THE COMPLEXITY OF BIOINFORMATICS: TECHNIQUES FOR ADDRESSING THE COMBINATORIAL EXPLOSION IN PROTEOMICS AND GENOMICS

BY

LEONID ZAMDBORG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Biophysics and Computational Biology
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

ABSTRACT

The last decade has seen an explosion of data arising from the development and proliferation of high-throughput data gathering and analysis pipelines. In order to transform this data into useful hypotheses and conclusions, it is necessary to determine which of it is pertinent to the problem being studied, and sometimes, conversely, which of many hypotheses being considered is best supported by the data at hand. In particular, the field of proteomics often grapples with this challenge, due to being at the confluence of a large number of high-throughput data pipelines. This work presents a series of computational frameworks that address this challenge in a manner that is both computationally efficient and biologically informative, acting as selective filters for the vast amount of data being processed.

A system is first presented to vastly reduce the potential combinatoric complexity of post-translational modifications (PTMs) and coding single nucleotide polymorphisms (cSNPs) for Top Down proteomics. Top Down proteomics is uniquely susceptible to a combinatorial explosion; as sequence length increases, the number of potential combinations of mass shift-inducing sequence features increases exponentially. This may be addressed to some extent by the process of shotgun annotation, where combinations of known PTMs and cSNPs are considered. This is in contrast to the rule-based variable modification approach prevalent in Bottom Up proteomics, where all residues of a given type are considered to be potentially modified in a specified manner. However, as high-throughput annotation pipelines vastly increase the number of known modifications and polymorphisms, the number of their combinations grows exponentially and eventually becomes unmanageable. It becomes necessary to restrict the potential combination space in a manner that does not unduly impinge on the identification and characterization capabilities of shotgun annotation. Built as part of a general framework for sequence transformation, the system being presented utilizes a genetic algorithm to identify a group of PTMs and cSNPs that is most suitable for inclusion in a shotgun-annotated sequence database. Additionally, a number of other advancements are presented in the bioinformatics of Top Down proteomics, including a cluster implementation of the ProSight search engine, and a design plan for the next generation of ProSight, built using the principles of online sequence transformation and optimization. This addresses the combinatorial explosion by providing means of efficiently restricting the search space, minimizing the amount of duplicated effort, and leveraging modern processor design to maximize throughput.

Second, genetic algorithms are applied to the problem of *de novo* peptide sequencing in Bottom Up proteomics by means of ultra-high-resolution mass spectrometry. Rather than detecting large numbers of less accurate fragment peaks as is presently typical in Bottom Up proteomics, detecting fragment ions at high resolution results in smaller numbers of highly accurate monoisotopic masses after deisotoping. This allows potential *de novo* sequence solutions to have exceedingly low fragment mass degeneracy. Presently, algorithms for *de novo* peptide sequencing that fully take advantage of this capability have been lacking. A system is presented for incorporating numerous metrics of solution quality simultaneously to evolve a sequence solution that best fits available data. The nature of proteomic data and its amenability to analysis by means of genetic algorithms is discussed. This system demonstrates highly confident automatic *de novo* peptide sequencing using a small number of confident fragment masses, potentially measured at the limits of detection.

Third, a system is presented for the efficient discovery of protein-DNA interactions by means of multiple simultaneous gene expression measurements. A major problem in discovering transcription factor binding motifs is that identifying overrepresented sequence motifs is insufficient; most are noise, and some only bind transcription factors under specific biological conditions. It is possible to identify real motifs by the correlation of their presence to differential gene expression under a particular biological condition. By employing multivariate penalized regression, the system described is capable of efficiently identifying transcription factor binding motifs whose presence strongly correlates with gene expression in the measured biological condition from amongst hundreds of candidates. A small, highly confident set of motifs is selected, which may be used for further bioinformatic studies, or as targets for *in vivo* or *in vitro* experiments.

ACKNOWLEDGEMENTS

This work could not have been possible without the input and inspiration of numerous co-workers. When I joined the Kelleher Research Group, the development of ProSight was guided by the hand of Richard LeDuc, and his ideas continue to have great impact on how ProSight evolves. The ProSight development team has waxed and waned over the years, but in particular, I would like to gratefully acknowledge the great efforts of Paul Thomas, Craig Wenger, Shannee Babai, Bryan Early, Kenneth Durbin, Michal Reuven, Joseph Stull, and Ben Wright, with whom I worked side-by-side to make ProSight a force to be reckoned with. The development of Refiner was directly inspired by a very stimulating collaboration with Ji Eun Lee and Timothy Cleland. All the work described in this dissertation was guided and supported by my advisors, Dr. Neil Kelleher and Dr. Ping Ma, who taught me several distinct dialects of the language of science, and how to translate between them. My studies and work were supported by a number of sources, including the Center for Biophysics and Computational Biology, the National Institutes of Health (R01 GM 067193), and the National Science Foundation (DMS 0800631). Extensions of this work led to an NIH fellowship (F30 DA 026672).

I would also like to take this opportunity to acknowledge the mentors who were absolutely instrumental to getting me to this point: Dr. Denis Moore, Dr. Semyon Rafalson, Dr. John Reinitz, Dr. Alexander Spirov, and Dr. Scott Tenner. Without your encouragement and advice, I would never have been able to begin this work.

TABLE OF CONTENTS

CHAPTER 1: Introduction

Biological systems are famous for their immense complexity. Simple building blocks come together to produce systems of breathtaking intricacy and tremendous variation. The *magnum opus* of biology in this century, if one can define it, would be to bring the analysis and comprehension of such systems in line with the rigor and thoroughness that phyisics and chemistry embraced in the previous. The complete understanding of biology would perhaps be far too great a demand; nonetheless, drawing biology away from its phenomenological roots and into the realm of fully-fledged theoretical science would be a laudable goal. In order to do so, however, the immense complexity of biology must be tamed.

This work attacks the great challenge of biology through the lens of proteomics, the study of complete proteomes. The term 'proteome' was coined by Marc Wilikins in 1994 [1] and first used in the literature in 1995 by Wasinger *et al.* [2] by analogy to 'genome': a proteome is the total protein complement of a genome. Unlike genomics, however, proteomics suffers tremendously from the curse of complexity. Even if one considers all unique molecular species that may be produced based on a single gene to be the same protein, a proteome contains many more proteins than there are genes in its underlying genome. If one considers every unique proteinaceous species to be distinct, their number becomes astronomical. If one can tame this complexity, however, one can reap great rewards; while the genome encodes the blueprint, the proteome is the completed edifice. The understanding of proteomics in its great complexity will finally allow 'closing the loop' on the Central Dogma of molecular biology: from DNA, to RNA, to protein, to function, to survival, and at last, to DNA. This dissertation will present a number of bioinformatic tools to address the great complexity of proteomics, and to begin linking it to function.

A Brief History of Proteomics

Before one can begin to discuss the computational tools with which one may tackle proteomics, one must first detail the nature of the science. In a sense, proteomics had existed for many years before it was named. Before computational techniques, before even programmable computers, the first attempts at unraveling the complexity of the proteome relied on separation. As is often the case with techniques used for proteomics, the goal separation is the reduction of complexity. It seeks to isolate, or at least fractionate into more manageable groups, the proteins in a given sample. The first efforts at protein separation, carried out in the 19th century, relied on fractional precipitation [3]; the resulting fractions were then characterized by various classical methods. This technique was used to great effect throughout the first third of the 20th century, and many of the classic model proteins of biology were first isolated through its use. However, it was, fundamentally, unsuitable as a general-purpose, proteome-scale separation technique due to its very low resolution. The number of distinct fractions resolvable by means of precipitation rarely exceeds ten. The key to proteomics in the middle third of the 20th century was electrophoresis.

Electrophoresis is the technique of propelling particles within some medium via the influence of an electric field. Electrophoresis was first used to study proteins in 1892 by Picton and Linder [4], when they observed the movement of hemoglobin in an electric field. They were also the first to use it for the separation of protein mixtures, in 1897 [5]. Three decades later, the dissertation work of Arne Tiselius in 1930 demonstrated the superiority of electrophoretic separation of proteins to the fractional precipitation methods in vogue at the time [6]. Tiselius would go on to win the Nobel Prize for this and later work, and is known to this day as the "Father of Electrophoresis". Electrophoretic separation of proteins relies on the fact that as proteins are propelled through some medium by a constant electric field, they do so at different rates, determined largely by their mass and shape. In polyacrylamide gel electrophoresis (PAGE), the typical technique used today, the application of an electric field to protein samples loaded onto a polyacrylamide gel for a fixed amount of time results in a characteristic pattern of bands on the gel. Proteins will migrate according to their charge-to-size ratio (typically simplified as molecular weight, MW), and one may identify them based on their reproducible retention factor (location) on the gel.

Basic PAGE is one-dimensional (1D-PAGE): it only separates based on one property, so proteins with the same charge-to-size ratio would typically have the same retention factor. The immense complexity of the proteome means that a single band on a 1D-PAGE gel is composed of hundreds, if not thousands of distinct proteins, making identification very difficult. In order to improve the separation ability of gel electrophoresis, it was necessary to add an additional dimension of separation. In other words, while proteins are being separated by their mass, they would simultaneously be separated by another, orthogonal (independent) property. The property usually used for the second dimension of separation is isoelectric point (pI), and the procedure employing it for separation is isoelectric focusing. The standard technique used for two-dimensional gel electrophoresis with isoelectric focusing for the last 35 years is 2D-PAGE, developed by O'Farrell in 1975 [7]. This vastly improved the separation ability of gel electrophoresis, allowing for hundreds of distinct spots to be observed. Like 1D-PAGE, the location of proteins on similarly-prepared gels run with the same parameters is reproducible. Due to this property, 2D-PAGE became the *de facto* standard for protein identification at the time. It is still often used for a quick determination of whether two samples differ in some reasonably-abundant component, especially when combined with computational methods for rapidly comparing two gel images. However, the limitations of 2D-PAGE, which include its bias to extremes in pI and MW, have been well-documented [8]. In addition, 2D-PAGE suffers from limitations as a separation platform for subsequent analytic stages. It has been noted to have poor recovery of hydrophobic proteins, and has a very low loading capacity. Given these limitations, there has been a general movement in modern proteomics away from gel-based separation techniques to solution-phase separation, such as chromatography.

Chromatography is another very common protein separation technique that gained prominence in the latter half of the 20th century. First described in the 19th century work of Schönbein and Goppelsröder, it is based on the differential adsorption of analytes in a mobile phase to a stationary phase [9-

10]. Goppelsröder used planar chromatography, where filter paper served as the stationary phase, to separate the small molecule and proteinaceous components of human urine, then characterized them using spectroscopic and colorimetric assays [11]. This was followed by the work of Tsvet, who developed adsorption column chromatography, wherein a column packed with a polar stationary phase (calcium carbonate) had a non-polar mobile phase (petroleum ether) with the analytes of interest passed through it. This process was applied on numerous occasions to successfully separate chlorophylls and carotenoids [12]. However, due to language barriers, the work of both Goppelsröder and Tsvet fell into obscurity. It wouldn't be until 1952 that the award of the Nobel Prize to Martin and Synge for their work on partition chromatography gave prominence to the use of chromatographic methods to separate amino acids, peptides, and proteins [13]. Despite being over a century after the initial work of Goppelsröder, modern high-performance reverse-phase liquid chromatography is used for much the same purpose: the protein constituents of the analyte mixture are eluted off the chromatographic column at differing times. The characteristic elution times are reproducible, and are relatively specific to particular protein constituents of the analyte mixture.

All of the separation methods detailed above, however, are dependent on some sort of detector to determine the presence of the analyte. The most basic detector is the human eye; if the analyte has a characteristic color, or an indicator dye is present, the investigator can observe and note distinct bands or elution events that arise from the separation procedure. More sophisticated detectors include UV spectroscopy and microchemical analysis. These methods can also be applied to putatively identifying the separated proteins, though their low specificity makes conclusions drawn from such techniques somewhat suspect. Nonetheless, highly-specific identification techniques do exist: for example, a very specific technique still used for proteomics is immunochemical staining. By reacting a protein fraction with a labeled antibody for a particular protein, one may observe whether or not that protein is present in that particular fraction. One could also look for an absorption peak at a particular frequency, characteristic of some particular protein. Nonetheless, all of these techniques have either poor specificity (such as UV spectroscopy) or poor throughput (immunochemistry). A highly-specific, highly-sensitive, and high-throughput technique was needed to fully identify the separated proteins, a technique that formed the basis of modern proteomics: mass spectrometry.

Mass spectrometry is a technique for measuring the mass-dependent properties of molecules in a sample. It consists of three components: an ionization method, a mass analysis method, and a detection method. The sample to be analyzed is first ionized. The resultant ions are introduced into the mass analyzer, which separates them based on their mass-to-charge (m/z) ratio. The presence of the separated ions is then measured by the detector. The combination of the calibrated mass analyzer and the detector enables the mass spectrometer to determine the intensity of ions present in the instrument at various m/z.

Mass spectrometry as it is understood today was developed by Aston in 1919, with his separation and measurement of distinct isotopes, first of neon [14], then of other elements including carbon, nitrogen, and oxygen [15] by means of what he called the "mass spectrograph", a work which led to him being awarded the Nobel Prize in 1922. Mass spectrometry was used extensively in the first half of the 20th century to characterize isotopes, even being used on a preparative scale to separate ^{235}U for the Manhattan Project [16]. The use of mass spectrometry for the characterization of proteinaceous material was, however, first attempted in 1959, when Biemann measured partial hydrolysates of various derivatized peptides, determining their sequence by means of observing sets of peaks separated from each other by the characteristic mass of an amino acid [17]. This method, the detection of peak sets separated by amino acid masses, is now known as *de novo* peptide sequencing, and remains in use to this day. One year later, Biemann laid the foundation of what would become mass spectrometric proteomics by mating a separation method (in this case, a gas chromatograph) to a mass spectrometer, enabling the characterization of complex peptide mixtures [18].

Mass spectrometric proteomics rapidly evolved as new techniques were developed. Biemann's partial hydrolysis method for *de novo* peptide sequencing was eventually replaced by gas-phase dissociation. It was noticed in early work on peptide mass spectrometry that molecular ions tended to fragment along the amide backbone, resulting in characteristic sets of daughter ions [19]. By analyzing the inter-peak distances of these fragments, it was possible to determine the peptide sequence without requiring prior partial hydrolysis. It was likewise immediately noticed that these inter-peak distances could be easily determined by a computer program that "walked" the spectrum; development of such a system by McLafferty [20] and Biemann [21] in 1966 could be considered the dawn of computational proteomics.

Though gas-phase dissociation of molecular ions was originally observed when metastable species fragmented immediately following ionization ("source-induced dissociation"), work soon focused on encouraging and controlling the formation of daughter ions. The most prominent dissociation method developed, and still the dominant method to this day, was collisionally-induced dissociation (CID) by McLafferty in 1973 [22]. This technique was of considerable use, but truly shone when implemented between the stages of a tandem mass spectrometer (MS/MS). In the MS/MS arrangement, there are two stages of mass analysis. In some systems, such as those using magnetic sectors or a triple-quadrupole arrangement, two mass analyzers are placed in series, one after the other. In others, such as linear ion traps, a mechanism is employed that enables daughter ions from a gas-phase dissociation reaction to be reintroduced back into the same mass analyzer. Both techniques allow the separation of parent ions in the first stage, selection of a particular precursor ion, fragmentation of that precursor, separation of the resultant daughter ions in the second stage, then measurement of those ions [23]. This forms the final conceptual piece of the standard system used to this day: a separation platform directly interfacing with an ionization source of a tandem mass spectrometer that is capable of selection, dissociation, and detection of parent and daughter ions.

Modern mass spectrometric proteomic investigations typically begin with various dimensions of separation using either electrophoresis or varying flavors of chromatography. They introduce the separated samples into an instrument by one of two "soft" ionization methods: electrospray ionization (ESI) [24] or matrix-assisted laser-induced desorption/ionization (MALDI) [25], both of which reliably ionize large proteinaceous species with minimal dissociation at the source. These ionization methods are used to introduce sample into what is typically a hybrid tandem mass spectrometer, consisting of two different types of mass analyzer. One very common configuration is ESI introducing multiply-charged ions into a relatively low-resolution linear quadrupole ion trap coupled to a high-resolution Fourier transform mass analyzer – either an ion cyclotron resonance (ICR) cell or Orbitrap. Another is MALDI introducing mostly singly-charged ions into a quadrupole mass analyzer coupled to a time-of-flight (TOF) tube. Both configurations offer data-dependent selectivity fragmentation, affording the analysis of complex samples and species. Nonetheless, the inherent complexity of proteins and the nature of the instrumentation results in two distinct, complementary approaches being used for modern proteomics: Top Down and Bottom Up.

The most common technique currently used for proteome-scale protein identification is known as Bottom Up Proteomics. The key to this technique is that either before or after protein separation, the sample is digested, typically with trypsin, to produce short peptides. These peptides are then usually separated and introduced into a tandem mass spectrometer, which isolates and fragments intense peaks, producing fragment ions that are used to identify peptides (and consequentially the proteins from which they came) [26]. This method typically produces sufficient sequence coverage to identify many proteins, and indeed, the current records for the number of proteins identified from a single sample are held by Bottom Up techniques [27].

However, Bottom Up is not without its flaws. Firstly, the peptides produced by enzymatic digestion are not necessarily unique to a particular protein. As proteins are composed of conserved domains, peptides may identify a group of proteins, rather than one unique protein. Secondly, enzymatic digestion destroys combinatorial positional information of sequence variants and post-translational modifications. If a peptide from a particular protein is detected with a phosphorylation on a given residue, and another peptide from the same protein is detected with another phosphorylation, the protein is not necessarily simultaneously phosphorylated on both sites. It merely indicates that it could potentially bear phosphorylations on one or both residues. Finally, quantitation of proteins by peptide levels is made highly problematic when one considers the first two issues – if one detects a change in abundance of a particular peptide, it is not necessarily the case that the protein from which it was produced had changed its abundance by the same amount.

These problems are resolved by the other major approach in proteomics, Top Down. In Top Down proteomics, separated intact proteins are introduced directly into the mass spectrometer. The gas-phase dissociation of these large molecular ions produces large fragment ions, which often produce 100% sequence coverage of the observed protein, enabling confident identification and full characterization [28]. Quantitation could likewise be done on the level of intact protein, eliminating problems with backtracking from peptide abundances. Top Down is an area of rapid development in all the major technical aspects of the methodology: separation, instrumentation, and data analysis. The latter component will be elaborated on in detail in the remainder of this work.

Fundamentals of Computational Proteomics

Computational analysis was a fundamental component of proteomics since the first peptide mass spectra were collected [20-21]. These initial *de novo* peptide sequencing approaches, however, were developed prior to large databases of predicted protein sequences becoming available in the post-genome era. Armed with these databases and a wealth of data arising from instrumentation utilizing ESI and MALDI, it became possible to identify and potentially characterize proteins in a sample without needing to fully sequence them. There are three major approaches: sequence tag matching, peptide mass fingerprinting, and fragment mass matching.

The sequence tag approach, though not the first to be implemented, is the one with perhaps the oldest pedigree. It uses partial *de novo* sequencing of short regions of the dissociated protein or peptide, generating a “sequence tag”. This sequence tag is then searched against a sequence database [29]. Because automated *de novo* sequencing algorithms have existed since the original work of McLafferty and Biemann, it is straightforward to couple this database search approach with automated *de novo* sequencing, resulting in a robust searching tool. Because this approach is less sensitive to mass shifts in regions not covered by the sequence tag, it remains useful for cases where sequence databases are sparse, or potential sequence variation and modification is unknown. Tools utilizing this approach are still under development today, such as PEAKS [30]. Applying this approach to cases of post-translational modifications within the sequence tag requires that the set of potential mass shifts between peaks when compiling the tag be expanded to support modified amino acids. This of necessity increases the potential space of potential sequences, and may significantly decrease specificity, especially on noisy spectra.

Peptide mass fingerprinting was developed simultaneously by a number of groups in 1993 [31-33]. The basic principle of peptide mass fingerprinting is that the observed intact masses of tryptic peptides resulting from a Bottom Up experiment form a "fingerprint" of the protein from which they came. By incorporating a probability-based score utilizing the number of peptide masses matched, it is possible to rank a database of potential protein hits by the likelihood of whether or not they were observed. This approach is computationally simple, and can be carried out on a standalone (not tandem) mass spectrometer; however, its lack of specificity has made it fall out of favor as more sophisticated techniques were developed. On the other hand, this approach is least affected by post-translational modifications, in the sense that it is incapable of, and not concerned with, their full localization. A peptide mass may be considered without post-translational modifications, or with potential modifications (either known or based on simple residue rules). The number of theoretical masses considered increases, but not nearly the exponential increase suffered by the final approach, direct fragment matching.

The last major approach for analyzing mass spectrometric data, fragment matching, was first implemented in the Sequest algorithm by the Yates group [34] in 1993. Sequest relies on the cross-correlation of a theoretical fragmentation spectrum and an observed fragmentation spectrum. Each protein being considered for scoring has a cross-correlation score computed between its theoretical spectrum and the observed spectrum. This results in an cross-correlation metric (XCorr), which when coupled with selective scoring of peptides based on their intact mass, allows for very high sensitivity. Although it is possible to apply the Sequest algorithm to in-source dissociation spectra on a single-stage mass spectrometer, akin to the data produced by the original *de novo* peptide sequencing attempts of McLafferty and Biemann in 1966 [20-21], such an approach is not recommended. Using the Sequest algorithm on data resulting from such experiments would entail scoring the entire protein sequence database against the observed spectrum, which would destroy the algorithm's specificity. Instead, Sequest is ideally suited for tandem mass spectrometric data, where the intact precursor mass can be used to restrict the database "window" being searched. However, it is very difficult to interpret the cross-correlation metric used by Sequest to determine what counts as a "hit". All that any search engine does is assign a score to the proteins being considered; it takes additional user assumptions to determine what scores constitute "hits" and what scores do not. The cross-correlation metric, while having exquisite sensitivity, has low specificity and no axiomatic way of determining cutoffs.

Another major algorithm in common use is Mascot, developed by Matrix Science in 1999 [35]. Based in part on MOWSE [31], one of the algorithms developed early on for peptide mass fingerprinting, it instead operates on the fragments of a peptide that result from gas-phase dissociation inside the mass spectrometer, rather than on peptides that result from the tryptic digestion of a protein. The same probabilistic score the enabled ranking proteins by how many theoretical peptides matched observed data was adapted to ranking peptide by how many theoretical fragments matched observed data. This made the Mascot score more comprehensible and analyzable than the opaque and highly charge-dependent Sequest cross-correlation metric.

Other, more recent algorithms for fragment matching exist, but in general, they use the same approach with more sophisticated probabilistic scoring functions to generate scoring metrics with reasonable, axiom-based or procedural ways of assigning score cutoffs. In general, modern computational proteomics relies on fragment matching algorithms for most of its heavy lifting, with the other two approaches relegated to specialized tasks. Fragment matching algorithms typically handle post-translational modifications as "variable modifications", assigned by rules. For instance, if one considers serine phosphorylation, all serines may be potentially phosphorylated. If a peptide being scored has 3 serines, then 8 theoretical peptides must be considered: with any reasonable fragment matching tolerance, a modified observed fragment will not match an unmodified theoretical fragment, and vice-versa. Despite this massive increase in the amount of work necessary to assign post-translational modifications, fragment matching algorithms are the only effective means of complete peptide and protein characterization, and together with the sequence tag approach, form the core of the first search engine developed specifically for Top Down experiments: ProSight.

References

1. Wilkins, M.R., *Proteomics data mining*. Expert Review of Proteomics, 2009. **6**(6): p. 599-603.
2. Wasinger, V.C., et al., *Progress with gene-product mapping of the Mollicutes: Mycoplasma genitalium*. Electrophoresis, 1995. **16**(7): p. 1090-1094.
3. Haslam, H.C., *The separation of proteids*. The Journal of Physiology, 1905. **32**(3-4): p. 267-298.
4. Picton, H. and S.E. Linder, *XI.—Solution and pseudo-solution. Part I.* Journal of the Chemical Society, Transactions, 1892. **61**: p. 148-172.
5. Picton, H. and S.E. Linder, *LVII.—Solution and pseudo-solution. Part III. The electrical convection of certain dissolved substances*. Journal of the Chemical Society, Transactions, 1897. **71**: p. 568-573.
6. Tiselius, A.W.K., *The moving boundary method of studying the electrophoresis of proteins*, in *Department of Chemistry*. 1930, Uppsala University: Uppsala.
7. O'Farrell, P.H., *High resolution two-dimensional electrophoresis of proteins*. The Journal of Biological Chemistry, 1975. **250**(10): p. 4007-4021.
8. Gygi, S.P., et al., *Evaluation of two-dimensional gel electrophoresis-based proteome analysis technology*. Proceedings of the National Academy of Sciences of the United States of America, 2000. **97**(17): p. 9390-9395.

9. Goppelsröder, F., *Ueber ein Verfahren Farbstoffe in ihren Gemischen zu erkennen.* Verhandlungen der Naturforschenden Gesellschaft zu Basel, 1861. **3**(2).

10. Goppelsröder, F., *Ueber ein neues Verfahren Farbstoffe in ihren Gemischen zu erkennen.* Annalen der Physik und Chemie, 1862. **191**(3): p. 487-494.

11. Goppelsröder, F., *Studien über die Anwendung der Capillaranalyse.* 1904: E. Birkhäuser.

12. Tsvet, M.S., *Physical chemical studies on chlorophyll adsorptions.* Berichte der Deutschen botanischen Gesellschaft, 1906. **24**: p. 316-323.

13. Martin, A.J.P. and R.L.M. Synge, *A new form of chromatogram employing two liquid phases: A theory of chromatography. 2. Application to the micro-determination of the higher monoamino-acids in proteins.* Biochem J, 1941. **35**(12): p. 1358-1368.

14. Aston, F.W., *Neon.* Nature, 1919. **104**(334).

15. Aston, F.W., *The Constitution of the Elements.* Nature, 1919. **104**(393).

16. Yergey, A.L. and A.K. Yergey, *Preparative Scale Mass Spectrometry: A Brief History of the Calutron.* Journal of the American Society for Mass Spectrometry, 1997. **8**(9): p. 943-953.

17. Biemann, K., F. Gapp, and J. Seibl, *Application of Mass Spectrometry to Structure Problems. I. Amino Acid Sequence in Peptides.* Journal of the American Chemical Society, 1959. **81**(9): p. 2274-2275.

18. Biemann, K. and W. Vetter, *Separation of peptide derivatives by gas chromatography combined with the mass spectrometric determination of the amino acid sequence.* Biochemical and Biophysical Research Communications, 1960. **3**(6): p. 578-584.

19. Wulfson, N.S., et al., *Mass spectrometric determination of the amino (hydroxy) acid sequence in peptides and depsipeptides.* Tetrahedron Letters, 1965. **6**(32): p. 2805-2812.

20. Senn, M. and F.W. McLafferty, *Automatic amino-acid-sequence determination in peptides.* Biochemical and Biophysical Research Communications, 1966. **23**(4): p. 381-385.

21. Biemann, K., C. Cone, and B.R. Webster, *Computer-Aided Interpretation of High-Resolution Mass Spectra. II. Amino Acid Sequence of Peptides.* Journal of the American Chemical Society, 1966. **88**(11): p. 2597-2598.

22. McLafferty, F.W., et al., *Metastable ion characteristics. XXII. Collisional activation spectra of organic ions.* Journal of the American Chemical Society, 1973. **95**(7): p. 2120-2129.

23. Kruger, T.L., et al., *Mixture analysis by mass-analyzed ion kinetic energy spectrometry.* Analytical Chemistry, 1976. **48**(14): p. 2113-2119.

24. Fenn, J.B., et al., *Electrospray ionization for mass spectrometry of large biomolecules.* Science, 1989. **246**(4926): p. 64-71.

25. Tanaka, K., et al., *Protein and polymer analyses up to m/z 100 000 by laser ionization time-of-flight mass spectrometry.* Rapid Communications in Mass Spectrometry, 1988. **2**(8): p. 151-153.

26. Hunt, D.F., et al., *Protein sequencing by tandem mass spectrometry.* Proceedings of the National Academy of Sciences of the United States of America, 1986. **83**(17): p. 6233-7.

27. Wisniewski, J.R., et al., *Universal sample preparation method for proteome analysis.* Nat Meth, 2009. **6**(5): p. 359-362.

28. Kelleher, N.L., et al., *Top Down versus Bottom Up Protein Characterization by Tandem High-Resolution Mass Spectrometry.* Journal of the American Chemical Society, 1999. **121**(4): p. 806-812.

29. Mann, M. and M. Wilm, *Error-Tolerant Identification of Peptides in Sequence Databases by Peptide Sequence Tags.* Analytical Chemistry, 1994. **66**(24): p. 4390-4399.

30. Ma, B., et al., *PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry.* Rapid Communications in Mass Spectrometry, 2003. **17**(20): p. 2337-2342.

31. Pappin, D.J.C., P. Hojrup, and A.J. Bleasby, *Rapid identification of proteins by peptide-mass fingerprinting.* Current Biology, 1993. **3**(6): p. 327-332.

32. Henzel, W.J., et al., *Identifying proteins from two-dimensional gels by molecular mass searching of peptide fragments in protein sequence databases.* Proceedings of the National Academy of Sciences of the United States of America, 1993. **90**(11): p. 5011-5015.

33. Mann, M., P. Højrup, and P. Roepstorff, *Use of mass spectrometric molecular weight information to identify proteins in sequence databases.* Biological Mass Spectrometry, 1993. **22**(6): p. 338-345.

34. Eng, J.K., A.L. McCormack, and J.R. Yates, III, *An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database.* Journal of the American Society for Mass Spectrometry, 1994. **5**(11): p. 976-989.

35. Perkins, D.N., et al., *Probability-based protein identification by searching sequence databases using mass spectrometry data.* Electrophoresis, 1999. **20**(18): p. 3551-3567.

CHAPTER 2: Bioinformatics of Top Down Proteomics

The work detailed here was done as part of the development of ProSight PTM 2.0, ProSightPC 2.0, and ProSightCluster, as well as other, ongoing work in the development and maintenance of ProSight. Reproduced in part, with permission, from "ProSight PTM 2.0: improved protein identification and characterization for top down mass spectrometry" Nucleic Acids Research Web Server Issue, Copyright 2007, Oxford Press. This work could not have been possible without the contributions of the entire ProSight development team, but especially Richard D. LeDuc, Bryan Early, and Joseph Stull. The work was supported by the National Institutes of Health through P30 DA 018310-039002 and R01 GM 067193.

The first developments in Top Down mass spectrometry were aimed squarely at detecting individual intact proteins. If identification was needed, two approaches could be attempted: a direct predicted-ion-matching approach and a sequence tag approach [1]. In the former, predicted charge-reduced theoretical fragment ion masses are directly compared to observed charge-reduced fragment ion masses. This technique allows for confirmation of complete protein characterization by total sequence coverage, but has great difficulty in working with highly modified or divergent protein sequences. In the latter, a consecutive series of inter-peak spacings characteristic of particular amino acids is detected, and a short 'sequence tag' compiled, which is then used to carry out a straightforward lexical search of a sequence database. This approach works better for highly divergent sequences or sequences where both termini are modified, but lacks the specificity of direct ion matching.

A number of qualitative differences exist between Top Down and Bottom Up proteomics that make analyzing Top Down data a unique challenge [Figure 2.1]. A typical Bottom Up data-dependent experiment acquires mass spectra of low-mass (rarely over 5000 Da) peptides in what is typically a 2+ or 3+ charge state using a high-resolution FT-ICR or Orbitrap instrument. The most intense peaks in this survey scan (often the monoisotopic peaks of their respective isotopic distributions) are selected for fragmentation using CID or ECD, and their fragment ions (rarely greater than 2+) measured in a low-resolution ion trap instrument. The sparse spectra resulting from this type of experiment may be analyzed by a search engine 'raw', without any pre-processing. The search engine would generate potential *m*/*z* values of theoretical fragment ions in 1+ and 2+ charge states and compare them to the observed peaks. Often, instead of attempting to match every isotopic peak, one either uses the average mass or the monoisotopic mass. Using monoisotopic mass data is preferred in proteomics due to the inherent variability of C12/C13 ratios in biomolecules, and the resultant limits on mass accuracy that average mass data imposes [2]. The precursor scan would have easily distinguishable monoisotopic peaks, as that is the most intense peak in low-mass species, allowing for monoisotopic neutral masses to be trivially assigned via charge inference, and then used to restrict the set of theoretical peptides considered for matching against the fragmentation spectrum. Alternatively, the search engine could simply consider all possible charge states for the parent peak, up to a predetermined maximum.

This type of strategy, however, cannot be used for Top Down data. An intact protein mass spectrum typically has a pronounced 'hump' [Figure 2.2], where chemical noise yields considerable intensity in every *m*/*z* channel. At higher charge states, monoisotopic peaks are typically below the noise floor, and isotopic peaks are spaced too closely for low-resolution instruments to resolve them. In order to determine the monoisotopic mass, a sophisticated transformation must be applied [3-4] to a high-resolution spectrum infer the charge and monoisotopic peak of an isotopic distribution. Alternatively, the charges may be inferred from low-resolution data by deconvoluting the characteristic multiple consecutive charge states produced by ESI [5-6]. Charge inference is even more vital for fragmentation spectra; Top Down CID and ECD spectra contain numerous multiply-charged fragments, and an exhaustive charge enumeration strategy for generating theoretical fragments is untenable. Top Down fragmentation spectra are often sufficiently dense due to chemical noise that attempting to match an arbitrary *m*/*z* value will almost always succeed; deconvolution and deisotoping are critical to having informative results.

At this writing, there are four search engines that are capable of analyzing Top Down data: ProSight [7], MascotTD [8], PIITA [9], and OMSSA [10]. These have a number of similarities imposed by the nature of the data. ProSight, the oldest search engine designed specifically for Top Down data, applies a neutral mass transform to both precursor and fragment spectra, as does MascotTD and PIITA.

Unlike the others, OMSSA can use either charge enumeration or external charge assignment for the precursor mass, and carries out charge enumeration for fragmentation spectra. The major difference in these search engines, however, is in how they handle post-translational modifications and sequence variations. MascotTD, PIITA, and OMSSA use simple Fasta-format data files as their back-end (converted to BLAST-format databases in the case of OMSSA), and thus do not incorporate any known information about sites of variation or modification. Instead, they use variable modifications, rule-based assignments of mass shifts to all residues based on their identity and general location (terminal or internal). For instance, a variable methionine oxidation would consider any methionine in a protein as potentially oxidized, and would enumerate all combinations of those oxidations, ranging from no oxidations to some user-defined maximum number of simultaneous oxidations. ProSight differs from the other Top Down-capable search engines in that it incorporates known sites of modification by using a rich dataset. Rather than using rule-based variable modifications, potential combinations of known post-translational modifications and sequence variations are stored in a rich sequence database, allowing them to be directly queried.

The origins of ProSight lie in the Database Retrieval Algorithm (DTA) that would eventually become the core of the search engine (Retriever) [11]. It was intended to make the identification and complete characterization of proteins by Top Down mass spectrometry feasible by using direct fragment matching, rather than the sequence tag approach. Retriever would form the core of a web-based system called ProSightPTM [12]. Although its underlying back-end differed somewhat from the modern implementation, the core search engine (Retriever 1.0) was the first publically-available system designed specifically to analyze deconvoluted, deisotoped Top Down mass spectral data. It had two fundamental search modes: an Absolute Mass search, where the observed neutral precursor mass is used to restrict the range of sequences in the database that are scored, and a Sequence Tag search, where manually- or automatically-compiled sequence tags are searched by direct text comparison to the database. The next generation of the search engine, Retriever 2.0 (deployed in ProSightPTM 2.0, neuroProSight, and ProSightPC) added nonspecific proteolysis support (Biomarker search), via a sliding-window algorithm.

Rather than being a single program, the current version of ProSight is a collection of related programs based on the same underlying technology [Figure 2.3], but dedicated to different tasks. ProSight is based on three primary components: Retriever, the core search engine (1) which uses deconvoluted, deisotoped mass spectral data to query a rich sequence database (2), when called by a user interface (3). The user interface is modular and several distinct systems were designed that call the same underlying search engine for different user-interaction scenarios. Two web-based user interfaces are currently in use: ProSightPTM 2.0 [7] for general-purpose searching and neuroProSight [13] for nonspecific searching and searching predicted neuropeptides using the NeuroPred algorithm [14]. A standalone user interface (ProSightPC 2.0) was also developed for high-throughput analysis. The latter wraps Retriever 2.0 in an iterative search logic which carries out additional searches on a particular query based on the results of previous searches, as well as supporting 'turnkey' data preprocessing and results reporting features.

As described earlier, the ProSight suite relies on a rich sequence database, encompassing as much known information as possible. The database schema used by the second-generation Retriever (the "Absolute Mass Schema" [Figure 2.4]) stores sequences with post-translational modifications embedded directly in prefix notation [Figure 2.5]. These post-translational modification codes refer to a general metadata schema ('DB_index') that contains information on the mass shifts that modifications apply to the following residue. Sequence variations are embedded without an explicit code, simply as a different sequence. Multiple sequences derived from a single source (due to sequence variants and/or PTMs) are grouped together with a single description field in the Gene table of the Absolute Mass Schema.

ProSight databases are typically constructed by drawing upon a rich data source such as UniProt [15]. UniProt stores information about potential sequence variation and post-translational modification in a per-site, rather than per-protein manner, and with the exception of alternative splice forms, does not describe which modifications are present simultaneously. In other words, a cSNP being noted on a protein means that it was observed in a certain number of individuals, but is not necessarily present all the time; a PTM being noted means that a given residue was observed to be modified in some cases, but not necessarily always. This means that in order to store sequences in the Absolute Mass Schema that encompass potential protein variation, one must store a sequence for every potential combination of potentially-observable sequence features, whose number scales exponentially with the number of sequence features. A novel database creation system was needed to tame this combinatorial explosion.

DBLoader 2.0 Overview

The database creation system ('DBLoader 1.0') was originally developed for ProSightPC 1.0 and then deployed to ProSightPTM 2.0 while the latter was under construction. DBLoader 1.0 was a monolithic program with a fixed set of functions. The original intent of the system was to generate semi-exhaustive combinatoric databases from rich UniProt [14] data under conditions of limited feature space; the feature set that it incorporated into sequences was limited to PTMs and a small set of cSNPs. Based on empirical evidence that numerous proteins were modified at the N-terminus (either by cleaving off the initial methionine, addition of an acetyl group, or both), DBLoader 1.0 assumed these sequence features to be present on all sequences, regardless of whether or not they were explicitly annotated in the source dataset. Handling of proteolysis was limited to an external pre-processing script that generated *in silico* digested files from input, which it then passed to DBLoader. If the number of sequence features on a protein exceeded what could be reasonably used to generate exhaustive combinatorics, a simple position ordering cutoff was applied, where all features in the sequence feature list past a given point were ignored.

As the number of annotated features in UniProt grew [Figure 2.6], it was clear that both a more sophisticated means of incorporating interesting sequence features, and a way of rejecting less-interesting features was necessary. Additionally, continued demand by experimentalists for new sequence processing stages was very difficult to meet with the existing monolithic database creation system. It was this rationale that led to the development of DBLoader 2.0, the database creation system that produced the databases that were released with ProSightPC 2.0 and were then deployed to ProSightPTM 2.0.

DBLoader 2.0 was designed as a modular pipeline [Figure 2.7], intended from the start to be modified and added to as time progressed. Instead of, *e.g.*, relying on an external pre-processing script to produce an *in-silico* enzymatically digested database, the new pipeline would be able to incorporate an *in-silico* digestion stage internally, capable of being deployed as-needed. By maximizing the potential features that could be implemented within the pipeline, as opposed to outside of it, the system could use internal state information to guide processing in later stages of the pipeline; this would be quite difficult to implement with external pre-processing scripts that needed to write output in a standard format, though as will be shown later, not impossible.

The DBLoader 2.0 pipeline was implemented in Perl [16], so as to leverage the considerable existing capability produced by the BioPerl project [17], such as parsers and abstract data types. The core of the pipeline is a set of three basic classes: MODULE, SEQ, and SEQFEATURE.

DBLoader Modules

A DBLoader pipeline is constructed of objects which implement the MODULE interface. A MODULE, at its core, is an entity that obeys a basic one-to-many input-output contract. It takes a SEQUENCE as input, and outputs zero or more SEQUENCES as output. A MODULE supports, to whatever degree feasible, lazy evaluation and generator semantics[1]. In other words, the MODULE interface provides a

1) The terms "lazy evaluation" and "generator semantics" appear often in this work. Lazy evaluation is a concept wherein computation is delayed until the last possible moment. In the context of this work, lazy evaluation specifically refers to the ability to define operations on the output of a previous processing stage without needing for all of that output to be present at the time that the next stage is defined. In lazy evaluation, a variable is not bound to data when it is defined, but only when its contents need to be read. This technique greatly reduces the memory footprint of a program, as results of one processing stage do not all need to be stored in memory before being supplied to the next processing stage. "Generator semantics" are a specific application of lazy evaluation, defined in

method – Execute – which will incrementally compute the next output for a given input, or return a null value indicating that no more output is pending. This, in turn, allows for a chain of modules to be strung together in a user-defined order and a set of fully-processed sequences produced from a limited set of inputs via a depth-first search along the data-flow graph. This construct is implemented by a pipeline utility class, which accepts a list of modules, and allows for generator semantics to be applied to an entire processing chain, whereby one can easily iterate over every output from a single input to the chain. The pipeline class is itself a module, enabling construction of processing chains possessing arbitrary complexity with minimal conceptual and computational overhead. Indeed, by leveraging lazy evaluation and generator semantics, the memory footprint of the pipeline was tightly controlled, a vital feature when dealing with a potential combinatoric explosion.

DBLoader Sequences and SeqFeatures

The processing pipeline operates on objects which describe a biological sequence along with its potential variations. The BioPerl library provided an abstract data type – Bio::Seq – which implemented much of this complexity. However, there were a number of design features necessary for the DBLoader pipeline that the existing data type lacked. Firstly, communications between processing stages needed to be kept in-band as much as possible, as this would permit ready parallelization if necessary, and would obviate the need for any external synchronization. Secondly, this information needed to be serializable to a standard format, in order to support external processing stages. Metadata stored by the pipeline needed to be transparent to external software, while being fully readable by the pipeline. Finally, any object passed down the pipeline needed to support efficient deep copying (clon-

this work as the application of an iterator-like interface to a lazily-evaluated data structure. In essence, it refers to the ability to define an iterator which produces a potentially-infinite output stream lazily, and then define operations on all of its output, without that output being present at that time. While possible to implement in Perl to an extent, the most accessible implementations of these concepts are currently present in Python and C#, where iterators employing lazy evaluation can be automatically created by using dedicated language syntax.

ing), in order to permit efficient forking in the pipeline. This last design requirement was a necessity due to the very heavily branched nature of the combinatoric data flow.

Nonetheless, the existing Bio::Seq class contained a number of useful features that made it useful as a basis for a more sophisticated class. It supported automatic construction from standard formats by existing parsers, as well as supporting lists of sequence features (via the Bio::SeqFeature class) that could be used to describe contiguous regions of the protein sequence. The existing class was extended into a new DBLoader::Root::Seq class, which retained all features of its parent, while adding support for the additional features necessary for the processing pipeline. Support for flexible inter-stage in-band communication was added by adding arbitrary 'flags' as a set of key/value pairs bound to the DBLoader::Root::Seq object. Because of Perl's flexible typing system, these flags could contain arbitrary information, ranging from simple strings and integers to objects of arbitrary complexity. These flags would be used to allow important details about the internal state of upstream processing modules to inform downstream sequence transformations. An example would be flags describing whether or not a given sequence contains the N- or C- terminus of a protein. If a sequence transformation such as *in-silico* digestion results in numerous peptide sequences from a single parent protein sequence, only those peptides that contain the parent's N-terminus would retain the N-terminal flag. These flags could be used to store any metadata generated during processing, or provided by the user during sequence input. The Bio::SeqFeature class was likewise extended into DBLoader::Root::SeqFeature, which supported analogous flags, but per each individual sequence feature.

Another type of in-band communication supported by the DBLoader::Root::Seq class is the transformation trace. The transformation trace is a simple list of 'notes' which describe the order and details of the transformations that resulted in a given DBLoader::Root::Seq. Every module was designed to write a descriptive note to the transformation trace of the sequences it generated, allowing for a detailed understanding of exactly how any given sequence that arrived at the end of the pipeline was produced.

In order to allow this information to be stored safely in standard formats, ignored by external parsers, but reconstituted by the pipeline if necessary, a serialization system needed to be implemented for DBLoader::Root::Seq and DBLoader::Root::SeqFeature. This was done by means of serializing the in-memory representations of the metadata fields into strings, and loading them into unstructured fields in the formats being written, analogously for both classes. Serializing a DBLoader::Root::Seq would, of necessity, serialize all DBLoader::Root::SeqFeatures that it contained. The final necessary feature of the biological sequence class, the capacity for deep copying, was implemented in a directly analogous fashion, via in-memory cloning of all constituent object fields using existing libraries.

DBLoader Pipeline Termini: Sources and Sinks

Once the basic features of the pipeline – the processing modules and the data they operated on – were defined, the general form of the processing pipeline took shape. The pipeline would accept a stream of biological sequences at its beginning, in the form of DBLoader::Root::Seq objects, apply various transformations to them, and output a stream of transformed biological sequences at the end. The nature of where that initial stream of sequences would arise from, and what the resulting output stream would be used for, is outside the context of the pipeline. Instead, a variety of 'sources' and 'sinks' could be placed at the start and end of the pipeline, and used for various tasks; in the case of DBLoader, this task would be to draw upon the information in a Fasta- or SwissProt-formatted flat file and construct a database obeying the ProSight Absolute Mass Schema.

The typical input source relied on the BioPerl library's parsers to construct Bio::Seq objects from a given file. Additional processing would then be carried out to convert these objects to DBLoader::Root::Seq objects; this took the form of removing unnecessary information, determining if additional metadata (either user- or program-supplied) was encoded in the file's unstructured fields, and deserializing this data should it be present. The input source could also be used as an initial filter for sequences or sequence features, restricting its output stream to sequences of interest by means of filters on arbitrary sequence and sequence feature fields. Like pipeline processing modules, the input source was written to use lazy evaluation and generator semantics; using it with the pipeline was as simple as writing a statement to the effect of 'for every sequence in input, apply pipeline processing'.

The output of the pipeline is similar to its input, a stream of objects representing sequences that are each the result of numerous consecutive transformations. This stream must then be loaded into the database schema. The output sink of the pipeline is thus what appears to be a processing module, but does not return output to the rest of the program. Instead, information in each sequence object is written to the database. Minimal sequence processing takes place in the output sink, only a final combinatorial expansion of selected post-translational modifications. This is largely a performance optimization; by placing this operation in the final sink, the number of potential sequence cloning operations is vastly reduced, increasing overall pipeline throughput. By keeping most sequence transformations out of the sink, the user-configurability of the pipeline is maximized. Most of the code in the sink is dedicated to database-specific operations, such as de-duplication, indexing, and compression. These features greatly improve the performance and footprint of the database, and will be described later. Like sources, a sink is completely independent of the processing occurring within the pipeline; this allows arbitrary sinks to be constructed for various database schemas or output formats.

Data Integration

Of particular interest during the initial development of DBLoader was the ability to reliably integrate additional information into the sequence stream that was not present in the original data file. Particularly, at the time UniProt was 'behind the curve' on integrating the thousands of newly-discovered coding single nucleotide polymorphisms (cSNPs) that were discovered in human genes by high-throughput projects such as HapMap [18]. Research efforts in the Kelleher Lab were focused at the time on 'proteotyping', the identification and quantitation of multiple genetic variants of a particular protein within single individuals [19] and thus a way of maximizing the number of variants considered during analysis was of high priority. Because processing modules are constructed independently of each other and communicate strictly in-band, it was possible to readily construct a module that queried mapping tables for possible cSNPs that may have existed on a given protein sequence, and carried out coordinate transformations to map them to the query sequence. First, a mapping table (provided by the iProClass project [20]) was used to map UniProt accession numbers to RefSeq [21] gene identifiers. The RefSeq identifier was then used to retrieve all applicable polymorphisms for that protein. However, as the polymorphisms were typically defined in reference to the RefSeq reference sequence, they were not guaranteed to match the canonical sequence provided by UniProt. As such, a coordinate transformation needed to be carried out by performing a global sequence alignment of the two sequences using the Needleman-Wunsch algorithm [22] implemented as part of the EMBOSS suite [23]. This mapping also allowed many cSNPs to be annotated with their observed heterozygosity values using the previously described feature flag system, permitting later inference and evaluation.

Another typical use of the processing pipeline's data integration capability was the assumption of N-terminal processing. Even if a given sequence did not encode N-terminal methionine cleavage or potential N-terminal acetylation or formylation, these modifications occur sufficiently frequently that it is useful to assume that they may occur for any protein. A module was constructed to add sequence features representing N-terminal acetylation and formylation, and another was constructed to add sequence features representing potential N-terminal methionine cleavage. Potential sequences were not generated at this point, but rather their possibility was noted; for ease of conceptual analysis and software engineering, it was decided to separate the pipeline informally into three broad conceptual stages: integration, complexity reduction, and combinatoric expansion [Figure 2.7]. In other words, first the pipeline adds all the additional information that it can, from whatever sources it has available; second, it determines which information is worth using; and finally, it generates a stream of sequences from all the potential combinations of sequence features.

Reducing Combinatorial Complexity

The reduction of biological complexity is at the heart of DBLoader's ability to deal with the ever-increasing amount of known information about protein variation and post-translational modification.

ProSight relies on a process whereby protein forms bearing all possible combinations of known post-translational modifications are constructed and loaded into a database for querying using observed data. This process has been termed *shotgun annotation*, in analogy to shotgun sequencing [24], as one cannot typically know in advance what protein forms exist in nature, only what sites could potentially be modified.

The nature of the problem as it originally presented itself is thus: a post-translational modification is either present on a particular residue of a protein when the protein is observed, or it is absent. If it is present, in the majority of cases, the residue cannot have any other modifications simultaneously present. If one considers a protein composed of k residues, and where n_i is the number of possible PTMs on the ith residue, then the total number of forms that a protein may take is:

$$N = \prod_{i=0}^{k} (n_i + 1) \qquad \textbf{Eq 2.1}$$

One can observe that ultimately, if the (slight) restriction of a residue bearing only a single PTM at a time is removed (or if all PTMs are on different residues), the number of protein forms resulting from an exhaustive enumeration of all possible combinations of PTMs scales as $O(2^N)$. If one considers human histone H3.1 (UniProtKB P68431, ver 66), an exhaustive enumeration of all combinations of known PTMs will result in 622,080,000 sequences. Consideration of potential N-terminal acetylation and N-terminal methionine cleavage (the former is not known to happen, while the latter is) raises the number of sequences to 2,488,320,000, a further fourfold increase. Considering that the lower bound on the size of an uncompressed Absolute Mass Schema database row describing one such sequence is 210 bytes, an uncompressed Absolute Mass Schema database containing only the possible forms of human histone H3.1 will take up at the very least 486 gigabytes, a currently untenable prospect. One may, perhaps, justify the creation of a fully-enumerated database for the targeted analysis of a single protein, but the use of unrestricted enumeration on the proteome scale is difficult to accept, even before one begins to consider sequence features other than PTMs.

In Top Down proteomics, one must consider the potential state that the observed protein will take as a whole. There are two relatively straightforward strategies for reducing the set of generated sequences produced during shotgun annotation without significantly reducing the ability of the resultant database to explain experimental data. The first is reasonably common in the Bottom Up world, and relies on reducing the total number of sequences by straightforwardly controlling the number of features that one would expect to be present on an observed protein simultaneously. In other words, one would consider all combinations of sequence features s taken t at a time, for all values of t from 0 up to a user-defined maximum value. This has the advantage of tight control over scaling; a closed-form solution exists for an upper bound on the number of total sequences generated, and one may tune the maximum value of t depending on one's space constraints and data-explanatory requirements. Certainly, any number of efficient algorithms exist for enumerating all such combinations [25]. However, this approach has a number of drawbacks. Firstly, it assumes that all sequence features known to exist on a protein have an equal probability of being observed; this assumption is blatantly false. Secondly, it assumes that equal amounts of detail are necessary for all locations on the sequence. This assumption is also often wrong; because lower-mass fragment ions are more likely to be observed than higher-mass fragment ions, and fragment ions contain sequences that are typically considered to be proper prefixes or suffixes of the intact sequence, a sequence feature is more likely to have fragment coverage the closer it is to the termini of a protein. Finally, this simple enumeration assumes that sequence features are independent, and ignores the restriction that typically, a residue may have only one feature at a time. This means that one will gain more 'explanatory power' (measured in the percentage of known features annotated) with less of an increase in the total number of generated sequences if one includes a set of sequence features all known to occur on the same residue than if one includes an equally-sized set of sequence features on different residues.

An alternative means of controlling the total number of generated sequences that that may arise during shotgun annotation is by restating the problem in terms of optimization. In this fashion, one can incorporate the additional information about and restrictions on particular sequence features, and generate a subset of known sequence features that maximizes the overall 'quality' of the sequence features while simultaneously minimizing the total number of generated sequences. One immediately runs into a significant challenge: because all the optimized variables are restricted to being 0 or 1, the overall problem immediately becomes a binary integer programming problem, one of Karp's original 21 NP-complete problems [26] and in fact NP-hard, at least as hard as the hardest problems in NP. If that wasn't enough, the fact that the constraint function describing the total number of sequences generated is exponential, not linear, makes this problem even more difficult to solve. However, in such a situation, one may take some inspiration from the data being analyzed. Protein sequences are the result of billions of years of evolution, undirected optimization under changing conditions. There exists an optimization technique that can provide approximate solutions to even a problem as difficult as this in bounded time: a genetic algorithm (GA).

A genetic algorithm is a global search heuristic that is implemented as a simulation of a population of candidate solutions. Candidate solutions are typically expressed as a bitstring, and usually referred to as 'chromosomes'. A fitness function is then defined that calculates each chromosome's overall 'quality'. If one can express a candidate solution as a chromosome, and can define a quality metric for it, one can use a genetic algorithm to optimize the candidate solutions. Although the term encompasses myriad variations, one can consider a 'typical' genetic algorithm to consist of three phases: initialization, selection, and reproduction. During initialization, the population is typically initialized to a set of randomly-generated chromosomes. Then, during the selection phase, some subset of the population (ranging from a small randomly-selected portion to the entire population) has its fitness evaluated by the fitness function, and some set of chromosomes (typically the best-scoring ones, though various selection techniques exist) is selected for reproduction. During the reproduction phase, the selected chromosomes have (again, typically) two biologically-inspired operators applied: crossover and mutation. During crossover, two 'parent' chromosomes have sections of themselves swapped with each other, forming two child chromosomes. Mutation randomly flips a random bit in a parent chromosome, forming a child chromosome. This process is repeated until a new child population, equal in size to the old parent population is formed. Afterwards, the selection and reproduction phases are repeated for some number of generations, making the resulting final population trend towards a maximization of the fitness function.

For the problem at hand, it is trivial to encode a potential solution into a bitstring chromosome: each position on the chromosome represents a specific sequence feature, while its value represents whether the sequence feature is selected or not. The fitness function is somewhat more involved, and takes the form of a preference function $p(\mathbf{x})$ weighted by an attenuation function $a(\mathbf{x})$:

$$f(\mathbf{x}) = a(\mathbf{x}) \cdot p(\mathbf{x}) \qquad \textbf{Eq 2.2}$$

The attenuation function, which encodes the constraint on the maximum number of sequences to be generated, takes the following form:

$$\begin{aligned} a(\mathbf{x}) &= e^{M(\mathbf{x})} \\ M(\mathbf{x}) &= \begin{cases} -\log(\mathit{TotalSeqs}(\mathbf{x}) - \mathit{MaxSeqs}), & \mathit{TotalSeqs}(\mathbf{x}) > \mathit{MaxSeqs} \\ 0, & \mathit{TotalSeqs}(\mathbf{x}) \leq \mathit{MaxSeqs} \end{cases} \end{aligned} \qquad \textbf{Eq 2.3}$$

The TotalSeqs function evaluates the candidate solution and returns the total number of sequences that may be generated by that particular set of active sequence features. One can observe that if the constraint is not exceeded, the attenuation function is equal to 1 and does not attenuate the preference function. However, if the constraint is exceeded, the attenuation function begins to penalize the preference function, at a rate of an order of magnitude of attenuation for every order of magnitude of excess sequences. The limit of the attenuation function as the number of excess sequences goes to infinity is 0; the function was constructed in this fashion in order to allow for informativeness even if the constraint is vastly exceeded. In a GA, as in many cases when making decisions based on value measurements, values of 0 are all equally uninformative. A GA shines when it can recombine partially-useful candidate solutions into good ones. A candidate solution whose fitness is 0 is judged to have no worth whatsoever, and that is rarely the correct decision to make. One always should give the algorithm some gradient to climb up.

The preference function, $p(\mathbf{x})$, encodes the quality-maximization objective and takes the following form:

$$
\begin{aligned}
p(\mathbf{x}) &= b_1 \sum_i h_1(x_i) + b_2 \sum_j h_2(x_j) \\
h_1(x) &= \begin{cases} 10^{1-\mathrm{tier}(x)}, & \mathrm{tier}(x) > 0 \\ 0, & \mathrm{tier}(x) \leq 0 \end{cases} \\
h_2(x) &= \frac{|S + 1 - [2 \bullet \mathrm{loc}(x)]|}{S - 1}
\end{aligned}
\qquad \textbf{Eq 2.4}
$$

Coefficients b_1 and b_2 are the respective weights of the two components of the preference function (set in practice to 0.5), S is the length of the sequence, and "loc" is the 1-indexed location of feature x on the sequence. "Tier" is the empirical integer-valued relative importance function which describes a PTM as "high importance" (1), "moderate importance" (2), "low importance" (3), or "unimportant" (-1). If the feature is a cSNP, its heterozygosity (if present) can be used instead of the tier function (undefined on cSNPs). Extension of the preference function is relatively simple; components need only obey the following rules:

$$
\begin{aligned}
&\{b_i \in \mathbb{R} : 0 \leq b_i \leq 1 \wedge \sum b_i = 1\} \\
&h : \{0, 1\} \rightarrow \{y \in \mathbb{R} : 0 \leq y \leq 1\}
\end{aligned}
\qquad \textbf{Eq 2.5}
$$

In other words, for the preference function to be valid, it must be a convex combination of component functions that map a binary digit (representing the state of a sequence feature) to a real number between 0 and 1. This makes the preference function itself have the range of real numbers between 0 and 1. Because the attenuation function is limited to this range as well, the overall fitness function is thus trivially a convex combination itself, on the range of 0 to 1.

Whenever optimization is needed, the algorithm is run for 1000 generations on a 100-individual population, using roulette selection, 0.95 probability of two-point crossover, and 0.05 probability of mutation. The sequence features that are not selected during the complexity reduction phase are marked as 'inactive' using the feature flag system; although this mark is a suggestion to later modules, not a mandate, these features are typically not used for combinatoric expansion in the next phase.

There are several major benefits to the optimization approach. First, it provides a framework by which multiple prioritization metrics can be applied simultaneously to sequence features. It is possible to use the preference function to effectively compare disparate sequence features that, while being radically different in nature, have the same potential effect on database size (*i.e.*, combinatoric explosion). If additional sequence features or metrics need to be considered in the future, it is straightforward to include their consideration into the preference function. Second, the fitness function evaluates the worth of a particular sequence feature relative to a given sequence and to its neighbors. When building a general-purpose proteomic database, if one has the choice, one would pick a commonly-encountered feature (*e.g.* acetylation) for shotgun annotation, rather than a rarely-encountered one (*e.g.*, lipidation). However, if there are no commonly-encountered features on a particular sequence, but there are several rare ones, one should certainly take the opportunity to annotate them; the cost is low, and the benefit may be high. Likewise, if a protein has many features at the termini, one would certainly annotate them first; however, if the only features a protein has are near its core, then that is what one should use. The optimization framework allows all these disparate concerns to be integrated into a single function, weighed, and a solution arrived at.

Sequence Transformations and Combinatoric Expansion

Once all possible information has been added to the sequence stream, and those sequence features unsuitable for expansion marked inactive, the task of generating all potential sequences may begin. There are, fundamentally, two types of sequence transformations that may occur: those that preserve the coordinate-system of a sequence and those that do not. We will define sequence transformations (and the sequence features that define them) that do not preserve the existing coordinate system as being *deforming* transformations, and those that do as *non-deforming*. The latter – PTMs and cSNPs – are all independent if on different residues, and are trivial to implement. The former, however, are more involved. Consider, for example, the following case. A protein is encoded by a gene made up of three exons. Alternative splicing produces two isoforms of the protein: one that contains all three exons, Isoform-1 and one that contains the first and third exon only, called Isoform-2 [Figure 2.8]. The N-terminus of the protein may be acetylated, while the region of the protein encoded by the third exon contains a residue which may be phosphorylated. The protein is specified in terms of the 'canonical sequence', chosen in a way that all sequence feature coordinates may be expressed relative to it – in this case, the clear choice for the canonical sequence is Isoform-1. In order to generate all possible combinations of exons (if desired), or just to consider all possible known isoforms, one must use an algorithm that keeps track of one's position on the canonical sequence and keep track of what coordinate maps may apply at any given moment. Additionally, the problem becomes much more complex if the restriction of all sequence features being initially specified relative to the canonical sequence is removed.

To address the problem of deforming transformations while preserving sequence features, one may consider a sequence generator to consist of a directed acyclic graph (DAG) of amino acid residues (nodes). Non-deforming sequence features would be defined as 'flags' attached to the residues, while deforming sequence features would be defined as additional paths (with added nodes if necessary) [Figure 2.9]. The set of all possible generable sequences would thus be the set of all paths between the set of nodes with no predecessors (potential starting points) and the set of nodes with no successors (potential ending points). If a non-deforming feature were defined as a range of residues, it could be handled either via splitting (marking each individual node with the feature, independent of all others), or defined as an entry/exit point (defining any path between those two points as annotated by the feature). Neither of these is a perfect solution; the first may introduce 'holes' in sequence features (such as an alpha helix being split into two pieces by an intercalated exon), while the second may result in sequence features spreading to the end of the sequence if their entry/exit point is disrupted (such as a binding region suddenly encompassing the entire sequence if its endpoint was in an exon that was removed). In all likelihood, neither approach would work in the general case, as annotating sequence features needs to be done by residue sequence and content, and sequence transformations may disrupt sequence features sufficiently that re-annotation using the original annotation algorithms is required.

Typically, one does not wish to enumerate *all* possible sequences resulting from alternative splicing. Instead, one wishes to enumerate known alternative splice forms, each resulting from applying a subset of coordinate-transforming features to the sequence at once. Within each of those, one would possibly wish to generate all possible sequence variants due to known mutations, expressed as either non-deforming or additional deforming transformations. One way that this may be implemented is to compile the information contained in a sequence with associated transformations into a program for a very simple virtual machine (VM). This VM would understand only a few instructions: DATA, JUMP, RTRN, NOOP, and HALT.

The VM as implemented in DBLoader is a two-stack machine, a very simple creature. It has a single, non-addressable register (the program counter, or PC). It has two stacks: a data stack and a return stack. The former is used for accumulating output, while the latter is used to allow the machine to recall where it jumped from. The VM simply executes the instruction at the memory address which is stored in the PC. When the VM is started, the PC is initialized to 0, beginning execution at the start of the program. If it encounters a DATA instruction, it pushes its operand (an amino acid residue with possibly some features attached) down onto the data stack and increments the program counter. If it encounters a NOOP instruction ("No-Op", no operation), it does nothing but increment the program counter. If it encounters a HALT instruction, the VM halts and outputs the contents of the data stack. The remaining two instructions are what allow a program to easily encode divergences from the usual path of execution; a JUMP instruction pushes the current value of PC+1 to the return stack and then sets the PC to its operand, while a RTRN instruction pops the last value from the return stack, adds its operand to it, and sets the PC to the result.

One can now observe how a sequence is generated by the VM-program. The JUMP and RTRN instructions act as divergence-points from the canonical sequence. The VM-program can, instead of going through a region of sequence, jump to an alternative sequence, then return to a spot just after the sequence it replaced. As the program executes, the data stack accumulates a sequence; when the VM halts, the resulting sequence is complete. However, as described, the VM-program will generate a single sequence, then terminate. An additional, external component makes it possible for the VM-program to generate multiple sequences. This component is not in the VM, but rather in the VM-program compiler, which is designed as a generator, as so many other elements in the pipeline are. When initialized with a sequence object, the compiler emits a stream of VM-program objects, each of which is executed by the VM, producing a sequence. The compiler maintains a table of deforming features, and the locations in the program where they are stored. It then simply replaces NOOP instructions with JUMP instructions as it generates VM-programs. Because the destinations of the jumps are fixed, no other modifications to the initial VM-program are necessary. All one needs to do is to determine which deforming features one needs to apply in a particular cycle of the compiler, and the resulting sequence is generated by the compiled VM-program.

Task-Dependent Custom Processing Modules

The flexibility of the DBLoader framework allows for processing modules to be easily developed for specific applications without needing to delve into the rest of the DBLoader codebase. For example, *in-silico* enzymatic digestion was easily supported as a processing module which generates peptide sequences given an input protein sequence, and allowed ProSightPC to be used for Middle Down and Bottom Up projects [27]. In another application of the framework, a custom module was developed to predict neuropeptides using the NeuroPred algorithm [14][28]. In the last major module to be added to the framework, signal peptide cleavage support for all alternative splice isoforms was implemented, enabling improved sequence identification based on nozzle-skimmer fragmentation of intact protein termini at high mass [29]. All of these applications are fully integrated into the processing pipeline, and support full information flow-through, allowing for accurate sequence prediction with minimal error.

Database Creation Challenges and Solutions

Once the task of determining how one will enumerate the set of sequences that one wishes to store in a database is complete, one must turn their attention to the database itself. ProSight uses a MySQL 5.0 database server as the back-end to Retriever. While the pipeline is entirely output-agnostic, merely outputting a sequence stream to an arbitrary data sink, one needs to consider the real-world performance issues that arise when constructing a MySQL 5.0 database. To do this, one first must evaluate the way that the resulting database is used, and the type of data likely to be loaded into the database.

First, one must realize that Absolute Mass Schema databases are WORM objects – once created, they are searched numerous times, but never modified (Write-Once, Read-Many). This means that the various aspects of database engines devoted to ACID compliance, row insertion and update performance are entirely useless for storing and querying an Absolute Mass Schema, and any performance tradeoff they applied to querying the database is a net detriment. This in turn means that the choice of database engine should be one that maximizes query performance at the cost of write performance and database integrity. In the case of MySQL 5.0, the fastest database engine for such applications is MyISAM.

Second, one must consider the types of queries issued by Retriever. The first type, the standard Absolute Mass Search, is a range query on a floating-point attribute (the intact mass). This query enumerates all rows whose value of the indexed attribute falls between some pair of endpoints. The second type, the Biomarker search, retrieves every sequence in the database that have a specific value for an integer attribute (the sequence type). This latter search is more expensive in terms of processing, so it is carried out less often than the bread-and-butter Absolute Mass Search.

Finally, one must consider the types of sequences that will be generated. Shotgun annotation generates vast numbers of very similar sequences, different only by a few characters. Additionally, it is often possible for an identical sequence to be produced in many different ways from different sources, especially considering potential database rot in the pipeline input source, and including various types of proteolytic processing in the pipeline.

These use-cases imply that any Absolute Mass Schema database must have a clustered index on the mass attribute, as this would maximize the performance of the most common query. However, the MyISAM database engine is a very simple engine optimized for speed, and does not support clustered indices. In order to create a MyISAM database with an (effectively) clustered index, a database must be first created, and then taken offline and the rows in its backing store physically re-ordered by an external program. This procedure is exactly what the Absolute Mass Schema sink module does in DBLoader.

Another important feature implied by these use-cases is database row compression. As shotgun annotation results in numerous very similar sequences being stored in the Absolute Mass Schema, considerable space-savings would be realized by row compression. MyISAM does support row compression, but with the caveats of it needing to be applied offline, and any resulting databases becoming immutable. As described previously, Absolute Mass Schema databases are WORM objects, making the latter restriction irrelevant. Like clustered index application, DBLoader supports row compression by default in the Absolute Mass Schema sink module.

Finally, the Absolute Mass Schema sink module addresses the problem of identical sequences being loaded into the database by using a stochastic hash-merge algorithm. Because the number of sequences being generated is likely very large, we cannot simply store them in memory and eliminate duplicates that way. Instead, a hash-merge algorithm relies on the principle that two identical sequences will, when a hash function is applied to them, result in the hash value. One then uses the remainder of this hash value divided by the desired number of 'buckets' to assign a file to which the sequence will be written. In this fashion, one may process the sequences being written to random buckets one at a time, and then only store the contents of a single bucket in memory when the bucket is read back in. If one chooses the number of buckets properly, the average bucket size will easily fit into memory, and furthermore, all members of any given set of identical sequences will be guaranteed to be found in the same bucket. This de-duplication step is carried out by default in the Absolute Mass Schema sink module, allowing for only a single instance of a given sequence to be stored in the database, with all possible ways that it could be generated noted in its description field. This feature reduces the size of Middle Down databases considerably [27], though it does provide some benefit to Top Down databases as well.

Implementing a clustered index and database compression improved Absolute Mass query performance by nearly two orders of magnitude over a non-clustered implementation, while shrinking database sizes by nearly 70%, making it feasible to use ProSightPC for high-throughput applications [19][27][30-32]. For instance, a dataset of several thousand scans searched against a human Top Down database built by the old methodology with an Absolute Mass search window of 2000 Da took weeks to complete [33], while an equivalent search using a database created by DBLoader 2.0 would take no more than hours. This degree of improvement permitted far wider search windows to be used; 50,000 Da windows and full-database searches are regularly used with modern Top Down searching [29]. Despite this improvement, however, concurrent increases in data-collection throughput demanded additional enhancements to existing infrastructure.

Retriever 2.0 Parallelization and ProSightCluster

The vast amount of data being generated by improved separation and data collection protocols necessitated a corresponding increase in throughput, above and beyond what existing implementations of the search engine could support, even with the aforementioned database back-end improvements. Thankfully, the problem of mass spectrometric searching is fundamentally one that is amendable to parallelization due to its separable nature. Consider a single spectrum being searched against a sequence database. In all modern mass spectrometric search engines, this search consists of some set of data points derived from that spectrum being used to score a set of sequences by means of some scoring function. There are at least as many scoring functions as there are search engines, but all of them are used as *metrics* to impose an ordering on the set of sequences relative to the query spectrum.

This ordering is then used by the search engine to return the top sequences, ones that were most likely (in some way) to produce the spectrum being analyzed. This process consists of independent steps that may be trivially performed concurrently; the act of scoring one sequence is entirely independent of scoring another sequence, and the act of querying one spectrum is (as far as the search engine is concerned) entirely independent of querying another spectrum. Such *embarrassingly parallel* problems are theoretically trivial to solve in a parallel fashion, limited only by engineering details.

Unfortunately, one cannot ignore engineering details in any real-world system, and parallelizing Retriever 2.0 posed a number of challenges. The nature of searching with ProSightPC is that an external search logic feeds experiments, one at a time, to Retriever 2.0 for searching [Figure 2.11]. Depending on the results of the search, the logic either carries out another search or stores the results in a database. Any parallelization framework would need to take this external search-management layer into account to be useful for existing search protocols. Additionally, the codebase of Retriever 2.0 has been under continuous maintenance for nearly seven years by a highly heterogeneous development team with high turnover. Combined with the nature of its design (a *very* highly side-effect-dependent pseudo-object-oriented monolithic codebase), this would make implementing parallelization within Retriever (by means of a threading library such as PTHREADS) a daunting task.

Instead, parallelization was implemented by means of multiprocessing and leveraging the first level of independence in search engine input, that of query independence. A Linux cluster was constructed utilizing the Rocks cluster-management framework [34], incorporating the Sun Grid Engine (SGE) job scheduling system [35]. This framework allows for batch scripts to be executed asynchronously on dynamically-managed computing resources. Batch scripts run in a consistent environment no matter where they were queued, an illusion maintained by accessing the filesystem via NFS [36]. This allows a single call to the search logic to be queued asynchronously by the job scheduling system, and run on any core of any machine in the cluster when resources are available. The search logic, once called, would execute Retriever as necessary, and when searching is completed, write the results. The search logic was modified slightly to support purely filesystem-based operations, replacing the final output to a database backend with a results file. As all queries are independent, each may be queued independently of others [Figure 2.12].

This parallel implementation of the search engine was used to support the high-throughput Top Down project. A web interface was developed enabling the user to submit a ProSight Upload Format (PUF) file containing their deconvoluted, deisotoped, averaged data and an XML file containing the search tree they wished to run on it to the cluster. The cluster would then split the PUF file into numerous PUF files, each containing an individual experiment, and queue a call to the search logic using the submitted search tree for every individual PUF file. Once each experiment was analyzed, the cluster would then merge the resulting PUF files (now containing search results) into a single output PUF file, copy it to the requesting user's home directory, and send them an email notifying them of job completion. By parallelizing the analysis across 48 cores, an entire 3D analysis run, consisting of 60 to 80 RPLC injections, could be analyzed in less than a day using a highly complex and wide-ranging search tree [29]. By combining this parallel implementation with the optimized databases described earlier in the chapter, new types of searching become feasible, such as full-database evaluation in situations where no observed precursor mass exists to limit the scope of the search.

The Future of ProSight

To a large degree, many of the challenges in implementing the features described earlier in the chapter were due to the legacy platform that they needed to be implemented on. In order to continue to develop and support Top Down proteomics in the future, a fresh start is needed to incorporate the lessons learned over the years with an eye towards continuous refinement and maintenance. The final section of this chapter will describe the presently ongoing efforts to design and implement an entirely new search engine, Retriever 3.0, and how it will fit into a new computational framework for Top Down proteomics.

One of the main challenges in implementing any new features with the Retriever 2.0 codebase was the design and implementation of the existing code. Retriever 2.0 was implemented in C++ with highly C-like semantics, with practically all instance methods being void. Essentially every method modified the data structures that its arguments pointed to, resulting in a tremendous number of side-effects; in fact, one may truthfully say that the Retriever 2.0 codebase worked because of side-effects, rather than despite of them. Most abstractions used in the code were highly leaky, leading to great difficulties in maintenance and adding additional features. Any replacement needed to be both fast (addressing a common concern with the existing codebase) and far more maintainable.

Initial work on Retriever 3.0 focused on addressing these issues by means of a streamlined, modern C++ implementation. After several months of work, a highly optimized, hand-tuned implementation of the core fragment-matching algorithm was discovered to be no faster than the existing implementation in Retriever 2.0, and nearly as opaque from a maintainer's perspective. However, during development of the prototype matcher, a curious property of the experimental codebase was noted: a Common Language Runtime (C# 3.0) wrapper written to permit simple crossplatform I/O was observed to be considerably faster than a (laboriously-written) native C++ implementation. Following a brief series of experiments with several CLR compilers on multiple platforms, it was conjenctured that compilers and libraries for the CLR had matured sufficiently that a pure managed-code implementation of the search engine would be sufficiently efficient and far more straightforward to extend and maintain. This decision led to a plethora of highly useful design features and implementation aspects that continue to benefit the project to this day.

Retriever 3.0 was designed to incorporate two fundamental design principles that were developed over the last five years of work. The first was lazy execution generator semantics, described earlier in this chapter; the second was the Automated Protein Characterization schema (APC), the novel database schema designed by my predecessor [30]. One fundamental structural problem with shotgun annotation as it is implemented by the Absolute Mass Schema is its static nature and the combinatorial explosion that it is, in truth, unsuitable for containing. As the number of features contained in a database grows, the total database size explodes [Figure 2.13]. Even with the GA-based optimizer described previously, one is fundamentally limited by the nature of the schema; one has to store as much data as possible, and the way that the data is stored (as already-applied combinations of features) is very highly inefficient. APC resolves this problem by annotating all features independently, rather than as sequences presenting combinations of features, permitting the search engine to implement optimizations that greatly reduce matching overhead. Unfortunately, previous attempts at implementing APC to replace the AMS were hampered by the great difficulty of maintaining the existing codebase. By incorporating the generator semantics used to such great effect in DBLoader 2.0, however, it is possible to utilize APC to its fullest potential.

A fundamental feature of modern C# (2.0 and above) is the ability to trivially construct generators. All that is necessary is to declare a method that returns an object which implements the IEnumerable (iterator) interface, and uses a specialized yield return statement to return a value. The compiler then automatically constructs the 'plumbing' that makes the returned object a generator. Every time the returned iterator is called, the underlying method is executed until it yield returns; the value is returned by the iterator, and execution of the underlying method pauses at the yield return statement. Later calls to the iterator resume execution of the underlying method where it left off the last time, until no more data is forthcoming. In this fashion, the custom implementation of one-to-many, lazily executed generators developed for DBLoader 2.0 can be replaced with a single, simple method. The resulting iterators may be operated on by Language-Integrated Query (LINQ) extension methods, trivially iterated on by built-in language constructs, processed through lambda functions, and trivially (and automatically!) parallelized by parallel implementations of LINQ (PLINQ). Thus, the complex sequence transformations that DBLoader 2.0 implemented using custom interface can now be accomplished via a simple nested chain of generators in Retriever 3.0, each taking input from the last. Retriever 3.0 enables the user to determine the degree of combinatorial complexity they desire at runtime by constructing a chain of generators, and supports matcher optimizations that take into account different types of sequence features.

APC fundamentally distinguished between transformations that modified the mass of a single residue (‘Monomer Mass Changes’ or MMCs) and those that modified multiple residues (‘Polymer Mass Changes’, PMCs) [30]. The difference between these is not the mass shift, but rather the co-ordinate shift. Anything that does not affect the coordinate system by which sequence features are mapped to a sequence can be treated purely arithmetically inside the matcher. Retriever 3.0 does so, doing simple mathematical operations on pre-allocated arrays, saving considerable cycles by not re-allocating them and using vectorized arithmetic – something that modern superscalar processors excel at. By avoiding checking observed and theoretical fragments unnecessarily, Retriever 3.0 can check for N-terminal methionine cleavage and N-terminal acetylation for all sequences, an operation that would take four times as long under the old paradigm, in only 35% more time than matching the unmodified sequence. Transformations which do modify the coordinate system by changing the length of the underlying sequence are handled via generators which return transformed ‘views’ of the parent sequence (analagous to views of parent tables in relational databases). These are lightweight objects defined as transformations of parent objects, but implementing the same ISequence interface as the parent. This enables them to be used anywhere where the parent could be, and their transformations are only applied when they would affect something downstream, ensuring that paying the cost of evaluation is only carried out when absolutely necessary.

Retriever 3.0 is thus designed as a stream of sequence objects being processed by the matcher. Any transformation that modifies sequence coordinates, such as alternative splicing or enzymatic digestion, is handled via nested generators, resulting in lazily-evaluated, lightweight objects interchangable with any other sequence object as far as the matcher is concerned. As shown, such a design is fast, has a low memory footprint, is trivially parallelizable, is easily maintainable and extensible, and may be used as a component in larger projects. Implementation of the full design is currently underway, but existing prototypes have performed very well in tests. The core Retriever 3.0 matching engine, when compared with Retriever 2.0 on the same dataset and queries, exhibited a substantial performance improvement, without factoring in any paralellization.

The codebase presently under development will be deployed in a number of ways. It will form the core of ProSightPTM 3.0, the next generation of the web-based public search engine. The same cluster implementation will support job submission not only via the web interface, but also from standalone systems such as a future ProSightPC 3.0. A Proteome Discoverer [37] node implementation is also under development, both in local and offsite-cluster job-submission variants (ProSightPD).

Figures

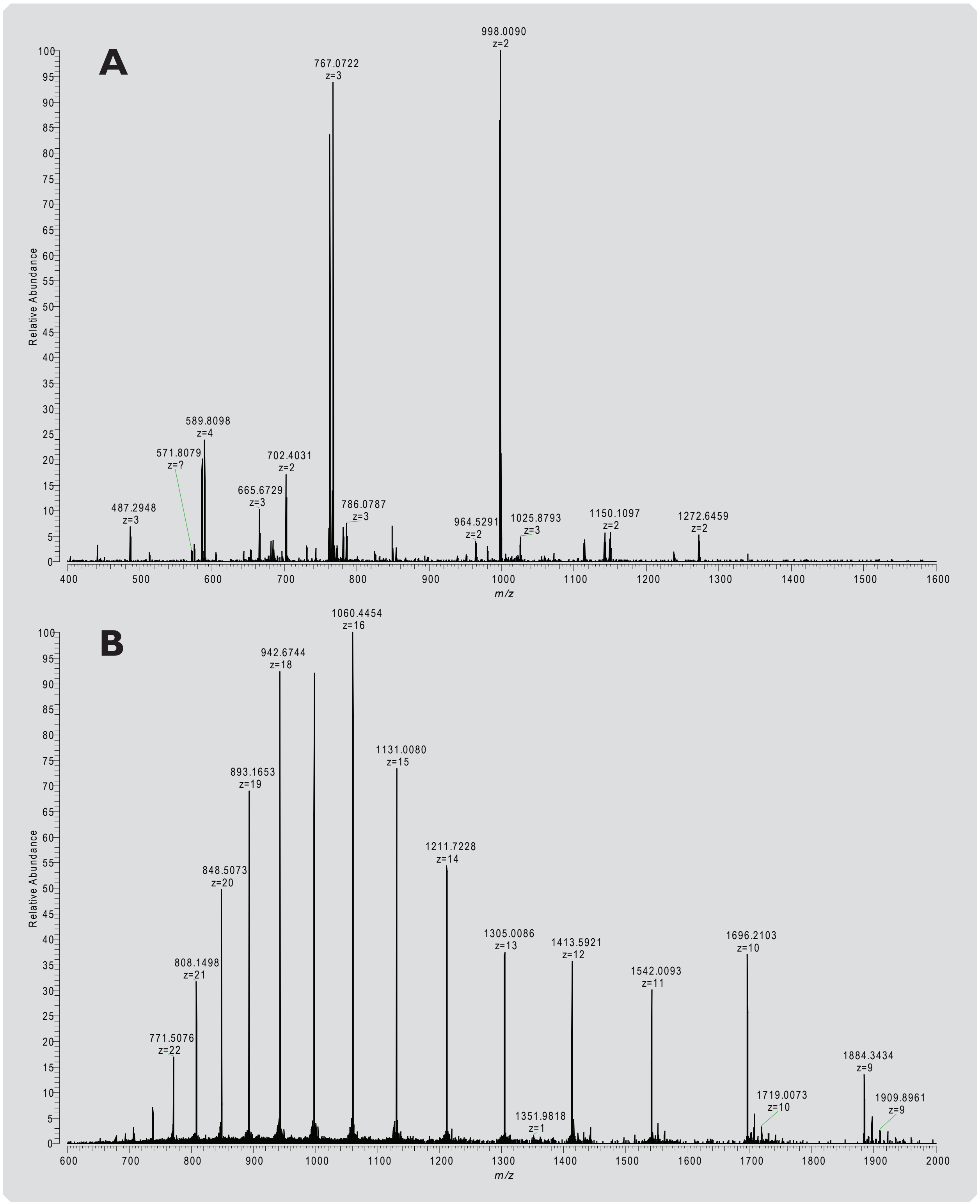


Figure 2.1
Typical Bottom Up (A) and Top Down (B) high-resolution survey scan mass spectra.

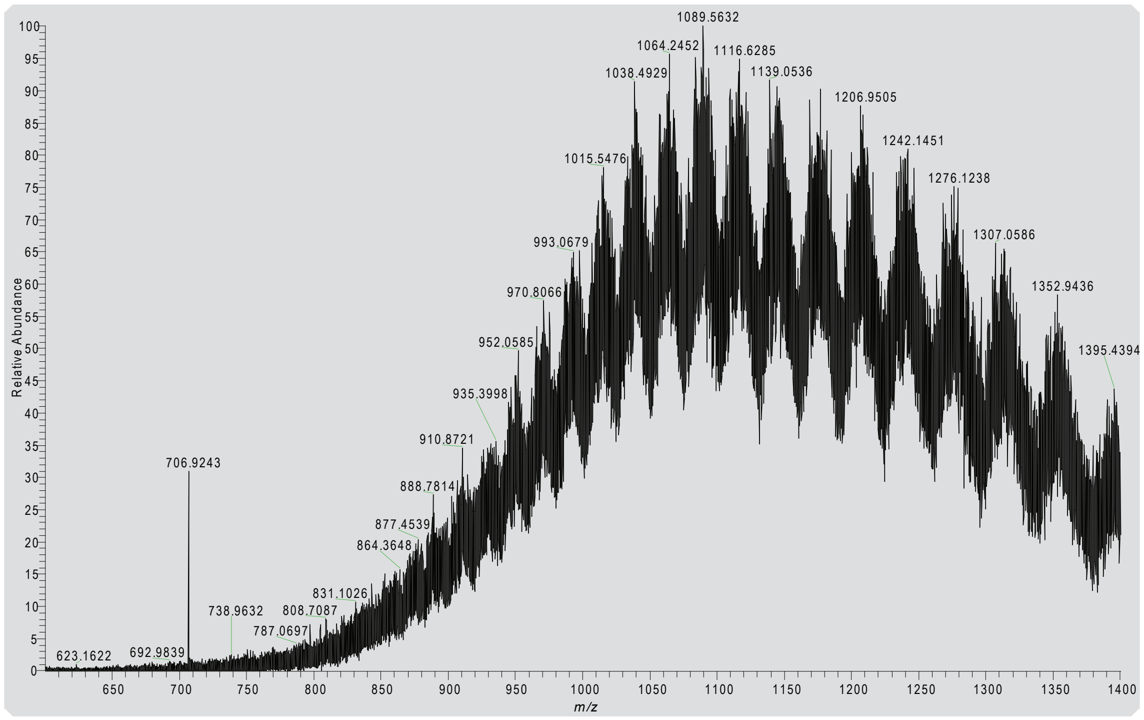


Figure 2.2
A highly-convoluted Top Down survey scan mass spectrum, exhibiting the characteristic high-mass "hump".

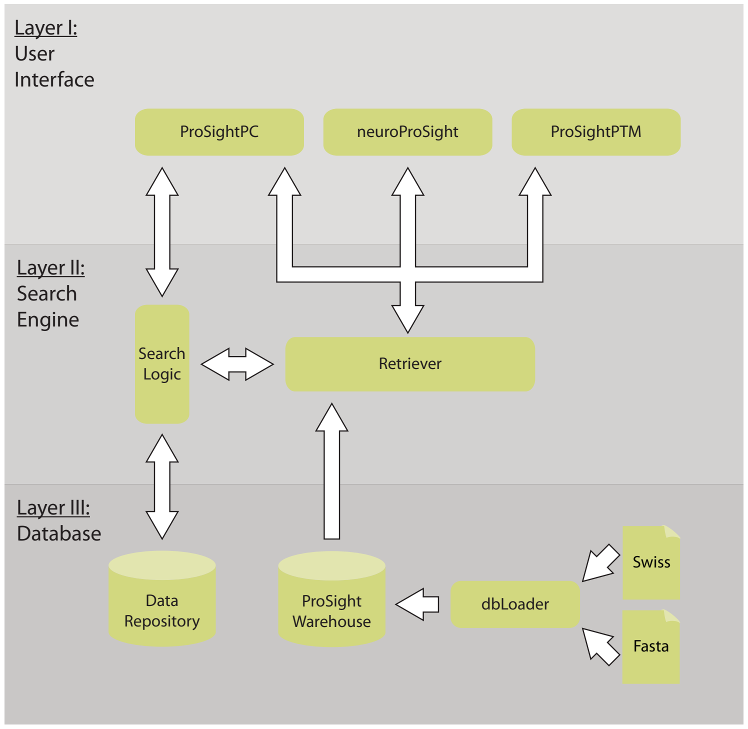


Figure 2.3
A diagram of the ProSight family of tools. All user interfaces rely on the same search technology, differing in how they interact with it.

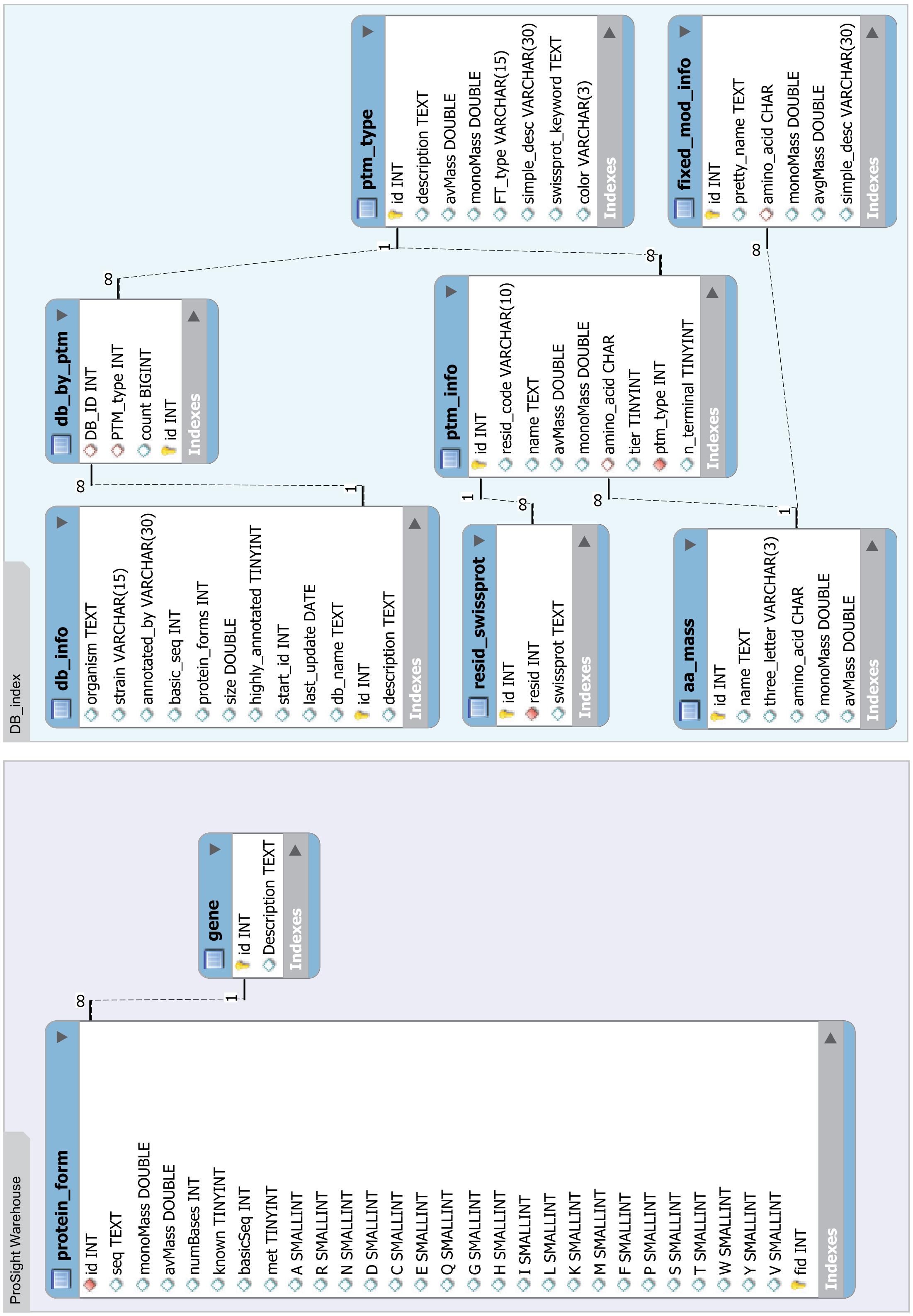
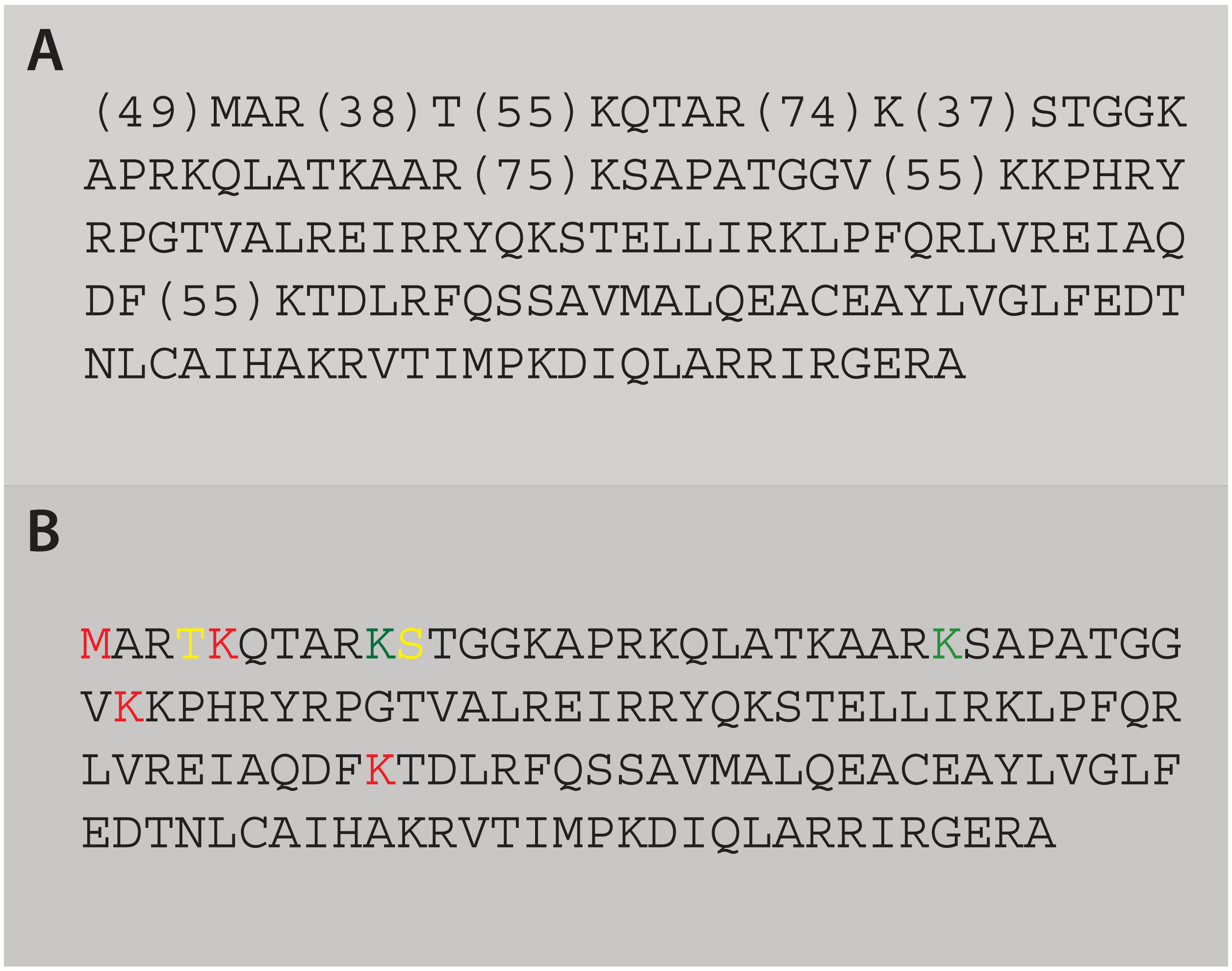


Figure 2.4

Entity-Relationship diagram of the Absolute Mass Schema. Individual ProSight Warehouses are stored as distinct MyISAM databases, with common information and metadata for all ProSight Warehouses stored in a separate 'DB_index' database.



A

```
(49)MAR(38)T(55)KQTAR(74)K(37)STGGK
APRKQLATKAAR(75)KSAPATGGV(55)KKPHRY
RPGTVALREIRRYQKSTELLIRKLPFQRLVREIAQ
DF(55)KTDLRFQSSAVMALQEACEAYLVGLFEDT
NLCAIHAKRVTIMPKDIQLARRIRGERA
```

B

```
MARTKQTARKSTGGKAPRKQLATKAARKSAPATGG
VKKPHRYRPGTVALREIRRYQKSTELLIRKLPFQR
LVREIAQDFKTDLRFQSSAVMALQEACEAYLVGLF
EDTNLCAIHAKRVTIMPKDIQLARRIRGERA
```

Figure 2.5

Example of post-translational modifications embedded in a sequence in prefix notation (A) and as color-codes (B). This particular form of Histone H3.2 has the following modifications: 1MAc, T4P, K5Ac, K10Me3, S11P, K28Me3, K37Ac, and K80Ac. The prefix codes are references to mass shift tables in the DB_index metadata schema.

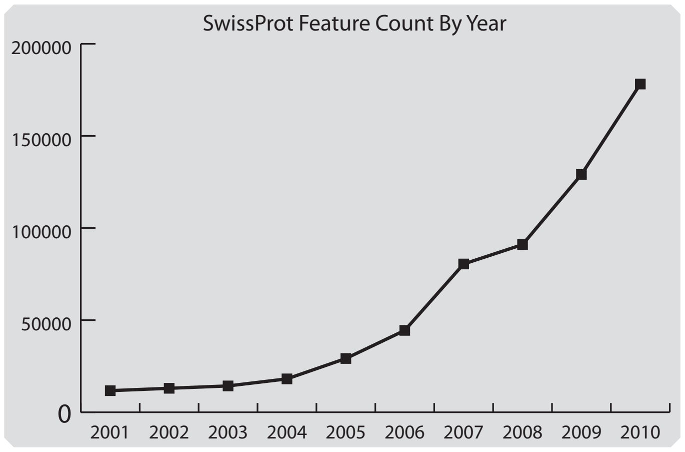


Figure 2.6
Number of features representing post-translational modifications in SwissProt, the manually-annotated component of UniProt. Growth in the number of protein forms is exponential relative to the number of sequence features on a given protein.

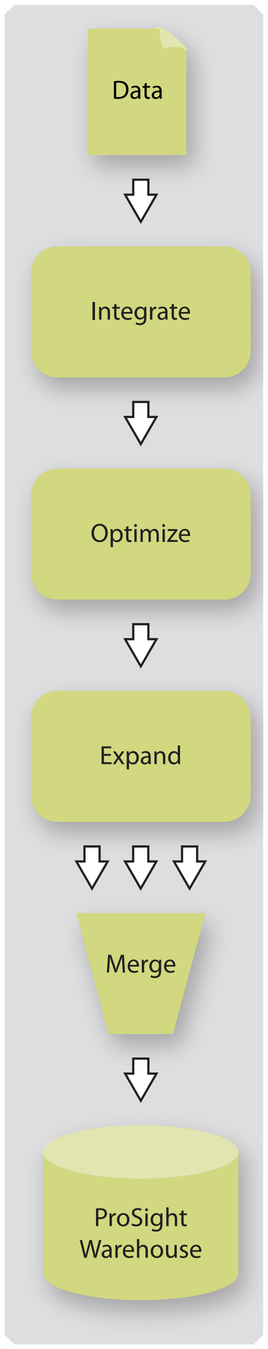


Figure 2.7
DBLoader abstract pipeline.

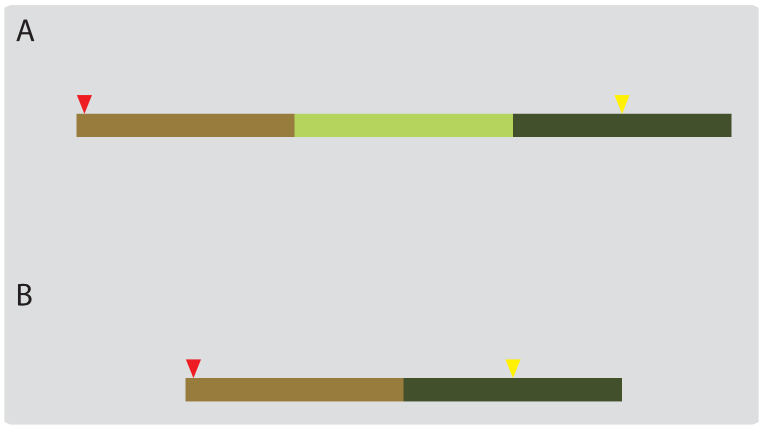


Figure 2.8
An example of a deforming transformation. Isoform-2 (B) differs from Isoform-1 (A) by the removal of the middle exon. This deforms the coordinate system, requiring a remapping of all feature coordinates past the point of deformation, such as the location of the phosphorylation in the third exon.

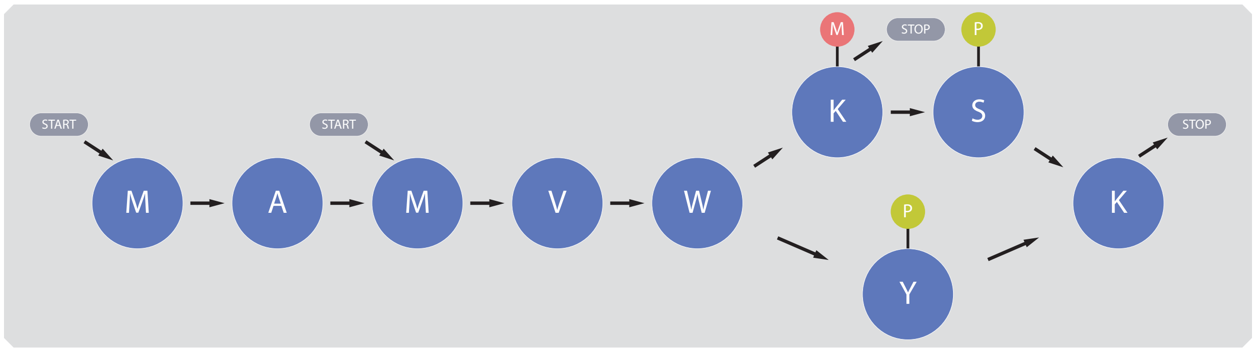


Figure 2.9
An example of a directed acyclic graph (DAG) representation of the variability of a single polypeptide. A polypeptide sequence is represented by a path from a start node to a stop node, while potential PTMs are represented as flags on individual amino acid nodes. The set of all flag combinations on all paths through the DAG is thus the set of all possible observable forms that this polypeptide may occur in.

```
----M-V-W-K----

----M-V-W-K----

----M-V-W-K-S-K

----M-V-W-K-S-K

----M-V-W-K-S-K

----M-V-W-K-S-K

----M-V-W---Y-K

----M-V-W---Y-K

M-A-M-V-W-K----

M-A-M-V-W-K----

M-A-M-V-W-K-S-K

M-A-M-V-W-K-S-K

M-A-M-V-W-K-S-K

M-A-M-V-W-K-S-K

M-A-M-V-W---Y-K

M-A-M-V-W---Y-K
```

Figure 2.10
All observable forms that may result from Figure 2-9.

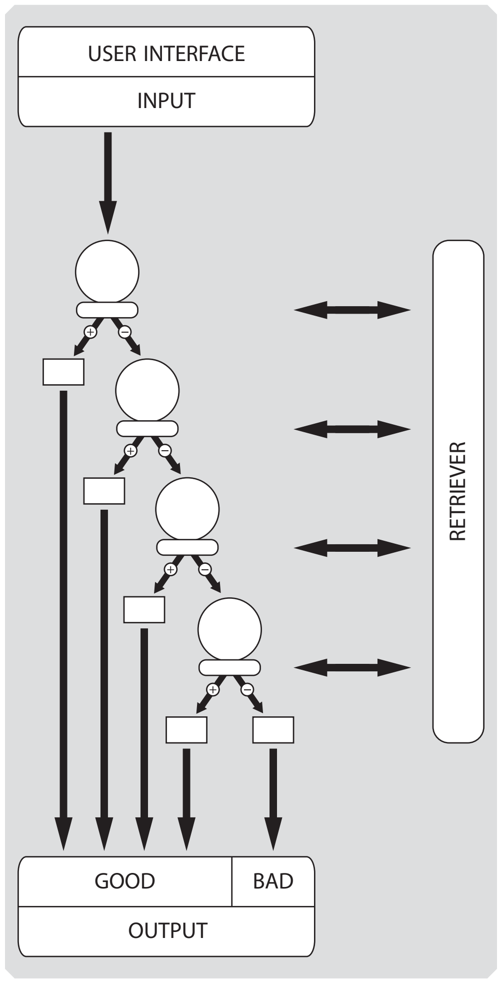


Figure 2.11
ProSightPC data flow diagram.

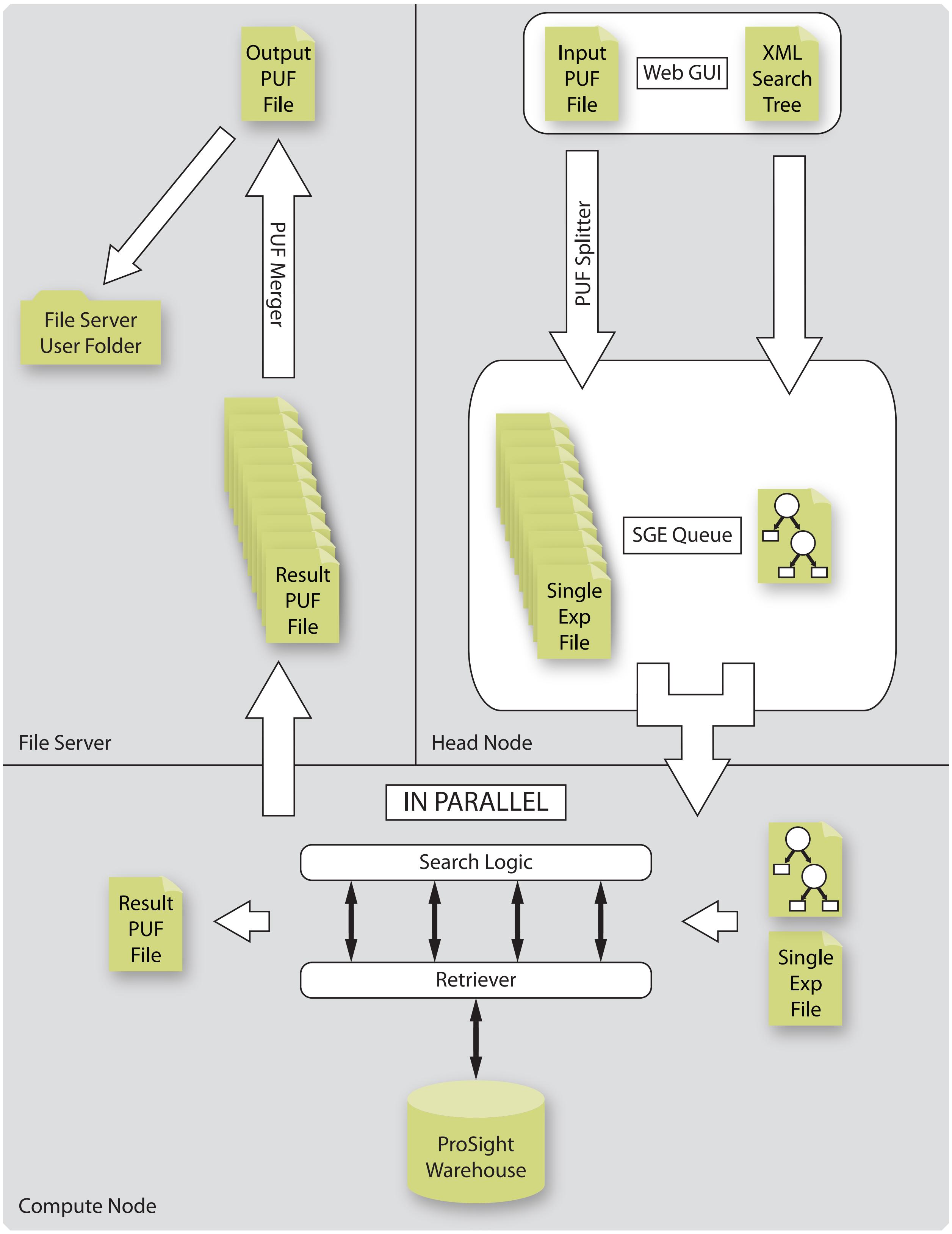


Figure 2.12
ProSightCluster data flow diagram. Users submit a PUF file and a search tree via the web interface, which is then split into individual experiments, processed in parallel, and results are delivered to their home directory.

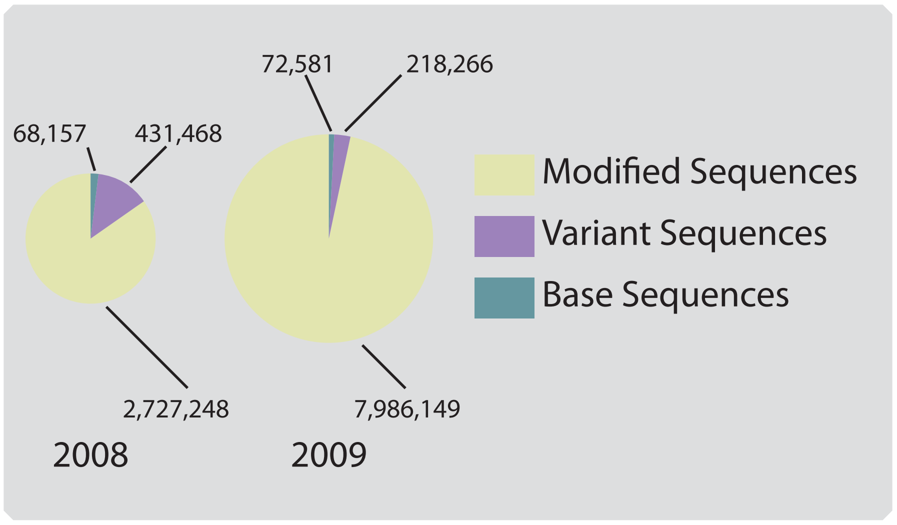


Figure 2.13
Human Top Down Absolute Mass Schema size for 2008 and 2009. Due to exponential scaling, a doubling of the number of annotated features caused a nearly fourfold increase in stored forms, even after optimization.

References

1. Mørtz, E., et al., *Sequence tag identification of intact proteins by matching tanden mass spectral data against sequence data bases.* Proceedings of the National Academy of Sciences of the United States of America, 1996. **93**(16): p. 8264-8267.

2. Beavis, R.C., *Chemical mass of carbon in proteins.* Analytical Chemistry, 1993. **65**(4): p. 496-497.

3. Horn, D.M., R.A. Zubarev, and F.W. McLafferty, *Automated reduction and interpretation of high resolution electrospray mass spectra of large molecules.* Journal of the American Society for Mass Spectrometry, 2000. **11**(4): p. 320-332.

4. Thermo Fisher Scientific, *Xtract.*

5. Fenn, J.B., et al., *Electrospray ionization for mass spectrometry of large biomolecules.* Science, 1989. **246**(4926): p. 64-71.

6. Durbin, K.R., et al., *Intact Mass Detection, Interpretation, and Visualization to Automate Top Down Proteomics on a Large Scale.* Proteomics, 2010. Accepted.

7. Zamdborg, L., et al., *ProSight PTM 2.0: improved protein identification and characterization for top down mass spectrometry.* Nucleic Acids Research, 2007.

8. Karabacak, N.M., et al., *Sensitive and Specific Identification of Wild Type and Variant Proteins from 8 to 669 kDa Using Top-down Mass Spectrometry.* Molecular & Cellular Proteomics, 2009. **8**(4): p. 846-856.

9. Tsai, Y.S., et al., *Precursor Ion Independent Algorithm for Top-Down Shotgun Proteomics.* Journal of the American Society for Mass Spectrometry, 2009. **20**(11): p. 2154-2166.

10. Geer, L.Y., et al., *Open Mass Spectrometry Search Algorithm.* Journal of Proteome Research, 2004. **3**(5): p. 958-964.

11. Meng, F., et al., *Informatics and multiplexing of intact protein identification in bacteria and the archaea.* Nat Biotech, 2001. **19**(10): p. 952-957.

12. LeDuc, R.D., et al., *ProSight PTM: an integrated environment for protein identification and characterization by top-down mass spectrometry.* Nucl. Acids Res., 2004. **32**(suppl_2): p. W340-345.

13. Bora, A., et al., *Neuropeptidomics of the Supraoptic Rat Nucleus.* Journal of Proteome Research, 2008. **7**(11): p. 4992-5003.

14. Southey, B.R., et al., *NeuroPred: a tool to predict cleavage sites in neuropeptide precursors and provide the masses of the resulting peptides.* Nucl. Acids Res., 2006. **34**(suppl_2): p. W267-272.

15. Apweiler, R., et al., *UniProt: the Universal Protein knowledgebase.* Nucl. Acids Res., 2004. **32**(suppl_1): p. D115-119.

16. Wall, L. and M. Loukides, *Programming Perl.* 2000: O'Reilly & Associates, Inc. Sebastopol, CA, USA.

17. Stajich, J., et al., *The Bioperl toolkit: Perl modules for the life sciences.* Genome research, 2002. **12**(10): p. 1611.

18. The International HapMap Consortium, *The International HapMap Project.* Nature, 2003. **426**(6968): p. 789-796.

19. Roth, M.J., et al., "Proteotyping": Population Proteomics of Human Leukocytes Using Top Down Mass Spectrometry. Analytical Chemistry, 2008. **80**(8): p. 2857-2866.

20. Wu, C.H., et al., *iProClass: an integrated, comprehensive and annotated protein classification database.* Nucl. Acids Res., 2001. **29**(1): p. 52-54.

21. Pruitt, K.D. and D.R. Maglott, *RefSeq and LocusLink: NCBI gene-centered resources.* Nucl. Acids Res., 2001. **29**(1): p. 137-140.

22. Needleman, S.B. and C.D. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins.* Journal of Molecular Biology, 1970. **48**(3): p. 443-453.

23. Rice, P., I. Longden, and A. Bleasby, *EMBOSS: The European Molecular Biology Open Software Suite.* Trends in Genetics, 2000. **16**(6): p. 276-277.

24. Pesavento, J.J., et al., *Shotgun Annotation of Histone Modifications: A New Approach for Streamlined Characterization of Proteins by Top Down Mass Spectrometry.* Journal of the American Chemical Society, 2004. **126**(11): p. 3386-3387.

25. Knuth, D.E., *The Art of Computer Programming.* 2005, Addison-Wesley Professional.

26. Karp, R.M., *Reducibility among combinatorial problems.* Complexity of computer computations, 1972. **43**: p. 85-103.

27. Boyne, M.T., et al., *Tandem Mass Spectrometry with Ultrahigh Mass Accuracy Clarifies Peptide Identification by Database Retrieval.* Journal of Proteome Research, 2008. **8**(1): p. 374-379.

28. Lee, J.E., et al., *Endogenous Peptide Discovery of the Rat Circadian Clock.* Molecular & Cellular Proteomics, 2010. **9**(2): p. 285-297.

29. Tran, J.C., et al., *Efficient Mapping of the Endogenous Human Proteome by Top Down Mass Spectrometry.* In preparation.

30. LeDuc, R.D., *Bioinformatics of High Throughput Proteomics Using Tandem Mass Spectrometry of Intact Proteins*, in *Department of Crop Sciences*. 2007, University of Illinois at Urbana-Champaign: Urbana. p. 143.

31. Ferguson, J.T., et al., *Top-Down Proteomics Reveals Novel Protein Forms Expressed in Methanosarcina acetivorans.* Journal of the American Society for Mass Spectrometry, 2009. **20**(9): p. 1743-1750.

32. Wenger, C.D., et al., *Versatile Online−Offline Engine for Automated Acquisition of High-Resolution Tandem Mass Spectra.* Analytical Chemistry, 2008. **80**(21): p. 8055-8063.

33. Roth, M.J., et al., *Precise and Parallel Characterization of Coding Polymorphisms, Alternative Splicing, and Modifications in Human Proteins by Mass Spectrometry.* Molecular & Cellular Proteomics, 2005. **4**(7): p. 1002-1008.

34. Papadopoulos, P.M., M.J. Katz, and G. Bruno, *NPACI Rocks: Tools and techniques for easily deploying manageable linux clusters.* Concurrency and Computation: Practice & Experience, 2003. **15**(7): p. 707-725.

35. Gentzsch, W. *Sun Grid Engine: Towards Creating a Compute Power Grid.* in *First IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01).* 2001. Brisbane, Australia.

36. Internet Engineering Task Force, *RFC 1813: NFS Version 3 Protocol Specification*. 1995.

37. Thermo Fisher Scientific, *Proteome Discoverer 1.1*. 2010.

CHAPTER 3: De-Novo Peptide Sequencing By Means Of High-Resolution Mass Spectrometry

This work was presented, in part, as "Automated Protein Characterization by High-Resolution Mass Spectrometry", a poster at the 2009 ASCI/AAP Joint Meeting. Elements of the genetic algorithm were implemented in collaboration with Lukasz Strzelecki.

Genetic algorithms have applications in proteomics beyond feature selection during database creation. In particular, it is possible to apply them to the problem of *de novo* sequencing using highly accurate mass spectrometry. Currently, the basic method of automated *de novo* sequencing, such as that used by ProSight, is a simple graph-theoretic algorithm [1-2]. The set of mass differences between all peaks in a deconvoluted and deisotoped spectrum is computed, and pairs of peaks separated by the mass of an amino acid are identified. This data is entered into a graph where the vertices are peaks, and the edges connecting them are amino acid masses. The set of paths through this graph, then, become the set of possible *de novo* sequences that can arise from the deconvoluted, deisotoped spectrum.

However, this approach has a number of problems. It does not attempt to characterize the sequence; nor does it quantify the quality of the resulting sequence tags; nor does it use any additional information to improve the sequence solution. It is possible to improve this simple sequence tag compiler by incorporating some techniques from sequencing tools for Hi-Lo data, such as PepNovo [3] and PEAKS [4] which incorporate sophisticated probabilistic models for amino acid composition and arrangement. However, because they are designed for Hi-Lo data, they are limited in their characterization power due to the inherent degeneracy of their fragmentation. Due to the high data locality and straightforwardness of encoding of GA-based protein sequence representations, it should be possible to use genetic algorithms to construct a novel Hi-Hi *de novo* sequencing tool which can use numerous pieces of information to arrive at a better sequence solution than previously possible. To investigate this, a prototype system for GA-based *de novo* sequencing using high-resolution data was developed, codenamed 'Refiner'.

Input and Data Filtering

Refiner accepts as input data consisting of a single neutral monoisotopic precursor mass and a set of neutral monoisotopic fragment masses. The system, as with most data processing systems detailed in this work, is composed of a number of distinct components which process data in sequence [Figure 3.1]. The first component, run prior to any analysis step, is a noise reduction filter. Unlike typical noise reduction filters, which filter by S/N ratio or local intensity, this prototype filter was designed specifically to remove peaks that, while valid, would potentially skew the results of the genetic algorithm. The filter eliminates fragments that were essentially identical (i.e., sets of fragments within a set tolerance of each other would be reduced to a single value), thereby eliminating the possibility that a single theoretical fragment could match multiple observed fragments. Additionally, the filter eliminates known neutral losses and adducts, i.e., peaks at known fixed offsets from other peaks. Water losses, ammonia losses, metal ions, non-standard fragments (e.g., a-ions) and the like can be easily removed from the peak list in this fashion. The precursor mass is also used for filtering. Nonsense fragments (i.e., fragment masses that exceed the mass of the precursor) are removed, as well as neutral loss chains from the precursor (a remarkably common occurrence). If desired, the filter can even infer additional data; if one assumes that the remainder of the fragment ions are real, one may locate complementary ion pairs, and infer the ghost ions for those peaks that are missing their partner, a process made possible only due to the high-accuracy measurement of both precursor and fragment ions. The resulting filtered peak list is then returned to Refiner for further processing and analysis.

Sequence Tag Compilation and Initial Conditions

Once the data is filtered, the next step prior to the main computation loop is determining initial conditions. A genetic algorithm does not start from an empty population of candidate solutions; typically, a random set of chromosomes is generated prior to the first evolution step. However, this is a somewhat naïve approach when additional information is present. Refiner was designed to be capable of characterizing putatively identified sequences. In other words, given a sequence solution which is partially correct, Refiner can manipulate it to fit the data better. To facilitate ease of testing and use, Refiner was built to use existing ProSight (PUF) files, which contain both data and prior analyses. When called on a particular ProSight experiment, it will use the data contained in that experiment for its own purposes, and can set the initial conditions to the results of some particular search. This constrains the overall search space and allows the process of evolution to ‘fill in the gaps’.

Another way by which Refiner populates initial conditions is by the compilation of sequence tags. While limited, the usual graph-theoretic algorithm for *de novo* sequencing is quite useful for coming up with an initial population of candidate solutions. A new implementation of this algorithm was developed for use within Refiner, extending the feature set found in the existing ProSight sequence tag compiler. Firstly, the new algorithm used an extended table of potential mass shifts, supporting not just single-residue mass shifts but dipeptidyl shifts as well. Because Refiner was developed for high-resolution fragmentation data, it can leverage the vastly improved mass accuracy provided to greatly reduce the degeneracy of the potential mass shift space, making dipeptidyl mass shifts feasible. Unfortunately, experiments with tripeptidyl and above mass shifts did not result in useful sequence solutions. The second improvement in the new sequence tag algorithm was the use of an explicit graph-theoretic formalism, rather than the implicit one used previously. If a mass shift was matched, the two peaks between which the mass shift was observed were added to the result graph as vertices, and an edge was created between them, annotated with the residue or oligopeptide that matched to the mass shift. Additionally, a numeric weight was assigned to the edge; if the match was to the mass of a single amino acid residue, the weight was -1.0, while dipeptidyl matches were assigned a weight of -0.5. The reasoning behind these negative edge weights merits some discussion. In the graph-theoretic formulation of the *de novo* sequencing problem, one needs to traverse the matching mass shift graph, where each edge corresponds to a known mass shift, and a path consisting of these mass shifts is a potential sequence solution. Since one does not wish to include proper subsequences in the output of the algorithm, one would naturally seek to enumerate the longest paths in the graph, between the set of vertices with no inbound paths (starting points) and the set of vertices with no outbound paths (ending points). This problem does not occur nearly as often in graph theory as its inverse, locating the *shortest* paths, which is straightforwardly and efficiently solvable by two classic algorithms, Dijkstra's Algorithm [5] and the Bellman-Ford Algorithm [6]. While both of these algorithms are taught routinely to undergraduates in computer science, typically all will use Dijkstra's Algorithm due to its higher efficiency. The only major advantage typically cited for the Bellman-Ford Algorithm is its ability to use negative edge weights, a capability that is rarely used. In this case, however, negative edge weights are a boon, because by using only negative edge weights, one can transform an algorithm that finds the *shortest* path into an algorithm that finds the *longest* path. The path between two vertices that has the "smallest" (i.e., most negative) aggregate weight is thus the "longest". The reason that two distinct edge weights (-1.0 and -0.5) were used was due to the use of dipeptidyl mass shifts. A dipeptidyl mass shift contains less information than a single-residue mass shift, and should not be weighted equivalently; as such, it is made to be worth half. Once the set of longest paths is enumerated by Bellman-Ford algorithm, the final step is resolving any remaining mass degeneracy, and expanding out any path components that could result from multiple sources, such as the mass shift that corresponds to both asparagine and glycine-glycine. After the set of all possible sequence tags is compiled, they are passed back to Refiner and added to the initial population of solutions, providing a useful source of variation – as well as serving as an additional component of the quality function, as will be detailed later.

Fitness Function

Once extraneous peaks are filtered, and initial conditions set, Refiner can begin evolving solutions. The encoding of potential solutions as virtual chromosomes is entirely straightforward, simple integer coding representing amino acids. The details of the genetic algorithm, however, are novel and merit explanation. Any genetic algorithm runs in phases: an evaluation step is followed by a reproduction step. Evaluation is carried out by a fitness function, while reproduction uses various operators. Refiner uses both a sophisticated fitness function that incorporates many distinct pieces of information about the problem, and a complex reproduction framework.

The fitness function used by Refiner consists of a linear combination of five components [Figure 3.2]. The first component represents the precursor mass difference. As the purpose of Refiner is to construct a sequence solution that fully characterizes the polypeptide being fragmented, it is vital to ensure that any such solution minimizes the mass difference between the theoretical mass of the solution and the observed mass to at least within measurement error. The first objective component uses a piecewise linear function of the precursor mass difference, where a sufficiently low mass difference gives full weight to the component function, a sufficiently large mass difference gives no weight at all, and values in-between give weight as a linear interpolation between the two. This function breaks a strong guideline for designing genetic algorithms, that is, always give the algorithm a gradient to traverse, and never clip at arbitrary maxima or minima. However, this algorithm is designed to function with real-world data, which is not infinite-precision. All mass differences within experimental error are equivalent; any ordering between them is false and may be misleading, so the fitness function treats all sufficiently low mass differences as being fully minimized. The clipping at sufficiently high mass differences is because at that point, not mass space degeneracy expands the number of potential sequences to the degree that any ordering of potential solutions is misleading. Instead, at high mass differences, other components of the fitness function are used to bring the solutions back into the 'real world'.

The second component of Refiner's fitness function is a simple proportion of matching fragments versus all observed fragments. The reasoning behind this is simple: we wish our sequence solution to explain the fragmentation pattern as comprehensively as possible.

The third component is a proteolysis specificity criterion. If the polypeptide being sequenced has resulted from a defined proteolytic method, one wishes to skew the optimization process to favor solutions which reflect the cleavage specificity of the proteolysis method. For example, tryptic peptides should terminate in lysine or arginine residues; when the proteolysis criterion is initialized to trypsin, sequence solutions ending in lysine or arginine are favored. Additionally, solutions should be reasonably free from internal cleavable residues; a tryptic peptide should have minimal, if any, internal lysines or arginines, as these would have been cleaved during sample digestion. The third component gives a constant value to a solution that ends in the correct residue, and penalizes the value by the same constant for each internal cleavable residue. This gives a low, but appreciably better fitness function value for solutions with no internal cleavage sites and the correct terminal residue, and strongly penalizes highly improbable (more than one internal cleavage site) solutions.

The fourth component of the fitness function is based on the sequence tags compiled in the previous stage. It gives benefit to the fitness function relative to how much of the sequence solution is explained by the sequence tags, via a coverage metric, ranging from no coverage to complete coverage, normalized to the sequence length.

The fifth and final component of the fitness function is a parsimony criterion. It is often possible to have two sequences that have the same fitness based on the previous four components, but one is clearly a better solution than the other. Due to degeneracy of the mass space as the number of residues increases, it is possible to have two sequences that differ in length, but have identical masses, numbers of fragment matches, coverage, and proteolytic specificity. The parsimony criterion simply penalizes the sequence for every additional residue, attempting to keep the sequence solution as short as possible.

These component functions are defined on the range of 0 to 1, and each has a coefficient that represents how much the component contributes to the overall fitness function. Typically, the precursor mass difference criterion receives the lion's share of the weight, at 70%. The fragmentation and proteolysis criteria each receive 10%, while the sequence tag coverage and parsimony criteria receive 5% respectively. The resulting fitness function value is used to order candidate solutions, so that selection and reproduction can take place.

Framework and Evolution Loop

The genetic algorithm framework used by Refiner is AI::Genetic::Pro [7], a comprehensive Perl [8] module for writing genetic algorithms. It runs for 50 generations in a pass, using a 5000-member population size. Selection is carried out using a roulette strategy, which uses the set of fittest solutions based on a proportional minimal fitness criterion, and selects candidates for reproduction from that set with a probability proportional to the candidate's fitness. Two parents are used for reproduction.

The reproduction process uses basic two-point crossover with 95% probability, and a 30% probability of point mutation. The latter merits comment; this mutation rate is an order of magnitude higher than typical mutation rates, and would for many problems cause the algorithm to diverge. The inherently highly local nature of this problem, however, appears to require a very high degree of point mutation, to the point where in experiments, normal mutation rates (3-5%) caused the algorithm to remain in its initial conditions and not discover any further local maxima. Additionally, several relatively unique operators were added to the standard mutation and crossover operators in order to take advantage of the unique conditions associated with this problem.

The first of the unique operators of Refiner's GA is variable-length chromosome support. In addition to the typical mutation operator, which converts a single random element in the chromosome array to another random value, it is possible for the mutation operator to select a terminal value (at the beginning or end of the chromosome), or even one step beyond it. In the former case, it is possible for the operator to convert the value at the end of the chromosome to nothing, thereby shrinking the chromosome at that end. Conversely, it is possible for the operator to convert the empty space beyond the edge of the chromosome to something, thereby growing the chromosome at that end. Two types of variable-length chromosomes are supported: single-ended, where all growth and shrinking takes place at one end of the chromosome, and double-ended, where it is supported at both ends. Variable-length chromosome support is vital to solving the *de novo* sequencing problem, as otherwise, one would be locked into sequences of the same length as the initial conditions, all of which must also be the same length. This would be sufficient to characterize cSNPs on an already identified sequence, but insufficient for true *de novo* sequencing.

The second unique operator that distinguishes Refiner's GA from most others is the adjacent-substitution operator. This operator runs on daughter chromosomes after they are produced by the crossover and mutation operators, and swaps adjacent residues in candidate sequences with a defined probability, typically set to 10%. The reason for this is that while a mutation event may produce a sequence that is close to the correct precursor mass, there is no sequence specificity to mutation operators. As such, given generally correct regions of sequence interspersed with incorrect regions, it is important to preserve those "building blocks" that work. The adjacent-substitution operator can convert incorrect sequences consisting of correct amino acids into the correct order with minimal disruption of correct subsequences. Accomplishing this with regular operators would require a highly improbable pair of adjacent mutation events, or an even more improbable chain of crossovers.

Refiner's final unique operator is in fact a modification to the typical reproduction procedure. Under normal circumstances, a given iteration of the evolution loop starts with a population of chromosomes that is strictly the result of selection and reproduction operators; in other words, each chromosome is a daughter chromosome of some set of parent chromosomes in the previous iteration of the loop. Because we need to sample many points in the sample space, however, we use a very high mutation and crossover probability, which has a chance of destroying the building blocks that developed in the last generation. The final unique feature of Refiner's evolution loop is elitist selection, which retains the top-scoring parent chromosomes from the last loop iteration and carries them over to the next loop iteration unchanged. This allows us to run at a very high mutation rate, analogous to a very high simulated annealing temperature, without worrying about solution divergence.

During the course of the evolution loop, Refiner applies all these operators, eventually producing a population with, one may hope, a significantly higher fitness than the initial conditions. However, in systems with very jagged fitness landscapes, it is easy to be caught in a local maximum, depending on both the initial conditions and the first few iterations of the evolution loop (path-dependence). A way of dealing with this needed to be developed.

Iterative Refinement

The process that allows the system some robustness to initial conditions is also the process that allows for parallelization, and coincidentally the same process that gave the system its name: iterative refinement [Figure 3.3]. It functions as follows – when the initial conditions are first set, the initial population is cloned several times, and evolution runs separately on each population, in parallel. Once evolution is complete for each population, the top-scoring members of each are pooled into a single population, identical in size to the initial conditions. This new, optimal population serves as the initial conditions for the next cycle of iterative refinement. A biological analogy for this process would be hills and islands. First, there is an ancestral population that lives in a hilly area. There are no barriers to movement, and members of that population mix freely all across the area. Then, the water level rises and the hills become islands. Members of the same, well-mixed population are trapped on each island, and each island begins to evolutionarily diverge. After some time, the water level falls, and members of the new, divergent populations can mix again, forming a new overall population. Then the process repeats. The advantage of this sort of hill-island architecture is that multiple paths can be taken from the same initial conditions, under the same selection pressure, without worrying that one will out-compete the others. Each sub-population finds its own local maximum, independent of the others, and populations only mix when they are stable. Multiple iterative refinement stages permit wide-ranging exploration from the best platforms, enabling a much more robust optimization than a single evolutionary loop.

Iterative refinement also lends itself to a number of tuning parameters. One can tune the number of subpopulations evolved in each step. By increasing this number, one samples more distinct evolutionary paths and increases the odds of finding some rare, interesting local maximum. One can also write an early termination rule – when the result of an iterative refinement stage is identical to the result of the previous stage, one can say that one has found the global maximum with far more confidence than with a similar rule within a single evolution loop.

Once evolution is completed, Refiner takes the top-scoring sequences from the final population, and evaluates them. Typically, not every bond has bidirectional fragmentation, so there is some degree of mass degeneracy. For example, there may be a dipeptidyl region without any fragmentation on the internal bond, so X-Y may be Y-X. Both would be generated during final combinatorial expansion, as well as all isobars equivalent to this. The resulting population of solutions is rescored in case extra fragments match, and solutions are returned.

Refiner, or a system similar to it, can be used in a number of ways. It can be embedded in a standalone characterization system and generate better solutions based on existing data. It can be used for purely *de novo* sequencing of very low-abundance sample, where every last erg of information must be squeezed out. It can also be used to generate candidate sequences for a downstream homology-based search platform, such as MS-BLAST [9], or just act as a much smarter sequence tag compiler. In the future, the technology behind Refiner will be integrated into the ProSight family of analysis tools to substantially improve the quality of characterization.

In tests, Refiner has demonstrated the ability to successfully recapitulate correct sequence solutions from error-prone initial guesses and several iterative refinement cycles. In addition to recovering correct sequence solutions from synthetically-perturbed sequence guesses, in tests Refiner was capable of generating the correct sequence given an initial guess of a homologous sequence from a different organism [Figure 3.4], in this case human when the real sample was bovine.

This gives an interesting application for Refiner: sequencing peptides from organisms that lack sequenced genomes. If it is possible to identify the peptide based on a database search with a related species, Refiner can then take the next step and determine the fully-characterized solution. This also suggests a number of future improvements for Refiner, such as a more sophisticated evolution system that instead of having an identical probability of point mutation for all amino acids, would instead be parameterized by a BLOSUM [10] or other position-specific scoring matrix, and allow for certain mutations to be more probable than others. Additionally, it may be possible to modify the mutation operator by making it position-dependent, ensuring that the probability of mutation in regions with excellent fragmentation is low. By making the *in-silico* evolutionary process carried out by Refiner more closely resemble biological evolution, it would be possible to better adapt the system for recapitulating real-world evolutionary relationships. In this fashion, one could begin analysis with a high-throughput tool that would assign putative identifications to observed peptides based on a sequence database from an evolutionarily-close species, and then use these as initial conditions to begin refining solutions. These solutions would be a sophisticated synthesis of numerous pieces of information, none of which could lead to a correct solution on their own, but which can, together, result in a useful answer.

Figures

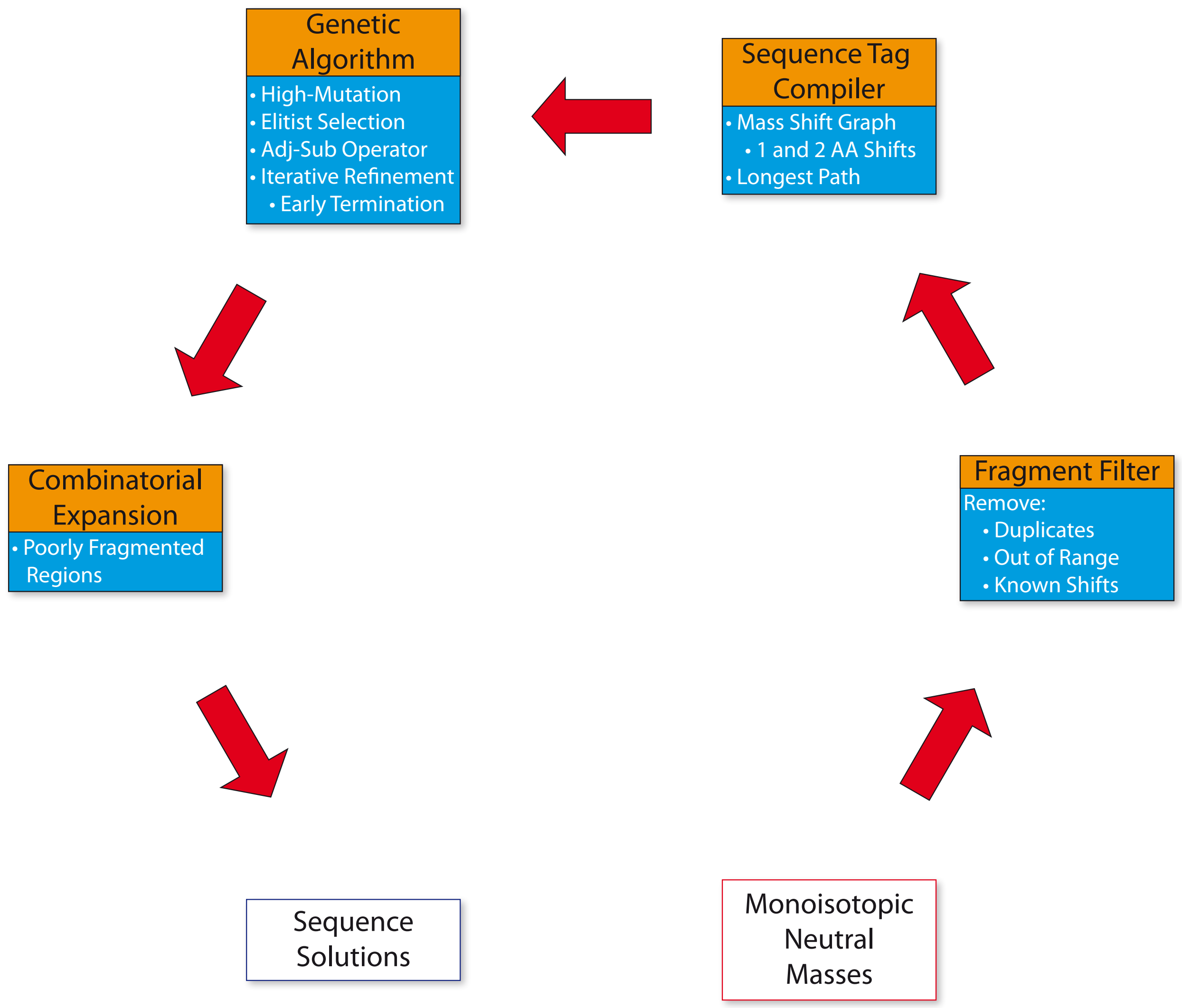


Figure 3.1
Refiner processing diagram.

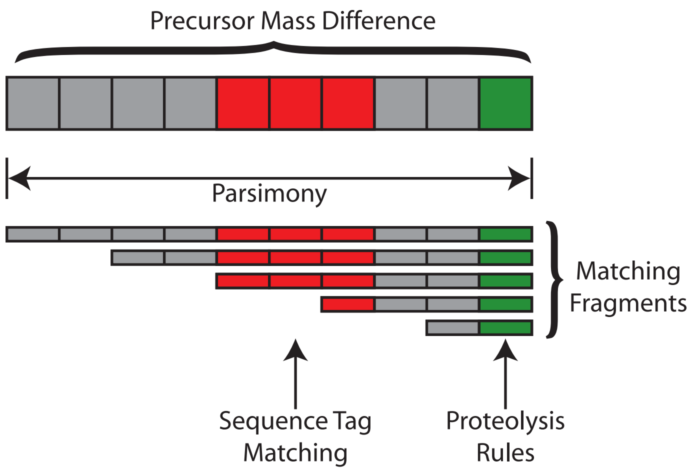


Figure 3.2
Refiner fitness function.

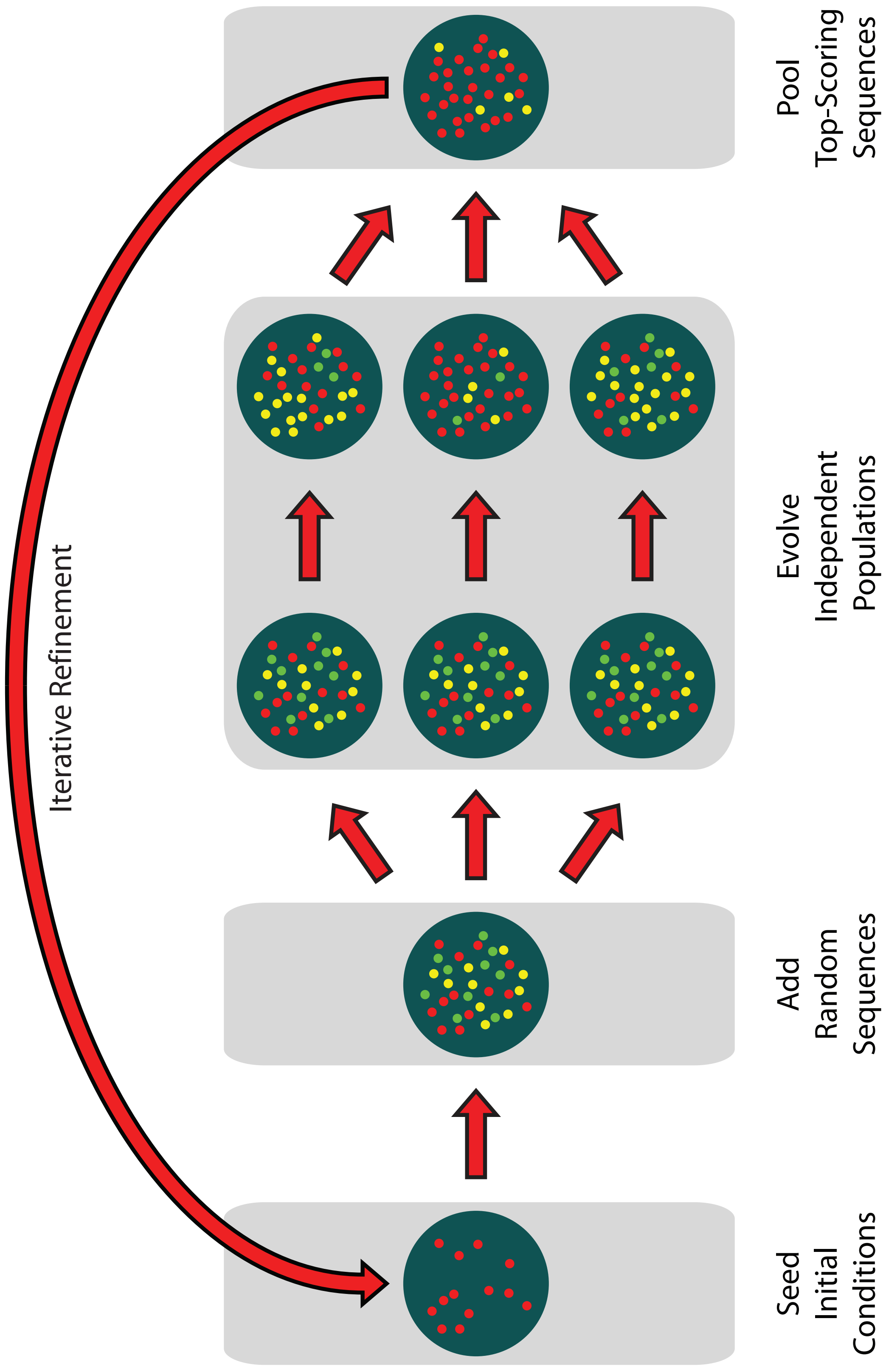


Figure 3.3
Iterative refinement diagram.

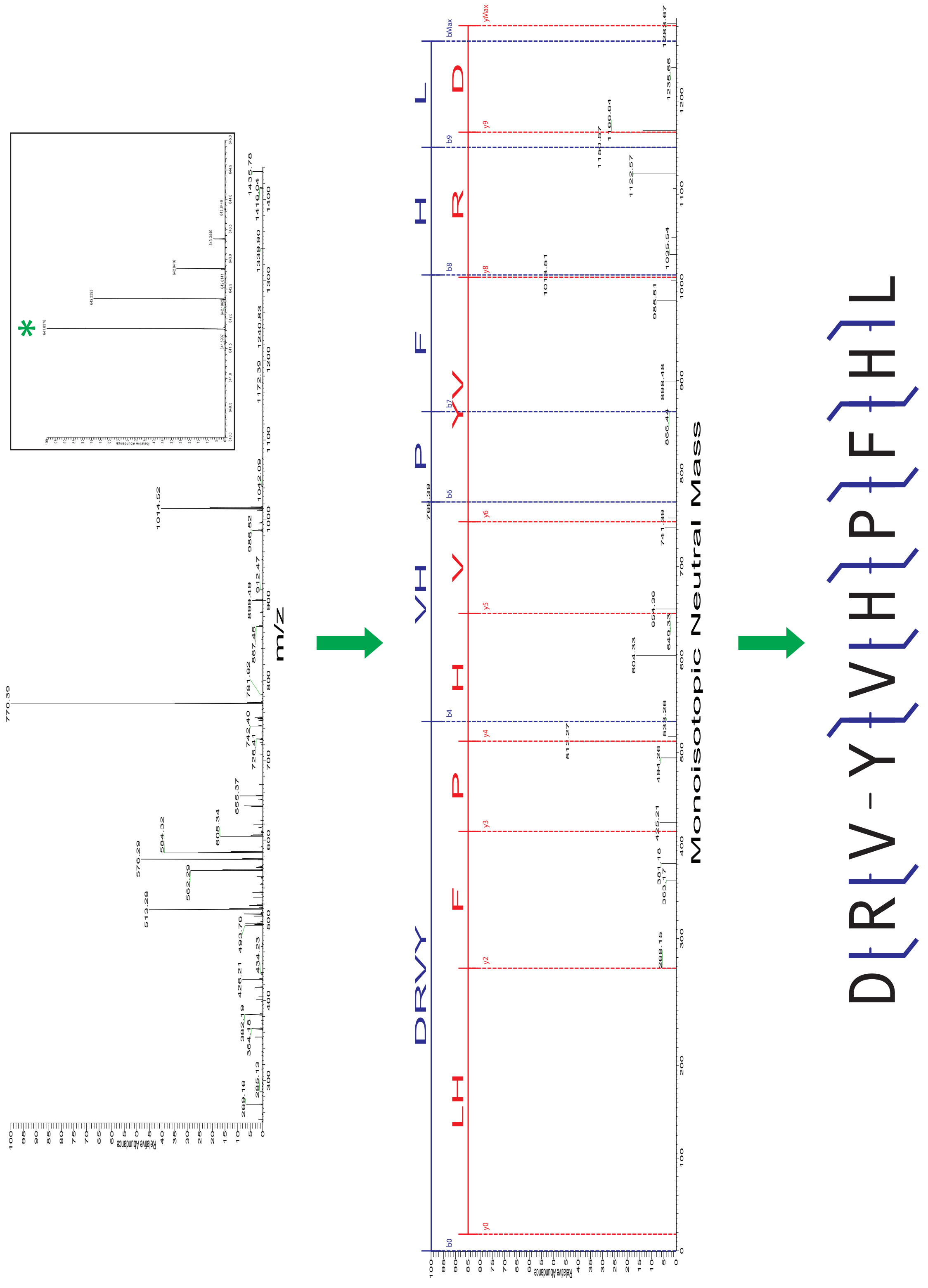


Figure 3.4
Bovine angiotensin sequence recapitulated from human initial hypothesis.

References

1. Bartels, C., *Fast algorithm for peptide sequencing by mass spectroscopy.* Biological Mass Spectrometry, 1990. **19**(6): p. 363-368.

2. Dancik, V., et al., *De novo peptide sequencing via tandem mass spectrometry.* J Comput Biol, 1999. **6**(3-4): p. 327-342.

3. Frank, A. and P. Pevzner, *PepNovo: de novo peptide sequencing via probabilistic network modeling.* Anal Chem, 2005. **77**(4): p. 964-973.

4. Ma, B., et al., *PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry.* Rapid Communications in Mass Spectrometry, 2003. **17**(20): p. 2337-2342.

5. Dijkstra, E., *A note on two problems in connexion with graphs.* Numerische mathematik, 1959. **1**(1): p. 269-271.

6. Bellman, R., *On a Routing Problem.* Quarterly of Applied Mathematics, 1958. **16**(1): p. 87.

7. Strzelecki, L., *AI::Genetic::Pro.* 2009.

8. Wall, L. and M. Loukides, *Programming Perl.* 2000: O'Reilly & Associates, Inc. Sebastopol, CA, USA.

9. Shevchenko, A., et al., *Charting the Proteomes of Organisms with Unsequenced Genomes by MALDI-Quadrupole Time-of-Flight Mass Spectrometry and BLAST Homology Searching.* Analytical Chemistry, 2001. **73**(9): p. 1917-1926.

10. Henikoff, S. and Henikoff, J.G., *Amino acid substitution matrices from protein blocks.* Proceedings of the National Academy of Sciences of the United States of America, 1992. 89(22): p. 10915-10919.

CHAPTER 4: Discovery of Protein-DNA Interactions

The work presented in this chapter is reproduced in part with permission from "Discovery of Protein-DNA Interactions", Nucleic Acids Research **37***(16), p. 5246-5254, Copyright 2009, Oxford Press. This work was supported by the National Science Foundation through DMS 0800631.*

Transcription factors (TFs) regulate the expression of target genes by binding in a DNA sequence-specific manner to their recognition sites in the promoter regions of these genes. The common pattern of the binding sites for a particular TF is called a transcription factor binding motif (TFBM), usually modeled by a position-specific weight matrix (PWM). Discovery of transcription factor binding sites (TFBSs) and TFBMs in TF-DNA interaction is essential for understanding the regulatory circuits that control cellular programs. In recent years, considerable progress has been made in developing both experimental and computational methods for elucidating TFBSs, and the mapping of their locations in a number of model organisms. Experimental techniques such as ChIP-chip [1] on promoter microarrays and whole genome tiling arrays and ChIP-seq [2] have been used to discover genome-wide TF-DNA binding sites for organisms ranging from Saccharomyces cerevisiae [3] to Homo sapiens [4]. Nonetheless, these experimental techniques are laborious and expensive, and require specialized antibodies which may be difficult to obtain [5]. As a given binding site is not necessarily occupied under all conditions [6], a single set of ChIP-chip/ChIP-seq experiments conducted under one experimental condition is insufficient to fully detect all sites to which a transcription factor may bind to in another experimental condition. Because of these limitations, computational methods provide appealing alternatives for pinning down TFBSs. De novo motif discovery algorithms based on probability models using PWMs, such as AlignACE [7], MDscan [8], MEME [9], and Weeder [10] have been accepted as important components of the computational biologist's toolkit. Moreover, rapid progress has been made to detect *cis*-regulatory modules which consist of highly coordinated TFBMs [11-12].

A recent trend to improve the aforementioned computational methods is to integrate information from relatively-inexpensive and easily-obtained gene expression data. The key idea to facilitate motif discovery using gene expression is that a gene's mRNA copy number is associated with active TFBMs' matching scores (or more intuitively, number of TFBM copies) in the promoter region of this gene. A number of attempts have been made along this line of thinking. For example, REDUCE [13] uses an exhaustive oligo-enumeration strategy to identify a potential set of candidate motifs, then 'reduces' this candidate set to a set of active motifs whose binding was best correlated with gene expression. As an improvement over REDUCE, Motif Regressor [14], incorporates gene expression in the initial identification of candidate motifs; the top-ranking genes in a single sample microarray experiment are used to identify an initial set of candidate motifs by MDscan [8]. This candidate list is then winnowed using stepwise regression with genome-wide gene expression serving as the response in a multiple regression model, resulting in a subset of motifs that are best correlated with gene expression across the genome.

These two approaches have stimulated many further studies in the past several years [15], and have generated a number of interesting results [16]. However, several issues remain that hinder their effective application in real practice. Since they rely on a single sample microarray measurement to carry out ranking and regression, they are sensitive to experimental and biological noise, especially in regards to low copy-number genes. Consequently, they may select different sets of motifs depending which single sample of microarray experiment is used in regression, requiring time-consuming manual merging and validation of the selected sets of motifs. Additionally, the stepwise motif selection in Motif Regressor relies on adding/deleting one motif at a time, a technique which is highly unstable and can only explore a small portion of all the possible models as the number of candidate motifs is usually hundreds [17].

To overcome these obstacles, we propose MotifExpress, a novel method that selects a set of motifs that best correlate with multiple samples of gene expression measured by microarrays simultaneously. We utilize multivariate regression to link gene expression (as responses) and candidate motifs (as predictors) together. In the multivariate regression framework, the selection of active motifs is very challenging as the number of parameters is much larger than the number of motifs. Thus we have a huge space to search for the globally-optimal model which gives rise to the set of active motifs. To score x_{ik} which indicates how likely motif k binds upstream of gene i in terms of both goodness of matching and number of sites as defined in Conlon et al [14].

$$x_{ik} = \log_2 \sum_{s \in S_{iw}} \frac{\Pr(s \text{ from } \theta_k)}{\Pr(s \text{ from } \theta_0)} \qquad \textbf{Eq 4.1}$$

Where $k=1,\ldots p$ and p is the total number of candidate motifs, θ_k is the probability matrix of motif k of width w, θ_0 is the third-order Markov model learned from intergenic sequences, and S_{iw} is the set of all w-mers in the upstream regulatory region of gene i.

Integration of Gene Expression with Motif Selection

The gene expression profile of gene i is denoted as $\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{im})$ where y_{ij} is the expression ratio in sample j for gene i, and m is the number of samples. Motif discovery further associates expression of gene i with all candidate motifs' matching scores x_{ik}, $k = 1,\ldots, p$. We assume a multivariate regression model between expression y and motif matching scores x of the form

$$y_{ij} = x_{i1}\beta_{1j} + x_{i2}\beta_{2j} + \ldots + x_{ip}\beta_{pj} + \varepsilon_{ij} \qquad \textbf{Eq 4.2}$$

Where random errors ε_{ij} are independently and identically distributed with mean zero and standard deviation σ_ε, and β_{kj} is the unknown coefficient which relates the expression of gene i to the motifs that putatively regulate it. We may rewrite [Eq 4.2] as

$$\begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{pmatrix} = \begin{pmatrix} x_{11} \\ \vdots \\ x_{n1} \end{pmatrix} (\beta_{11}\ldots\beta_{1m}) + \begin{pmatrix} x_{12} \\ \vdots \\ x_{n2} \end{pmatrix} (\beta_{21}\ldots\beta_{2m}) + \ldots + \begin{pmatrix} x_{1p} \\ \vdots \\ x_{np} \end{pmatrix} (\beta_{p1}\ldots\beta_{pm}) + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix} \qquad \textbf{Eq 4.3}$$

Note that for any motif k, if $\beta_{k1} = \ldots = \beta_{km} = 0$, then motif k is not associated with gene expression. We thus infer that motif k is not active in the biological conditions under which gene expression was measured. Otherwise, motif k is inferred to be active. Identifying active motifs thus becomes a variable selection problem in [Eq 4.3].

Motif Selection Using Penalized Multivariate Regression with CAP

By combining regression with variable selection, it is possible to select a set of active motifs which is significantly associated with gene expression. Classically, stepwise regression is used for variable selection; however, this technique is sensitive to perturbation of the data and can only explore a small portion of all the possible models as the number of candidate motifs is usually in the hundreds. A more modern technique, Lasso [20], has recently received significant attention as an efficient variable selection method. Lasso estimates the coefficients of predictors through minimization of the following expression

$$RSS + \lambda \sum |\beta_i| \qquad \textbf{Eq 4.4}$$

where the residual sum of squares (RSS) and the sum of absolute values of coefficients are two conflicting measures: the model with a smaller residual sum of squares tends to have more nonzero coefficients, which in turn results in a higher sum of absolute values of coefficients. λ is a regularization parameter controlling the trade-off between these two goals. Compared to a solution that minimizes only the residual sum of squares, i.e., the least squares estimate, the estimated coefficients in Lasso are closer to zero, which is referred to as 'shrinking'. It has been shown that the Lasso can select predictors consistently, i.e., to select correct predictors with probability approaching unity asymptotically, by shrinking the coefficients of the insignificant predictors to zero. However, Lasso was developed for regression with a single response rather than with multivariate responses as in [Eq 4.2]. Moreover, we are interested in eliminating any inactive motif in [Eq 4.2], which requires simultaneously shrinking all m coefficients corresponding to that motif to zero, rather than shrinking an individual coefficient. Recently, the simultaneous variable selection [23] and group Lasso [21-22] methods have been developed for selecting groups of variables. These methods have been nicely summarized in a unified shrinkage method, the composite absolute penalty (CAP) approach [22]. We apply the idea of a composite absolute penalty to a multivariate regression model in [Eq 4.2].

Our estimate is defined as the minimizer of

$$\sum_{i=1}^{n}\sum_{j=1}^{m}(y_{ij} - x_{i1}\beta_{1j} - x_{i2}\beta_{2j} - \ldots - x_{ip}\beta_{pj})^2 + \lambda \times \mathrm{pen}(\beta_{11}, \beta_{12}, \ldots, \beta_{pm}) \qquad \textbf{Eq 4.5}$$

where λ is a regularization parameter. The composite absolute penalty uses a combination of various metrics to achieve the objective of group predictor selection. One possible choice of penalty to achieve this goal is $\mathrm{pen}(\beta_{11}, \beta_{12}, \ldots, \beta_{pm}) = \sum_{k=1}^{p}\sqrt{\beta_{k1}^2 + \beta_{k2}^2 + \ldots + \beta_{km}^2}$, which reduces to the group Lasso penalty. However, the computation of group Lasso relies on a shooting algorithm, and the cost of computation is very high [21]. We instead elected to use the following CAP:

$$\mathrm{pen}(\beta_{11}, \beta_{12}, \ldots, \beta_{pm}) = \sum_{k=1}^{p}\max(|\beta_{k1}|, |\beta_{k2}|, \ldots, |\beta_{km}|) \qquad \textbf{Eq 4.6}$$

which is the penalty used in [23]. For each motif, the corresponding m coefficients are grouped through their maximum absolute values. As λ is increased, the group of motif coefficients β shrink simultaneously; a motif whose group of coefficients have shrunk to zero falls out of the model.

Since minimizing [Eq 4.5] is a convex optimization problem, a solution satisfying the Karush-Kuhn-Tucker conditions is a global minimum [23-24]. A beneficial feature of the proposed method [Eq 4.5] with penalty [Eq 4.7] is that the solution has a piecewise linear solution path for all values of λ. We adopt the homotopy algorithm [25-26], also known as the LARS/Lasso algorithm [27] to find the solutions for all values of λ. Even though the solution path for all values of λ can be effectively computed, it is still highly desirable that one solution is given for a fine-tuned value of λ. To choose a value of λ with a good balance of goodness-of-fit of the model and model parsimony, we minimize the Bayesian Information Criterion (BIC) [19].

$$\mathrm{BIC} = nm \ln \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\left(y_{ij} - x_{i1}\hat{\beta}_{1j}(\lambda) - x_{i2}\hat{\beta}_{2j}(\lambda) - \ldots - x_{ip}\hat{\beta}_{pj}(\lambda)\right)^2}{nm} + p_A(\lambda) \times \ln(nm) \qquad \textbf{Eq 4.7}$$

where $\hat{\beta}_{1j}(\lambda), \hat{\beta}_{2j}(\lambda), \ldots, \hat{\beta}_{pj}(\lambda)$ are CAP estimates of $\beta_{1j}, \beta_{2j}, \ldots, \beta_{pj}, p_A(\lambda)$ is the number of estimated active motifs, *i.e.*, non-zero coefficients, that depend on the tuning parameter λ. Since the solution path is piecewise linear, the minimum BIC can be found by comparing [Eq 4.7] for a number of λ values.

Once the model is selected, we test the significance of each selected motif by calculating for each of them a pooled *P*-value which combines all *P*-values of the corresponding *m* coefficients using Stouffer's method [28-29].

Functional Annotation of Discovered Motifs

To verify identifications and further elucidate biological relationships, manual analysis and functional annotation was carried out on discovered motifs. Results were validated where possible by comparison to known TF binding sites by ChIPCodis [30] against the MacIsaac *et al* yeast TFBM dataset [3] as well as additional STAMP [32] comparison to the common ribosome-associated RRPE and PAC motifs [33].

Validation – Simulations

Extensive simulations were carried out to examine the effectiveness of MotifExpress in identifying active motifs. We set number of genes, n = 5000, and number of the motif candidates as 100, among which 10 motifs were active. We generated motif scores x_{ik} from N(0,1) and random error from N(0,σ_ε). We gradually increased the standard deviation of random error from $\sigma_\varepsilon = 1$ to $\sigma_\varepsilon = 5$. Gene expression was generated as the summation of linear combinations of the active motif scores and random error as in [Eq 4.2]. We let $m = 3,4,5$. We generated 100 datasets for each setting, i.e., each combination of σ_ε and m. The Matthews Correlation Coefficient (MCC) [34], was calculated for each dataset, where a value of 1 indicates perfect selection of active motifs and rejection of spurious motifs, while a value of 0 is average random selection.

The summary statistics of the resulting MCCs are given in [Table 4.1]. It can be seen that our proposed method consistently performs very well across all settings. Since AIC with second order correction (AICc) was also suggested as an alternative criterion to choose the regularization parameter, we tested the performance of our proposed method using AICc. MotifExpress with BIC-minimization consistently outperformed that with AICc-minimization; average MCCs were significantly higher across all simulation regimes. MotifExpress with AICc was observed to identify more motifs spuriously; its performance was robust as σ_ε increased, but had a lower MCC throughout the settings [Figure 4.2]. In comparison, the performance of MotifExpress with BIC did degrade as σ_ε increased, but still had much higher MCC as we see in Fig 4.2

Simulations were also carried out to compare the performance of MotifExpress and Motif Regressor. Simulations were carried out as previously, with 100 datasets generated for each level of random error from $\sigma_\varepsilon = 1$ *to* $\sigma_\varepsilon = 5$. MotifExpress made its selections based on $m = 3$ responses, while Motif Regressor used a single response. While MCC performance remained similar for both, a considerable difference was observed in the false discovery rate (FDR) [Figure 4.3]. MotifExpress controlled the FDR efficiently and consistently, while Motif Regressor's performance varied across the tested error levels; as observed in later experiments on real data, MotifExpress consistently returned more parsimonious sets of confident motifs, especially as the noise was increased.

Validation – Experiments

In addition to simulations, experiments were carried out with real-world data to gauge the effectiveness of MotifExpress under known conditions. For all experiments, microarray data was retrieved from the Gene Expression Omnibus (GEO) [35] and $\log_2$ transformed. Missing values were estimated using k-nearest-neighbor imputation [36], implemented in the R [37] *impute* package. The upstream 800 bp sequence for each gene was used, with repetitive sequences masked using RepeatMasker [38]. Two experiments were carried out: one probing GCN2 pathway, and the other investigating heat shock response, both in the budding yeast *Saccharomyces cerevisiae*.

GCN2 Constitutive Activation Analysis

The protein kinase GCN2 has drawn attention in recent years due to its extensive regulatory impact. The Gcn2p homodimer associates with the large ribosome subunit in the cytosol, and when activated, phosphorylates Ser-52 of eIF2α [39]. In yeast, this has two immediate effects; firstly, the repression of general translation by sequestration of eIF2β, and secondly the derepression of GCN4 translation. Gcn4p then acts as a transcription factor that modulates the expression of numerous stress- and starvation-related genes [40]. Gcn2p may be activated by numerous signals via multiple pathways, including the drug rapamycin [41].

We elected to use constitutively-active Gcn2 as a means of validating our method. As active Gcn2p results in Gcn4p activation, consequentially leading to an activation of downstream genes, it follows that the presence of the GCN4 motif should be strongly correlated with gene expression under the condition of constitutively-active Gcn2p. A four-sample dataset of cDNA microarrays, which hybridized four biological replicates of GCN2c constitutively-active mutant samples to a common reference wild-type sample, was retrieved from GEO (GSE8111) [42]. The data retrieved was the log-transformed gene expression ratios between mutant and wild-type, and was used by MotifExpress as the response to fit a multivariate regression model. Three motifs were selected by MotifExpress, identified by STAMP as GCN4, PAC, and RAP1.

The results obtained by MotifExpress were compared with those obtained by running Motif Regressor [14] separately on each sample. In contrast to three motifs discovered by MotifExpress, Motif Regressor discovered 130 motifs. For Motif Regressor running on each sample, the smallest *P*-value of the motifs identified by STAMP [32] as GCN4 and the highest rank of the GCN4 motifs sorted by *P*-values in the result is reported in [Table 4.2]. MotifExpress analysis resulted in a parsimonious set of results, where the GCN4 motif had the lowest *P*-value and ranked first, while analyzing individual samples in the dataset by Motif Regressor yielded highly heterogeneous results. The *P*-value was much lower in the MotifExpress results than in any of the Motif Regressor results.

The presence of the PAC and RAP1 motifs is likewise unsurprising; the RP regulon (strongly associated with the RAP1 motif) and the RRB regulon (strongly associated with the PAC motif) [43] are both known to be repressed by treatment with rapamycin, which is also known to induce Gcn4p synthesis [41]. The GSE8111 dataset shows a transcription profile quite similar to rapamycin treatment, with the RRB and RP genes downregulated, and amino acid biogenesis genes upregulated. Analysis of genes used for motif discovery via ChIPCODIS revealed a significant overrepresentation of genes to which Gcn4p binds under rapamycin treatment, with a pooled P-value of 1.56×10^{-18}.

Heat Shock Analysis

The heat shock response is a conserved and concerted cellular program in eukaryotes. Temperature changes above the physiological optimum induce the synthesis of heat shock proteins, a diverse class of proteins that have effects on protein folding, metabolism, and antioxidant response. Expression of the genes coding for these proteins is regulated by a set of stress-related transcription factors, most importantly Hsf1p and Msn2/4p.

A three-sample dataset of cDNA microarrays, comparing wild-type cells in mid-log-phase grown at 30C to wild-type cells heat-shocked at 39C for 15 minutes, was downloaded from GEO (GSE7665) [44]. The data was the log-transformed gene expression ratios between heat shock and control conditions, and was used by MotifExpress as the response to fit a multivariate regression model. A set of 15 active motifs was selected by MotifExpress, among them HSF1, RPH1, MSN2/4, SFP1, FHL1, and PAC; of these, the HSF1 motif was the most significant, with a pooled *P*-value of 7.8×10^{-37}. In contrast, Motif Regressor discovered 113 motifs, many of which were redundant.

It is known that many of the genes regulated by Hsf1p encode chaperones, proteins responsible for inducing and maintaining protein conformation and preventing unwanted protein aggregation, such as HSP82 [45]. It would be expected that cells undergoing heat shock would experience an upregulation of genes regulated by Hsf1p and Msn2/4p; analysis by MotifExpress analysis demonstrates the detection of this response. The PAC motif, as mentioned in the previous example, is a signature of the RRB regulon, which is down-regulated under heat shock conditions [46]. The genes in the dataset were confirmed to be significantly overrepresented for Hsf1p, Msn2/4p, Fhlp, and Sfp1p binding under stress conditions by ChIPCODIS (*P*-values ranging from 8.82×10^{-62} to 3.63×10^{-18}).

It is interesting to note that the HSF1 motif is known to consist of repeats of a 5-bp consensus sequence 5'-NGAAN-3' and its reverse complement 5'-NTTCN-3' [47]. In Fig 4.4, we plotted the motif logos of the most significant HSF1 motifs discovered by MotifExpress, by Motif Regressor via analyzing each sample independently, and by MacIsaac *et al* using ChIP-chip and phylogenetic methods [3]. The HSF1 motif discovered by MotifExpress is notably the closest to the consensus sequence reported in [47] among the five motifs [Figure 4.4].

Discussion

In this chapter, we presented a novel method, MotifExpress, for identifying transcription factor binding motifs (TFBMs) strongly associated with multiple samples of gene expression. Existing methods for identifying TFBMs correlate sequence information to a single sample of gene expression, one sample at a time, which results in a redundant set of active motifs with many spurious results [48]. Additionally, existing methods rely on classical variable selection techniques such as stepwise regression, which is highly unstable and can only explore a small portion of all the possible models as the number of candidate motifs is usually in the hundreds. Our method is designed to integrate multiple samples of gene expression via multivariate regression. Using the CAP approach and selecting the regularization parameter using BIC, we can effectively identify a parsimonious set of active motifs. We examined the performance of MotifExpress using synthetic data under an array of settings with different numbers of samples and various variance magnitudes of random error. In these simulations, MotifExpress performed consistently well throughout all settings. We then analyzed two real experiments using MotifExpress, identifying active motifs correlated with expression. In both experiments, the set of discovered motifs agreed well with current literature.

surmount this challenge, we fit a model using a composite absolute penalty (CAP). Unlike the stepwise regression procedure, the CAP procedure selects motifs via a convex optimization and can effectively find the globally-optimal set of active motifs. We use the Bayesian information criterion (BIC) to select the regularization parameter. We demonstrate the excellent performance of MotifExpress by applying it to synthetic data as well as GCN2 constitutive activation and heat shock experiments in Saccharomyces cerevisiae. It is evident from these results that incorporating multiple samples of gene expression substantially reduced the number of spurious results.

MotifExpress Framework Overview

The MotifExpress framework consists of several stages. First, significance analysis of microarrays (SAM) [18], a microarray analysis algorithm, is used to identify genes that are differentially expressed between the treatment and control conditions. The upstream promoter sequences of significantly differentially-expressed genes are then used as input in a de novo motif discovery algorithm (MDscan) to search for candidate motifs. The upstream promoter sequences of all genes for which expression was measured are then scored for matches to each candidate motif; the algorithm then uses multivariate regression to link gene expression in all samples with the motif matching scores of all candidate motifs. Finally, the active motifs that are significantly associated with gene expression are identified using a composite absolute penalty approach with the regularization parameter selected through minimizing BIC [19] [Figure 4.1].

Motif Discovery

SAM [18] analysis is run on the gene expression profiles to determine which genes are differentially expressed between treatment and control conditions at a pre-specified false discovery rate. MDscan [8] is then run on the upstream promoter sequences of the significantly differentially expressed genes to discover motifs ranging from 7 to 15 bp in width. The 30 most significant motifs for each motif width are combined to form a set of candidate motifs. We then calculate the motif matching

The MotifExpress framework is easily extensible to support other TF-DNA binding discovery methods, especially *cis*-regulatory module discovery methods. Statistically, penalized multivariate regression with CAP is ready to incorporate additional structural information about motifs. Likewise, as high-throughput transcriptomic studies transition from hybridization-based microarrays to rapid whole-transcriptome sequencing, this new data is easily integrated in MotifExpress.

Aside from motif selection, another challenge is to identify the regulatory targets of a TF. In principle, given the motifs (including promoter sequence) and estimated coefficients, we can predict gene expression. Once this is done, the genes with significantly high or low predicted expression could be considered as potential regulatory targets. However, such prediction in practice typically has too large an uncertainty to be used for identifying regulatory targets. A possible alternative is to build a prediction model with gene cluster membership as the response, as Beer and Tavazoie [49]. The variable selection method that we employed in this paper can be adapted to that model.

Tables

σ_ε	$m = 3$				$m = 4$				$m = 5$			
	MCC (AICc)		MCC (BIC)		MCC (AICc)		MCC (BIC)		MCC (AICc)		MCC (BIC)	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
1	0.582	0.143	0.928	0.056	0.620	0.159	0.933	0.057	0.587	0.143	0.940	0.058
2	0.590	0.158	0.923	0.068	0.572	0.147	0.918	0.058	0.639	0.156	0.939	0.057
3	0.580	0.142	0.895	0.069	0.591	0.150	0.904	0.072	0.614	0.148	0.922	0.068
4	0.577	0.149	0.872	0.077	0.556	0.132	0.854	0.073	0.609	0.139	0.864	0.086
5	0.576	0.150	0.839	0.091	0.575	0.135	0.831	0.081	0.598	0.144	0.838	0.078

Table 4.1

Mean (ave) and standard deviation (std) of Matthews Correlation Coefficient (MCC) for MotifExpress with regularization parameters selected through AICc-minimization and BIC-minimization as the random error's standard deviation σ_ε and the number of samples m are varied in the simulation study.

Analysis	Number of Motifs Identified	GCN4 Motif with Lowest *P*-value	Rank of Result
MotifExpress	3	1.73×10^{-40}	1st in 3
MotifRegressor Sample 1	42	8.86×10^{-3}	9th in 42
MotifRegressor Sample 2	15	4.37×10^{-4}	3rd in 15
MotifRegressor Sample 3	55	1.22×10^{-3}	7th in 55
MotifRegressor Sample 4	18	8.28×10^{-12}	2nd in 18

Table 4.2

GCN4 motif discovery on constitutively-activated Gcn2 mutant dataset by MotifExpress on all samples simultaneously and by Motif Regressor, on each sample individually.

Figures

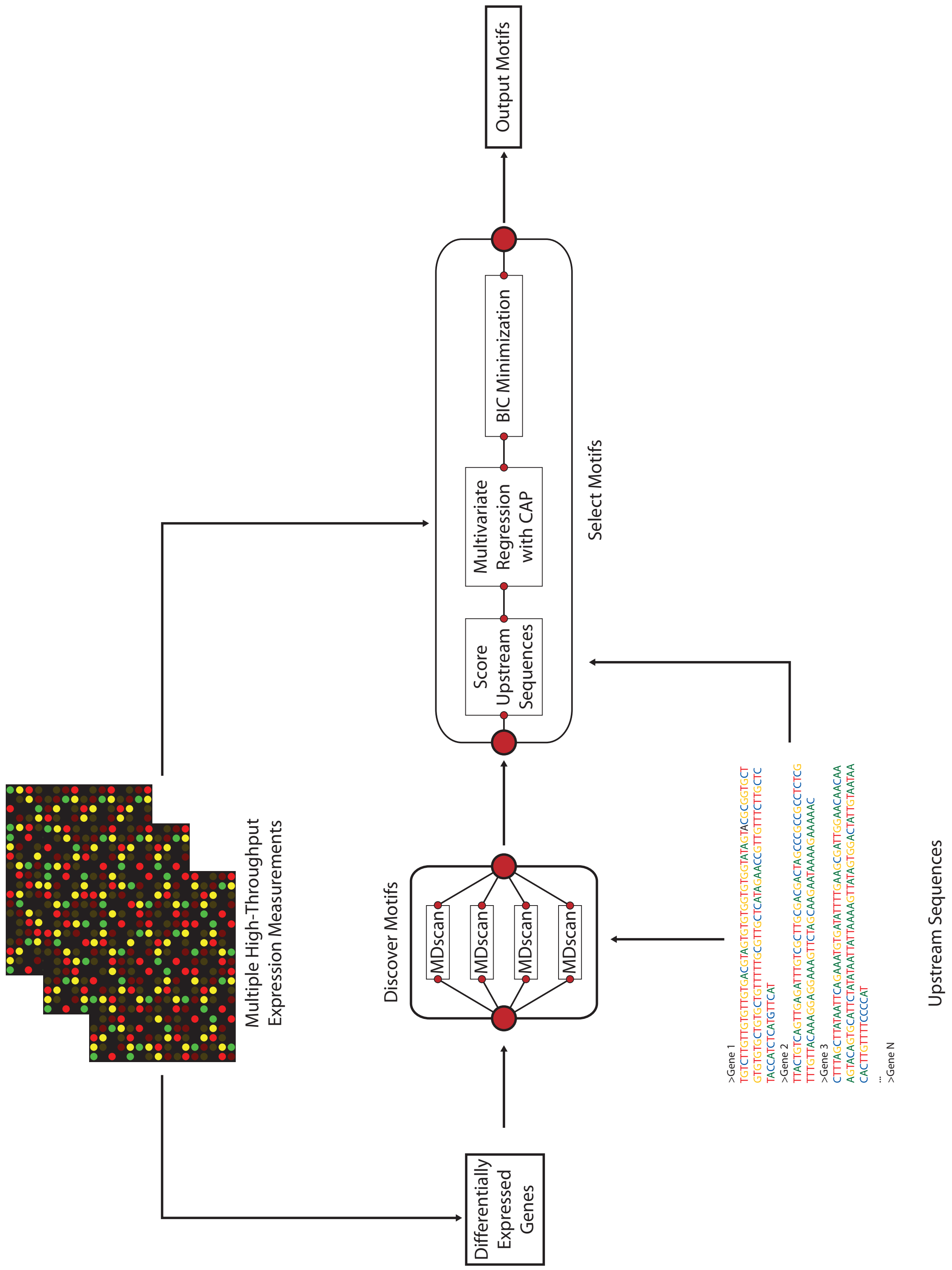


Figure 4.1
MotifExpress conceptual diagram.

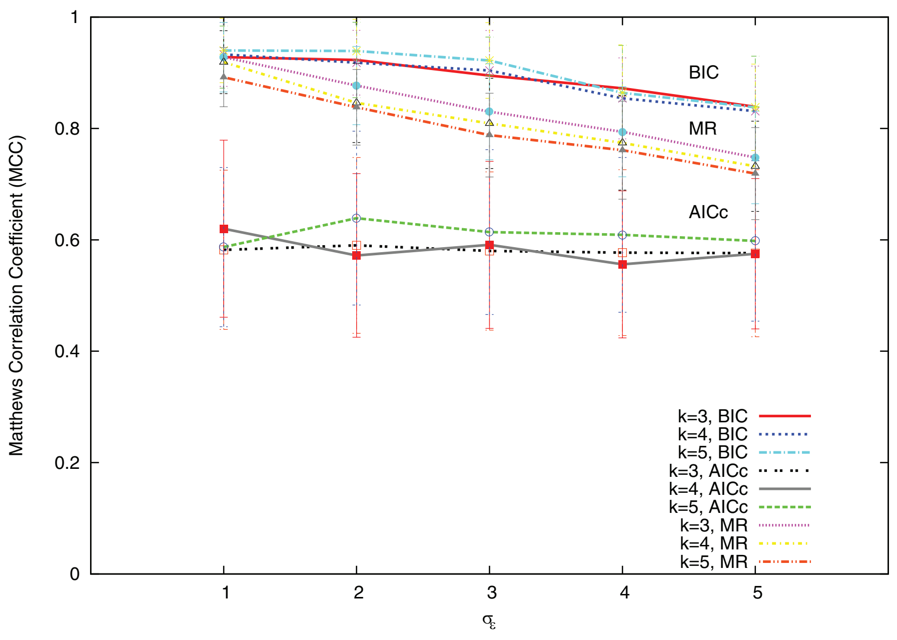


Figure 4.2

Summary plot of Matthews Correlation Coefficient (MCC) for MotifExpress with regularization parameters selected through AICc-minimization and BIC-minimization as the random error's standard deviation σ_ε and the number of samples m are varied in the simulation study. MotifRegressor performance in the same simulation was computed by pooling results from independent runs.

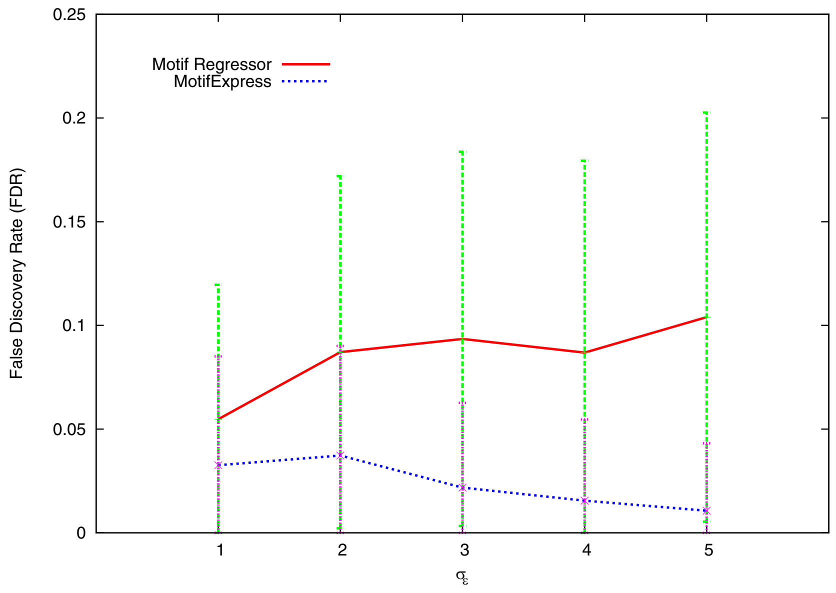


Figure 4.3

False Discovery Rate (FDR) of MotifRegressor and MotifExpress as the error standard deviation σ_ε was varied. 100 runs were carried out at each error level, each with 10 real and 90 spurious motifs; MotifRegressor selected motifs based on a single simulated response, while MotifExpress selected motifs based on three simulated responses simultaneously.

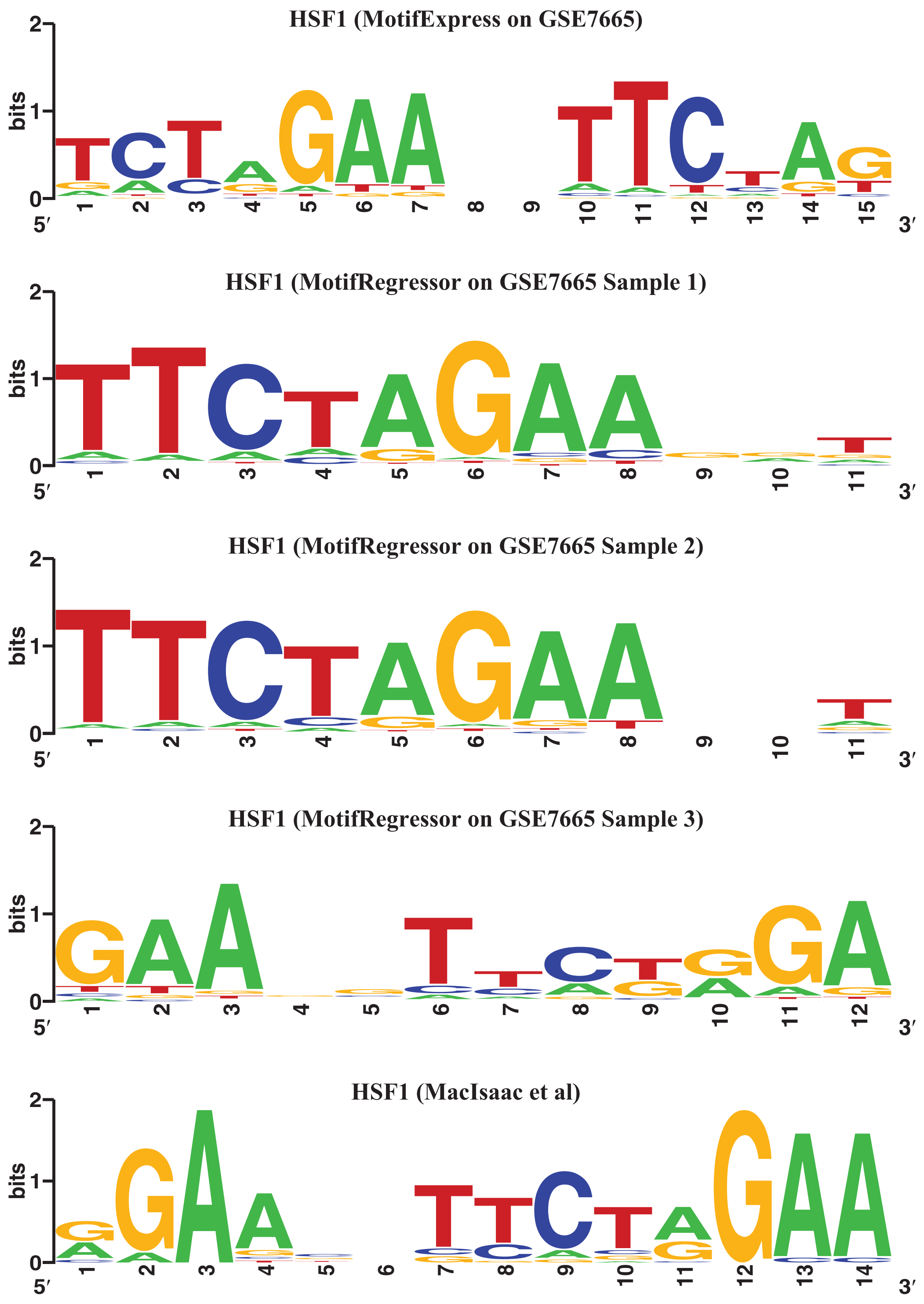


Figure 4.4
HSF1 binding motif discovered by MotifExpress by analyzing all samples in GSE7665 simultaneously compared to MotifRegressor analyzing each sample individually, as well as current literature. The head-to-head inverted NGAAN motif is prominent in the MotifExpress results.

References

1. Buck, M.J. and J.D. Lieb, *ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments.* Genomics, 2004. **83**(3): p. 349-360.

2. Johnson, D.S., et al., *Genome-wide mapping of in vivo protein-DNA interactions.* Science, 2007. **316**(5830): p. 1497-502.

3. MacIsaac, K.D., et al., *An improved map of conserved regulatory sites for Saccharomyces cerevisiae.* BMC Bioinformatics, 2006. **7**(113).

4. Wei, C.-L., et al., *A Global Map of p53 Transcription-Factor Binding Sites in the Human Genome.* Cell, 2006. **124**(1): p. 207-219.

5. Wu, J., et al., *ChIP-chip comes of age for genome-wide functional analysis.* Cancer Res, 2006. **66**(14): p. 6899-902.

6. Harbison, C.T., et al., *Transcriptional regulatory code of a eukaryotic genome.* Nature, 2004. **431**(7004): p. 99-104.

7. Hughes, J.D., et al., *Computational identification of Cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae.* Journal of Molecular Biology, 2000. **296**(5): p. 1205-1214.

8. Liu, X.S., D.L. Brutlag, and J.S. Liu, *An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments.* Nature Biotechnology, 2002. **20**(8): p. 835-839.

9. Bailey, T. and C. Elkan, *Fitting a mixture model by expectation maximization to discover motifs in biopolymers.* Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, 1994: p. 28-36.

10. Pavesi, G., et al., *Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes.* Nucleic Acids Research, 2004. **32**(Web Server issue): p. W199-W203.

11. Zhou, Q. and W.H. Wong, *CisModule: de novo discovery of cis-regulatory modules by hierarchical mixture modeling.* Proc Natl Acad Sci U S A, 2004. **101**(33): p. 12114-9.

12. Gupta, M. and J.S. Liu, *De novo cis-regulatory module elicitation for eukaryotic genomes.* Proc Natl Acad Sci U S A, 2005. **102**(20): p. 7079-84.

13. Roven, C. and H.J. Bussemaker, *REDUCE: An online tool for inferring cis-regulatory elements and transcriptional module activities from microarray data.* Nucleic Acids Res, 2003. **31**(13): p. 3487-90.

14. Conlon, E.M., et al., *Integrating regulatory motif discovery and genome-wide expression analysis.* Proceedings of the National Academy of Sciences of the United States of America, 2003. **100**(6): p. 3339-3344.

15. Das, D., M. Pellegrini, and J.W. Gray, *A primer on regression methods for decoding cis-regulatory logic.* PLoS Comput

Biol, 2009. **5**(1): p. e1000269.

16. Ben-Yehuda, S., et al., *Defining a centromere-like element in Bacillus subtilis by Identifying the binding sites for the chromosome-anchoring protein RacA.* Mol Cell, 2005. **17**(6): p. 773-82.

17. Breiman, L., *Heuristics of instability and stabilization in model selection.* Annals of Statistics, 1996. **24**(6): p. 2350-2383.

18. Tusher, V.G., R. Tibshirani, and G. Chu, *Significance analysis of microarrays applied to the ionizing radiation response.* Proceedings of the National Academy of Sciences of the United States of America, 2001. **98**(9): p. 5116-5121.

19. Schwarz, G., *Estimating the Dimension of a Model.* The Annals of Statistics, 1978. **6**(2): p. 461-464.

20. Tibshirani, R., *Regression Shrinkage and Selection via the Lasso.* Journal of the Royal Statistical Society, Series B (Methodological), 1996. **58**(1): p. 267-288.

21. Yuan, M. and Y. Lin, *Model selection and estimation in regression with grouped variables.* Journal of the Royal Statistical Society Series B-Statistical Methodology, 2006. **68**: p. 49-67.

22. Zhao, P., G.V. da Rocha, and B. Yu, *The Composite Absolute Penalties family for Grouped and Hierarchical Variable Selection.* Annals of Statistics, In Press.

23. Karush, W., *Minima of Functions of Several Variables with Inequalities as Side Constraints.* Master's Thesis., 1939.

24. Kuhn, H.W. and A.W. Tucker, *Nonlinear programming.* Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability, 1951: p. 481-492.

25. Turlach, B.A., W.N. Venables, and S.J. Wright, *Simultaneous Variable Selection.* Technometrics, 2005. **47**(3): p. 349-363.

26. Osborne, M.R., B. Presnell, and B.A. Turlach, *A new approach to variable selection in least squares problems.* Ima Journal of Numerical Analysis, 2000. **20**(3): p. 389-403.

27. Efron, B., et al., *Least angle regression.* Annals of Statistics, 2004. **32**(2): p. 407-451.

28. Burns, P., *Random Portfolios for Performance Measurement*, in *Optimization, Econometric and Financial Analysis*, E.J. Kontoghiorghes and C. Gatu, Editors. 2007, Springer: Berlin. p. 239-240.

29. Whitlock, M.C., *Combining probability from independent tests: the weighted Z-method is superior to Fisher's approach.* J Evol Biol, 2005. **18**(5): p. 1368-73.

30. Abascal, F., et al., *ChIPCodis: mining complex regulatory systems in yeast by concurrent enrichment analysis of chip-on-chip*

data. Bioinformatics, 2008. **24**(9): p. 1208-9.

31. Gupta, S., et al., *Quantifying similarity between motifs.* Genome Biol, 2007. **8**(2): p. R24.

32. Mahony, S. and P.V. Benos, *STAMP: a web tool for exploring DNA-binding motif similarities.* Nucleic Acids Research, 2007. **35**(Web Server Issue): p. W253-W258.

33. Hughes, J.D., et al., *Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae.* J Mol Biol, 2000. **296**(5): p. 1205-14.

34. Matthews, B.W., *Comparison of the predicted and observed secondary structure of T4 phage lysozyme.* Biochim Biophys Acta, 1975. **405**(2): p. 442-51.

35. Edgar, R., M. Domrachev, and A.E. Lash, *Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.* Nucl. Acids Res., 2002. **30**(1): p. 207-210.

36. Troyanskaya, O., et al., *Missing value estimation methods for DNA microarrays.* Bioinformatics, 2001. **17**(6): p. 520-525.

37. R Development Core Team, *R: A Language and Environment for Statistical Computing*. 2008, R Foundation for Statistical Computing: Vienna, Austria.

38. Smit, A.F.A., R. Hubley, and P. Green, *RepeatMasker Open-3.0*. 1996-2004.

39. Padyana, A.K., et al., *Structural basis for autoinhibition and mutational activation of eukaryotic initiation factor 2alpha protein kinase GCN2.* J Biol Chem, 2005. **280**(32): p. 29289-99.

40. Hinnebusch, A.G. and K. Natarajan, *Gcn4p, a master regulator of gene expression, is controlled at multiple levels by diverse signals of starvation and stress.* Eukaryot Cell, 2002. **1**(1): p. 22-32.

41. Kubota, H., et al., *Rapamycin-induced translational derepression of GCN4 mRNA involves a novel mechanism for activation of the eIF2 alpha kinase GCN2.* J Biol Chem, 2003. **278**(23): p. 20457-60.

42. Menacho-Marquez, M., et al., *Gcn2p regulates a G1/S cell cycle checkpoint in response to DNA damage.* Cell Cycle, 2007. **6**(18): p. 2302-5.

43. Wade, C.H., M.A. Umbarger, and M.A. McAlear, *The budding yeast rRNA and ribosome biosynthesis (RRB) regulon contains over 200 genes.* Yeast, 2006. **23**(4): p. 293-306.

44. Shivaswamy, S. and V.R. Iyer, *Stress-dependent dynamics of global chromatin remodeling in yeast: dual role for SWI/SNF in the heat shock stress response.* Mol Cell Biol, 2008. **28**(7): p. 2221-34.

45. Lindquist, S. and E.A. Craig, *The heat-shock proteins.* Annu Rev Genet, 1988. **22**: p. 631-77.

46. Warner, J.R., *The economics of ribosome biosynthesis in yeast.* Trends Biochem Sci, 1999. **24**(11): p. 437-40.

47. Bonner, J.J., C. Ballou, and D.L. Fackenthal, *Interactions between DNA-bound trimers of the yeast heat shock factor.* Mol Cell Biol, 1994. **14**(1): p. 501-8.

48. Foat, B.C., et al., *Profiling condition-specific, genome-wide regulation of mRNA stability in yeast.* Proc Natl Acad Sci U S A, 2005. **102**(49): p. 17675-80.

49. Beer, M.A. and S. Tavazoie, *Predicting gene expression from sequence.* Cell, 2004. **117**(2): p. 185-98.

CHAPTER 5: Conclusions and Future Directions

High-throughput data-gathering modalities caused an explosion in available data about biological systems. The last decade has seen a new 'omic' field arise for practically every aspect of biology, ranging from investigations into the central dogma of molecular biology (genomics, transcriptomics, and proteomics) to investigations of primary and secondary metabolism (metabolomics), to other, stranger beasts such as regulomes, interactomes, ionomes, kinomes, mechanomes, and physiomes. Practically every 'omic' field has had, at its founding, some core technology that produced a qualitatively larger volume of data than any predecessor, and formed the nucleus around which the field accreted. Genomics as we know it now could not be possible without the high-throughput automated Sanger sequencer, now several generations old. The core of transcriptomics was the hybridization microarray, which catalyzed numerous fruitful collaborations between biologists and statisticians. Proteomics and metabolomics as we know it could not have existed without high-throughput LC-MS technology, and so on. Each of these technologies catalyzed the creation of a fundamentally new field, but one of the things that separate the 'omic' fields from their low-throughput predecessors is the realization that high volumes of data behave fundamentally differently than low volumes. One cannot treat a microarray experiment, which measures the expression of ten thousand genes, as simply a somewhat larger set of measurements, tractable with the same analytic tools as those used on simple multi-well PCR-based assays. One cannot evaluate the results of a proteomic experiment which produced data on ten thousand peptides in the same way as one would the results of spraying a sample containing a single purified peptide into a mass spectrometer. Every 'omic' field required new tools and new ways of looking at data in order to make sense of itself.

One vital aspect of this was the interactions within the data, both within individual data points and between them, potential and actual. When one only has access to measurements that could result from the interactions of hundreds of 'hidden' variables, one requires very careful analysis. While vast amounts of data could be a great boon, this could just as easily turn into a terrible curse that saps statistical power from experiments and destroys conclusions before they are formed. The tools that one uses must take this into account, and the classical methods of yesteryear often have great difficulty in coping with the vast search spaces opened up by the combinatoric interactions of vast arrays of data points. My work has focused on developing tools to make some sense of this, and on validating fundamental algorithmic and statistical approaches to determine their suitability for answering real biological questions given the results of a high-throughput data pipeline. In particular, it has focused on two fundamental approaches for addressing a potential combinatorial explosion: genetic algorithms and penalized regression.

Genetic Algorithms

Genetic algorithms have broad applications for optimization, but are especially suitable for cases where one is attempting to optimize a system that may be easily represented as a sequence which may have its state summarized in a simple metric. This makes GAs directly applicable to many situations in proteomics, where one observes a simple function (mass) of a complex underlying system (a protein with potential variations and modifications). One significant problem that is tractable by means of genetic algorithms is the problem of Top Down protein characterization. If, after fragmenting a protein, one has identified both the gene that encoded it and the specific splice form that was produced from it, then in most cases one has fixed the overall sequence length of the protein. The remainder of the characterization work is to determine which set of known and/or unknown cSNPs and PTMs is present on this instance of that particular protein. This overall problem can be potentially quite difficult. It is solved in Bottom Up proteomics by exhaustive enumeration of potential mass shifts, and choosing the combination of mass shifts that results in the best value of the scoring metric that one uses. This approach will simply break down in Top Down proteomics, where the combinatorial explosion results in billions of potential combinations to consider. A GA, however, may be the ideal way of solving this problem in the general case of no limitations on the number of simultaneously applied mass shifts. The set of mass shifts applied to the residues of a protein is trivially encoded as a virtual chromosome. Fragmentation data exhibits excellent locality, where a given fragment in the vast majority of cases represents a proper prefix or suffix of the sequence. This means that it would be simple for the genetic algorithm to converge towards a set of useful 'building blocks', components of the solution that are relatively independent of each other and each of which encode a significant improvement to the scoring metric. There are numerous scoring metrics in proteomics, each of which is designed to reflect the degree to which a given sequence hypothesis fits the data, which is exactly what a GA needs to function. By taking advantage of the tremendous locality of mass spectrometric data, it is possible to use such a GA to convert the exponential enumeration algorithms used by Bottom Up search engines to a heuristic with a constant maximum runtime. The results of this heuristic may then be straightforwardly evaluated in polynomial time to identify regions of potential variation, which enables the algorithm to output explicit bounds on the specificity of the returned solution. Because the algorithm may be designed to operate in constant time with a constant population of candidate sequences, the overall space of evaluated candidate sequences is vastly reduced, with a corresponding increase in statistical power for the score returned with the final output of the algorithm. By leveraging an existing high-throughput identification platform, the amount of time spent on characterization can be vastly decreased, increasing the throughput of analysis even further.

This approach for automated protein characterization for Top Down proteomics forms the first aspect of planned future work. The existing framework provided by ProSight's iterative search logic may be trivially extended in order to support this functionality, providing a useful means of taking the results of a high-throughput identification run to the next level. It would, in many ways, be comparable to the current way that gene-restricted searches work in the context of the iterative search logic; by using a fast, statistically-strong search for identification, one can greatly reduce the search space that is evaluated by a subsequent slow and statistically-weaker characterization stage. In addition, nothing obligates such a system to operate downstream of an identification stage. All that is necessary for efficiency is a small number of initial candidate sequences. While these can be provided by a high-throughput identification pipeline, they may also be provided by the user directly. In this way, the GA-based automatic protein characterization system can act as a core algorithm underlying a powerful standalone characterization system. Such a system would be highly useful for many types of experiments where the user has a purified protein of known provenance, but lacks information about variation or post-translational modification.

In addition to characterizing on a protein sequence resulting from the presence of cSNPs or PTMs, a GA-based characterization system may be used for further analysis. If one localizes a mass shift resulting from a glycosylation event, it is possible to use further analysis to determine the structure of the sugar. If one has an additional level of fragmentation data (MS3) that separates and cleaves the sugar moiety from its bound residue, one can use a GA to characterize the complex branching pattern present in such structures. These branching patterns form a tree structure, which may be trivially represented as an array, which, in turn, can be encoded as a virtual chromosome. By carefully constructing the GA to generate possible branching patterns from a set of possible sugar residues, one can fit the observed fragments of the tree, and the total mass of the tree, to the generated branching patterns.

Genetic algorithms may in fact be the ideal means of dealing with complex feature combinations on large protein sequences. Their bounded runtime and highly effective optimization properties should make them a vital component of any future Top Down analysis toolkit.

Penalized Regression

Penalized regression techniques, as described in [Chapter 3], have broad applicability for variable selection and dealing with ill-conditioned problems. Unlike genetic algorithms, which can optimize a hypothesis to best fit the available data, penalized regression is best suited for variable selection. In other words, given a posited relationship between a multidimensional dataset and some function of it, penalized regression can choose the set of dimensions that best fit that relationship. While this may potentially be useful in direct proteomic applications, such as carrying out exploratory data analysis to determine which information is sufficient to predict some outcome, it is better suited for analyzing the relationships between disparate datasets, such as those between genomic, transcriptomic and proteomic data. [Chapter 3] detailed the development of a system that used penalized regression to identify putative motifs in an overpopulated set of candidates based on their ability to predict gene expression. This is hardly the only possible application of this technique; another, currently under development, is the use of penalized regression to discover chromatin-associated motifs.

It has been hypothesized that just as nucleosome positioning is significantly encoded by the DNA, so too are the patterns of post-translational modification present on those nucleosomes. By extending the MotifExpress framework, it is possible to identify the DNA elements that are significantly correlated with the presence of particular histone PTMs, and thus gathering evidence in support of this hypothesis. The first step is replacing gene expression as the response with a measurement that represents the presence of a PTM of interest. This may be accomplished by using ChIP-chip measurements. By first carrying out chromatin immunoprecipitation for a particular histone PTM, and then hybridizing the selected DNA to a microarray, it is possible to measure the overall presence of a given PTM for the promoter regions of every gene. In particular, a straightforward analysis would be to use an antibody against hyperacetylated histone H4, thus getting good sensitivity and eliminating the specificity problems that plague antibodies raised against specific modified sites. Genes with significant H4 hyperacetylation in their promoter regions (identified in much the same way as previously in MotifExpress) would then be used as a training set for *de novo* motif discovery, creating a set of candidate motifs putatively associated with H4 hyperacetylation. Using these motifs as the explanatory part of a design matrix, and the set of ChIP-chip measurements as the response part, one is capable of then carrying out penalized regression to determine which candidate motifs are significantly correlated with histone hyperacetylation, genome-wide.

However, this is simply the first step. It is known that many histone PTMs, and H4 hyperacetylation in particular, are correlated with gene expression; likewise, overrepresented motifs may be transcription-factor binding motifs, and not H4 hyperacetylation-related motifs. The motifs selected in the manner described above may simply be TFBMs, and may not necessarily be acetylation motifs. In order to identify which motifs are associated strictly with H4 hyperacetylation and not gene expression, a second stage of selection may be employed, utilizing gene expression as the response and the selected subset of putative acetylation motifs as the explanatory variables. This selection stage would identify which, if any, of that subset of motifs are significantly correlated with gene expression, and would then eliminate them, retaining only those which are correlated with only H4 hyperacetylation. In preliminary synthetic tests, such an approach works very well, achieving strong sensitivity and specificity under high noise conditions. However, due to the aforementioned interrelationship between H4 hyperacetylation and gene expression, it may be that all acetylation motifs are correlated with both H4 hyperacetylation and gene expression, and two stages of selection cannot distinguish them; under these conditions, one would need to use a database of known TFBMs to eliminate them from the set of candidates.

Conclusion and Final Words

Computational biology and bioinformatics are still in their infancy. As new means of data collection are invented, and richer datasets are generated, so too will new analysis techniques keep pace. Just as computation and mathematical analysis made physics and chemistry fundamental sciences in the last century, so too will biology achieve the same in this century. Quantity has a quality all of its own, and as datasets become richer and larger, novel analysis techniques will permit us to probe deeper into them, teasing out subtle interactions and building useful models of biological systems. The presence of quantitative models will permit the generation of useful hypotheses and give *in silico* experiments as prominent a place in biology as they hold in chemistry and physics.