_column_number + 1)
SETVAR comparisons_column_name %cat($comparison_column_prefix $comparisons_column_number)

TABLE DEFAULT comparisons
TABLE COLUMN_APPEND EXPLICIT_DATA $comparisons_column_name
TABLE DEFAULT $tbl1
SETVAR column_number %MATH($column_number + 1)


# table one row loop

TABLE DEFAULT $tbl1
SETVAR points_to_row_name_tbl1 0
SETVAR current_row_in_spreadsheet1 1
SETVAR row_tbl1 0


FOR current_row_in_spreadsheet1 IN %table((*) row number)
```

```
TABLE DEFAULT $tbl1
SETVAR row_tbl1 %MATH($row_tbl1 + 1)
SETVAR points_to_row_name_tbl1 %MATH($points_to_row_name_tbl1 + 1)
SETVAR row_name_tbl1 %table($points_to_row_name_tbl1 row name)

SETVAR cell_value_tbl1 %rcell($row_tbl1 $current_column_in_spreadsheet)
SETVAR cell_value_tbl1 %ABS($cell_value_tbl1)

# table two row loop

TABLE DEFAULT $tbl2
SETVAR points_to_row_name_tbl2 0
SETVAR current_row_in_spreadsheet2 1
SETVAR row_tbl2 0


FOR current_row_in_spreadsheet2 IN %table((*) row number)

TABLE DEFAULT $tbl2
SETVAR row_tbl2 %MATH($row_tbl2 + 1)
SETVAR points_to_row_name_tbl2 %MATH($points_to_row_name_tbl2 + 1)
SETVAR row_name_tbl2 %table($points_to_row_name_tbl2 row name)

SETVAR cell_value_tbl2 %rcell($row_tbl2 $current_column_in_spreadsheet)
SETVAR cell_value_tbl2 %ABS($cell_value_tbl2)

#first comparison

SETVAR first_difference %MATH($cell_value_tbl1 - $cell_value_tbl2)
SETVAR absolute_value_first_difference %ABS($first_difference)
SETVAR first_comparison %MATH($absolute_value_first_difference / (($cell_value_tbl1 + $cell_value_
tbl2) / 2))


# table three row loop

TABLE DEFAULT $tbl3
SETVAR points_to_row_name_tbl3 0
SETVAR current_row_in_spreadsheet3 1
SETVAR row_tbl3 0


FOR current_row_in_spreadsheet3 IN %table((*) row number)

TABLE DEFAULT $tbl3
SETVAR row_tbl3 %MATH($row_tbl3 + 1)
SETVAR points_to_row_name_tbl3 %MATH($points_to_row_name_tbl3 + 1)
SETVAR row_name_tbl3 %table($points_to_row_name_tbl3 row name)

SETVAR cell_value_tbl3 %rcell($row_tbl3 $current_column_in_spreadsheet)
SETVAR cell_value_tbl3 %ABS($cell_value_tbl3)

# second comparison
```

```
SETVAR second_difference_1 %MATH($cell_value_tbl1 - $cell_value_tbl3)
SETVAR absolute_value_second_difference_1 %ABS($second_difference_1)
SETVAR second_comparison_1 %MATH($absolute_value_second_difference_1 / (($cell_value_tbl1 +
$cell_value_tbl3) / 2))

SETVAR second_difference_2 %math($cell_value_tbl2 - $cell_value_tbl3)
SETVAR absolute_value_second_difference_2 %ABS($second_difference_2)
SETVAR second_comparison_2 %MATH($absolute_value_second_difference_2 / (($cell_value_tbl2 +
$cell_value_tbl3) /2))


# table four row loop

TABLE DEFAULT $tbl4
SETVAR points_to_row_name_tbl4 0
SETVAR current_row_in_spreadsheet4 1
SETVAR row_tbl4 0


FOR current_row_in_spreadsheet4 IN %table((*) row number)

TABLE DEFAULT $tbl4
SETVAR row_tbl4 %MATH($row_tbl4 + 1)
SETVAR points_to_row_name_tbl4 %MATH($points_to_row_name_tbl4 + 1)
SETVAR row_name_tbl4 %table($points_to_row_name_tbl4 row name)

SETVAR cell_value_tbl4 %rcell($row_tbl4 $current_column_in_spreadsheet)
SETVAR cell_value_tbl4 %ABS($cell_value_tbl4)

# third comparison

SETVAR third_difference_1 %MATH($cell_value_tbl1 - $cell_value_tbl4)
SETVAR absolute_value_third_difference_1 %ABS($third_difference_1)
SETVAR third_comparison_1 %MATH($absolute_value_third_difference_1 / (($cell_value_tbl1 +
$cell_value_tbl4) / 2))

SETVAR third_difference_2 %MATH($cell_value_tbl2 - $cell_value_tbl4)
SETVAR absolute_value_third_difference_2 %ABS($third_difference_2)
SETVAR third_comparison_2 %MATH($absolute_value_third_difference_2 / (($cell_value_tbl2 +
$cell_value_tbl4) / 2))

SETVAR third_difference_3 %MATH($cell_value_tbl3 - $cell_value_tbl4)
SETVAR absolute_value_third_difference_3 %ABS($third_difference_3)
SETVAR third_comparison_3 %MATH($absolute_value_third_difference_3 / (($cell_value_tbl3 +
$cell_value_tbl4) / 2))


SETVAR total_comparison %MATH(($first_comparison + $second_comparison_1 + $second_compari-
son_2 + $third_comparison_1 + $third_comparison_2 + $third_comparison_3)/6)


# row labelling loop
```

```
TABLE DEFAULT comparisons
SETVAR comparisons_column $current_column_in_spreadsheet


IF %EQ($current_column_in_spreadsheet 1)

SETVAR comparisons_row %math($comparisons_row + 1)
TABLE ROW_APPEND $comparisons_row

SETVAR newval1 %wcell($comparisons_row $comparisons_column $row_name_tbl1)
SETVAR comparisons_column %MATH($comparisons_column + 1)
SETVAR newval2 %wcell($comparisons_row $comparisons_column $row_name_tbl2)
SETVAR comparisons_column %MATH($comparisons_column + 1)
SETVAR newval3 %wcell($comparisons_row $comparisons_column $row_name_tbl3)
SETVAR comparisons_column %MATH($comparisons_column + 1)
SETVAR newval4 %wcell($comparisons_row $comparisons_column $row_name_tbl4)
SETVAR comparisons_column %MATH($comparisons_column + 1)

SETVAR newval5 %wcell($comparisons_row $comparisons_column $total_comparison)

ELSE

SETVAR comparisons_row %math($comparisons_row + 1)
SETVAR comparisons_column %MATH($current_column_in_spreadsheet + 4)
SETVAR newval6 %wcell($comparisons_row $comparisons_column $total_comparison)

ENDIF


ENDFOR

ENDFOR

ENDFOR

ENDFOR

ENDFOR



TABLE DEFAULT comparisons
SETVAR comparisons_column_number %MATH($comparisons_column_number + 1)
SETVAR comparisons_column_name %cat($comparison_column_prefix $comparisons_column_number)
TABLE COLUMN_APPEND EXPLICIT_DATA $comparisons_column_name
SETVAR comparisons_row 1


FOR comparisons_row IN %table((*) row number)
SETVAR total_total 0
SETVAR column_number 5


WHILE %LT($column_number $comparisons_column_number)
```

```
SETVAR cell_value %rcell($comparisons_row $column_number)
SETVAR total_total %MATH($total_total + $cell_value)
SETVAR column_number %MATH($column_number + 1)
ENDWHILE


SETVAR divisor %MATH($column_number - 5)
SETVAR total_total %MATH($total_total / $divisor)
SETVAR newval7 %wcell($comparisons_row $comparisons_column_number $total_total)
SETVAR comparisons_row %MATH($comparisons_row + 1)


ENDFOR
```

.

Appendix 4.2 The four molecule Perl script. Calculates and saves the similarity scores for four molecule comparisons.

```
#!/usr/bin/perl

# Program:  msc_4_molecules.pl
# authors:  Paul A. Wilson, Larry Beintema
# email:  pwilson@cobremail.itrc.umt.edu
# date of latest version:  August 22, 2004
# date of origination: late 1990s
#
#  Sybyl Programming Language - run from within the Sybyl program by Tripos
#
#  Change the path and name of the input files below in the code as appropriate.
#  As well, changethe name of the ouput files as appropriate.
#  space, until the system crashes making this program unusable in most cases.
#
#  In Short:  This program compares the rows in data files developing a score
#  for each set of rows (set - one row from each data file).  A text file is
#  written containing all comparisons.  Each row in the output file contains
#  in comma delimited format the label from each label field followed by the
#  score for each field and finally the total score.  This version of the
#  program does not sort the similarity scores.
#
#  similarity score = (summation from measure 1 to total number of measures (
#  summation of from data file 1 to total number of data files ( (|Vab - Vacl /
#  ((|Vabl + |Vacl)/2)) / number of combinatorial )))
#  Vab = measure a of datafile b
#  Vac = measure a of datafile c
#


# Do compares a row at a time on the following 4 files (spreadsheets)
# These are comma delimited files, first column is label, rest are numbers
# NOTE: Perl array element indices start at 0 !


# Time file
# Change the path and name as appropriate
open( TI, ">times_4_molecule.out" )
 || die "Unable to open file times.record for time recording";

# start time
$start = time();


# Input files
#  Change the path and name of these input files as appropriate
$f1 = "./MCN_5652_nq3.txt";
$f2 = "./Sertraline_nq3.txt";
$f3 = "./Indatraline_nq3.txt";
$f4 = "./S-Citalopram_nq3.txt";

# Output file
```

```
# Data output file, change the path and name as appropriate
open(OF,">msic_4_output.txt") || die "Unable to open file test.output for output";

# Error output file, change the path and name as appropriate
# If everything is going well this file will remain empty
open(EF,">test_4_molecule.error") || die "Unable to open file test.error for output";


# Open file for row loop through first file
open(FH1,"<$f1") || die "Unable to open file $f1 for input";
$f1cnt = 0;
while ( <FH1> )
 {
 chop;
 ($label1,@row_fh1) = split(",",$_);
 $f1cnt++;

 # Open file for row loop through second file
 open(FH2,"<$f2") || die "Unable to open file $f2 for input";
 $f2cnt = 0;
 while ( <FH2> )
  {
  chop;
  ($label2,@row_fh2) = split(",",$_);
  $f2cnt++;

  # First comparison, do entire row
  for ($cn=0; $cn<=$#row_fh2;$cn++ )
   {
   # molecule 1 compared with molecule 2
   $comp1_1[$cn] = abs($row_fh1[$cn] - $row_fh2[$cn]) /
    ( (abs($row_fh1[$cn]) + abs($row_fh2[$cn])) / 2 );
   }

  # Open file for row loop through third file
  open(FH3,"<$f3") || die "Unable to open file $f3 for input";
  $f3cnt = 0;
  while ( <FH3> )
   {
   chop;
   ($label3,@row_fh3) = split(",",$_);
   $f3cnt++;

   # Second comparison, do entire row
   for ($cn=0; $cn<=$#row_fh3;$cn++ )
    {
    # molecule 1 compared with molecule 3
    $comp2_1[$cn] = abs($row_fh1[$cn] - $row_fh3[$cn]) /
     ( (abs($row_fh1[$cn]) + abs($row_fh3[$cn])) / 2 );
    # molecule 2 compared with molecule 3
    $comp2_2[$cn] = abs($row_fh2[$cn] - $row_fh3[$cn]) /
     ( (abs($row_fh2[$cn]) + abs($row_fh3[$cn])) / 2 );
    }
```

```
    # Open file for row loop through fourth file
    open(FH4,"<$f4") || die "Unable to open file $f4 for input";
    $f4cnt = 0;
    while ( <FH4> )
     {
     chop;
     ($label4,@row_fh4) = split(",",$_);
     $f4cnt++;
     $ocnt++;
     $total_total = 0;
     $format = "%s,%s,%s,%s";

     # Third comparison, do entire row
     for ($cn=0; $cn<=$#row_fh4;$cn++ )
      {
      # molecule 1 compared with molecule 4
      $comp3_1[$cn] = abs($row_fh1[$cn] - $row_fh4[$cn]) /
       ( (abs($row_fh1[$cn]) + abs($row_fh4[$cn])) / 2 );
      # molecule 2 compared with molecule 4
      $comp3_2[$cn] = abs($row_fh2[$cn] - $row_fh4[$cn]) /
       ( (abs($row_fh2[$cn]) + abs($row_fh4[$cn])) / 2 );
      # molecule 3 compared with molecule 4
      $comp3_3[$cn] = abs($row_fh3[$cn] - $row_fh4[$cn]) /
       ( (abs($row_fh3[$cn]) + abs($row_fh4[$cn])) / 2 );

      $tot_comp[$cn] = ( $comp1_1[$cn] + $comp2_1[$cn] +
       $comp2_2[$cn] + $comp3_1[$cn] + $comp3_2[$cn] +
       $comp3_3[$cn] ) / 6;
      $total_total += $tot_comp[$cn];
      $format .= ",%s";
      }

     # similarity score
     # Do the final new column now, we have the data for the row already
     $divisor = $#row_fh4 + 1; # Number of numerical columns
     $total_total = $total_total / $divisor;

     $format .= ",%s";
     $format .= "\n";
     # Print out the row for the new spreadsheet
     printf OF $format,$label1,$label2,$label3,$label4,@tot_comp,$total_total;


     } #end FH4
     close FH4;
    } #end FH3
    close FH3;
   } #end FH2
   close FH2;
  } #end FH1
  close FH1;


# end time
```

```
$end = time();
$truns = $end - $start;
#start time, end time, total time written to file
print TI "start run time is ", scalar localtime($start), "\n";
print TI "end run time is ", scalar localtime($end), "\n";
print TI "total run time is ", $truns, " seconds \n";
close TI;
```

Appendix 4.3 The three molecule Perl script. Calculates and saves the similarity scores for three molecule comparisons.

```
#!/usr/bin/perl

# Program:  msc_4_molecules.pl
# authors:  Paul A. Wilson, Larry Beintema
# email:  pwilson@cobremail.itrc.umt.edu
# date of latest version:  August 22, 2004
# date of origination: late 1990s
#
#  Sybyl Programming Language - run from within the Sybyl program by Tripos
#
#  Change the path and name of the input files below in the code as appropriate.
#  As well, changethe name of the ouput files as appropriate.
#  space, until the system crashes making this program unusable in most cases.
#
#  In Short:  This program compares the rows in data files developing a score
#  for each set of rows (set - one row from each data file).  A text file is
#  written containing all comparisons.  Each row in the output file contains
#  in comma delimited format the label from each label field followed by the
#  score for each field and finally the total score.  This version of the
#  program does not sort the similarity scores.
#
#  similarity score = (summation from measure 1 to total number of measures (
#  summation of from data file 1 to total number of data files ( (|Vab - Vacl /
#  ((|Vabl + |Vacl)/2)) / number of combinatorial )))
#  Vab = measure a of datafile b
#  Vac = measure a of datafile c
#


# Do compares a row at a time on the following 3 files (spreadsheets)
# These are comma delimited files, first column is label, rest are numbers
# NOTE: Perl array element indices start at 0 !


# Time file
# Change the path and name as appropriate
open( TI, ">times_3_molecule.out" )
 || die "Unable to open file times.record for time recording";

# start time
$start = time();



# Input files
$f1 = "./Sertraline_nq3.txt";
$f2 = "./Indatraline_nq3.txt";
$f3 = "./S-Citalopram_nq3.txt";

# Output file
#  Data output file, change the path and name as appropriate
```

```
open(OF,">msc_3_output.txt") || die "Unable to open file test.output for output";

#  Error output file, change the path and name as appropriate
#  If everything is going well this file will remain empty
open(EF,">test_3_molecules.error") || die "Unable to open file test.error for output";


# Open file for row loop through first file
open(FH1,"<$f1") || die "Unable to open file $f1 for input";
$f1cnt = 0;
while ( <FH1> )
 {
 chop;
 ($label1,@row_fh1) = split(",",$_);
 $f1cnt++;

 # Open file for row loop through second file
 open(FH2,"<$f2") || die "Unable to open file $f2 for input";
 $f2cnt = 0;
 while ( <FH2> )
  {
  chop;
  ($label2,@row_fh2) = split(",",$_);
  $f2cnt++;

  # First comparison, do entire row
  for ($cn=0; $cn<=$#row_fh2;$cn++ )
   {
   # molecule 1 compared with molecule 2
   $comp1_1[$cn] = abs($row_fh1[$cn] - $row_fh2[$cn]) /
    ( (abs($row_fh1[$cn]) + abs($row_fh2[$cn])) / 2 );
   }

  # Open file for row loop through third file
  open(FH3,"<$f3") || die "Unable to open file $f3 for input";
  $f3cnt = 0;
  while ( <FH3> )
   {
   chop;
   ($label3,@row_fh3) = split(",",$_);
   $f3cnt++;
   $ocnt++;

   $total_total = 0;
   $format = "%s,%s,%s";

   # Second comparison, do entire row
   for ($cn=0; $cn<=$#row_fh3;$cn++ )
    {
    $comp2_1[$cn] = abs($row_fh1[$cn] - $row_fh3[$cn]) /
     ( (abs($row_fh1[$cn]) + abs($row_fh3[$cn])) / 2 );
    $comp2_2[$cn] = abs($row_fh2[$cn] - $row_fh3[$cn]) /
     ( (abs($row_fh2[$cn]) + abs($row_fh3[$cn])) / 2 );
```

```
        $tot_comp[$cn] = ( $comp1_1[$cn] +
         $comp2_1[$cn] + $comp2_2[$cn] ) / 3;
        $total_total += $tot_comp[$cn];
        $format .= ",%s";
        }


      # similarity score
      # Do the final new column now, we have the data for the row already
      $divisor = $#row_fh3 + 1; # Number of numerical columns
      $total_total = $total_total / $divisor;

      $format .= ",%s";
      $format .= "\n";
      # Print out the row for the new spreadsheet
      printf OF $format,$label1,$label2,$label3,@tot_comp,$total_total;


      } #end FH3
      close FH3;
     } #end FH2
     close FH2;
    } #end FH1
    close FH1;


# end time
$end = time();
$truns = $end - $start;
#start time, end time, total time written to file
print TI "start run time is ", scalar localtime($start), "\n";
print TI "end run time is ", scalar localtime($end), "\n";
print TI "total run time is ", $truns, " seconds \n";
close TI;
```

A Practical Comparison of Multiprocessing Libraries

Application of MPI and OpenMP

Paul A. Wilson
pwilson@cobremail.itrc.umt.edu

Abstract

The comparative similarity between two or more mutable entities can be determined by assigning a similarity score, based on relative difference, to every possible variant combination between the entities. In this specific case, a program was written to determine the most similar low energy conformations of four molecules which have high efficacy for selective inhibition at the serotonin transporter. The large data set, 1.3 million conformational clusters, used in this endeavor have made multithreading and multiprocessing inviting opportunities. The initial program has been rewritten to either take advantage of OpenMP or MPI. These two versions allow for the comparison of serial, OpenMP, and MPI strategies on dual processor Power Macs (Mac OS 10.2) and on a nine node, eighteen processor linux Beowulf.

Introduction

Situations arise where sets of data need to be compared in order to determine subsets possessing maximum similarities or dissimilarities. When these data sets consist of comparable measurements, which are relevant to the problem at hand, one possible solution is to calculate the relative difference within each subset of similar data pairs. Typically the relative difference we are familiar with is used to compare how close an experimentally derived value is to the known value. This is calculated by taking a ratio of the difference between the experimental and known value to the known value (figure 1a). However, in situations where the known value does not exist this ratio can still be developed from two experimentally derived values of the same measure. It becomes the ratio of the difference of the two values to the average of the two values (figure 1b). The numeric values derived from these ratios are without units and of comparable scale.

$$\text{relative difference} = \frac{V_{measured} - V_{known}}{V_{known}}$$

a) relative difference when the correct value is known

$$\text{relative difference} = \frac{\left|V_{measured} - V_{known}\right|}{\left(\left|V_{measured} - V_{known}\right|\right) / 2}$$

b) relative difference when the correct value is unknown

Figure 1. Relative difference equations

By comparing multiple common measures between three objects, the two most similar objects can be determined. The sum of values provided by the relative difference between each pair of common measures provides an objective value to the amount of similarity between the two objects. This value solely relies on the degree to which the objects can be described by their common measures. The magnitude or units of the measures do not affect the outcome.

In the specific case where the work presented here initially took shape, four conformationally dynamic molecules were identified and compared. These four molecules, commonly known as serotonin selective reuptake inhibitors (SSRIs), inhibit the reuptake of serotonin from the synaptic cleft by blocking the serotonin transporter (SERT) and are commonly employed in the treatment of depression, anxiety and

Appendix 4.4 "A Practical Comparison of Multiprocessing Libraries: Application of MPI and OpenMP" presented at MacHack 2003, June 2003.

obsessive-compulsive disorder (OCD) [Vaswani2002]. It is thought that each molecule blocks the SERT through the same mechanism and, accordingly, interacts with the transporter at the same binding domain. Therefore, if the most similar conformation of each molecule is found and subsequently compared with the most similar conformation of each individual molecule within the molecular subset, key points in transporter-ligand interaction can be identified. These established structural commonalities then facilitate mapping of the SERT-SSRI binding domain. These established structural commonalities then facilitate mapping of the SERT-SSRI binding domain and aid in the development of a SERT pharmacophore to be utilized in the design of superior SERT ligands.

$$\frac{\sum_{k=1}^{n_{measures}} \frac{\sum_{i,j} \frac{|V_k^i - V_k^j|}{\left(\frac{|V_k^i + V_k^j|}{2}\right)}}{n_{rdc}}}{n_{measures}}$$

where $1 \le i < j \le$ number of molecules

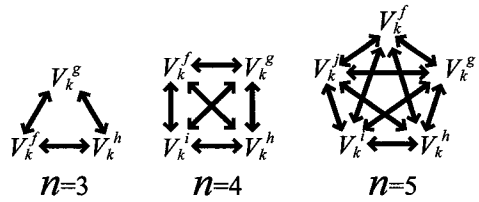
n_{rdc} = number of relative difference calculatior within a conformational comparison gro

$n_{measures}$ = number of compared measures

V_k^{α} = measure k of molecules α

Figure 2. Similarity score

Every conformation of each molecule must be compared with all of the conformations of the other molecules in order to determine the conformational subset that is most similar. The molecules are described using common measures in three-dimensional distance space. The common measurement pairs of each four conformational sets are combinatorially compared (figure 2) using relative difference. The relative differences for each measurement subset of four conformations are then summed (figure 3). These sums can then be used as an index of similarity between subsets of four molecular conformations. The smallest sum equates to the most similar and the largest sum equates to the most dissimilar group of within the four conformations. The smallest sum equates to the most similar and the largest sum equates to the most dissimilar group within the four conformations.



V_k^{α} = measure k of molecules α

$\longleftrightarrow$ = one relative difference calculation

$$\text{total number of relative difference calculations} = \sum_{n=1}^{\text{number of comparison pairs -1}} n$$

Figure 3. The increase in the number of combinatorial relative difference calculations in relation to the increase in comparison pairs, follows an arithmetic series.

Serial Implementation

The program written to perform the relative difference and similarity index calculations reads the initial measured data from comma delimited files. The program uses command line arguments to identify three or more comma delimited data files. The other required data, such as number of columns, number of rows in each file, and output file name, are obtained through interaction with the user. Once this information is known, memory is dynamically set aside for three one-dimensional arrays to store 1) the row labels from each input file, 2) numerical data from

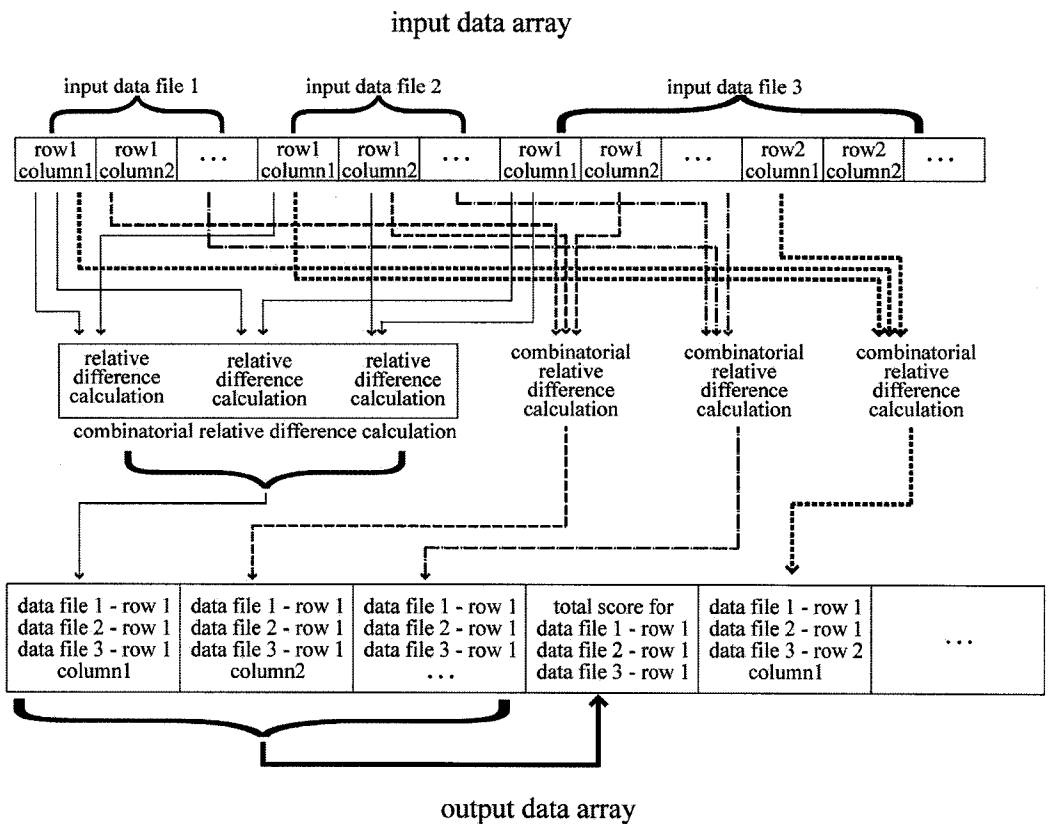


Figure 4. Graphical view of data moving from the input array through the relative difference calculation and into the output array.

each input file and 3) the numerical results of the calculations.

Conceptually, the combinatorial relative difference is calculated between the matching columns of the first row of data in each input file as depicted figure 4. After completion of the first row, the last input file then increments to the second row and the other files reset to the first column of their first rows. This continues until the last data file reaches its final row. At which point the next to the last data file increments to the second row and the last data file resets to its first row. These calculations proceed until every row of each data file has been compared with every other row of the other data files. The result from each calculation is incrementally stored in the results array. Theoretically, the problem is relatively simple with one only having to keep track of indexes for the two one-dimensional input and output arrays. Following completion of all calculations, the labels from the label array and the numerical data from the results array are matched and written to a single file. Each comma delimited row in this file contains the row labels of the rows compared, the combinatorial relative difference for each column and the sum of the combinatorial relative difference.

Four comma delimited text files were produced during the initial stages of the SERT pharmacophore development. Each of these text files had 15 columns of data and one label column. The four files respectively had 12, 16, 29, and 233 rows of data, representing the low energy

```
...

for(ii=1; ii<=oi; ii++)
  {
  for(i=1; i<=(nc); i++)
    {
    for(j=1; j<=argc-2; j++)
      {
      a = index[j];
      for(k=j+1; k<=argc-1; k++)
        {
        b = index[k];

        // relative difference
        //  calculation

        singlemeasurerd = \
          (((fabs(dataarray[a] - \
          dataarray[b])) / (0.5* \
          (fabs(dataarray[a] \
          + dataarray[b])))) \
          + singlemeasurerd);

       }  // end k
      }  // end j

      avgSingleMeasureRD = \
      (singlemeasurerd/(divisor));

      totalmeasurerd = \
        singlemeasurerd/(divisor) \
        + totalmeasurerd;

      reldif[m]=avgSingleMeasureRD;
      m = m + 1;
      singlemeasurerd = 0.0;
      avgSingleMeasureRD = 0.0;
      for(j=1; j<=argc-1; j++)
        {
        a = index[j];
        index[j] = a + 1;
        }
      }  // end i

   buffer3 = totalmeasurerd / nc;

    ...

    }  //  end ii
```

Listing 1. Code fragment showing the nested loop structure of the serial implementation utilized in the relative difference calculation.

conformations identified for each of the four molecules. The product of the number of rows, equaling 1,297,344, represents the total number of combinatorial possibilities. Each of these combinatorial possibilities consists of 15 columns, with each column containing 6 relative difference calculations (as depicted in figure 3, n=4). Collectively, this results in 116,760,960 relative difference calculations for this specific data set. Figure 5 shows the typical amount of wall time required to complete the calculations in serial. The times indicate the amount of time required to calculate 15 columns of combinatorial relative differences and, on a row-by-row basis, the average of these 15 columns.

Parallel Implementation

Demonstrable parallel code can be a key component leading to a project being granted supercomputer time. Most supercomputers follow either a distributed memory model or a shared memory model. It is important to understand these models before writing parallel code.

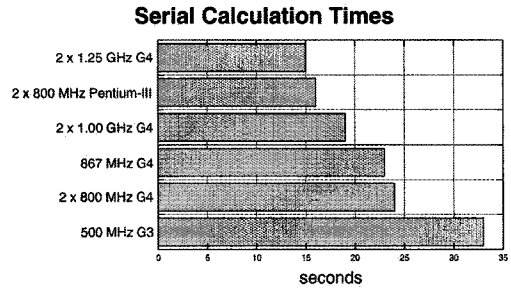


Figure 5. Average times required to process the SERT data set in serial. Test computers: Power Mac dual 1.25 GHz G4 Mac OS 10.2.5, dual processor 800 MHz Pentium-III linux, Power Mac dual 1 GHz G4 Mac OS 10.2.5, PowerBook 12" 867 MHz G4 Mac OS 10.2.5, Power Mac dual 800 MHz G4 Mac OS 10.2.5, PowerBook Pismo 500 MHz G3 Mac OS 10.2.5.

```
index[1] = (((numberOfRows[1] \
     / numprocs) * myid * nc) );
trackindex[1] = (((numberOfRows[1] \
     / numprocs) * myid * nc) );
...
  oi = (prt / numprocs);
...
for(ii=(myid*oi)+1; \
    ii<=((myid+1)*oi); ii++)
  {
  for(i=1; i<=(nc); i++)
    {
    for(j=1; j<=argc-2; j++)
      {
      a = index[j];
      for(k=j+1; k<=argc-1; k++)
        {
        b = index[k];

        // relative difference
        //  calculation

        singlemeasurerd = \
          (((fabs(dataarray[a] - \
          dataarray[b])) / (0.5* \
          (fabs(dataarray[a] \
          + dataarray[b]))))) \
          + singlemeasurerd);

       }  // end k
      }  // end j

      avgSingleMeasureRD = \
      (singlemeasurerd/(divisor));

      totalmeasurerd = \
        singlemeasurerd/(divisor) \
        + totalmeasurerd;

      reldif[m]=avgSingleMeasureRD;
      m = m + 1;
      singlemeasurerd = 0.0;
      avgSingleMeasureRD = 0.0;
      for(j=1; j<=argc-1; j++)
        {
        a = index[j];
        index[j] = a + 1;
        }
      }  // end i

   buffer3 = totalmeasurerd / nc;
     ...
    }  //  end ii
```

Many supercomputers, including Beowulf clusters follow a distributed memory model. Each processor has its own memory and its own local data. Processors, or nodes, can communicate using common networking technology, such as ethernet. In theory, code following this model will reach completion, or process data faster by allotting more processors to the task. In the case of a cluster, this can be achieved by connecting more computers, or nodes.

In contrast, the shared memory model refers to all processors sharing common memory space. This type of common memory sharing is exemplified in dual processor computers. Adding processors to a shared memory space has been shown to be challenging and incurs prohibitive expense. In other words, unless access is granted to a supercomputer, one is limited to the two processors found in the typical multiprocessor personal computer.

Communication between processes is the slow step in parallel computing. Looking at the nested loops of the serial code (listing 1), the two inner most loops calculate the combinatorial relative difference for one column of a row. In the case of the SERT data set, every time these two loops are completed six relative difference

Listing 2, to the left. A code fragment showing parallelization using MPI. MPI numbers processes starting at zero and going to total number of processes minus one. The first two lines of the code fragment cause unique equidistant starting points within the data from the first data file to be assigned to each process. Each process then cycles through the outermost `for` *loop from their unique starting point to the point which immediately precedes the starting point of the next process (or the point which is equivalent to the last row in the first data file). The obvious problem with this particular code occurs when the number of rows in the first data file are not evenly divisible by the number of processes.*

calculations have been performed. Any benefit from parallelizing these two loops may not be seen until the number of files being compared approaches 50. A similar situation is present in the second most outer loop. Every cycle through this loop accounts for one row of the data sets being processed. Again, benefit would not be evident from parallelizing this loop until the input files approach the neighborhood of a thousand columns. For every cycle through the outer most loop, one row of the first data file has been compared with all rows

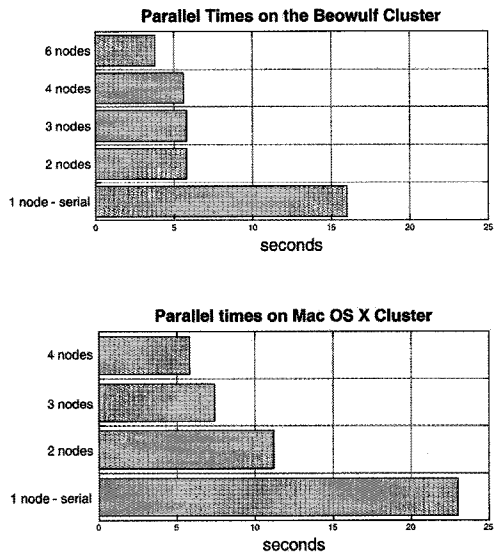


Figure 6. Average time required to process data in serial and in parallel. The Beowulf cluster consists nine dual 800 MHz Pentium-III nodes. The heterogeneous Mac OS X cluster consisted of: node 1) PowerBook G4 12" 867 MHz G4, node 2) Power Mac dual 1.25 GHz G4, node 3) Power Mac dual 1 GHz G4, node 4) Power Mac dual 800MHz G4. All Apple computers were running Mac OS 10.2.5. MPICH was the implementation of MPI running on both clusters. In the case of the Mac OS X cluster, a common shared file system had to be created. This was simulated by mounting the PowerBook's hard drive on each Power Mac and then creating a symbolic link from each PowerMac's root directory to the MPI program residing on the PowerBook's hard drive.

```
...
for(ii=1; ii<=oi; ii++)
  {
  for(i=1; i<=(nc); i++)
    {
    #pragma omp parallel for \
            private(a, b, k)
    for(j=1; j<=argc-2; j++)
      {
      a = index[j];
      for(k=j+1; k<=argc-1; k++)
        {
        b = index[k];
          // relative difference
            // calculation
        singlemeasurerd = \
          (((fabs(dataarray[a] - \
          dataarray[b])) / (0.5* \
          (fabs(dataarray[a] \
          + dataarray[b])))) \
          + singlemeasurerd);
       }  // end k
      }  // end j

      avgSingleMeasureRD = \
      (singlemeasurerd/(divisor));

      totalmeasurerd = \
        singlemeasurerd/(divisor) \
        + totalmeasurerd;
          .
      reldif[m]=avgSingleMeasureRD;
      m = m + 1;
      singlemeasurerd = 0.0;
      avgSingleMeasureRD = 0.0;
      for(j=1; j<=argc-1; j++)
        {
        a = index[j];
        index[j] = a + 1;
        }
      }  // end i
    buffer3 = totalmeasurerd / nc;
     ...
    }  //  end ii
```

Listing 3. Code fragment with the OpenMP directive for parallelizing a loop in bold. The loop is automatically parallelized with the number of threads used during execution defined by the environment variable OMPC_NUM_PROCS. The OMP_NESTED environment variable can be used to control parallelization of nested loops. The code was compiled with the Omni OpenMP compiler.

of the other data files. In the case of the SERT data set one cycle through the outer most loop accounts for 7,297,560 relative difference calculations. A substantial time savings could be attained through the parallelization of this loop.

Message Passing Interface

The Message Passing Interface (MPI) standard was developed, in part, to allow portability of parallel code between supercomputers. MPI is a library for enabling interprocess communication. An implementation of MPI is generally available at supercomputer centers on distributed memory machines. Code can easily be written and tested with limited computing facilities and remains highly portable to computers of greater computational capacity.

Using MPI requires consideration of how a program can efficiently be split into separate processes and what communication needs to take place between these processes. In the case of the relative difference program the data from the first input file will be divided by the number of processes. Depending on the number of rows in this input file being evenly divisible by the number of processes (processors assign to this task), the work load will be equally split amongst processors for the duration of the relative difference calculations. The variables used to keep track of which piece of data is currently being acted upon will need to be changed to indicate the start point within the data set for each process. This can be seen in the code fragment in listing 2 with the impact of this MPI parallelization on run times shown in figure 6.

OpenMP

The OpenMP compiler directives and library provide a mechanism for shared memory parallel computing. As well, OpenMP can be used in conjunction with MPI to take advantage of the processing capabilities available in clusters of dual processor nodes. OpenMP provides easily accessible methods for parallelizing loops, as seen in listing 3. These methods for loop parallelization work well with simple loops. Mechanisms exist in OpenMP which allow complex sections of code and nested loops to be parallelized. In the code shown here, the loop that cycles through the calculation of the combinatorial relative difference for one data point from each input file has been parallelized. As previously mentioned this may prove to be useful with an increased number of input files. In the case of the SERT data set, there is diminished performance, probably due to initial communication overhead. The computation time increased on one, two processor Beowulf node from 16 second for serial to 80 seconds for the OpenMP version.

Conclusion

When putting in the effort to effectively parallelize code, it is gratifying to see the performance gains from running on four, eight, or thirty two processors. Given the opportunity, which is largely dependant on the available equipment, pursuing a MPI solution will afford a notable increase in performance. If limited to a dual processor machine or a shared memory supercomputer, putting the effort into OpenMP would be the optimum choice.

Acknowledgements

Don Morton - The Montana Rockies Center for Computational Science (MRoCCS) at the University of Montana
John Gerdes - The Molecular Computational Core Facility of the NIH-COBRE Center for Structural and Functional Neuroscience at the University of Montana
Jennifer Parham - Department of Computer Science, University of Montana
Melodie Weller - Department of Pharmaceutical Science, University of Montana

Bibliography

Chandra, Rohit, et al. Parallel Programming in OpenMP. Morgan Kaufmann Publishers, San Francisco, CA. 2001.

Foster, Ian. Designing and Building Parallel Programs. Addison-Wesley Publishing Company, Reading, MA. 1995.

Gropp, William, et al. Using MPI. The MIT Press, Cambridge, MA. 1999.

[Vaswani2002] Vaswani, Meera, et al. Role of Selective Serotonin Reuptake Inhibitors in Psychiatric Disorders: A Comprehensive Review. Progress in Neuro-Psychopharmacology and Biological Psychiatry 27, 85-102. Elsevier, New York, NY 2003

Resources

High Performance Computing on Mac OS X
http://gravity.psu.edu/~khanna/hpc.html

MPI
http://www-unix.mcs.anl.gov/mpi/

MPICH
http://www-unix.mcs.anl.gov/mpi/mpich/

OpenMP
http://www.openmp.org/

Omni OpenMP
http://phase.etl.go.jp/Omni/

GNU Portable Threads and Multithreading Libraries
http://www.gnu.org/software/pth/related.html

Appendix 4.5 The compsortall program, written in C. Calculates, sorts and saves all similarity scores.

```
/*

Program: compsortall
version: 1.01
author: Paul A. Wilson
email: pwilson@cobremail.itrc.umt.edu
date of latest version: August 22, 2004
date of origination: 2003

command: program name data file 1 data file 2 data file 3
example: ./compsortall ./MCN-5652.txt ./sertraline.txt ./indatraline.txt ./s-citalopram.txt

minimum of three data files required, no limit to maximum number of data
files to be compared - within reason

required data file format: comma delimited text, first field of each row is
equivalent to a label, the remaining fields in each row are numbers, each
field is separated by a comma, no quotes around the labels, no spaces between
fields (commas only)

output file name is limited to a maximum of 64 characters

In Short: This program compares the rows in data files developing a score
for each set of rows (set - one row from each data file). Then sorts the
scores from low to high. Low scores represent the sets of rows which are
most similar. A text file is written containing all comparisons in
ascending order. Each row in the output file contains the label from each
label field (first field in each data file) concatenated together followed
in comma delimited form by the score for each field and finally the total
score. Two text files are created containing intermediary data. These
two files, called tempfile_1 and tempfile_2, can be deleted after the
program has completed.

This program was written in response to to a project where distance
space descriptions needed to be compared in order to determine the
conformations of four molecules which were the most similar.
(see SFN 2003 Annual Meeting, Poster Presentation 371.4 and SFN
2004 Annual Meeting, Poster Presentation 922.1, also see Machack 18,
Wilson, P.A. "A Practical Comparison of Multiprocessing Libraries",
MacHack 2003, June 2003, the code is different but a lot of the ideas
remain) I am willing to share manuscript versions of these posters in
pdf format.

Comparisons are carried out combinatorially using relative difference.
The relative difference equation was modified slightly to enable creating
and weighting the comparison of a positive and negative measurement as less
similar. Each row in each data file is compared against each row of every
other data file. Within each row, each field is compared against the
corresponding field from the rows being compared in the other data files.
If there are 4 data files than there are 6 comparisons between each field
of the four rows being compared. The relative difference score from each
```

field is added together and divided by the number of fields to produce a similarity score for the for the rows being compared.

similarity score = (summation from measure 1 to total number of measures (summation of from data file 1 to total number of data files ((|Vab - Vac| / ((|Vab| + |Vac|)/2)) / number of combinatorial)))
Vab = measure a of datafile b
Vac = measure a of datafile c

The program programs structure has a legacy stemming from a personal interest in out-of-core sorting, dynamic memory allocation, and parallel computing. There are three versions of this program. This one is sorts all scores using an out of core sorting mechanism. The other two versions of the program 1) does not sort, and 2) one sorts all but only keeps a user determined portion of the scores. The out of core version, here, sorts a group of calculated scores to a file, calculates and sorts the next group of score, reads part of the sorted file back into memory, merges the two sets of sorted scores together maintaining ascending order, and writes out to a new temporary file. The process is repeated until all of the first temporary file is read in and all of the scores have been written out to the second file. The algorithm works well and is fast, except the file I/O step. It would be interesting at some point to look into this again, using a parallel file system (or at least a high speed file system).

Pointers are used extensively as port of some of my original notions on dynamically allocating memory and sorting.

The program was written in C to provide maximum portability. I have not tried compiling this code on a Linux, or any other, machine. I have noticed this code runs great under Mac OS 10.3 and failed when running under Mac OS 10.2.

*/

```
//#include <dirent.h>
#include <fcntl.h>
#include <math.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/errno.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/uio.h>

// structure for holding data read from files, label and numerical descriptors
struct descriptions
 {
 char * labels;
```

```
 double * descriptivedata;
 };

// structure for holding concatenated label and score for each descriptor
// and a total score value
struct labelscores
 {
 char * label;
 double * reldif;
 };

// from here to main are a set of function used for sorting
// the sorting algorithm is a quicksort, with a mean of 3 used for determining
// the first pivot point - see Data Structures and Algorithm
// Analysis in C by Mark Allen Weiss


// swap redirects pointers - u will point to v and v will point to u
// Swap - needs to be compiled inline for efficiency

void Swap(struct labelscores ** u, struct labelscores ** v)
 {
 struct labelscores temp;
 temp = **u;
 **u = **v;
 **v = temp;
 }


// insertion sort - used as part of quicksort
// scoreisort - struct of labelscores containing scores to be sorted by the insertion sort method
// n is the upper edge of of the chunk data the insertion sort is occurring on
// numdecriptsis - number of description insertion sort,
//    is equal to the number of descriptive fields in the data files
//    used for for malloc of temporary insertion sort struct of labelscores
// totallabelelngthis - total label length insertion sort,
//    is equal to the maximum number of characters in concatenated labels
//    used for for malloc of temporary insertion sort struct of labelscores

void InsertionSort(struct labelscores ** scoreisort, int n, int numdescriptsis, int totlablengthis)
 {
 int j, p;    // j and p are indexes used in the insertion sort
 int descriptindexis;    // an index for descriptive field
 char * nullstring = "\0";    // nullstring needed at the end of strings

 struct labelscores tempisort;    // temporary labelscore struct used in insertion sort

 tempisort.label = (char *) malloc(sizeof(char) * (totlablengthis));  // allocate memory
 tempisort.reldif = (double *) malloc((numdescriptsis + 1) *sizeof(double));
 (char *)tempisort.label = strcpy(tempisort.label, nullstring);
 for(descriptindexis=0; descriptindexis < numdescriptsis+1; descriptindexis++)
  {
  tempisort.reldif[descriptindexis] = 0.0;
  }
```

```
// insertion - copy one struct out, move the rest, then put the struct back were it goes
for(p=1; p<n; p++)
 {
 tempisort.label = strcpy(tempisort.label, scoreisort[p]->label);
 for(descriptindexis=0; descriptindexis <= numdescriptsis; descriptindexis++)
  {
  tempisort.reldif[descriptindexis] = scoreisort[p]->reldif[descriptindexis];
  }
 for(j=p; ((j > 0) && (scoreisort[j-1]->reldif[numdescriptsis] > tempisort.reldif[numdescriptsis])); j--)
  {
  scoreisort[j]->label = strcpy(scoreisort[j]->label, scoreisort[j-1]->label);
  for(descriptindexis=0; descriptindexis <= numdescriptsis; descriptindexis++)
   {
   scoreisort[j]->reldif[descriptindexis] = scoreisort[j-1]->reldif[descriptindexis];
   }
  }
  scoreisort[j]->label = strcpy(scoreisort[j]->label, tempisort.label);
  for(descriptindexis=0; descriptindexis <= numdescriptsis; descriptindexis++)
   {
   scoreisort[j]->reldif[descriptindexis] = tempisort.reldif[descriptindexis];
   }
 }
 free(tempisort.label);   // free memory previously malloced
 free(tempisort.reldif);
}


// median of three  - selecting the pivot for the quicksort
//
// scoremedian - struct of labelscores, postion of left(first) score, postion of right(last) score
// numdescriptsm - number of descripts median of 3, the last descript field - the total similarity score
// the median value of the three sampled values is determined and return to the quicksort routine
// to be used as the pivot point

float Median3(struct labelscores **scoremedian, int left, int right, int numdescriptsm, int totlablengthm)
 {
 int center;
 center = (left + right ) / 2;   // position of the center score - median position

 if (scoremedian[left]->reldif[numdescriptsm] > scoremedian[center]->reldif[numdescriptsm])
  {
  Swap(&scoremedian[left], &scoremedian[center]);
  }
 if (scoremedian[left]->reldif[numdescriptsm] > scoremedian[right]->reldif[numdescriptsm])
  {
  Swap(&scoremedian[left], &scoremedian[right]);
  }
 if (scoremedian[center]->reldif[numdescriptsm] > scoremedian[right]->reldif[numdescriptsm])
  {
  Swap(&scoremedian[center], &scoremedian[right]);
  }

 Swap(&scoremedian[center], &scoremedian[right-1]);
```

```
 return scoremedian[right-1]->reldif[numdescriptsm];
 }


// quicksort
// scoreqsort - the struct of labelscores to be sorted by quicksort
// leftqs, rightqs - the lower and upper boundaries of the quicksort - quicksort is recursive
// numdescriptsqs - number of descriptive fields quicksort - used for allocating necessary memory
// totlablength - total label length quick sort - maximum number of characters in a label
//   used for allocating necessary memory

void Qsort(struct labelscores **scoreqsort, int leftqs, int rightqs, int numdescriptsqs, int totlablengthqs)
 {
 int i, j;   // indexes used in quicksort

 struct labelscores pivot;   // labelscores struct which hold the pivot

 pivot.label = (char *) malloc(sizeof(char) * (totlablengthqs));   // allocating memory for pivot
 pivot.reldif = (double *) malloc((numdescriptsqs+1) *sizeof(double));

 // heart of quicksort, pick pivot swap appropriate values to either side of pivot
 if ((leftqs + 3) <= rightqs)
  {
  pivot.reldif[numdescriptsqs] = Median3(scoreqsort, leftqs, rightqs, numdescriptsqs, totlablengthqs);
  i = leftqs;
  j = rightqs-1;
  for( ; ; )
   {
   while(scoreqsort[++i]->reldif[numdescriptsqs] < pivot.reldif[numdescriptsqs] && i < rightqs-1){}
   while(scoreqsort[--j]->reldif[numdescriptsqs] > pivot.reldif[numdescriptsqs] && j > 0){}
   if(i < j)
    Swap(&scoreqsort[i], &scoreqsort[j]);
   else
    break;
   }
  Swap(&scoreqsort[i], &scoreqsort[rightqs-1]);

  Qsort(scoreqsort, leftqs, i-1, numdescriptsqs, totlablengthqs);
  Qsort(scoreqsort, i+1, rightqs, numdescriptsqs, totlablengthqs);
  }
 else   // if left and right a close together do an insertion sort
  {
  InsertionSort(scoreqsort+leftqs,  rightqs-leftqs+1, numdescriptsqs, totlablengthqs);
  }

 free(pivot.label);   // freeing allocated memory
 free(pivot.reldif);
 }


// start of main

int main (int argc, const char * argv[])
 {
```

```
int bufferidx;   // used when determining if end of array reached
int compindex;   // comparison index - which comparison set is being compared
int compindex2;   // index used for resetting score array
int compstart;   // comparison start - the starting point for each set of comparisons
int confindex;   // conformation index
int descriptindex;   // desciptor index - data file columns
int descriptindex2;   // index used for resetting score array
int entityindex;   // id of entity one in the comparison
int entityindex2;   // id of entity two in the comparison
int entityidxbuff;   // entity index buffer
int fileinidx;   // file in index, index for reading temp file
int fileoutidx;   // file out idx, index for writing temp out
int fileindex=1;   // index used to indicate which data file is open
int i;   // simple index used in loop calculating the divisor
int lastarraysz;   // the size of the last array, the remainder of calculations
int lseekbuffer;   // buffer for file position
int lseekposition;   // file position in data file being read
int maxarraysz;   // maximum array size = numlinescalc + numbestscore
int maxfinsz;   // max file in size, size of array to be read from temporary file
int maxfoutsz;   // max file out size, size of array to be filled before writing to file
int maxscoreidx;   // the last comparison in a comparison set
int numdescripts;  // total number of descriptions = number of columns in each data file
int numcomps;   // total number of comparison sets
int numentities;   // the total number of entities to be compared = number of data files
int numcalcsets;   // the number of times temporary files need to be created
int reldifidx;   // relative diference index
int tmpfileflag;   // used to write to the correct temporary file
int calcsetidx;  // index for the number of times temporary file created
int totlabel;   // total length, number of characters, of concatenated label

int * conformationidx[argc-1];   //  index of conformations
int * filedescript[argc-1];   // array of file descriptors - data files from command line
int * labellength[argc-1];   //  array of the label lengths for each data file
int * maxlabellength[argc-1];   //  array of the maximum label lengths for each data file
int * numcolumns;   // number of columns
int * numconfs[argc-1];   //  array of the number of rows in each data file
int * tmpfile1fdp;   // temp file 1 file descriptor
int * tmpfile2fdp;   // temp file 2 file descriptor
int * tmpfile1pos;   // temporarily holds file position in temp file 1
int * tmpfile2pos;   // temporarily holds file position in temp file 2

char * nullstring = "\0";   // nullstring for adding the end of lines and initialization
char * readbuffer;   // read buffer for reading data files
char * strnumber;   // numerical data read from file stored as string
char * tmpfile1;   // stores name of temporary file 1 - tempfile_1
char * tmpfile2;   // stores name of temporary file 2 - tempfile_2

char outfilename[64];   // character array for output file name - max 65 character name

double descriptscore = 0.0;   // the relative difference score for one combinatorial
double divisor = 0.0;   // the divisor used in calculating the single measure relative difference
double doublenumber = 0.0;   // numerical string read from data file is converted to double number
double singlemeasureRD = 0.0;   // the Relative Difference score for a single set of description
```

```
double totalmeasureRD = 0.0;   // the Relative Difference score for a comparison set
double totaltime =0.0;   // the total time in seconds used for timing sorts and run time
double totaltimecalc = 0.0;   // the total time in seconds used for timing calculation time
double totaltimemerge = 0.0;   // the total time in seconds used for timing merge sort time
double totaltimeqsort = 0.0;   // the total time in seconds used for timing quick sort time
double totaltimereada = 0.0;   // the total time in seconds used for timing initial file read time
double totaltimereadb = 0.0;   // the total time in seconds used for timing post initial file read time
double totaltimewrite = 0.0;   // the total time in seconds used for timing file write time
double totaltimeoocsort = 0.0;   // the total time in seconds used for timing out of core sort time

FILE *outfilefd;   // the output file descriptor
FILE *tmpfile1fd;   // temp file 1 file descriptor
FILE *tmpfile2fd;   // temp file 2 file descriptor

time_t starttimecalc;  // start time for set of calculations
time_t endtimecalc;  // end time for set of calculations
time_t starttimemerge;  // start time for merge sort
time_t endtimemerge;  // end time for merge sort
time_t starttimeoocsort;  // start time for out of core sort
time_t endtimeoocsort;  // end time forout of core sort
time_t starttimeread;  // start time for a temporary file read
time_t endtimeread;  // end time for a temporary file read
time_t starttimeqsort;  // start time for quicksort
time_t endtimeqsort;  // end time for quicksort
time_t starttimetotal;  // start time for total run time
time_t endtimetotal;  // end time for total run timet
time_t starttimewrite;  // start time for a temporary file write
time_t endtimewrite;  // end time for a temporary file write

struct descriptions ***conformation;  // struct which stores all the data from the data files

struct labelscores **score;   // struct which stores concatenated labels and scores
struct labelscores **filein;   // struct which stores labels and scores from temp file read
struct labelscores **fileout;   // struct which stores labels and scores to write to temp file
struct labelscores scorebuffer;   // a buffer to temporarily store scores in

//  initial memory allocation and variable initialization
readbuffer = malloc(sizeof(char));
strnumber = malloc(sizeof(char) * 16);  // this needs to be increased if numbers are
                                        // more than 15 digits plus 1 NULL string

tmpfile1 = malloc(10 * sizeof(char)); // 10 characters - tempfile_1
tmpfile2 = malloc(10 * sizeof(char)); // 10 characters - tempfile_2
tmpfile1 = "tempfile_1";
tmpfile2 = "tempfile_2";
tmpfileflag = 1;

tmpfile1fdp = (int *) malloc(sizeof(int));
tmpfile2fdp = (int *) malloc(sizeof(int));

tmpfile1fd = (FILE *) malloc(sizeof(int));
tmpfile2fd = (FILE *) malloc(sizeof(int));

numcolumns = (int *) malloc(sizeof(int));
```

```
*filedescript = (int *) malloc(sizeof(int) * (argc - 1));

tmpfile1pos = (int *) malloc(sizeof(int));
tmpfile2pos = (int *) malloc(sizeof(int));

numentities = argc - 1;
confindex = 0;
numcomps = 1;
totlabel = 1;

// print information to the screen which will help the user answer the
// first three questions they are asked by the program
printf("\n\n\nThis program is going to calculate X number of scores, ");
printf("\nsort those scores, and writes them to a temporary file, and then ");
printf("\ncalculates the next X number of scores. This latest set of X");
printf("\nscores are sorted together with X scores read from the temporary ");
printf("\nfile, and as soon as X scores have been sorted they are written to ");
printf("\na new temporary file. This continues until all X calculated scores ");
printf("\nand the first temporary file have been sorted and written to ");
printf("\nthe second temporary file. This process continues calculating, ");
printf("\nreading from, and writing to two temporary files until all scores ");
printf("\nhave been calculated, sorted and written to an output file in ");
printf("\nascending order.\n");

printf("\nThe answers given to the next three question will determine ");
printf("\nhow much memory is allocated. Remember there are limits ");
printf("\nto how much memory can be allocated to a single application ");
printf("\nand virtual memory is slower than physical memory. ");
printf("\nYou are about to be asked for the number of scores to be ");
printf("\ncalculated between writes to a temporary file. This is also ");
printf("\nthe number of scores to read from the temporary file at a time, ");
printf("\nand the number of scores to write to the second temporary file ");
printf("\nat a time.\n");
printf("\nThe product of 3X accounts for a majority of the memory used ");
printf("\nby this program. Calculating X number of scores, reading X");
printf("\nnumber of scores, and writing X number of scores at a time");
printf("\n allows for the program to remain within a limited memory");
printf("\nfootprint. An example value that has proven useful to the");
printf("\nauthor is: X = 500000. ");
printf("\nThis values should be changed according to your data set,");
printf("\nneeds, and physical memory.\n");

// ask for and obtain the number of conformational sets to calculate
// between the "sort and store" steps
printf("\nHow many scores do you want to calculate before sorting, ");
printf("\nreading from, and writing to temporary files? ");
scanf("%i", &maxarraysz);

maxfinsz = maxarraysz;  // setting out of core file in and file out arrays
maxfoutsz = maxarraysz;  // the same size as maxarraysz, prevents possible
                // problems from having different size arrays

// allocating memory for input data array
conformation = (struct descriptions ***) malloc (sizeof(struct \
```

```
 descriptions **) * numentities);

// determined from arguments on comand line
printf("\nThe number of data files = %i\n", numentities);

// asking for the number of columns per data file
printf("\nWhat is the total number of columns in each file? ");
scanf("%i", (int *)&numcolumns);

numdescripts = (int)numcolumns - 1;
printf("\nEach file has %i columns, 1 label column and %i description columns.\n", \
 (int)numcolumns, numdescripts);

// allocating memory and loading the data arrays

for (entityindex=0; entityindex < numentities; entityindex++)
 {
 maxlabellength[entityindex] = (int *)malloc(sizeof(int));
 *maxlabellength[entityindex] = 0;
 if (((int)filedescript[fileindex] = (open(argv[fileindex], O_RDONLY))) < 0)
  {
  perror(argv[fileindex]);
  exit(EXIT_FAILURE);
  }
 // asking for the number of rows in the current data file being read
 printf("\n\nfile %s is open\n", argv[fileindex]);
 printf("How many rows are in this file? ");
 scanf("%i", (int *)&numconfs[entityindex]);
 printf("There are %i rows representing %i conformations in this file.\n", \
 (int)numconfs[entityindex], (int)numconfs[entityindex]);

 // allocating memory for the array of pointers to conformations
 conformation[entityindex] = (struct descriptions **) malloc \
 (( sizeof(struct descriptions *) * (int)numconfs[entityindex]));

 lseekposition = 0;

 // allocating memory for a conformation s structure and descriptors
 for(confindex=0; confindex < (int)numconfs[entityindex]; confindex++)
  {
  conformation[entityindex][confindex] = (struct descriptions *) malloc \
   (( sizeof(struct descriptions)));
  conformation[entityindex][confindex]->descriptivedata = (double *) \
   malloc(sizeof(double) * numdescripts);

  // allocating memory for a conformation s structure label
  labellength[entityindex] = (int *)malloc(sizeof(int));
  fflush(NULL);
  *readbuffer = '\0 ;
  *labellength[entityindex] = 0;
  lseekbuffer=lseekposition;

  // counting number of characters in label
  // fscan would be faster and was used for file I/O in the
```

```
// out-of-core sortng program
while(*readbuffer != ' ' && *readbuffer != ', )
 {
 lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
 read((int)filedescript[fileindex], readbuffer, 1);
 *labellength[entityindex] = *labellength[entityindex] + 1;
 lseekposition++;
 }

if (*maxlabellength[entityindex] < *labellength[entityindex])
 {
 *maxlabellength[entityindex] = *labellength[entityindex];
 // printf("\nmaximum label length = %i\n", *maxlabellength[entityindex]);
 }

// allocating memory for label - depends on label maximum length
conformation[entityindex][confindex]->labels = \
(char *) malloc(sizeof(char) * *labellength[entityindex]);

// reading, concatenating, and storing label
fflush(NULL);
*readbuffer = '\0 ;
lseekposition = lseekbuffer;
descriptindex = 0;
while(*readbuffer != ' ' && *readbuffer != ', )
 {
 lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
 read((int)filedescript[fileindex], readbuffer, 1);
 if(*readbuffer != ', )
  {
  (char *)conformation[entityindex][confindex]->labels = \
  strncat( conformation[entityindex][confindex]->labels, readbuffer, 1);
  }
 lseekposition++;
 }

// resetting and reading numerical descriptor data from data files
// data read as string and converted to double - strtod
while(*readbuffer != '\n  && *readbuffer != '\0 )
 {
 fflush(NULL);
 *readbuffer = \0 ;
 *strnumber = '\0 ;
 while(*readbuffer != ',  && *readbuffer != ' ' && *readbuffer != '\n )
  {
  lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
  read((int)filedescript[fileindex], readbuffer, 1);
  if (*readbuffer != ',  && *readbuffer != ' ' && *readbuffer != '\n )
   {
   strnumber = strncat(strnumber, readbuffer, 1);
   lseekposition++;
   }
  }
 doublenumber = strtod(strnumber, NULL);
```

```
    conformation[entityindex][confindex]->descriptivedata[descriptindex]=\
    doublenumber;
    descriptindex++;
    lseekposition++;
    }
   }
  fileindex = fileindex + 1;  // finished reading from data file move to next file
  }

// asking for and obtaining name of output file
fflush(NULL);
printf("\n\nEnter file name for results to be stored in: ");
scanf("%s", (char *)&outfilename);
printf("The results will be saved in %s\n", outfilename);

starttimetotal = time(NULL);  // start time for total time

// calculating total number of comparisons and maximimum total label length
for (entityindex=0; entityindex < numentities; entityindex++)
  {
  numcomps = numcomps * (int)numconfs[entityindex];
  totlabel = totlabel + *maxlabellength[entityindex];
  }

numcalcsets = (numcomps/maxarraysz);  // number of sets of calculations
                    // number of out of core sorts

// if number of comparisons is less than the size of a set of calculations
if (maxarraysz > numcomps)
  {
  maxarraysz = numcomps;
  maxfinsz = numcomps;
  maxfoutsz = numcomps;
  numcalcsets = 0;
  tmpfileflag = 3;
  }

// calculate the size of the last set of comparisons
lastarraysz = numcomps - (numcalcsets * maxarraysz);

// allocate memory for score array and initialize
score = (struct labelscores **) malloc (sizeof(struct labelscores *) \
 * (int)maxarraysz);

for(compindex=0; compindex <= maxarraysz; compindex++)
  {
  score[compindex] = (struct labelscores *) malloc(sizeof(struct\
   labelscores));
  score[compindex]->label = (char *) malloc(sizeof(char) * (totlabel + numentities));
  score[compindex]->reldif = (double *) malloc((numdescripts+1) * sizeof(double));
  for(descriptindex=0; descriptindex < numdescripts+1; descriptindex++)
    {
    score[compindex]->reldif[descriptindex] = 0.0;
    }
```

```
 }

// allocate memory for file in array and initialize
filein = (struct labelscores **) malloc (sizeof(struct labelscores *) \
 * (int)maxfinsz);

for(compindex=0; compindex <= maxfinsz; compindex++)
 {
 filein[compindex] = (struct labelscores *) malloc(sizeof(struct\
  labelscores));
 filein[compindex]->label = (char *) malloc(sizeof(char) * (totlabel + numentities));
 filein[compindex]->reldif = (double *) malloc((numdescripts+1) * sizeof(double));
 for(descriptindex=0; descriptindex < numdescripts+1; descriptindex++)
  {
  filein[compindex]->reldif[descriptindex] = 0.0;
  }
 }

// allocate memory for file out array and initialize
fileout = (struct labelscores **) malloc (sizeof(struct labelscores *) \
 * (int)maxfoutsz);

for(compindex=0; compindex <= maxfoutsz; compindex++)
 {
 fileout[compindex] = (struct labelscores *) malloc(sizeof(struct\
  labelscores));
 fileout[compindex]->label = (char *) malloc(sizeof(char) * (totlabel + numentities));
 fileout[compindex]->reldif = (double *) malloc((numdescripts+1) * sizeof(double));
 for(descriptindex=0; descriptindex < numdescripts+1; descriptindex++)
  {
  fileout[compindex]->reldif[descriptindex] = 0.0;
  }
 }

// allocating memory for scorebuffer
scorebuffer.label = (char *) malloc(sizeof(char) * (totlabel + numentities));
scorebuffer.reldif = (double *) malloc((numdescripts+1) *sizeof(double));


// memory allocation for main complete
printf("\nmalloc complete - memory allocated\n");

// calculate and print the number of combinatorial comparisons per each set of
// desriptors in a comparison set
divisor = 3;
for(i=3; i<numentities; i++)
 {
 divisor = divisor + i;
 }
printf("\n\nThere are %.0f possible comparisons for each descriptive field", divisor);
printf("\nin each comparison set\n\n");


// initializing variables before relative difference calculation
```

```
entityindex = 1;  // first entity
entityindex2 = 0;  // second entity
descriptscore = 0.0;
descriptindex = 1;  // description being compared
singlemeasureRD = 0.0;
totalmeasureRD = 0.0;
compstart = 0;
(int)conformationidx[entityindex] = 0;  // conformation of entity one
(int)conformationidx[entityindex2] = 0;  // conformation of entity two

for(entityindex=0; entityindex < numentities; entityindex++)
 {
 (int)conformationidx[entityindex] = 0;
 }

// loop through number of calculation sets appropriate amount of times
for(calcsetidx=0; calcsetidx < numcalcsets; calcsetidx++)
 {
 starttimecalc = time(NULL);  //  start time for calculation set
 // loop through the appropriate space in the score array for
 // storing newly calculated relative difference score
 for(compindex=0; compindex < maxarraysz; compindex++)
  {
  // loop through descriptions being compared
  for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
   {
   // loop through entity x
   for (entityindex=0; entityindex < numentities - 1; entityindex++)
    {
    // loop through entity y
    for (entityindex2=entityindex+1; entityindex2 < numentities; entityindex2++)
     {
     // calculate relative difference score for each combinatorial
     // descriptor comparison
     descriptscore = \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex] - \
      (float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) / \
      (0.5 * \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex]) + \
      fabs((float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) + \
      descriptscore;
     }  //  end entity y
    }  //  end entity x

   // calculate and store single measure relative difference score
   singlemeasureRD = descriptscore / divisor;
   scorebuffer.reldif[descriptindex] = singlemeasureRD;
   singlemeasureRD = 0.0;
   // keep running total for total measure relative difference score
```

```
 totalmeasureRD = (descriptscore / divisor) + totalmeasureRD;
 descriptscore = 0.0;
 } // end description loop

// calculate and store total measure relative difference score
// this is the similarity score
scorebuffer.reldif[numdescripts] = totalmeasureRD / numdescripts;
totalmeasureRD = 0.0;

// concatenate and store the label of the comparison set aka comparison group
(char *)scorebuffer.label = strcpy(scorebuffer.label, nullstring);
for (entityidxbuff = (entityindex - (numentities - 1)); \
 entityidxbuff <= entityindex; entityidxbuff++)
 {
 (char *)scorebuffer.label = \
  strcat(scorebuffer.label, \
  conformation[entityidxbuff][(int)conformationidx[entityidxbuff]]->labels);
 if (entityidxbuff < entityindex)
  {
  strcat(scorebuffer.label, "_");
  }
 }

// heart of keeping the indexes pointing to the right place in the
// conformation data - loop through conformations of last entity
// until the last conformation of the last entity is reached. Then
// increment the second to the last entity to the next conformation
// and decrement the last entity to its first conformation. Following
// this through all of the entities will cause every conformation to
// be compared with every conformation of the other entities.
// This is some what analogous to a mechanical odometer (or counter)
(int)conformationidx[argc-2] = (int)conformationidx[argc-2] + 1;

if((int)conformationidx[argc-2] >= (int)numconfs[argc-2])
 {
 for (confindex = (argc-2); confindex > 0; confindex--)
  {
  if((int)conformationidx[confindex] >= (int)numconfs[confindex])
   {
   (int)conformationidx[confindex] = 0;
   (int)conformationidx[confindex-1] = (int)conformationidx[confindex-1] + 1;
   }
  }
 }

// putting a label with the scores
score[compindex]->label = strcpy(score[compindex]->label, scorebuffer.label);
for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
 {
 score[compindex]->reldif[descriptindex] = scorebuffer.reldif[descriptindex];
 }
} // end of numcomps computational group loop

endtimecalc = time(NULL); // end time for set of comparison claculations
```

```
totaltimecalc = endtimecalc - starttimecalc;  // total time for calculations
printf("\n\n%i comparisons have been completed", maxarraysz * (calcsetidx+1));

starttimeqsort = time(NULL);  // start time for quicksort

// quicksort
Qsort(score, 0, maxarraysz-1, numdescripts, totlabel + numentities);
endtimeqsort = time(NULL);  // end time for quicksort
totaltimeqsort = endtimeqsort - starttimeqsort;  // total time for quicksort
printf("\n%i comparisons of been calculated in %.0f seconds and sorted in %.0f seconds", \
 maxarraysz, totaltimecalc, totaltimeqsort);


starttimewrite = time(NULL);  // start time for first temp file write

// the results of the first set of comparisons are written to tempfile_1
if (calcsetidx == 0)
 {
 if ((tmpfile1fd = fopen(tmpfile1, "wt")) == NULL)
  {
  printf("\ncan not open %s\n", tmpfile2);
  exit(2);
  }
 fflush(NULL);
 for(compindex=0; compindex < maxarraysz; compindex++)
  {
  fprintf(tmpfile1fd, "%s\n", score[compindex]->label);
  for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
   {
   fprintf(tmpfile1fd, "%.12f\n", score[compindex]->reldif[descriptindex]);
   }
  }
 tmpfileflag = 2;
 fclose(tmpfile1fd);

 endtimewrite = time(NULL);  // end time for first tempfile_1 write
 totaltimewrite = endtimewrite - starttimewrite;  // total time for write
 printf("\n%i lines written to tempfile_1 in %.0f seconds", maxarraysz, totaltimewrite);

 }

// after the first set of comparisons temp files are read from and written to
else
 {
 starttimeoocsort = time(NULL);  // start time for out of core sort
 starttimeread = time(NULL);  // start time for out of core sort
 fflush(NULL);


// if temp file flag == 1 write to tempfile_1 and read from tempfile_2
  if (tmpfileflag == 1)
   {
   // open tempfile_1 for writing
   if ((tmpfile1fd = fopen(tmpfile1, "wt")) == NULL)
```

```
 {
 printf("\ncan not open %s\n", tmpfile1);
 exit(2);
 }
//open tempfile_2 for reading
if (((FILE *)tmpfile2fdp = (fopen(tmpfile2, "r"))) < 0)
 {
 perror(tmpfile2);
 exit(EXIT_FAILURE);
 }

// initialize variables and rewind tempfile_2
bufferidx = 0;
rewind ((FILE *)tmpfile2fdp);
bufferidx = 0;

// read maxfinsz lines from tempfile_2 into filein array
for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
 {
 reldifidx = 0;
 fflush(NULL);
 *readbuffer = '\0 ;
 *filein[fileinidx]->label = '\0 ;
 fscanf((FILE *)tmpfile2fdp, "%s", filein[fileinidx]->label);
 for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
  {
  fscanf((FILE *)tmpfile2fdp, "%s", strnumber);
  doublenumber = strtod(strnumber, NULL);
  filein[fileinidx]->reldif[reldifidx] = doublenumber;
  }
 }
if (fgetpos ((FILE *)tmpfile2fdp, (fpos_t *)tmpfile2pos) != 0)
 {
 perror("fgetpos error");
 }

endtimeread = time(NULL);  // end file read time
totaltimereada = endtimeread - starttimeread; // total time for initial file read
printf("\n%i lines read from tempfile_2 in %.0f seconds", maxfinsz, totaltimereada);

fileinidx = 0;
//  read, merge, and write until all scores calculated to this point
for(compindex=0; compindex <= calcsetidx; compindex++)
 {
 starttimemerge = time(NULL);
 for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)
  {
  // if the calculated scores have all been put into the
  // fileout array, then write the data left in the
  // filein array into the fileout array
  if (bufferidx >= maxarraysz)
   {
   fileout[fileoutidx]->label = filein[fileinidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
```

```
  {
  fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
  }
 if (fileinidx < maxfinsz-1)
  {
  fileinidx++;
  }
 }
// if the data in the filein array has been written to the file out array
// then read more data from the temp file into the filein array
else if (fileinidx >= maxfinsz)
 {
 // if there is still new data to be read from the temporary file
 // read it into the filein array. if the compindex is less than
 // the calcsetidx then this is true
 if (compindex < calcsetidx)
  {
  starttimeread = time(NULL);  // start time for reading from tempfile
  if (fsetpos ((FILE *)tmpfile2fdp, (fpos_t *)tmpfile2pos) != 0)
   {
   perror ("fsetpos error");
   }
  for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
   {
   reldifidx = 0;
   fflush(NULL);
   *readbuffer = '\0 ;
   *filein[fileinidx]->label = '\0 ;

   fscanf((FILE *)tmpfile2fdp, "%s", filein[fileinidx]->label);
   for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
    {
    fscanf((FILE *)tmpfile2fdp, "%s", strnumber);
    doublenumber = strtod(strnumber, NULL);
    filein[fileinidx]->reldif[reldifidx] = doublenumber;
    }
   }
  if (fgetpos ((FILE *)tmpfile2fdp, (fpos_t *)tmpfile2pos) != 0)
   {
   perror("fgetpos error");
   }
  fileinidx = 0;
  fileoutidx--;

  endtimeread = time(NULL); // end time for reading
  totaltimereadb = endtimeread - starttimeread;  // total read time
  printf("\n%i lines read from tempfile_2 in %.0f seconds", maxfinsz, totaltimereadb);
  }
 // if all of the data has been read from the temporary file then
 // put what is left of the calculated data into the fileout array
 else
  {
  fileout[fileoutidx]->label = score[bufferidx]->label;
  for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
```

```
      {
      fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
      }
      bufferidx = bufferidx + 1;
     }
    }
   // if score is less than filein then write it to fileout
   else if (score[bufferidx]->reldif[numdescripts] < filein[fileinidx]->reldif[numdescripts])
    {
    // printf("\n less than");
    fileout[fileoutidx]->label = score[bufferidx]->label;
    for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
     {
     fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
     }
     bufferidx = bufferidx + 1;
    }
   // if score is greater the filein then write filein to fileout
   else if (filein[fileinidx]->reldif[numdescripts] <= score[bufferidx]->reldif[numdescripts])
    {
    // printf("\n greater than or equal");
    fileout[fileoutidx]->label = filein[fileinidx]->label;
    for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
     {
     fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
     }
     fileinidx++;
    }
   }
  endtimemerge = time(NULL); // end merge time
  totaltimemerge = endtimemerge - starttimemerge; // total merge time
  if (totaltimereadb > 0)
   {
   // if read occurred in middle of merge then subtract read time from
   // merge time to produce time for just the merging activity
   totaltimemerge = totaltimemerge - totaltimereadb;
   totaltimereadb = 0;
   }
  printf("\nmerge sorting %i scores complete in %.0f seconds", maxfoutsz, totaltimemerge);

  starttimewrite = time(NULL); // start time for temp file write
  for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++) // write to temp file
   {
   fprintf(tmpfile1fd, "%s\n", fileout[fileoutidx]->label);
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fprintf(tmpfile1fd, "%.12f\n", fileout[fileoutidx]->reldif[descriptindex]);
    }
   }
  endtimewrite = time(NULL); // end time for temp file write
  totaltimewrite = endtimewrite - starttimewrite; // total writing time
  printf("\n%i lines written to tempfile_1 in %.0f seconds", maxfoutsz, totaltimewrite);
  }
 tmpfileflag = 2;
```

```
fclose(tmpfile1fd); // close tempfile_1
fclose(tmpfile2fd); // close tempfile_2

endtimeoocsort = time(NULL); // end out of core sort time
totaltimeoocsort = endtimeoocsort - starttimeoocsort; // total ooc sort tome
printf("\n%i similarity scores sorted out of core in %.0f seconds", \
 maxarraysz * (calcsetidx+1), totaltimeoocsort);
}


// if temp file flag == 2 write to tempfile_2 and read from tempfile_1
  else if (tmpfileflag == 2)
  {
  // open tempfile_2 for writing
  if ((tmpfile2fd = fopen(tmpfile2, "wt")) == NULL)
   {
   printf("\ncan not open %s\n", tmpfile2);
   exit(2);
   }
  // open tempfile_1 for reading
  if (((FILE *)tmpfile1fdp = (fopen(tmpfile1, "r"))) < 0)
   {
   perror(tmpfile1);
   exit(EXIT_FAILURE);
   }

  // initialize variables and rewind tempfile_2
  rewind ((FILE *)tmpfile1fdp);
  bufferidx = 0;

  // read maxfinsz lines from tempfile_1 into filein array
  for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
   {
   reldifidx = 0;
   fflush(NULL);
   *readbuffer = '\0' ;
   *filein[fileinidx]->label = '\0' ;
   fscanf((FILE *)tmpfile1fdp, "%s", filein[fileinidx]->label);
   for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
    {
    fscanf((FILE *)tmpfile1fdp, "%s", strnumber);
    doublenumber = strtod(strnumber, NULL);
    filein[fileinidx]->reldif[reldifidx] = doublenumber;
    }
   }
  if (fgetpos ((FILE *)tmpfile1fdp, (fpos_t *)tmpfile1pos) != 0)
   {
   perror("fgetpos error");
   }

  endtimeread = time(NULL); // end file read time
  totaltimereada = endtimeread - starttimeread; // total time for initial file read
  printf("\n%i lines read from tempfile_1 in %.0f seconds", maxfinsz, totaltimereada);
```

```
fileinidx = 0;
// read, merge, and write until all scores calculated to this point
for(compindex=0; compindex <= calcsetidx; compindex++)
 {
 starttimemerge = time(NULL);
 for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)
  {
  // if the calculated scores have all been put into the
  // fileout array, then write the data left in the
  // filein array into the fileout array
  if (bufferidx >= maxarraysz)
   {
   fileout[fileoutidx]->label = filein[fileinidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
    }
   if (fileinidx < maxfinsz-1)
    {
    fileinidx++;
    }
   }
  // if the data in the filein array has been written to the file out array
  // then read more data from the temp file into the filein array
  else if (fileinidx >= maxfinsz)
   {
   // if there is still new data to be read from the temporary file
   // read it into the filein array. if the compindex is less than
   // the calcsetidx then this is true
   if (compindex < calcsetidx)
    {
    starttimeread = time(NULL); // start time for reading from tempfile
    if (fsetpos ((FILE *)tmpfile1fdp, (fpos_t *)tmpfile1pos) != 0)
     {
     perror ("fsetpos error");
     }
    for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
     {
     reldifidx = 0;
     fflush(NULL);
     *readbuffer = '\0 ;
     *filein[fileinidx]->label = '\0 ;

     fscanf((FILE *)tmpfile1fdp, "%s", filein[fileinidx]->label);
     for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
      {
      fscanf((FILE *)tmpfile1fdp, "%s", strnumber);
      doublenumber = strtod(strnumber, NULL);
      filein[fileinidx]->reldif[reldifidx] = doublenumber;
      }
     }
    if (fgetpos ((FILE *)tmpfile1fdp, (fpos_t *)tmpfile1pos) != 0)
     {
     perror("fgetpos error");
```

```
    }
   fileinidx = 0;
   fileoutidx--;
   endtimeread = time(NULL); // end time for reading
   totaltimereadb = endtimeread - starttimeread;  // total read time
   printf("\n%i lines read from tempfile_1 in %.0f seconds", maxfinsz, totaltimereadb);
   }
  // if all of the data has been read from the temporary file then
  // put what is left of the calculated data into the fileout array
  else
   {
   fileout[fileoutidx]->label = score[bufferidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
    }
   bufferidx = bufferidx + 1;
   }
  }
 // if score is less than filein then write it to fileout
 else if (score[bufferidx]->reldif[numdescripts] < filein[fileinidx]->reldif[numdescripts])
  {
  // printf("\n less than");
  fileout[fileoutidx]->label = score[bufferidx]->label;
  for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
   {
   fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
   }
   bufferidx = bufferidx + 1;
  }
 // if score is greater the filein then write filein to fileout
 else if (filein[fileinidx]->reldif[numdescripts] <= score[bufferidx]->reldif[numdescripts])
  {
  // printf("\n greater than or equal");
  fileout[fileoutidx]->label = filein[fileinidx]->label;
  for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
   {
   fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
   }
   fileinidx++;
  }
 }
endtimemerge = time(NULL);  //  end merge time
totaltimemerge = endtimemerge - starttimemerge;  // total merge time
if (totaltimereadb > 0)
 {
 // if read occurred in middle of merge then subtract read time from
 // merge time to produce time for just the merging activity
 totaltimemerge = totaltimemerge - totaltimereadb;
 totaltimereadb = 0;
 }
printf("\nmerge sorting %i scores complete in %.0f seconds", maxfoutsz, totaltimemerge);

starttimewrite = time(NULL);  // start time for temp file write
```

```
      for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)  // write to temp file
       {
       fprintf(tmpfile2fd, "%s\n", fileout[fileoutidx]->label);
       for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
        {
        fprintf(tmpfile2fd, "%.12f\n", fileout[fileoutidx]->reldif[descriptindex]);
        }
       }
      endtimewrite = time(NULL);  // end time for temp file write
      totaltime = endtimewrite - starttimewrite;  // total writing time
      printf("\n%i lines written to tempfile_2 in %.0f seconds", maxfoutsz, totaltime);
      }
     tmpfileflag = 1;
     fclose(tmpfile2fd);  // close tempfile_2
     fclose(tmpfile1fd);  // close tempfile_1

     endtimeoocsort = time(NULL);  //  end out of core sort time
     totaltimeoocsort = endtimeoocsort - starttimeoocsort;  // total ooc sort tome
     printf("\n%i similarity scores sorted out of core in %.0f seconds", \
      maxarraysz * (calcsetidx+1), totaltimeoocsort);
     }
    }

   //  reset the combinatorial calculation array
   compstart = compstart + maxarraysz;
   for(compindex2=0; compindex2 < maxarraysz; compindex2++)
    {
    for(descriptindex2=0; descriptindex2 < numdescripts + 1; descriptindex2++)
     {
     score[compindex2]->reldif[descriptindex2] = 0.0;
     }
    }
   //  printf("\narray reset");
   }


  endtimetotal = time(NULL);  // end time before last array
  totaltime = endtimetotal - starttimetotal;  // total time before last array
  printf("\n\ntotal time before last array %.0f seconds", totaltime);



 // start of last array
  printf("\nstarting last set of %i calculations", lastarraysz);

  starttimecalc = time(NULL);  // start time for last array calculation time

  // loop through the appropriate space in the score array for
  // storing newly calculated relative difference score
  for(compindex=0; compindex < numcomps-compstart; compindex++)
   {
   // loop through descriptions being compared
   for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
    {
```

```
// loop through entity x
for (entityindex=0; entityindex < numentities - 1; entityindex++)
 {
 // loop through entity y
 for (entityindex2=entityindex+1; entityindex2 < numentities; entityindex2++)
  {
  // calculate relative difference score for each combinatorial
  // descriptor comparison
  descriptscore = \
   (fabs((float)conformation[entityindex]\
   [(int)conformationidx[entityindex]]->descriptivedata[descriptindex] - \
   (float)conformation[entityindex2]\
   [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) / \
   (0.5 * \
   (fabs((float)conformation[entityindex]\
   [(int)conformationidx[entityindex]]->descriptivedata[descriptindex]) + \
   fabs((float)conformation[entityindex2]\
   [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex]))) + \
   descriptscore;
  } // end entity y
 } // end entity x

// calculate and store single measure relative difference score
singlemeasureRD = descriptscore / divisor;
scorebuffer.reldif[descriptindex] = singlemeasureRD;
singlemeasureRD = 0.0;

// keep running total for total measure relative difference score
totalmeasureRD = (descriptscore / divisor) + totalmeasureRD;
descriptscore = 0.0;
} // end of description loop

// calculate and store total measure relative difference score
// this is the similarity score
scorebuffer.reldif[numdescripts] = totalmeasureRD / numdescripts;
totalmeasureRD = 0.0;

// concatenate and store the label of the comparison set aka comparison group
(char *)scorebuffer.label = strcpy(scorebuffer.label, nullstring);
for (entityidxbuff = (entityindex - (numentities - 1)); \
entityidxbuff <= entityindex; entityidxbuff++)
 {
 (char *)scorebuffer.label = \
  strcat(scorebuffer.label, \
  conformation[entityidxbuff][(int)conformationidx[entityidxbuff]]->labels);
 if (entityidxbuff < entityindex)
  {
  strcat(scorebuffer.label, "_");
  }
 }


// heart of keeping the indexes pointing to the right place in the
// conformation data - loop through conformations of last entity
```

```
// until the last conformation of the last entity is reached. Then
// increment the second to the last entity to the next conformation
// and decrement the last entity to its first conformation. Following
// this through all of the entities will cause every conformation to
// be compared with every conformation of the other entities.
// This is some what analogous to a mechanical odometer (or counter)
(int)conformationidx[argc-2] = (int)conformationidx[argc-2] + 1;
if((int)conformationidx[argc-2] >= (int)numconfs[argc-2])
 {
 for (confindex = (argc-2); confindex > 0; confindex--)
  {
  if((int)conformationidx[confindex] >= (int)numconfs[confindex])
   {
   (int)conformationidx[confindex] = 0;
   (int)conformationidx[confindex-1] = (int)conformationidx[confindex-1] + 1;
   }
  }
 }

// putting a label with the scores
score[compindex]->label = strcpy(score[compindex]->label, scorebuffer.label);
for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
 {
 score[compindex]->reldif[descriptindex] = scorebuffer.reldif[descriptindex];
 }

} // end of compindex computational group loop

endtimecalc = time(NULL);  // end time for final set of calculations
totaltimecalc = endtimecalc - starttimecalc;  // total time final set of calcs

// all combinatorial combination of conformations have been compared at this point
printf("\n\n%i comparisons have been completed", numcomps);

starttimeqsort = time(NULL);  // start time for quicksort

//  quicksort
Qsort(score, 0, numcomps-compstart-1, numdescripts, totlabel + numentities);
endtimeqsort = time(NULL);  // end time for quicksort
totaltimeqsort = endtimeqsort - starttimeqsort;  //  total time for quicksort
printf("\n%i comparisons of been calculated in %.0f seconds and sorted in %.0f seconds", \
 lastarraysz, totaltimecalc, totaltimeqsort);

maxscoreidx = numcomps - compstart;  // redefining maxscoreidx to be representative of
                        // the smaller last array size

starttimeoocsort = time(NULL);  //  start time for out of core sort
starttimeread = time(NULL);  //  start time for out of core sort
fflush(NULL);


//  if temp file flag == 1 write to outfile and read from tempfile_2
  if (tmpfileflag == 1)
   {
```

```
// open outfile for writing
if ((outfilefd = fopen(outfilename, "wt")) == NULL)
 {
 printf("\ncan not open %s\n", outfilename);
 exit(2);
 }
//open tempfile_2 for reading
if (((FILE *)tmpfile2fdp = (fopen(tmpfile2, "r"))) < 0)
 {
 perror(tmpfile2);
 exit(EXIT_FAILURE);
 }

// initialize variables and rewind tempfile_2
bufferidx = 0;
reldifidx = 0;
*readbuffer = '\0 ;
rewind ((FILE *)tmpfile2fdp);
bufferidx = 0;

// read maxfinsz lines from tempfile_2 into filein array
for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
 {
 reldifidx = 0;
 fflush(NULL);
 *readbuffer = '\0 ;
 *filein[fileinidx]->label = '\0 ;

 fscanf((FILE *)tmpfile2fdp, "%s", filein[fileinidx]->label);
 for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
  {
  fscanf((FILE *)tmpfile2fdp, "%s", strnumber);
  doublenumber = strtod(strnumber, NULL);
  filein[fileinidx]->reldif[reldifidx] = doublenumber;
  }
 }
if (fgetpos ((FILE *)tmpfile2fdp, (fpos_t *)tmpfile2pos) != 0)
 {
 perror("fgetpos error");
 }

endtimeread = time(NULL);  // end file read time
totaltimereada = endtimeread - starttimeread; // total time for initial file read
printf("\n%i lines read from tempfile_2 in %.0f seconds", maxfinsz, totaltimereada);


fileinidx = 0;
// read, merge, and write until all scores calculated to this point
for(compindex=0; compindex <= calcsetidx; compindex++)
 {
 starttimemerge = time(NULL);
 for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)
  {
  // if the calculated scores have all been put into the
```

```
// fileout array, then write the data left in the
// filein array into the fileout array
if (bufferidx >= maxscoreidx)
 {
 fileout[fileoutidx]->label = filein[fileinidx]->label;
 for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
  {
  fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
  }
 if (fileinidx < maxfinsz-1)
  {
  fileinidx++;
  }
 }
// if the data in the filein array has been written to the file out array
// then read more data from the temp file into the filein array
else if (fileinidx >= maxfinsz)
 {
 // if there is still new data to be read from the temporary file
 // read it into the filein array. if the compindex is less than
 // the calcsetidx then this is true
 if (compindex < calcsetidx)
  {
  starttimeread = time(NULL);  // start time for reading from tempfile
  if (fsetpos ((FILE *)tmpfile2fdp, (fpos_t *)tmpfile2pos) != 0)
   {
   perror ("fsetpos error");
   }
  for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
   {
   reldifidx = 0;
   fflush(NULL);
   *readbuffer = '\0 ;
   *filein[fileinidx]->label = '\0 ;

   fscanf((FILE *)tmpfile2fdp, "%s", filein[fileinidx]->label);
   for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
    {
    fscanf((FILE *)tmpfile2fdp, "%s", strnumber);
    doublenumber = strtod(strnumber, NULL);
    filein[fileinidx]->reldif[reldifidx] = doublenumber;
    }
   }
  if (fgetpos ((FILE *)tmpfile2fdp, (fpos_t *)tmpfile2pos) != 0)
   {
   perror("fgetpos error");
   }

  fileinidx = 0;
  fileoutidx--;

  endtimeread = time(NULL); // end time for reading
  totaltimereadb = endtimeread - starttimeread;  // total read time
  printf("\n%i lines read from tempfile_2 in %.0f seconds", maxfinsz, totaltimereadb);
```

```
    }
   // if all of the data has been read from the temporary file then
   // put what is left of the calculated data into the fileout array
   else
    {
    fileout[fileoutidx]->label = score[bufferidx]->label;
    for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
     {
     fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
     }
    bufferidx = bufferidx + 1;
    }
   }
  // if score is less than filein then write it to fileout
  else if (score[bufferidx]->reldif[numdescripts] < filein[fileinidx]->reldif[numdescripts])
   {
   // printf("\n less than");
   fileout[fileoutidx]->label = score[bufferidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
    }
    bufferidx = bufferidx + 1;
   }
  // if score is greater the filein then write filein to fileout
  else if (filein[fileinidx]->reldif[numdescripts] <= score[bufferidx]->reldif[numdescripts])
   {
   // printf("\n greater than or equal");
   fileout[fileoutidx]->label = filein[fileinidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
    }
    fileinidx++;
   }
  }
 endtimemerge = time(NULL);  // end merge time
 totaltimemerge = endtimemerge - starttimemerge;  // total merge time
 if (totaltimereadb > 0)
  {
  // if read occurred in middle of merge then subtract read time from
  // merge time to produce time for just the merging activity
  totaltimemerge = totaltimemerge - totaltimereadb;
  totaltimereadb = 0;
  }
 if (compindex < calcsetidx)  // maxfoutsz merge time not last merge cycle
  {
  printf("\nmerge sorting %i scores complete in %.0f seconds", maxfoutsz, totaltimemerge);
  }
 if (compindex == calcsetidx)  // lastarraysz merge time last merge cycle
  {
  printf("\nmerge sorting %i scores complete in %.0f seconds", lastarraysz, totaltimemerge);
  }
```

```
    starttimewrite = time(NULL);  // start time for temp file write
    if (compindex < calcsetidx)  // write all fileout if not last write cycle
     {
     for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)  // write to temp file loop
      {
      fprintf(outfilefd, "%s,", fileout[fileoutidx]->label);
      for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
       {
       fprintf(outfilefd, "%.12f,", fileout[fileoutidx]->reldif[descriptindex]);
       }
      fprintf(outfilefd, "%.12f\n", fileout[fileoutidx]->reldif[numdescripts]);
      }
     endtimewrite = time(NULL);  // end time for temp file write
     totaltime = endtimewrite - starttimewrite;  // total writing time
     printf("\n%i lines written to output file in %.0f seconds", maxfoutsz, totaltime);
     }
    else if (compindex == calcsetidx)  // write part of fileout on last write cycle
     {
     for(fileoutidx=0; fileoutidx < lastarraysz; fileoutidx++)  // write to temp file loop
      {
      fprintf(outfilefd, "%s,", fileout[fileoutidx]->label);
      for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
       {
       fprintf(outfilefd, "%.12f,", fileout[fileoutidx]->reldif[descriptindex]);
       }
      fprintf(outfilefd, "%.12f\n", fileout[fileoutidx]->reldif[numdescripts]);
      }
     endtimewrite = time(NULL);  // end time for temp file write
     totaltime = endtimewrite - starttimewrite;  // total writing time
     printf("\n%i lines written to output file in %.0f seconds", lastarraysz, totaltime);
     }
    }
   fclose(outfilefd);  // close outfile
   fclose(tmpfile2fd);  // close tempfile_2

   endtimeoocsort = time(NULL);  // end out of core sort time
   totaltimeoocsort = endtimeoocsort - starttimeoocsort;  // total ooc sort tome
   printf("\n%i similarity scores sorted out of core in %.0f seconds", \
    lastarraysz, totaltimeoocsort);
   }


// if temp file flag == 2 write to outfile and read from tempfile_1
  else if (tmpfileflag == 2)
   {
   // open outfile for writing
   if ((outfilefd = fopen(outfilename, "wt")) == NULL)
    {
    printf("\ncan not open %s\n", outfilename);
    exit(2);
    }
   // open tempfile_1 for reading
   if (((FILE *)tmpfile1fdp = (fopen(tmpfile1, "r"))) < 0)
    {
```

```
 perror(tmpfile1);
 exit(EXIT_FAILURE);
 }

// initialize variables and rewind tempfile_1
bufferidx = 0;
reldifidx = 0;
*readbuffer = '\0 ;
rewind ((FILE *)tmpfile1fdp);
bufferidx = 0;

// read maxfinsz lines from tempfile_1 into filein array
for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
 {
 reldifidx = 0;
 fflush(NULL);
 *readbuffer = '\0 ;
 *filein[fileinidx]->label = '\0 ;

 fscanf((FILE *)tmpfile1fdp, "%s", filein[fileinidx]->label);
 for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
  {
  fscanf((FILE *)tmpfile1fdp, "%s", strnumber);
  doublenumber = strtod(strnumber, NULL);
  filein[fileinidx]->reldif[reldifidx] = doublenumber;
  }
 }
if (fgetpos ((FILE *)tmpfile1fdp, (fpos_t *)tmpfile1pos) != 0)
 {
 perror("fgetpos error");
 }

endtimeread = time(NULL);  // end file read time
totaltimereada = endtimeread - starttimeread; // total time for initial file read
printf("\n%i lines read from tempfile_1 in %.0f seconds", maxfinsz, totaltimereada);

fileinidx = 0;
//  read, merge, and write until all scores calculated to this point
for(compindex=0; compindex <= calcsetidx; compindex++)
 {
 starttimemerge = time(NULL);
 for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)
  {
  // if the calculated scores have all been put into the
  // fileout array, then write the data left in the
  // filein array into the fileout array
  if (bufferidx >= maxscoreidx)
   {
   fileout[fileoutidx]->label = filein[fileinidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
    }
   if (fileinidx < maxfinsz-1)
```

```
  {
  fileinidx++;
  }
 }
// if the data in the filein array has been written to the file out array
// then read more data from the temp file into the filein array
else if (fileinidx >= maxfinsz)
 {
 // if there is still new data to be read from the temporary file
 // read it into the filein array. if the compindex is less than
 // the calcsetidx then this is true
 if (compindex < calcsetidx)
  {
  starttimeread = time(NULL);  // start time for reading from tempfile
  if (fsetpos ((FILE *)tmpfile1fdp, (fpos_t *)tmpfile1pos) != 0)
   {
   perror ("fsetpos error");
   }
  for(fileinidx = 0; fileinidx < maxfinsz; fileinidx++)
   {
   reldifidx = 0;
   fflush(NULL);
   *readbuffer = '\0 ;
   *filein[fileinidx]->label = '\0 ;

   fscanf((FILE *)tmpfile1fdp, "%s", filein[fileinidx]->label);
   for(reldifidx=0; reldifidx <= numdescripts; reldifidx++)
    {
    fscanf((FILE *)tmpfile1fdp, "%s", strnumber);
    doublenumber = strtod(strnumber, NULL);
    filein[fileinidx]->reldif[reldifidx] = doublenumber;
    }
   }
  if (fgetpos ((FILE *)tmpfile1fdp, (fpos_t *)tmpfile1pos) != 0)
   {
   perror("fgetpos error");
   }

  fileinidx = 0;
  fileoutidx--;
  endtimeread = time(NULL); // end time for reading
  totaltimereadb = endtimeread - starttimeread;  // total read time
  printf("\n%i lines read from tempfile_1 in %.0f seconds", maxfinsz, totaltimereadb);
  }
 // if all of the data has been read from the temporary file then
 // put what is left of the calculated data into the fileout array
 else
  {
  fileout[fileoutidx]->label = score[bufferidx]->label;
  for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
   {
   fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
   }
  bufferidx = bufferidx + 1;
```

```
    }
   }
  // if score is less than filein then write it to fileout
  else if (score[bufferidx]->reldif[numdescripts] < filein[fileinidx]->reldif[numdescripts])
   {
   // printf("\n less than");
   fileout[fileoutidx]->label = score[bufferidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = score[bufferidx]->reldif[descriptindex];
    }
    bufferidx = bufferidx + 1;
   }
  // if score is greater the filein then write filein to fileout
  else if (filein[fileinidx]->reldif[numdescripts] <= score[bufferidx]->reldif[numdescripts])
   {
   // printf("\n greater than or equal");
   fileout[fileoutidx]->label = filein[fileinidx]->label;
   for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
    {
    fileout[fileoutidx]->reldif[descriptindex] = filein[fileinidx]->reldif[descriptindex];
    }
    fileinidx++;
   }
  }
endtimemerge = time(NULL);  // end merge time
totaltimemerge = endtimemerge - starttimemerge;  // total merge time
if (totaltimereadb > 0)
 {
 // if read occurred in middle of merge then subtract read time from
 // merge time to produce time for just the merging activity
 totaltimemerge = totaltimemerge - totaltimereadb;
 totaltimereadb = 0;
 }
if (compindex < calcsetidx)  // maxfoutsz merge time not last merge cycle
 {
 printf("\nmerge sorting %i scores complete in %.0f seconds", maxfoutsz, totaltimemerge);
 }
if (compindex == calcsetidx)  // lastarraysz merge time last merge cycle
 {
 printf("\nmerge sorting %i scores complete in %.0f seconds", lastarraysz, totaltimemerge);
 }

starttimewrite = time(NULL);  // start time for temp file write
if (compindex < calcsetidx)  // write all fileout if not last write cycle
 {
 for(fileoutidx=0; fileoutidx < maxfoutsz; fileoutidx++)  // write to temp file loop
  {
  fprintf(outfilefd, "%s,", fileout[fileoutidx]->label);
  for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
   {
   fprintf(outfilefd, "%.12f,", fileout[fileoutidx]->reldif[descriptindex]);
   }
  fprintf(outfilefd, "%.12f\n", fileout[fileoutidx]->reldif[numdescripts]);
```

```
        }
       endtimewrite = time(NULL);  // end time for temp file write
       totaltime = endtimewrite - starttimewrite;  // total writing time
       printf("\n%i lines written to output file in %.0f seconds", maxfoutsz, totaltime);
       }
      else if (compindex == calcsetidx)  // write part of fileout on last write cycle
       {
       for(fileoutidx=0; fileoutidx < lastarraysz; fileoutidx++)  // write to temp file loop
        {
        fprintf(outfilefd, "%s,", fileout[fileoutidx]->label);
        for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
         {
         fprintf(outfilefd, "%.12f,", fileout[fileoutidx]->reldif[descriptindex]);
         }
        fprintf(outfilefd, "%.12f\n", fileout[fileoutidx]->reldif[numdescripts]);
        }
       endtimewrite = time(NULL);  // end time for temp file write
       totaltime = endtimewrite - starttimewrite;  // total writing time
       printf("\n%i lines written to output file in %.0f seconds", lastarraysz, totaltime);
       }
      }
     fclose(outfilefd);  // close outfile
     fclose(tmpfile1fd);  // close tempfile_2

     endtimeoocsort = time(NULL);  // end out of core sort time
     totaltimeoocsort = endtimeoocsort - starttimeoocsort;  // total ooc sort tome
     printf("\n%i similarity scores sorted out of core in %.0f seconds", \
     lastarraysz, totaltimeoocsort);
     }


// if tmpfileflag == 3 then the number of calculations are less than the size
// of the score array - all of the out of core stuff is skipped and this
// section writes out the scores to the outfiles - outfile is opened for writing
   else if(tmpfileflag == 3)
     {
     printf("\ntemp files not used\n");
     // open outfile for writing
     if ((outfilefd = fopen(outfilename, "wt")) == NULL)
      {
      printf("\ncan not open %s\n", tmpfile2);
      exit(2);
      }
     // loop writes all calculations to outfile
     for(compindex=0; compindex < maxfoutsz; compindex++)
      {
      fprintf(outfilefd, "%s,", score[compindex]->label);
      for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
       {
       fprintf(outfilefd, "%.12f,", score[compindex]->reldif[descriptindex]);
       }
      fprintf(outfilefd, "%.12f\n", score[compindex]->reldif[numdescripts]);
      }
     fclose(outfilefd);  // close outfile
```

```
    }

  endtimetotal = time(NULL);  // end total time
  // total run time minus user input part of the program and time to free memory
  totaltime = endtimetotal - starttimetotal;  // total run time
  printf("\n\ntotal time %.0f seconds\n\n", totaltime);


// freeing memory
  printf("\nstarting to free memory\n");
  for(entityindex=0; entityindex < numentities; entityindex++)
    {
    for(confindex=0; confindex < (int)numconfs[entityindex]; confindex++)
      {
      free(conformation[entityindex][confindex]->labels);
      free(conformation[entityindex][confindex]->descriptivedata);
      free(conformation[entityindex][confindex]);
      }
    free(conformation[entityindex]);
    }
  free(conformation);

  for(compindex=0; compindex < maxarraysz; compindex++)
    {
    free(score[compindex]->label);
    free(score[compindex]->reldif);
    free(score[compindex]);
    }
  free(score);

  free(scorebuffer.reldif);
  free(scorebuffer.label);

 printf("\nmemory free\n\n");

 return(0);
 }
```

Appendix 4.6 The compsort program, written in C. Calculates and sorts all similarity scores. Saves a user determined number of best, most similar, scores.

```
/*

program: compsort
version: 1.01
author: Paul A. Wilson
email: pwilson@cobremail.itrc.umt.edu
date of latest version: August 11, 2004
date of origination: 2003

command: program name data file 1 data file 2 data file 3
example: ./compsort ./MCN-5652.txt ./sertraline.txt ./indatraline.txt ./s-citalopram.txt

minimum of three data files required, no limit to maximum number of data
files to be compared - within reason

required data file format: comma delimited text, first field of each row is
equivalent to a label, the remaining fields in each row are numbers, each
field is separated by a comma, no quotes around the labels, no spaces between
fields (commas only)

output file name is limited to a maximum of 64 characters

In Short: This program compares the rows in data files developing a score
for each set of rows (set - one row from each data file). Then sorts the
scores from low to high. Low scores represent the sets of rows which are
most similar. A text file is written containing X number of the lowest
scoring sets of rows. The number of rows X is determine by the user.
Each row in the output file contains the label from each label field
(first field in each data file) concatenated together followed in comma
delimited form by the score for each field and finally the total score.

This program was written in response to to a project where distance
space descriptions needed to be compared in order to determine the
conformations of four molecules which were the most similar.
(see SFN 2003 Annual Meeting, Poster Presentation 371.4 and SFN
2004 Annual Meeting, Poster Presentation 922.1, also see Machack 18,
Wilson, P.A. "A Practical Comparison of Multiprocessing Libraries",
MacHack 2003, June 2003, the code is different but a lot of the ideas
remain) I am willing to share manuscript versions of these posters in
pdf format.

Comparisons are carried out combinatorially using relative difference.
The relative difference equation was modified slightly to enable creating
and weighting the comparison of a positive and negative measurement as less
similar. Each row in each data file is compared against each row of every
other data file. Within each row, each field is compared against the
corresponding field from the rows being compared in the other data files.
If there are 4 data files than there are 6 comparisons between each field
of the four rows being compared. The relative difference score from each
field is added together and divided by the number of fields to produce a
similarity score for the for the rows being compared.
```

similarity score = (summation from measure 1 to total number of measures (
summation of from data file 1 to total number of data files ((|Vab - Vacl /
((|Vabl + |Vacl)/2)) / number of combinatorial)))
Vab = measure a of datafile b
Vac = measure a of datafile c

The program programs structure has a legacy stemming from a personal interest in out-of-core sorting, dynamic memory allocation, and parallel computing. The program was originally written to produce an output file containing all scores in sorted order from low to high (copies of this program can be made available, the algorithm was interesting, fast (except the file I/O step)). Due to the out-of-core mechanism I used, though fast, was not fast enough to be useful for large data sets. It would be interesting at some point to look into this again, using a parallel file system (or at least a high speed file system). The version here is a compromise which calculates a set number of scores, sorts the scores and retains a set number of the lowest scores, calculate the next set number of scores, sorts the new score with the saved scores and again only retains a set number of the lowest scores.

Pointers are used extensively as port of some of my original notions on dynamically allocating memory and sorting.

The program was written in C to provide maximum portability. I have not tried compiling this code on a Linux, or any other, machine. I have noticed this code only runs under Mac OS 10.3 and failed when running under Mac OS 10.2.

*/

```
//#include <dirent.h>
#include <fcntl.h>
#include <math.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/errno.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/uio.h>

// structure for holding data read from files, label and numerical descriptors
struct descriptions
 {
 char * labels;
 double * descriptivedata;
 };

// structure for holding concatenated label and score for each descriptor
```

```
// and a total score value
struct labelscores
 {
 char * label;
 double * reldif;
 };


// from here to main are a set of function used for sorting
// the sorting algorithm is a quicksort, with a mean of 3 used for determining
// the first pivot point - see Data Structures and Algorithm
// Analysis in C by Mark Allen Weiss


// swap redirects pointers - u will point to v and v will point to u
// Swap - needs to be compiled inline for efficiency

void Swap(struct labelscores ** u, struct labelscores ** v)
 {
 struct labelscores temp;
 temp = **u;
 **u = **v;
 **v = temp;
 }


// insertion sort - used as part of quicksort
// scoreisort - struct of labelscores containing scores to be sorted by the insertion sort method
// n is the upper edge of of the chunk data the insertion sort is occurring on
// numdecriptsis - number of description insertion sort,
//   is equal to the number of descriptive fields in the data files
//   used for for malloc of temporary insertion sort struct of labelscores
// totallabelelngthis - total label length insertion sort,
//   is equal to the maximum number of characters in concatenated labels
//   used for for malloc of temporary insertion sort struct of labelscores

void InsertionSort(struct labelscores ** scoreisort, int n, int numdescriptsis, int totlablengthis)
{
 int j, p;   // j and p are indexes used in the insertion sort
 int  descriptindexis;   // an index for descriptive field
 char * nullstring = "\0";   // nullstring needed at the end of strings

 struct labelscores tempisort;   // temporary labelscore struct used in insertion sort

 tempisort.label = (char *) malloc(sizeof(char) * (totlablengthis));  // allocate memory
 tempisort.reldif = (double *) malloc((numdescriptsis + 1) *sizeof(double));

 // initialize tempisort
 (char *)tempisort.label = strcpy(tempisort.label, nullstring);
 for(descriptindexis=0; descriptindexis < numdescriptsis+1; descriptindexis++)
  {
  tempisort.reldif[descriptindexis] = 0.0;
  }
```

```
// insertion - copy one struct out, move the rest, then put the struct back were it goes
for(p=1; p<n; p++)
  {
  tempisort.label = strcpy(tempisort.label, scoreisort[p]->label);
  for(descriptindexis=0; descriptindexis <= numdescriptsis; descriptindexis++)
    {
    tempisort.reldif[descriptindexis] = scoreisort[p]->reldif[descriptindexis];
    }
  for(j=p; ((j > 0) && (scoreisort[j-1]->reldif[numdescriptsis] > tempisort.reldif[numdescriptsis])); j--)
    {
    scoreisort[j]->label = strcpy(scoreisort[j]->label, scoreisort[j-1]->label);
    for(descriptindexis=0; descriptindexis <= numdescriptsis; descriptindexis++)
      {
      scoreisort[j]->reldif[descriptindexis] = scoreisort[j-1]->reldif[descriptindexis];
      }
    }
    scoreisort[j]->label = strcpy(scoreisort[j]->label, tempisort.label);
    for(descriptindexis=0; descriptindexis <= numdescriptsis; descriptindexis++)
      {
      scoreisort[j]->reldif[descriptindexis] = tempisort.reldif[descriptindexis];
      }
  }

  free(tempisort.label);   // free memory previously malloced
  free(tempisort.reldif);
}


// median of three  - selecting the pivot for the quicksort
//
//  scoremedian - struct of labelscores, postion of left(first) score, postion of right(last) score
//  numdescriptsm - number of descripts median of 3, the last descript field - the total similarity score
//  the median value of the three sampled values is determined and return to the quicksort routine
//  to be used as the pivot point

float Median3(struct labelscores **scoremedian, int left, int right, int numdescriptsm)
{
 int center;
 center = (left + right ) / 2;   // position of the center score - median position

 if (scoremedian[left]->reldif[numdescriptsm] > scoremedian[center]->reldif[numdescriptsm])
   {
   Swap(&scoremedian[left], &scoremedian[center]);
   }
 if (scoremedian[left]->reldif[numdescriptsm] > scoremedian[right]->reldif[numdescriptsm])
   {
   Swap(&scoremedian[left], &scoremedian[right]);
   }
 if (scoremedian[center]->reldif[numdescriptsm] > scoremedian[right]->reldif[numdescriptsm])
   {
   Swap(&scoremedian[center], &scoremedian[right]);
   }

 Swap(&scoremedian[center], &scoremedian[right-1]);
```

```
 return scoremedian[right-1]->reldif[numdescriptsm];
}


// quicksort
// scoreqsort - the struct of labelscores to be sorted by quicksort
// leftqs, rightqs - the lower and upper boundaries of the quicksort - quicksort is recursive
// numdescriptsqs - number of descriptive fields quicksort - used for allocating necessary memory
// totlablength - total label length quick sort - maximum number of characters in a label
//   used for allocating necessary memory

void Qsort(struct labelscores **scoreqsort, int leftqs, int rightqs, int numdescriptsqs, int totlablengthqs)
{
 int i, j;   // indexes used in quicksort

 struct labelscores pivot;   // labelscores struct which hold the pivot

 pivot.label = (char *) malloc(sizeof(char) * (totlablengthqs));   // allocating memory for pivot
 pivot.reldif = (double *) malloc((numdescriptsqs+1) *sizeof(double));

 // heart of quicksort, pick pivot swap appropriate values to either side of pivot
 if ((leftqs + 3) <= rightqs)
  {
  pivot.reldif[numdescriptsqs] = Median3(scoreqsort, leftqs, rightqs, numdescriptsqs);
  i = leftqs;
  j = rightqs-1;
  for( ; ; )
   {
   while(scoreqsort[++i]->reldif[numdescriptsqs] < pivot.reldif[numdescriptsqs] && i < rightqs-1){}
   while(scoreqsort[--j]->reldif[numdescriptsqs] > pivot.reldif[numdescriptsqs] && j > 0){}
   if(i < j)
    Swap(&scoreqsort[i], &scoreqsort[j]);
   else
    break;
   }
  Swap(&scoreqsort[i], &scoreqsort[rightqs-1]);

  Qsort(scoreqsort, leftqs, i-1, numdescriptsqs, totlablengthqs);
  Qsort(scoreqsort, i+1, rightqs, numdescriptsqs, totlablengthqs);
  }
 else   // if left and right a close together do an insertion sort
  {
  InsertionSort(scoreqsort+leftqs,  rightqs-leftqs+1, numdescriptsqs, totlablengthqs);
  }

 free(pivot.label);   // freeing allocated memory
 free(pivot.reldif);
}


// start of main

int main (int argc, const char * argv[])
 {
```

```
int calcsetidx;   // index for which set of calculation (comparisons) is current
int compindex;   // comparison index - which comparison set is being compared
int compstart;   // comparison start - the starting point for each set of comparisons
int confindex;   // conformation index
int descriptindex;   // desciptor index - data file columns
int entityidxbuff;   // entity index buffer
int entityindex;   // id of entity one in the comparison
int entityindex2;   // id of entity two in the comparison
int fileindex=1;   // index used to indicate which data file is open
int i;   // simple index used in loop calculating the divisor
int lastarraysz;   // the size of the last array, the remainder of calculations
int lseekbuffer;   // buffer for file position
int lseekposition;   // file position in data file being read
int maxarraysz;   // maximum array size = numlinescalc + numbestscore
int numbestscore;   // the number of best scores to keep after sorting
int numcalcsets;   // the number of sets of calculations
int numcomps;   // total number of comparison sets
int numdescripts;  // total number of descriptions = number of columns in each data file
int numentities;   // the total number of entities to be compared = number of data files
int numlinescalc;   // the number of scores to calculate before sorting
int totlabel;   // total length, number of characters, of concatenated label

int * conformationidx[argc-1];   // index of conformations
int * filedescript[argc-1];   // array of file descriptors - data files from command line
int * labellength[argc-1];   // array of the label lengths for each data file
int * maxlabellength[argc-1];   // array of the maximum label lengths for each data file
int * numcolumns;   // number of columns
int * numconfs[argc-1];   // array of the number of rows in each data file

char * nullstring = "\0";   // nullstring for adding the end of lines and initialization
char * readbuffer;   // read buffer for reading data files
char * strnumber;   // numerical data read from file stored as string

char outfilename[64];   // character array for output file name - max 65 character name

double descriptscore;   // the relative difference score for one combinatorial
double divisor;   // the divisor used in calculating the single measure relative difference
double doublenumber;   // numerical string read from data file is converted to double number
double singlemeasureRD;   // the Relative Difference score for a single set of description
double totalmeasureRD;   // the Relative Difference score for a comparison set
double totaltime;   // the total time in seconds used for timing run time
double totaltimecalc;   // the total time in seconds used for timing calculation time
double totaltimesort;   // the total time in seconds used for timing sorts  time

FILE *outfilefd;   // the output file descriptor

time_t starttimeintcalc;   // marks start time of combinatorial calculations
time_t endtimeintcalc;   // marks end time of combinatorial calculations
time_t starttimeintsort;   // marks start time of sort
time_t endtimeintsort;   // marks end time of sort
time_t starttimeinttot;   // marks start time before calculations being
time_t endtimeinttot;   // marks end time after output file is written
```

```
struct descriptions ***conformation;  // struct which stores all the data from the data files

struct labelscores **score;    // struct which stores concatenated labels and scores
struct labelscores scorebuffer;    // a buffer to temporarily store scores in


//  some initial memory allocation and variable initialization
readbuffer = malloc(sizeof(char));
strnumber = malloc(sizeof(char) * 16);

numcolumns = (int *) malloc(sizeof(int));
*filedescript = (int *) malloc(sizeof(int) * (argc - 1));

numentities = argc - 1;
confindex = 0;
numcomps = 1;
totlabel = 1;

// print information to the screen which will help the user answer the
// first two questions they are asked by the program
printf("\n\n\nThis program is going to calculate X number of scores, ");
printf("\nsort those scores, save the best Y scores and then ");
printf("\ncalculates the next X number of scores.\n");

printf("\nThe answers given to the next two question will determine ");
printf("\nhow much memory is allocated.  Remember there are limits ");
printf("\nto how much memory can be allocated to a single application ");
printf("\nand virtual memory is slower than physical memory.  ");
printf("\nYou are about to be asked for the number of scores to be ");
printf("\nsaved and the the number of scores to calculate between sorts ");
printf("\nThese two number added together will account for a majority ");
printf("\nof the memory used by this program.  Calculating only X many ");
printf("\nscores at a time allows for the program top remain within a ");
printf("\nlimited memory footprint.  Values that have proven useful to ");
printf ("\nthe author are:  X = 1000000 and Y = 200000.  These values ");
printf("\nshould be changed according to your data set, needs, and ");
printf("\nphysical memory.\n");

// ask for and obtain the number of conformational sets to calculate
// between the "sort and store" steps
printf("\nHow many scores do you want ");
printf("\nto calculate between sorts? ");
scanf("%i", &numlinescalc);

// ask for and obtain the number of lowest scores to keep
printf("\nHow many of the lowest (best) ");
printf("\nscores would you like to save? ");
scanf("%i", &numbestscore);

maxarraysz = numbestscore + numlinescalc;  // maximum size of array of structs

// allocating memory for struct conformation
conformation = (struct descriptions ***) malloc (sizeof(struct \
  descriptions **) * numentities);
```

```
printf("\n\nThe number of data files = %i\n", numentities);

// asking for the number of columns per data file
printf("\n\nWhat is the total number of columns in each file? ");
scanf("%i", (int *)&numcolumns);
printf("\nThere are %i columns.\n", (int)numcolumns);

numdescripts = (int)numcolumns - 1;
printf("\nEach file has 1 lable column and %i description columns.", \
 numdescripts);


// allocating memory and loading the data arrays

for (entityindex=0; entityindex < numentities; entityindex++)
 {
 maxlabellength[entityindex] = (int *)malloc(sizeof(int));
 *maxlabellength[entityindex] = 0;
 if (((int)filedescript[fileindex] = (open(argv[fileindex], O_RDONLY))) < 0)
  {
  perror(argv[fileindex]);
  exit(EXIT_FAILURE);
  }

 // asking for the number of rows in the current data file being read
 printf("\n\n\nfile %s is open", argv[fileindex]);
 printf("\nHow many rows are in this file? ");
 scanf("%i", (int *)&numconfs[entityindex]);

 printf("\nloading %i rows of data from file", (int)numconfs[entityindex]);

// allocating memory for the array of pointers to conformations
 conformation[entityindex] = (struct descriptions **) malloc \
 (( sizeof(struct descriptions *) * (int)numconfs[entityindex]));

 lseekposition = 0;

// allocating memory for a conformation s structure and descriptors
 for(confindex=0; confindex < (int)numconfs[entityindex]; confindex++)
  {
  conformation[entityindex][confindex] = (struct descriptions *) malloc \
   (( sizeof(struct descriptions)));
  conformation[entityindex][confindex]->descriptivedata = (double *) \
   malloc(sizeof(double) * numdescripts);

//  allocating memory for a conformation s structure label
  labellength[entityindex] = (int *)malloc(sizeof(int));
  fflush(NULL);
  *readbuffer = '\0 ;
  *labellength[entityindex] = 0;
  lseekbuffer=lseekposition;

  // counting number of characters in label
```

```
// fscan would be faster and was used for file I/O in the
// out-of-core sortng program
while(*readbuffer != ' ' && *readbuffer != ', )
 {
 lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
 read((int)filedescript[fileindex], readbuffer, 1);
 *labellength[entityindex] = *labellength[entityindex] + 1;
 lseekposition++;
 }

if (*maxlabellength[entityindex] < *labellength[entityindex])
 {
 *maxlabellength[entityindex] = *labellength[entityindex];
//   printf("\nmaximum label length = %i\n", *maxlabellength[entityindex]);
 }

// allocating memory for label - depends on label maximum length
conformation[entityindex][confindex]->labels = \
(char *) malloc(sizeof(char) * *labellength[entityindex]);

// reading, concatenating, and storing label
fflush(NULL);
*readbuffer = '\0 ;
lseekposition = lseekbuffer;
descriptindex = 0;
while(*readbuffer != ' ' && *readbuffer != ', )
 {
 lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
 read((int)filedescript[fileindex], readbuffer, 1);
 if(*readbuffer != ', )
  {
  (char *)conformation[entityindex][confindex]->labels = \
  strncat( conformation[entityindex][confindex]->labels, readbuffer, 1);
  }
 lseekposition++;
 }

// resetting and reading numerical descriptor data from data files
// data read as string and converted to double - strtod
while(*readbuffer != '\n  && *readbuffer != '\0 )
 {
 fflush(NULL);
 *readbuffer = \0 ;
 *strnumber = '\0 ;
 while(*readbuffer != ',  && *readbuffer != ' ' && *readbuffer != '\n )
  {
  lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
  read((int)filedescript[fileindex], readbuffer, 1);
  if (*readbuffer != ',  && *readbuffer != ' ' && *readbuffer != '\n )
   {
   strnumber = strncat(strnumber, readbuffer, 1);
   lseekposition++;
   }
  }
```

```
    doublenumber = strtod(strnumber, NULL);
    conformation[entityindex][confindex]->descriptivedata[descriptindex]=\
    doublenumber;
    descriptindex++;
    lseekposition++;
    }
   }
 fileindex = fileindex + 1;  // finished reading from data file move to next file
 }

// asking for and obtaining name of output file
fflush(NULL);
printf("\n\n\nEnter file name for results to be stored in: ");
scanf("%s", (char *)&outfilename);
printf("\nThe results will be saved in %s\n", outfilename);


starttimeinttot = time(NULL);  // start time for total time


// calculating total number of comparisons and maximimum total label length
for (entityindex=0; entityindex < numentities; entityindex++)
 {
 numcomps = numcomps * (int)numconfs[entityindex];
 totlabel = totlabel + *maxlabellength[entityindex];
 }

numcalcsets = (numcomps/numlinescalc);  // number of sets of calculations

// if number of comparisons is less than the size of a set of calculations
if (numlinescalc > numcomps)
 {
 maxarraysz = numbestscore + numcomps;
 numcalcsets = 0;
 compstart = numcomps;
 }

// calculate and print the size of the last set of comparisons
lastarraysz = numcomps - (numcalcsets * numlinescalc);
printf("\n\n%i sets of %i calculations and", numcalcsets, numlinescalc);
printf("\none last set of %i calculations", lastarraysz);
printf("\nwill be done");

// allocate memory for score array and initialize
score = (struct labelscores **) malloc (sizeof(struct labelscores *) \
 * (int)maxarraysz);

for(compindex=0; compindex < numbestscore; compindex++)
 {
 score[compindex] = (struct labelscores *) malloc(sizeof(struct\
  labelscores));
 score[compindex]->label = (char *) malloc(sizeof(char) * (totlabel + numentities));
 score[compindex]->reldif = (double *) malloc((numdescripts+1) * sizeof(double));
 for(descriptindex=0; descriptindex < numdescripts+1; descriptindex++)
```

```
  {
  // initializing with a big number helps guarantee low scores move into this
  // part of the array where the lowest scores are kept
  score[compindex]->reldif[descriptindex] = 1000000.0;
  }
 }
for(compindex=numbestscore; compindex <= maxarraysz; compindex++)
 {
 score[compindex] = (struct labelscores *) malloc(sizeof(struct\
  labelscores));
 score[compindex]->label = (char *) malloc(sizeof(char) * (totlabel + numentities));
 score[compindex]->reldif = (double *) malloc((numdescripts+1) * sizeof(double));
 for(descriptindex=0; descriptindex < numdescripts+1; descriptindex++)
  {
  score[compindex]->reldif[descriptindex] = 0.0;
  }
 }

// allocating memory for scorebuffer
scorebuffer.label = (char *) malloc(sizeof(char) * (totlabel + numentities));
scorebuffer.reldif = (double *) malloc((numdescripts+1) *sizeof(double));


// memory allocation for main complete
printf("\n\n\nmalloc complete - memory allocated");

// calculate and print the number of combinatorial comparisons per each set of
// desriptors in a comparison set
divisor = 3;
for(i=3; i<numentities; i++)
 {
 divisor = divisor + i;
 }
printf("\n\n\nThere are %.0f possible comparisons for each descriptive field", divisor);
printf("\nin each comparison set\n\n");


// initializing variables before relative difference calcualtion
entityindex = 1;  // first entity
entityindex2 = 0;  // second entity
descriptscore = 0.0;
descriptindex = 1;  // description being compared
singlemeasureRD = 0.0;
totalmeasureRD = 0.0;
compstart = 0;
(int)conformationidx[entityindex] = 0;  // conformation of entity one
(int)conformationidx[entityindex2] = 0;  // conformation of entity two

for(entityindex=0; entityindex < numentities; entityindex++)
 {
 (int)conformationidx[entityindex] = 0;
 }

// loop through number of calculation sets appropriate amount of times
```

```
for(calcsetidx=0; calcsetidx < numcalcsets; calcsetidx++)
 {
 starttimeintcalc = time(NULL);  // start time for calculation set
 lseekposition = 0;
 // loop through the appropriate space in the score array for
 // storing newly calculated relative difference score
 for(compindex=numbestscore; compindex < maxarraysz; compindex++)
  {
  // loop through descriptions being compared
  for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
   {
   // loop through entity x
   for (entityindex=0; entityindex < numentities - 1; entityindex++)
    {
    // loop through entity y
    for (entityindex2=entityindex+1; entityindex2 < numentities; entityindex2++)
     {
     // calculate relative difference score for each combinatorial
     // descriptor comparison
     descriptscore = \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex] - \
      (float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) / \
      (0.5 * \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex]) + \
      fabs((float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex]))) + \
      descriptscore;
     }  // end entity y
    }  // end entity x

   // calculate and store single measure relative difference score
   singlemeasureRD = descriptscore / divisor;
   scorebuffer.reldif[descriptindex] = singlemeasureRD;
   singlemeasureRD = 0.0;

   // keep running total for total measure relative difference score
   totalmeasureRD = (descriptscore / divisor) + totalmeasureRD;
   descriptscore = 0.0;
   }  // end description loop

  // calculate and store total measure relative difference score
  // this is the similarity score
  scorebuffer.reldif[numdescripts] = totalmeasureRD / numdescripts;
  totalmeasureRD = 0.0;

  // concatenate and store the label of the comparison set aka comparison group
  (char *)scorebuffer.label = strcpy(scorebuffer.label, nullstring);
  for (entityidxbuff = (entityindex - (numentities - 1)); \
  entityidxbuff <= entityindex; entityidxbuff++)
   {
   (char *)scorebuffer.label = \
```

```
  strcat(scorebuffer.label, \
  conformation[entityidxbuff][(int)conformationidx[entityidxbuff]]->labels);
  if (entityidxbuff < entityindex)
   {
   strcat(scorebuffer.label, "_");
   }
 }


// heart of keeping the indexes pointing to the right place in the
// conformation data - loop through conformations of last entity
// until the last conformation of the last entity is reached. Then
// increment the second to the last entity to the next conformation
// and decrement the last entity to its first conformation. Following
// this through all of the entities will cause every conformation to
// be compared with every conformation of the other entities.
// This is some what analogous to a mechanical odometer (or counter)
(int)conformationidx[argc-2] = (int)conformationidx[argc-2] + 1;

if((int)conformationidx[argc-2] >= (int)numconfs[argc-2])
 {
 for (confindex = (argc-2); confindex > 0; confindex--)
  {
  if((int)conformationidx[confindex] >= (int)numconfs[confindex])
   {
   (int)conformationidx[confindex] = 0;
   (int)conformationidx[confindex-1] = (int)conformationidx[confindex-1] + 1;
   }
  }
 }

// putting a label with the scores
score[compindex]->label = strcpy(score[compindex]->label, scorebuffer.label);
for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
 {
 score[compindex]->reldif[descriptindex] = scorebuffer.reldif[descriptindex];
 }
} // end of numcomps computational group loop

// tracking the correct starting place in the data after each calculation set
compstart = compstart + (maxarraysz-numbestscore);


printf("\n%i total comparisons have been completed", compstart);

endtimeintcalc = time(NULL); // end calculation time
totaltimecalc = difftime(endtimeintcalc, starttimeintcalc); // calculation time
starttimeintsort = time(NULL); // start time for sort

// call quick sort - sort all of the low scores to the beginning of the array
Qsort(score, 0, maxarraysz-1, numdescripts, totlabel + numentities);

// end time of sort, subtract from start time and print total time of sort
endtimeintsort = time(NULL); // end time for sort
```

```
  totaltimesort = difftime(endtimeintsort, starttimeintsort);  // sort time
  printf("\n\ncalculation time = %.0f seconds, sort time = %.0f seconds\n\n", \
   totaltimecalc, totaltimesort);
  }  //  end of calcset calculation set loop


// start of last array
  printf("\nstarting last set of %i calculations", lastarraysz);

  starttimeintcalc = time(NULL);  // start time for last array calculation time

   // loop through the appropriate space in the score array for
   // storing newly calculated relative difference score
   for(compindex=numbestscore; compindex < numcomps-compstart+numbestscore; compindex++)
    {
    // loop through descriptions being compared
    for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
     {
     // loop through entity x
     for (entityindex=0; entityindex < numentities - 1; entityindex++)
      {
     // loop through entity y
      for (entityindex2=entityindex+1; entityindex2 < numentities; entityindex2++)
       {
       // calculate relative difference score for each combinatorial
       // descriptor comparison
       descriptscore = \
        (fabs((float)conformation[entityindex]\
        [(int)conformationidx[entityindex]]->descriptivedata[descriptindex] - \
        (float)conformation[entityindex2]\
        [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) / \
        (0.5 * \
        (fabs((float)conformation[entityindex]\
        [(int)conformationidx[entityindex]]->descriptivedata[descriptindex]) + \
        fabs((float)conformation[entityindex2]\
        [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex]))) + \
        descriptscore;
       }  //  end entity 2
      }  //  end entity

     // calculate and store single measure relative difference score
     singlemeasureRD = descriptscore / divisor;
     scorebuffer.reldif[descriptindex] = singlemeasureRD;
     singlemeasureRD = 0.0;

     // keep running total for total measure relative difference score
     totalmeasureRD = (descriptscore / divisor) + totalmeasureRD;
     descriptscore = 0.0;
     }  // end of description loop

    // calculate and store total measure relative difference score
    // this is the similarity score
    scorebuffer.reldif[numdescripts] = totalmeasureRD / numdescripts;
```

```
totalmeasureRD = 0.0;

// concatenate and store the label of the comparison set aka comparison group
(char *)scorebuffer.label = strcpy(scorebuffer.label, nullstring);
for (entityidxbuff = (entityindex - (numentities - 1)); \
entityidxbuff <= entityindex; entityidxbuff++)
 {
 (char *)scorebuffer.label = \
  strcat(scorebuffer.label, \
  conformation[entityidxbuff][(int)conformationidx[entityidxbuff]]->labels);
 if (entityidxbuff < entityindex)
  {
  strcat(scorebuffer.label, "_");
  }
 }


// heart of keeping the indexes pointing to the right place in the
// conformation data - loop through conformations of last entity
// until the last conformation of the last entity is reached. Then
// increment the second to the last entity to the next conformation
// and decrement the last entity to its first conformation. Following
// this through all of the entities will cause every conformation to
// be compared with every conformation of the other entities.
// This is some what analogous to a mechanical odometer (or counter)
(int)conformationidx[argc-2] = (int)conformationidx[argc-2] + 1;
if((int)conformationidx[argc-2] >= (int)numconfs[argc-2])
 {
 for (confindex = (argc-2); confindex > 0; confindex--)
  {
  if((int)conformationidx[confindex] >= (int)numconfs[confindex])
   {
   (int)conformationidx[confindex] = 0;
   (int)conformationidx[confindex-1] = (int)conformationidx[confindex-1] + 1;
   }
  }
 }

// putting a label with the scores
score[compindex]->label = strcpy(score[compindex]->label, scorebuffer.label);
for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
 {
 score[compindex]->reldif[descriptindex] = scorebuffer.reldif[descriptindex];
 }

} // end of compindex computational group loop

// all combinatorial combination of conformations have been compared at this point
printf("\n\n%i total comparisons have been completed", compstart + lastarraysz);


endtimeintcalc = time(NULL);  // end last array calculation time
totaltimecalc = difftime(endtimeintcalc, starttimeintcalc);  // calculation time
starttimeintsort = time(NULL);  // start time for sort
```

```
// call quick sort - sort all of the low scores to the beginning of the array
Qsort(score, 0, numcomps-compstart-1+numbestscore, numdescripts, totlabel + numentities);

// end time of sort, subtract from start time and print total time of sort
endtimeintsort = time(NULL);  // end time for sort
totaltimesort = difftime(endtimeintsort, starttimeintsort);  // sort time
printf("\n\ncalculation time = %.0f seconds, sort time = %.0f seconds\n\n", \
 totaltimecalc, totaltimesort);


// writing lowest scores and corresponding labels to output file
printf("\n\n\nwriting %i lowest scores to file %s", numbestscore, outfilename);
 if ((outfilefd = fopen(outfilename, "wt")) == NULL)
   {
   printf("\ncan not open %s\n", outfilename);
   exit(2);
   }
 for(compindex=0; compindex < numbestscore; compindex++)
   {
   fprintf(outfilefd, "%s,", score[compindex]->label);
   for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
     {
     fprintf(outfilefd, "%.12f,", score[compindex]->reldif[descriptindex]);
     }
   fprintf(outfilefd, "%.12f\n", score[compindex]->reldif[numdescripts]);
   }
 fclose(outfilefd);


// end time for all calculation, sorting, and writing output file
endtimeinttot = time(NULL);
totaltime = difftime(endtimeinttot, starttimeinttot);
printf("\n\n\ntotal time to calculate, sort and write output file = %.0f seconds\n\n", totaltime);


// freeing memory

printf("\nstarting to free memory\n");
for(entityindex=0; entityindex < numentities; entityindex++)
 {
 for(confindex=0; confindex < (int)numconfs[entityindex]; confindex++)
   {
   free(conformation[entityindex][confindex]->labels);
   free(conformation[entityindex][confindex]->descriptivedata);
   free(conformation[entityindex][confindex]);
   }
 free(conformation[entityindex]);
 }
free(conformation);

for(compindex=0; compindex < maxarraysz; compindex++)
 {
 free(score[compindex]->label);
```

```
    free(score[compindex]->reldif);
    free(score[compindex]);
    }
  free(score);

  free(scorebuffer.reldif);
  free(scorebuffer.label);

printf("\nmemory free\n");

return(0);
}
```

Appendix 4.7 The comp program, written in C. Calculates and saves all similarity scores.

```
/*

Program: comp
author: Paul A. Wilson
email: pwilson@cobremail.itrc.umt.edu
date of latest version: August 15, 2004
date of origination: 2003

command: program name data file 1 data file 2 data file 3
example: ./comp ./MCN-5652.txt ./sertraline.txt ./indatraline.txt ./s-citalopram.txt

minimum of three data files required, no limit to maximum number of data
files to be compared - within reason

required data file format: comma delimited text, first field of each row is
equivalent to a label, the remaining fields in each row are numbers, each
field is separated by a comma, no quotes around the labels, no spaces between
fields (commas only)

output file name is limited to a maximum of 64 characters

In Short: This program compares the rows in data files developing a score
for each set of rows (set - one row from each data file). A text file is
written containing all comparisons. Each row in the output file contains
the label from each label field (first field in each data file)
concatenated together followed in comma delimited form by the score for
each field and finally the total score. This version of the program does
not sort the similarity scores. See compsort and compsortall for score
sorting in ascending order.

This program was written in response to to a project where distance
space descriptions needed to be compared in order to determine the
conformations of four molecules which were the most similar.
(see SFN 2003 Annual Meeting, Poster Presentation 371.4 and SFN
2004 Annual Meeting, Poster Presentation 922.1, also see Machack 18,
Wilson, P.A. "A Practical Comparison of Multiprocessing Libraries",
MacHack 2003, June 2003, the code is different but a lot of the ideas
remain) I am willing to share manuscript versions of these posters in
pdf format.

Comparisons are carried out combinatorially using relative difference.
The relative difference equation was modified slightly to enable creating
and weighting the comparison of a positive and negative measurement as less
similar. Each row in each data file is compared against each row of every
other data file. Within each row, each field is compared against the
corresponding field from the rows being compared in the other data files.
If there are 4 data files than there are 6 comparisons between each field
of the four rows being compared. The relative difference score from each
field is added together and divided by the number of fields to produce a
similarity score for the for the rows being compared.
```

similarity score = (summation from measure 1 to total number of measures (
summation of from data file 1 to total number of data files ((|Vab - Vac| /
((|Vab| + |Vac|)/2)) / number of combinatorial)))
Vab = measure a of datafile b
Vac = measure a of datafile c

The program programs structure has a legacy stemming from a personal interest in out-of-core sorting, dynamic memory allocation, and parallel computing. There are three versions of this program. This one does not sort similarity scores. The other two versions of the program 1) sorts all scores using an out of core sorting mechanism, and 2) one sorts all but only keeps a user determined portion of the scores.

Pointers are used extensively as port of some of my original notions on dynamically allocating memory and sorting.

The program was written in C to provide maximum portability. I have not tried compiling this code on a Linux, or any other, machine. I have noticed this code runs great under Mac OS 10.3 and failed when running under Mac OS 10.2.

*/

```
//#include <dirent.h>
#include <fcntl.h>
#include <math.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/errno.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/uio.h>

// start of main
int main (int argc, const char * argv[])
  {

// structure for holding data read from files, label and numerical descriptors
  struct descriptions
    {
    char * labels;
    double * descriptivedata;
    };

// structure for holding concatenated label and score for each descriptor
// and a total score value
  struct labelscores
    {
    char * label;
```

```
double * reldif;
};

int calcsetidx;   // index for which set of calculation (comparisons) is current
int confindex;   // comparison index - which comparison set is being compared
int compindex;   // comparison index - which comparison set is being compared
int compindex2;   // index used for resetting score array
int compstart;   // comparison start - the starting point for each set of comparisons
int descriptindex;   // desciptor index - data file columns
int descriptindex2;   // index used for resetting score array
int entityindex;   // id of entity one in the comparison
int entityindex2;   // id of entity two in the comparison
int entityidxbuff;   // entity index buffer
int fileindex=1;   // index used to indicate which data file is open
int i;   // simple index used in loop calculating the divisor
int lastarraysz;   // the size of the last array, the remainder of calculations
int lseekposition;   // file position in data file being read
int lseekbuffer;   // buffer for file position
int maxarraysz;   // maximum array size = numlinescalc + numbestscore
int numcalcsets;   // the number of sets of calculations
int numcomps;   // total number of comparison sets
int numdescripts;  // total number of descriptions = number of columns in each data file
int numentities;   // the total number of entities to be compared = number of data files
int totlabel;   // total length, number of characters, of concatenated label

int * conformationidx[argc-1];   //  index of conformations
int * filedescript[argc-1];   // array of file descriptors - data files from command line
int * labellength[argc-1];   //  array of the label lengths for each data file
int * maxlabellength[argc-1];   //  array of the maximum label lengths for each data file
int * numcolumns;   // number of columns
int * numconfs[argc-1];   //  array of the number of rows in each data file

char * nullstring = "\0";   // nullstring for adding the end of lines and initialization
char * readbuffer;   // read buffer for reading data files
char * strnumber;   // numerical data read from file stored as string

char outfilename[64];   // character array for output file name - max 65 character name

double descriptscore;   // the relative difference score for one combinatorial
double divisor;   // the divisor used in calculating the single measure relative difference
double doublenumber;   // numerical string read from data file is converted to double number
double singlemeasureRD;   // the Relative Difference score for a single set of description
double totalmeasureRD;   //  the Relative Difference score for a comparison set
double totaltime;   // the total time in seconds used for timing sorts and run time

FILE *outfilefd;   //  the output file descriptor

time_t starttimecalc;  // start time for set of calculations
time_t endtimecalc;  // end time for set of calculations
time_t starttimetotal;  // start time for total run time
time_t endtimetotal;  // end time for total run timet
time_t starttimewrite;  // start time for a temporary file write
time_t endtimewrite;  // end time for a temporary file write
```

```
struct descriptions ***conformation;  // struct which stores all the data from the data files

struct labelscores **score;    // struct which stores concatenated labels and scores
struct labelscores scorebuffer;    // a buffer to temporarily store scores in

//  initial memory allocation and variable initialization
readbuffer = malloc(sizeof(char));
strnumber = malloc(sizeof(char) * 16); // this needs to be increased if numbers are
                                       // more than 15 digits plus 1 NULL string

numcolumns = (int *) malloc(sizeof(int));
*filedescript = (int *) malloc(sizeof(int) * (argc - 1));

numentities = argc - 1;
confindex = 0;
numcomps = 1;
totlabel = 1;

// print information to the screen which will help the user answer the
// first three questions they are asked by the program
printf("\n\n\nThis program is going to calculate X number of scores, ");
printf("\nwrite the results to the output file, and then calculate the ");
printf("\nsthe next X number of scores. This latest set of X scores are");
printf("\nappended to the end of the output file. This repeats until all ");
printf("\nscores are calculated and written to the output file.\n");

printf("\nThe answer given for the next question will determine ");
printf("\nhow much memory is allocated. Remember there are limits ");
printf("\nto how much memory can be allocated to a single application ");
printf("\nand virtual memory is slower than physical memory. ");
printf("\nYou are about to be asked for the number of scores to be ");
printf("\ncalculated between writes to the output file.\n");

printf("\nThe number of scores (X) calculated between writes to the output ");
printf("\nfile accounts for a majority of the memory used by this program.");
printf("\nCalculating X number of scores at a time allows the program ");
printf("\nallows for the program to remain within a limited memory");
printf("\nfootprint. An example value that has proven useful to the");
printf("\nauthor is:  X = 500000. This value should be changed according");
printf("\nto your data set, needs, and physical memory\n");

// ask for and obtain the number of conformational sets to calculate
// between writes to the output file
printf("\nHow many scores do you want to calculate before ");
printf("\nwriting to the output file?  ");
scanf("%i", &maxarraysz);

// allocating memory for input data array
conformation = (struct descriptions ***) malloc (sizeof(struct \
  descriptions **) * numentities);

// determined from arguments on comand line
printf("\nThe number of data files = %i\n", numentities);
```

```
// asking for the number of columns per data file
printf("\nWhat is the total number of columns in each file? ");
scanf("%i", (int *)&numcolumns);
printf("\nThere are %i columns.\n in each file", (int)numcolumns);

numdescripts = (int)numcolumns - 1;
printf("\nEach file has 1 lable column and %i description columns.\n", \
 numdescripts);

// allocating memory and loading the data arrays

for (entityindex=0; entityindex < numentities; entityindex++)
 {
 maxlabellength[entityindex] = (int *)malloc(sizeof(int));
 *maxlabellength[entityindex] = 0;
 if (((int)filedescript[fileindex] = (open(argv[fileindex], O_RDONLY))) < 0)
  {
  perror(argv[fileindex]);
  exit(EXIT_FAILURE);
  }
 // asking for the number of rows in the current data file being read
 printf("\n\nfile %s is open\n", argv[fileindex]);
 printf("How many rows are in this file? ");
 scanf("%i", (int *)&numconfs[entityindex]);
 printf("There are %i rows representing %i conformations in this file.\n", \
 (int)numconfs[entityindex], (int)numconfs[entityindex]);

 // allocating memory for the array of pointers to conformations
 conformation[entityindex] = (struct descriptions **) malloc \
 (( sizeof(struct descriptions *) * (int)numconfs[entityindex]));

 lseekposition = 0;

 // allocating memory for a conformation s structure and descriptors
 for(confindex=0; confindex < (int)numconfs[entityindex]; confindex++)
  {
  conformation[entityindex][confindex] = (struct descriptions *) malloc \
   (( sizeof(struct descriptions)));
  conformation[entityindex][confindex]->descriptivedata = (double *) \
   malloc(sizeof(double) * numdescripts);

  // allocating memory for a conformation s structure label
  labellength[entityindex] = (int *)malloc(sizeof(int));
  fflush(NULL);
  *readbuffer = '\0 ;
  *labellength[entityindex] = 0;
  lseekbuffer=lseekposition;

  // counting number of characters in label
  // fscan would be faster and was used for file I/O in the
  // out-of-core sortng program
  while(*readbuffer != ' ' && *readbuffer != ', )
   {
```

```
 lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
 read((int)filedescript[fileindex], readbuffer, 1);
 *labellength[entityindex] = *labellength[entityindex] + 1;
 lseekposition++;
 }

if (*maxlabellength[entityindex] < *labellength[entityindex])
 {
 *maxlabellength[entityindex] = *labellength[entityindex];
 // printf("\nmaximum label length = %i\n", *maxlabellength[entityindex]);
 }

// allocating memory for label - depends on label maximum length
conformation[entityindex][confindex]->labels = \
(char *) malloc(sizeof(char) * *labellength[entityindex]);

// reading, concatenating, and storing label
fflush(NULL);
*readbuffer = '\0 ;
lseekposition = lseekbuffer;
descriptindex = 0;
while(*readbuffer != ' ' && *readbuffer != ', )
 {
 lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
 read((int)filedescript[fileindex], readbuffer, 1);
 if(*readbuffer != ', )
  {
  (char *)conformation[entityindex][confindex]->labels = \
  strncat( conformation[entityindex][confindex]->labels, readbuffer, 1);
  }
 lseekposition++;
 }

// resetting and reading numerical descriptor data from data files
// data read as string and converted to double - strtod
while(*readbuffer != '\n  && *readbuffer != '\0 )
 {
 fflush(NULL);
 *readbuffer = \0 ;
 *strnumber = '\0 ;
 while(*readbuffer != ',  && *readbuffer != ' ' && *readbuffer != '\n )
  {
  lseek((int)filedescript[fileindex], lseekposition, SEEK_SET);
  read((int)filedescript[fileindex], readbuffer, 1);
  if (*readbuffer != ',  && *readbuffer != ' ' && *readbuffer != '\n )
   {
   strnumber = strncat(strnumber, readbuffer, 1);
   lseekposition++;
   }
  }
 doublenumber = strtod(strnumber, NULL);
 conformation[entityindex][confindex]->descriptivedata[descriptindex]=\
 doublenumber;
 descriptindex++;
```

```
    lseekposition++;
    }
   }
  fileindex = fileindex + 1;  // finished reading from data file move to next file
  }

// asking for and obtaining name of output file
fflush(NULL);
printf("\n\nEnter file name for results to be stored in: ");
scanf("%s", (char *)&outfilename);
printf("\nThe results will be saved in %s\n", outfilename);

starttimetotal = time(NULL);  // start time for total time

// calculating total number of comparisons and maximimum total label length
for (entityindex=0; entityindex < numentities; entityindex++)
  {
  numcomps = numcomps * (int)numconfs[entityindex];
  totlabel = totlabel + *maxlabellength[entityindex];
  }

numcalcsets = (numcomps/maxarraysz);

// if number of comparisons is less than the size of a set of calculations
if (maxarraysz > numcomps)
  {
  maxarraysz = numcomps;
  numcalcsets = 0;
  }

// calculate the size of the last set of comparisons
lastarraysz = numcomps - (numcalcsets * maxarraysz);

// allocate memory for score array and initialize
score = (struct labelscores **) malloc (sizeof(struct labelscores *) \
 * (int)maxarraysz);

for(compindex=0; compindex < maxarraysz; compindex++)
  {
  score[compindex] = (struct labelscores *) malloc(sizeof(struct\
   labelscores));
  score[compindex]->label = (char *) malloc(sizeof(char) * (totlabel + numentities));
  score[compindex]->reldif = (double *) malloc((numdescripts+1) * sizeof(double));
  for(descriptindex=0; descriptindex < numdescripts+1; descriptindex++)
    {
    score[compindex]->reldif[descriptindex] = 0.0;
    }
  }

// allocating memory for scorebuffer
scorebuffer.label = (char *) malloc(sizeof(char) * (totlabel + numentities));
scorebuffer.reldif = (double *) malloc((numdescripts+1) *sizeof(double));
```

```
// memory allocation for main complete
printf("\nmalloc complete - memory allocated\n");

// calculate and print the number of combinatorial comparisons per each set of
// desriptors in a comparison set
divisor = 3;
for(i=3; i<numentities; i++)
 {
 divisor = divisor + i;
 }
printf("\n\nThere are %.0f possible comparisons for each descriptive field", divisor);
printf("\nin each comparison set\n\n");


// initializing variables before relative difference calculation
entityindex = 1;  // first entity
entityindex2 = 0;  // second entity
descriptscore = 0.0;
descriptindex = 1;  // description being compared
singlemeasureRD = 0.0;
totalmeasureRD = 0.0;
compstart = 0;
(int)conformationidx[entityindex] = 0;  // conformation of entity one
(int)conformationidx[entityindex2] = 0;  // conformation of entity two

for(entityindex=0; entityindex < numentities; entityindex++)
 {
 (int)conformationidx[entityindex] = 0;
 }

// loop through number of calculation sets appropriate amount of times
for(calcsetidx=0; calcsetidx < numcalcsets; calcsetidx++)
 {
 starttimecalc = time(NULL);  //  start time for calculation set
 // loop through the appropriate space in the score array for
 // storing newly calculated relative difference score
 for(compindex=compstart; compindex < maxarraysz+compstart; compindex++)
  {
  // loop through descriptions being compared
  for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
   {
   // loop through entity x
   for (entityindex=0; entityindex < numentities - 1; entityindex++)
    {
    // loop through entity y
    for (entityindex2=entityindex+1; entityindex2 < numentities; entityindex2++)
     {
     // calculate relative difference score for each combinatorial
     // descriptor comparison
     descriptscore = \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex] - \
      (float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) / \
```

```
    (0.5 * \
    (fabs((float)conformation[entityindex]\
    [(int)conformationidx[entityindex]]->descriptivedata[descriptindex]) + \
    fabs((float)conformation[entityindex2]\
    [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) + \
    descriptscore;
   } // end entity 2
  } // end entity

 // calculate and store single measure relative difference score
 singlemeasureRD = descriptscore / divisor;
 scorebuffer.reldif[descriptindex] = singlemeasureRD;
 singlemeasureRD = 0.0;

 // keep running total for total measure relative difference score
 totalmeasureRD = (descriptscore / divisor) + totalmeasureRD;
 descriptscore = 0.0;
 } // end descriptindex

// calculate and store total measure relative difference score
// this is the similarity score
scorebuffer.reldif[numdescripts] = totalmeasureRD / numdescripts;
totalmeasureRD = 0.0;

// concatenate and store the label of the comparison set aka comparison group
(char *)scorebuffer.label = strcpy(scorebuffer.label, nullstring);
for (entityidxbuff = (entityindex - (numentities - 1)); \
entityidxbuff <= entityindex; entityidxbuff++)
 {
 (char *)scorebuffer.label = \
  strcat(scorebuffer.label, \
  conformation[entityidxbuff][(int)conformationidx[entityidxbuff]]->labels);
 if (entityidxbuff < entityindex)
  {
  strcat(scorebuffer.label, "_");
  }
 }

// heart of keeping the indexes pointing to the right place in the
// conformation data - loop through conformations of last entity
// until the last conformation of the last entity is reached. Then
// increment the second to the last entity to the next conformation
// and decrement the last entity to its first conformation. Following
// this through all of the entities will cause every conformation to
// be compared with every conformation of the other entities.
// This is some what analogous to a mechanical odometer (or counter)
(int)conformationidx[argc-2] = (int)conformationidx[argc-2] + 1;

if((int)conformationidx[argc-2] >= (int)numconfs[argc-2])
 {
 for (confindex = (argc-2); confindex > 0; confindex--)
  {
  if((int)conformationidx[confindex] >= (int)numconfs[confindex])
   {
```

```
      (int)conformationidx[confindex] = 0;
      (int)conformationidx[confindex-1] = (int)conformationidx[confindex-1] + 1;
      }
     }
    }

 // putting a label with the scores
 score[compindex-compstart]->label = strcpy(score[compindex-compstart]->label, scorebuffer.label);
 for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
  {
  score[compindex-compstart]->reldif[descriptindex] = scorebuffer.reldif[descriptindex];
  }
 } // end of numcomps computational group loop

endtimecalc = time(NULL); // end time for set of comparison claculations
totaltime = endtimecalc - starttimecalc; // total time for calculations
printf("\n\n%i total comparisons have been completed", maxarraysz * (calcsetidx+1));
printf("\n%i comparisons of been calculated in %.0f seconds", \
 maxarraysz, totaltime);

starttimewrite = time(NULL);  // start time for write to output file

// open outfile for writing
if ((outfilefd = fopen(outfilename, "a")) == NULL)
 {
 printf("\ncan not open %s\n", outfilename);
 exit(2);
 }

// loop writes/appends calculations to outfile
for(compindex=0; compindex < maxarraysz; compindex++)
 {
 fprintf(outfilefd, "%s,", score[compindex]->label);
 for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
  {
  fprintf(outfilefd, "%.12f,", score[compindex]->reldif[descriptindex]);
  }
 fprintf(outfilefd, "%.12f\n", score[compindex]->reldif[numdescripts]);
 }
fclose(outfilefd); // close outfile

endtimewrite = time(NULL); // end time for write to output file
totaltime = endtimewrite - starttimewrite; // total time for file write
printf("\n%i lines written to the output file in %.0f seconds", \
 maxarraysz, totaltime);

// reset the combinatorial calculation array
compstart = compstart + maxarraysz;
for(compindex2=0; compindex2 < maxarraysz; compindex2++)
 {
 for(descriptindex2=0; descriptindex2 < numdescripts+1; descriptindex2++)
  {
  score[compindex2]->reldif[descriptindex2] = 0.0;
  }
```

```
  }
 // printf("\narray reset");
 }


endtimetotal = time(NULL); // end time before last array
totaltime = endtimetotal - starttimetotal; // total time before last array
printf("\n\ntotal time before last array %.0f seconds", totaltime);



// start of last array
 printf("\nstarting last set of %i calculations", lastarraysz);

 starttimecalc = time(NULL); // start time for last array calculation time

 // loop through the appropriate space in the score array for
 // storing newly calculated relative difference score
 for(compindex=compstart; compindex < compstart+lastarraysz; compindex++)
  {
  // loop through descriptions being compared
  for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
   {
   // loop through entity x
   for (entityindex=0; entityindex < numentities - 1; entityindex++)
    {
    // loop through entity y
    for (entityindex2=entityindex+1; entityindex2 < numentities; entityindex2++)
     {
     // calculate relative difference score for each combinatorial
     // descriptor comparison
     descriptscore = \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex] - \
      (float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex])) / \
      (0.5 * \
      (fabs((float)conformation[entityindex]\
      [(int)conformationidx[entityindex]]->descriptivedata[descriptindex]) + \
      fabs((float)conformation[entityindex2]\
      [(int)conformationidx[entityindex2]]->descriptivedata[descriptindex]))) + \
      descriptscore;
     } // end entity 2
    } // end entity

   // calculate and store single measure relative difference score
   singlemeasureRD = descriptscore / divisor;
   scorebuffer.reldif[descriptindex] = singlemeasureRD;
   singlemeasureRD = 0.0;

   // keep running total for total measure relative difference score
   totalmeasureRD = (descriptscore / divisor) + totalmeasureRD;
   descriptscore = 0.0;
   } // end of description loop
```

```
// calculate and store total measure relative difference score
// this is the similarity score
scorebuffer.reldif[numdescripts] = totalmeasureRD / numdescripts;
totalmeasureRD = 0.0;

// concatenate and store the label of the comparison set aka comparison group
(char *)scorebuffer.label = strcpy(scorebuffer.label, nullstring);
for (entityidxbuff = (entityindex - (numentities - 1)); \
 entityidxbuff <= entityindex; entityidxbuff++)
  {
  (char *)scorebuffer.label = \
   strcat(scorebuffer.label, \
   conformation[entityidxbuff][(int)conformationidx[entityidxbuff]]->labels);
  if (entityidxbuff < entityindex)
    {
    strcat(scorebuffer.label, "_");
    }
  }


// heart of keeping the indexes pointing to the right place in the
// conformation data - loop through conformations of last entity
// until the last conformation of the last entity is reached. Then
// increment the second to the last entity to the next conformation
// and decrement the last entity to its first conformation. Following
// this through all of the entities will cause every conformation to
// be compared with every conformation of the other entities.
// This is some what analogous to a mechanical odometer (or counter)
(int)conformationidx[argc-2] = (int)conformationidx[argc-2] + 1;
if((int)conformationidx[argc-2] >= (int)numconfs[argc-2])
  {
  for (confindex = (argc-2); confindex > 0; confindex--)
    {
    if((int)conformationidx[confindex] >= (int)numconfs[confindex])
      {
      (int)conformationidx[confindex] = 0;
      (int)conformationidx[confindex-1] = (int)conformationidx[confindex-1] + 1;
      }
    }
  }

// putting a label with the scores
score[compindex-compstart]->label = strcpy(score[compindex-compstart]->label, scorebuffer.label);
for(descriptindex=0; descriptindex <= numdescripts; descriptindex++)
  {
  score[compindex-compstart]->reldif[descriptindex] = scorebuffer.reldif[descriptindex];
  }

 } // end of compindex computational group loop
endtimecalc = time(NULL);
totaltime = endtimecalc - starttimecalc;
printf("\n\n%i comparisons have been completed", numcomps);
printf("\n%i comparisons of been calculated in %.0f seconds", \
```

```
 lastarraysz, totaltime);


 starttimewrite = time(NULL);  // start output file write time
 // open outfile for writing
 if ((outfilefd = fopen(outfilename, "a")) == NULL)
  {
  printf("\ncan not open %s\n", outfilename);
  exit(2);
  }
 fflush(NULL);
 // loop appends last array calculations to outfile
 for(compindex=0; compindex < lastarraysz; compindex++)
  {
  fprintf(outfilefd, "%s,", score[compindex]->label);
  for(descriptindex=0; descriptindex < numdescripts; descriptindex++)
   {
   fprintf(outfilefd, "%.12f,", score[compindex]->reldif[descriptindex]);
   }
  fprintf(outfilefd, "%.12f\n", score[compindex]->reldif[numdescripts]);
  }
 fclose(outfilefd);  // close outfile

 endtimewrite = time(NULL);  // end output file write time
 totaltime = endtimewrite - starttimewrite;  //  total file write time
 printf("\n%i lines written to the output file in %.0f seconds", \
 lastarraysz, totaltime);


 endtimetotal = time(NULL);  // end total time
 totaltime = difftime(endtimetotal, starttimetotal);
 printf("\n\ntotal time %.0f seconds\n\n", totaltime);  // total run time


// freeing memory

 printf("\nstarting to free memory\n");
 for(entityindex=0; entityindex < numentities; entityindex++)
  {
  for(confindex=0; confindex < (int)numconfs[entityindex]; confindex++)
   {
   free(conformation[entityindex][confindex]->labels);
   free(conformation[entityindex][confindex]->descriptivedata);
   free(conformation[entityindex][confindex]);
   }
  free(conformation[entityindex]);
  }
 free(conformation);

 for(compindex=0; compindex < lastarraysz; compindex++)
  {
  free(score[compindex]->label);
  free(score[compindex]->reldif);
  free(score[compindex]);
```

```
  }

free(score);
free(scorebuffer.reldif);
free(scorebuffer.label);

printf("\nmemory free\n");

printf("\nDone!\n");

return(0);
}
```

Bibliography

Accelrys, Inc., San Diego, California, www.accelrys.com

AESOP developed by B.B. Masek, Zeneca, Wilmington, Delaware.

Baumann, B.H., Pable, J.P., Ali, S.F., Rothman, R.B., and Mash, D.C. Noribogaine (12-hydroxyibogaine0: a biologically active metabolite of the antiaddicitive drug ibogaine. *Ann. N.Y. Acad. Sci.* 2000, **914**, 354-368.

Baase, S., Van Gelder, A. *Computer Algorithms Introduction to Design and Analysis.* Addison-Wesley, Menlo Park, CA, 2000

Blumenthal, L.M. *Theory and applications of distance geometry.* Chelsea Publishing, Bronx, 1970.

Crippen, G.M. A novel approach to calculations of conformation: distance geometry. *J. Comp. Phys.* 1997, **24**, 96-107.

Dean, P.M., and Perkins, T.D.J. Calculation of three-dimensional similarity. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 199-218.

Gerdes, J.M., DeFina, S.C., Wilson, P.A., and Taylor,S.E. Serotonin transporter inhibitors: synthesis and binding potency of 2′-methyl- and 3′-methyl-6-nitroquipazine. *Bioorg. Med. Chem. Lett.* 2000, **10**, 2643-2646.

Greco, G., Novellino, E., and Martin, Y.C. 3D-QSAR methods. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 219-252.

Gundertofte, K., Bøgesø, K.P., and Liljefors, T. A stereoselective pharmacophore model of the serotonin re-uptake site. In: *Computer-assisted lead finding and optimization*, Waterbeemd, H., Testa, B., and Folkers, G., Eds., VHCA, Basil, and Wiley-VHC, Weinheim, 1997, pp. 445-459.

Jin, B., and Hopfinger, A.J. A proposed common spatial pharmacophore and the corresponding active conformations of some TxA_2 receptor antagonists. *J. Chem. Inf. Comput. Sci.* 1994, **34**, 1014-1021.

Martin, Y.C. Pharmacophore Mapping. In: *Designing bioactive molecules: three-dimensional techniques and applications*, Martin, Y.C., and Willet, P., Eds., ACS, Washington DC, 1998, pp. 121-148.

Microsoft Corporation, Redmond, Washington, www.microsoft.com

Mottola, D.M., Laiter, S., Watts, V.J., Tropsha, A., Wyrick, S.D., Nichols, D.E., and Mailman, R.B. Conformational analysis of D_1 dopamine receptor agonists: pharmacophore assessment and receptor mapping. *J. Med Chem.* 1996, **39**, 285-296.

Nicklaus, M.C., Wang, S., Driscoll, J.S., and Milne, G.W. Conformational changes of small molecules binding to proteins. *Bioorg. Med. Chem.* 1995, **3**, 411-428.

Papadopoulos, M.C., and Dean, P.M. Molecular structure matching by simulated annealing. IV. Classification of atom correspondences in sets of dissimilar molecules. *J. Comput.-Aided Mol. Design* 1991, **4**, 119-133.

Perkins, T.D.J., and Dean, P.M. Molecular partial similarity using surface-volume comparisons. In: *Computer-assisted lead finding and optimization*, Waterbeemd, H., Testa, B., and Folkers, G., Eds., VHCA, Basil, and Wiley-VHC, Weinheim, 1997, pp. 421-432.

Rupp, A., Kovar, K., Beuerle, G., Ruf, C., and Folkers, G. A new pharmacophoric model for 5HT reuptake-inhibitors: differentiation of amphetamine analogues. *Pharma. Acta Helv.* 1994, **68**, 235-244.

Tripos, Inc., St. Louis, Missouri, www.tripos.com

Vaswani, M., Linda, F.K., and Ramesh, S. Role of Selective Serotonin Reuptake Inhibitors in Psychiatric Disorders: a Comprehensive Review. Progress in Neuro-Psychopharmacology & Biological Psychiatry. 2002, 27, 85-102

WaveMetrics, Inc., Lake Oswego, Oregon, www.wavemetrics.com

Weiss, M.A. *Data Structures and Algorithm Analysis in C.* Menlo Park, CA, 1997.

Wilson, J.D. Physics Laboratory Experiments. D.C. Heath and Company, Lexington, 1986, pp. 9-10.