

# A Grid Service for Pattern Extraction from Mass Spectrometry Data

Mario Cannataro  
University Magna Græcia of Catanzaro  
88100 Catanzaro, Italy  
cannataro@unicz.it

Pierangelo Veltri  
University Magna Græcia of Catanzaro  
88100 Catanzaro, Italy  
veltri@unicz.it

## Abstract

*The paper presents a Grid Service allowing to detect and extract the longest common sub-spectrum among a set of mass spectrometry spectra data. The service is based on a novel pattern extraction algorithm named LCSS (Longest Common Spectra SubString) that adapts a very popular string matching technique based on Suffix Trees to spectra data. The core of the algorithm and a first performance evaluation of the related Grid Service are discussed.*

**Keywords:** Pattern Extraction, Mass Spectrometry, Suffix Tree, Grid Services

## 1 Introduction

Mass Spectrometry (MS) is a technique allowing to identify the masses of macromolecules in a compound. The mass spectrometer separates gas phase ions according to their  $m/z$  (mass to charge ratio) values [1]. Mass Spectrometry output can be represented as a (large) sequence of value pairs, said spectrum. Each pair contains a measured *intensity*, which depends on the quantity of the detected biomolecule, and a mass to charge ratio ( $m/z$ ), which depends on the molecular mass of the detected biomolecule. Spectra may be affected by errors and noise so they are usually preprocessed before conducting any further data mining or pattern extraction analysis, e.g. classification of diseased vs healthy patients. Classifiers usually separate classes through few peaks. A complementary approach could require that entire sub-spectrum are repeated among samples of the same class, thus it could be important to extract from a set of spectra some common pattern (i.e. a common sub-spectrum) that could be representative of a class.

Pattern extraction is a technique widely used in bioinformatics and especially in protein sequence analysis. String-based pattern extraction works on strings representing protein sequences and a lot of algorithms have been developed for detecting and extracting common sub-strings or sub-sequences, such as those based on Suffix Trees [10]. The

paper presents a novel pattern extraction algorithm allowing to detect and extract the longest common sub-spectrum among a set of MALDI-TOF spectra. The proposed algorithm, named LCSS (Longest Common Spectra SubString), adapts a very popular string matching technique, the Suffix Tree, to spectra data. It has been implemented as a Grid Service and a first performance evaluation is presented. The main reason to use string-based algorithms to face the problem of pattern extraction from spectra is their high efficiency and their suitability to solve a lot of pattern extraction problems, such as exact string matching, common substring, common subsequence, longest common substring, longest common subsequence, repetitive structures in biological data, etc. [10].

On the other hand, the Grid offers the storage and computational power to store huge spectra data and perform spectra preprocessing and analysis algorithms. The Life Science Grid Research Group [9] of the Global Grid Forum [7] aims to investigate how bioinformatics requirements can be fitted and satisfied by Grid services and standards. Emerging BioGrids will leverage bioinformatics tools wrapped as Grid Services as well as semantic middleware services offered by Knowledge Grids [5] [15] and Semantic Grids [2] [6]. The proposed Grid Service goes along this direction. Section 2 introduces MS data. Section 3 presents the proposed pattern extraction algorithm. Section 4 discusses preliminary results, while Section 5 summarizes the paper and describes future work.

## 2 Mass Spectrometry Data

Mass Spectrometry (MS) is a technique allowing to identify the masses of macromolecules in a compound. The mass spectrometer separates gas phase ions according to their  $m/z$  (mass to charge ratio) values [1]. The sample can be inserted directly into the ionization source or can be separated into different components which enter the spectrometer sequentially, e.g. by using Liquid Chromatography (LC). Common ionization techniques are Electrospray Ionization (ESI), Surface Enhanced Laser Desorp-

tion/Ionization (SELDI), and Matrix-Assisted Laser Desorption/Ionization (MALDI), coupled with different kind of mass analyzers such as Time Of Flight (TOF) or quadrupole ion traps. Mass Spectrometry output can be represented as a (large) sequence of value pairs (spectrum). Each pair contains a measured *intensity*, which depends on the quantity of the detected biomolecule, and a mass to charge ratio ( $m/z$ ), which depends on the molecular mass of the detected biomolecule.

Mass spectrometry is emerging as an important tool for biomarker discovery. Body fluids as well as tissues can be routinely used to generate protein profiles, containing potential disease markers whether individual proteins or sets of interacting proteins [13]. Data Mining is commonly used to discover biomarker patterns in spectra data [8], but a number of technical challenges need to be faced, among which is the extremely high-dimensionality of mass spectra. A typical mass spectrum has several thousands of attributes that exhibit a high degree of spatial redundancy. The exact number depends on the type of mass spectrometry instrument that is used, its resolution, and the mass range it covers.

Spectra preprocessing aims to correct intensity and  $m/z$  values in order to reduce noise, reduce the amount of data, and make spectra comparable [3], [12], [11], [14]. *Binning* performs data dimensionality reduction by aggregating measured data into *bins*: a set of peaks from a spectrum is substituted with a unique peak ( $I, m/z$ ), whose intensity  $I$  is an aggregate function of the original intensities (e.g. their sum), and the mass  $m/z$  is usually chosen among the original mass values (e.g. the median value). *Peaks alignment* corrects errors on  $m/z$  measurements finding a common set of peak locations in a set of spectra, in such a way that all aligned spectra will have common  $m/z$  values for the same biological entities. Moreover, a small number of individuals are chosen to be included in clinical study, so pattern extraction has to deal with the problem of high dimensionality small sample size. Indeed, spectra can yet exhibit hundreds or thousands of peaks after preprocessing, thus string-based pattern extraction can complement data mining methods.

### 3 String-based Spectra Pattern Extraction

The main idea of the paper is to apply a string-based pattern matching algorithm, the Longest Common Substring algorithm based on Suffix Tree [10], to spectra data. After recalling Suffix Trees the proposed LCSS algorithm is described.

#### 3.1 Background on Suffix Trees

Any string of length  $m$  can be represented by its  $m$  suffixes, and these suffixes can be stored in a data structure said

Suffix Tree (ST). Given the string ‘mississippi’, ‘miss’ is a prefix, ‘ippi’ is a suffix, and ‘issi’ is a substring. Note that a substring is a prefix of a suffix.

If  $T = t_1, t_2, \dots, t_i, \dots, t_m$  is a string, then  $T_i = t_i, t_{i+1}, \dots, t_m$ , is the suffix of  $T$  that starts at position  $i$ . All suffixes of ‘mississippi’ are showed in Figure 1.

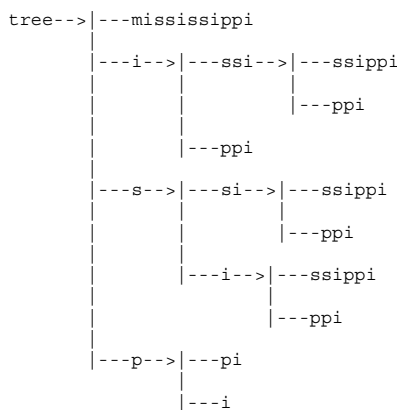
```
T1 = mississippi = T
T2 = ississippi
T3 = sissippi
T4 = sissippi
T5 = issippi
T6 = ssippi
T7 = sippi
T8 = ippi
T9 = ppi
T10 = pi
T11 = i
T12 = (empty)
```

Figure 1. All suffixes of ‘mississippi’

Creating the ST requires time  $O(m)$  and searching for a pattern  $P$  in it requires time  $O(n)$ , where  $n$  is the length of the pattern. An important aspect is the possibility to decouple the creation of the ST from the search of the pattern. These properties make the suffix tree an interesting data structure for implementing many string-based applications, e.g. multiple genome alignment in bioinformatics. The first linear-time suffix tree algorithm was developed by Weiner in 1973, after McCreight realized a more space efficient algorithm in 1976, and Ukkonen produced an “online” variant of it in 1995 [10]. The key to search speed in a suffix tree is that there is a path from the root for each suffix of the text. This means that at most  $n$  comparisons are needed to find a pattern of length  $n$ .

If the non-empty suffixes are sorted it is evident that some of them (may) share common prefixes. Here there are substrings starting with ‘i’, ‘m’, ‘p’ and ‘s’, but all of those starting ‘is’, in fact start ‘issi’. Two or more common prefixes share a common path from the root of the suffix tree. Now, a search (sub)string  $P$  must be a prefix of a suffix of  $T$ , if it occurs in  $T$ . The suffix tree for ‘mississippi’ is showed in Figure 2.

When there is more than one string to be searched a Generalized Suffix Tree (GST) can be created [10]. As the GST contains more than one string, each leaf node contains an identifier indicating the string associated with this suffix, and a second identifier indicating the position of the suffix. The Longest Common Substring (LCS) of two strings,  $T_1$  and  $T_2$ , can be found by building a generalized suffix tree for  $T_1$  and  $T_2$ : each node is marked to indicate if it represents a suffix of  $T_1$  or  $T_2$  or both. The deepest node marked for both  $T_1$  and  $T_2$  represents the longest common substring. Equivalently, one can build a (basic) suffix tree for the string  $T_1\$T_2\#$ , where ‘\$’ is a special terminator for  $T_1$  and ‘#’ is a special terminator for  $T_2$ . The longest com-



**Figure 2. The Suffix Tree for ‘mississippi’**

mon substring is indicated by the deepest fork node that has both ‘...\$...’ and ‘...#...’. Note that the ‘longest common substring problem’ is different to the ‘longest common subsequence problem’: an instance of a subsequence can have gaps where it appears in  $T_1$  and in  $T_2$ , but an instance of a substring cannot have gaps. The algorithm for two strings can easily be extended to more strings.

### 3.2 The LCSS Algorithm

Since spectra data are couples of real numbers, (*intensity, m/z*), a first problem is to transform spectra into strings so that suffix trees and generalized suffix trees can be computed. The simple approach proposed here is to sample intensity values mapping them onto discrete values each one corresponding to a letter of an alphabet. Let be  $A$  an alphabet of  $N$  symbols, and  $maxIntensity$  and  $minIntensity$  the maximum and minimum intensity values, the intensity domain can be partitioned into  $N$  quantization intervals  $[minIntensity + (i * K), minIntensity + ((i + 1) * K)]$ ,  $i = 0, \dots, N - 1$ , where  $K = (maxIntensity - minIntensity) / N$  is the quantization level. Thus, a peak of intensity  $I$  it can be associated to the symbol  $A_j$ , if  $I$  belongs to the  $j$ th quantization interval. Figure 3 shows the quantization procedure when the ASCII coded alphabet is used.

To effectively compare different spectra whose intensities are represented through strings spectra must be aligned, i.e. the same molecular component need to have the same mass on each sample. Thus, before intensities can be quantified, spectra must be aligned with respect to masses. The work [3] surveys some recent spectra preprocessing algorithms including spectra alignment.

We name *Longest Common Spectra Substring* (LCSS) the algorithm that finds the longest common sub-spectrum in a set of mass spectra. Its pseudo-code is showed in Fig-

```

spectrum2string(N: integer; S: spectrum): spectrum
Input: N; //size of the alphabet, e.g. 64, 128, 256
      S; //input spectrum
Output: T; //converted spectrum

{
  Compute maxIntensity, minIntensity;
  K=(maxIntensity-minIntensity)/N
  for i=0 to length(S) {
    T[i].intensity = char(S[i].intensity / K)
    T[i].mass = S[i].mass
  }
  return T;
}
  
```

**Figure 3. The quantization procedure**

ure 4. The current version of the algorithm compares two spectra  $S_1$  and  $S_2$ . After the two spectra are converted into strings  $T_1$  and  $T_2$ , the proposed algorithm builds the GST and then the Longest Common Substring (LCS) is computed by finding the deepest node marked for both  $T_1$  and  $T_2$ . Since the algorithm provides the position of the LCS in  $T_1$  and  $T_2$ , namely  $i$  and  $j$ , it is possible to extract the corresponding sub-spectrum from each spectrum  $S_1$  and  $S_2$  simply selecting the peaks (intensity,  $m/z$ ) starting respectively at positions  $i$  and  $j$ . The two sub-spectrum can be compared to find how the quantization error impacts on the length of discovered sub-spectrum. Moreover, since the position of the sub-spectrum in both spectra deals with masses of molecules, the significance of the discovered common pattern can be evaluated with respect to the positions  $i$  and  $j$ . Very close positions mean that the common pattern refers to common masses, whereas very different positions may have few of significance.

```

LCSS(N: integer; S1, S2: spectrum; l, i, j: integer)
Input: N; //size of the alphabet, e.g. 64, 128, 256
      S1, S2; //input spectra
Output: l; //length of the common sub-spectrum
        i, //position of the common sub-spectrum in S1
        j; //position of the common sub-spectrum in S2

{
  T1 = spectrum2string(N, S1);
  T2 = spectrum2string(N, S2);
  GST = GenSuffixTree(T1, T2); //build the GST
  LCS (l, i, j); //find the longest common substring in GST;
}
  
```

**Figure 4. The LCSS algorithm**

## 4 Performance Evaluation

The LCSS algorithm has been implemented as a Globus Grid Service and deployed on a Grid. A first performance evaluation regards the behavior of the core algorithm when used to find the longest common sub-spectrum on a real spectra dataset. In such study we measured respectively

the LCSS execution time, the length and the positions of the discovered LCSS, and the difference (error) between the sub-spectrum evaluated on the first and the second input spectra, after recovering real intensity values. Such measurement were taken varying the quantization parameter  $N$  (i.e. the alphabet size).

The second performance study regarded the response time of different instances of the algorithm wrapped as a Grid Service deployed on a small Grid composed by three computers. Execution times and spectra transfer times were measured considering one Grid Service receiving two input spectra from, respectively, 1, 3 and 6 clients installed on such a Grid.

### 4.1 Algorithm Evaluation

The input dataset is obtained by two set of experiments. In the first experiment, a biological human serum has been processed by a MALDI-TOF mass spectrometer obtaining the  $A_1$  spectrum. Such processing has been repeated four times obtaining the spectra  $A_2$ ,  $A_3$ ,  $A_4$ , and  $A_5$ . Although they are replicas of the same sample, their values are not exact replicas of the  $A_1$  values, due to noise and instrument perturbation. In the second experiment, the same serum of the first experiment has been mixed with two known proteins and, using the same approach, the spectrum  $B_1$  and replicas  $B_2$ ,  $B_3$ ,  $B_4$ , and  $B_5$ , have been obtained. The adding of two proteins to the original serum produces some perturbation on the spectrum, yet maintaining some common sub-spectrum. Each spectrum contains 34671 peaks. The masses range between 3999.749927 and 20000.324102 Dalton. Spectra  $A_1$  and  $B_1$  are respectively showed in Figures 5 and 6. The discovered LCSS is indicated by a circle. Table 1 shows the maximum and minimum intensity values for the ten spectra.

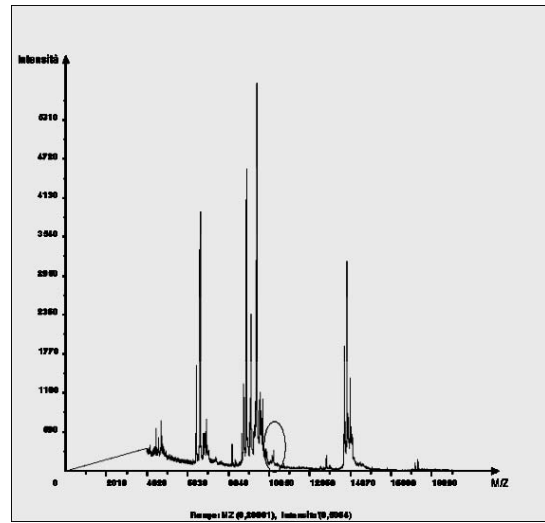


Figure 5. Spectrum A1

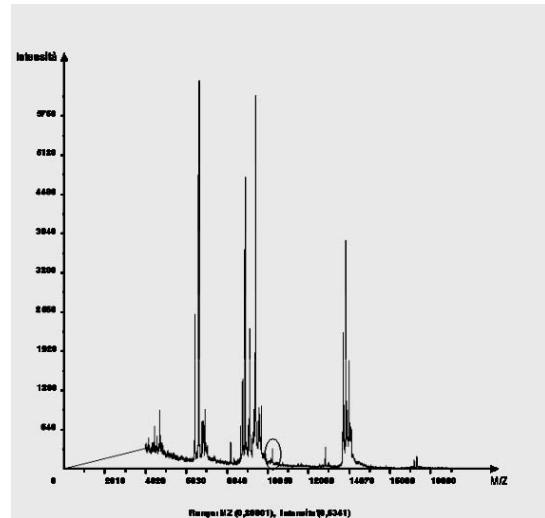


Figure 6. Spectrum B1

Table 1. Max/Min peak intensities

Spectrum	maxIntensity	minIntensity
A1	5863.23	5.01483
A2	8024.88	10.5301
A3	7211.13	4.62012
A4	6796.67	2.69962
A5	5186.37	3.63926
B1	8059.8	6.3976
B2	6340.75	5.03147
B3	8454.74	9.80456
B4	7598.2	7.88486
B5	8359.63	9.69859

Table 2 shows for the couple of spectra  $A_1$  and  $B_1$  respectively the LCSS length, its initial positions in  $A_1$  and  $B_1$ , namely  $i$  and  $j$ , the gap  $|i - j|$ , and 2-norm average relative error, when varying  $N$ . Similar results are obtained when searching the LCSS on the remaining spectra.

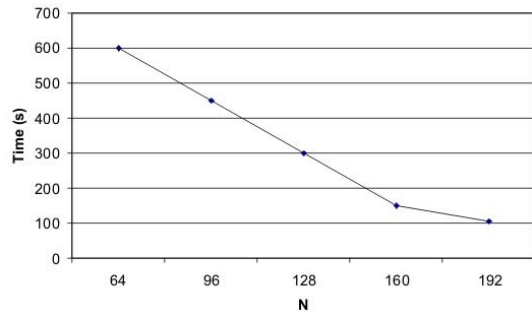
Figure 7 shows the LCSS execution time when varying  $N$ . Execution time decreases when the quantization level

increases. In fact, for lower values of  $N$  the alphabet comprises a small set of characters so the strings obtained by the source spectra contain many similar characters. This increases the time requested to find the LCSS on the GST. Instead, an higher value of  $N$ , i.e. a richer alphabet, simplifies the search of the LCSS on the GST since there are less identical characters.

On the other hand, a low  $N$  introduces great approximation and quantization error, so it happens that for low values of  $N$  the length of the LCSS increases (due to a worse quantization), the positions  $i$  and  $j$  of the sub-spectrum in the input spectra tend to be greatly unaligned and finally, the difference (error) between the sub-spectrum computed on the two spectra increases. On the contrary, large  $N$  in-

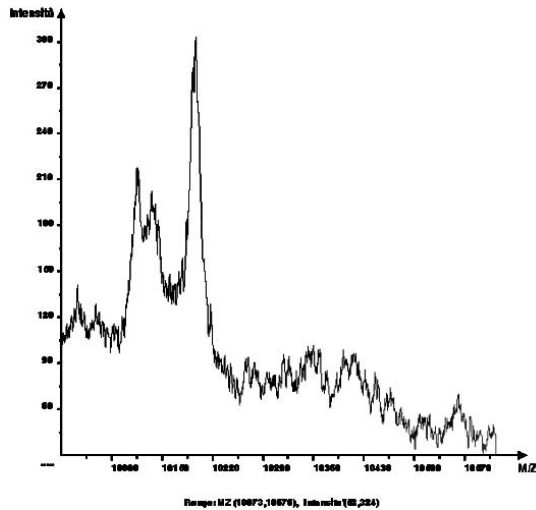
**Table 2. Characteristics of LCSS varying N**

N	LCSS len	i	j	i - j	IAR	2AR
64	4236	17992	17989	3	142.5866	11.94096
96	1616	20060	19584	476	140.5634	11.85594
128	495	22715	21936	779	130.4989	11.423612
160	121	18762	18008	754	102.2511	10.11193
192	30	20063	20060	3	102.363	10.11746



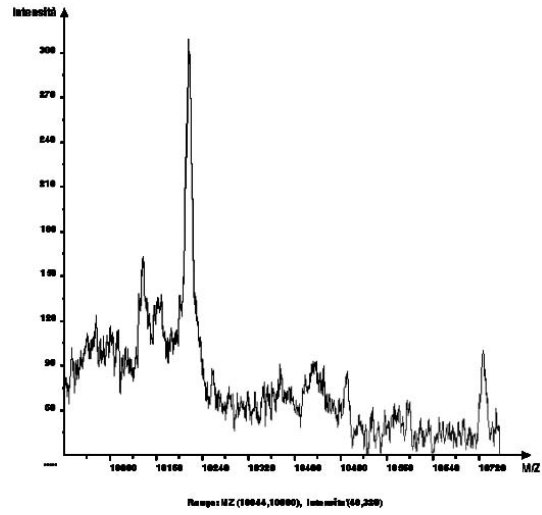
**Figure 7. LCSS execution times**

increases precision (i.e.  $i$  and  $j$  alignment), reduces errors, but discover shorten sub-spectra. Figures 8 and 9 respectively show the details of the LCSS discovered in ( $A_1$ ,  $B_1$ ) for  $N = 192$ . It is possible to see a very high similarity between the sub-spectrum on each input spectra.



**Figure 8. Detail of LCSS on spectrum  $A_1$**

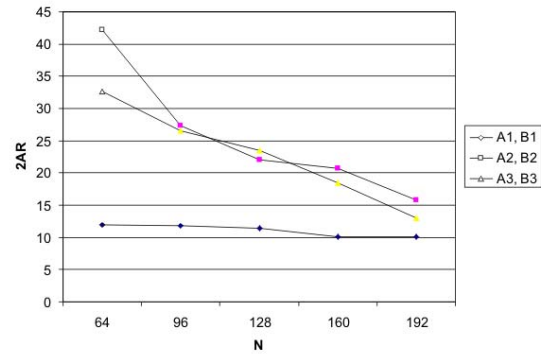
Figure 10 shows the 2-norm average relative (2AR) error when varying  $N$ . Let  $k$  be the length of the LCSS, the sub-spectrum starting on position  $i$  on the  $A_1$  spectrum is  $Y = [y_1, \dots, y_k]$ , while the sub-spectrum starting on position  $j$  on the  $A_1$  spectrum is  $Y' = [y'_1, \dots, y'_m]$ . The 2-norm average



**Figure 9. Detail of LCSS on spectrum  $B_1$**

relative (2AR) error is defined as:

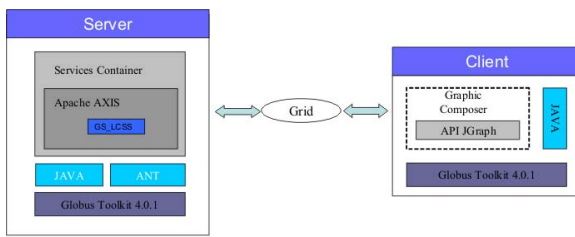
$$2AR = \sqrt{\frac{1}{k} \sum_{i=1,k} \left( \frac{|y_i - y'_i|}{y_i} \right)^2}$$



**Figure 10. Error vs Alphabet Size**

## 4.2 Grid Service Evaluation

The basic environment used for the evaluation on the Grid comprises a Grid Service implementing the LCSS algorithm and a client able to load two spectra from the file system (e.g.  $A_1$  and  $B_1$ ), send them to the server and receiving the discovered LCSS (see Figure 11). We conducted a set of experiments varying the number of clients that contemporarily sent search requests to just one LCSS Grid Service. The number of clients  $C$  was respectively 1, 3, and 6.



**Figure 11. LCSS Grid Service and Client**

Table 3 shows respectively the average times needed to send the spectra ( $T_s$ ) to the Grid Service, to execute the LCSS algorithm ( $T_e$ ), to receive the results ( $T_r$ ), i.e. length and position of the discovered LCSS, on the client, and the overall response time ( $T$ ). It is possible to note that although the number of clients increases from 1 to 6, the overall response time per search increases only slightly: this is due to the parallel multi-threaded implementation of the LCSS Grid Service.

**Table 3. LCSS Execution Time**

C	$T_s$	$T_e$	$T_r$	T
1	0.1405	588.745	0.0131	588.8975
3	0.1825	669.627	0.02295	669.837
6	0.2695	803.654	0.025	803.9485

## 5 Conclusions and Future Work

The main contribution of the proposed system is the combination of a very popular string matching technique based on Suffix Trees with functions for the preprocessing and management of mass spectrometry data. The first performance results are encouraging so future work will regard the extension of the algorithm to face more than two spectra and to allow the contemporary detection of more sub-spectra, and its combination with spectra preprocessing techniques. The proposed Grid Service will be made available through MS-Analyzer [4], a software platform for spectra management and analysis developed at University of Catanzaro, Italy.

## References

- [1] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, 13 March 2003.
- [2] M. Cannataro. Next-generation Grids: requirements and knowledge-based services. *Concurrency and Computation: Practice and Experience*, 18(8):887–898, 2006.
- [3] M. Cannataro, P. Guzzi, T. Mazza, G. Tradigo, and P. Veltri. Preprocessing of mass spectrometry proteomics data on the grid. In *CBMS'05*, pages 549–554, 2005.
- [4] M. Cannataro, P. H. Guzzi, T. Mazza, G. Tradigo, and P. Veltri. Using ontologies for preprocessing and mining spectra data on the Grid. *Future Generation Computer Systems*, in press, 2006.
- [5] M. Cannataro and D. Talia. KNOWLEDGE GRID An Architecture for Distributed Knowledge Discovery. *Communication of ACM*, 46(1), 2003.
- [6] D. De Roure, N.R. Jennings, and N.R. Shadbolt. The semantic grid: A future e-science infrastructure. In F. Berman, G. Fox, and A. J. G. Hey, editors, *Grid Computing - Making the Global Infrastructure a Reality*, pages 437–470. John Wiley and Sons Ltd., 2003.
- [7] Global Grid Forum. - <http://www.gridforum.org>.
- [8] V. Gopalakrishnan, E. William, S. Ranganathan, R. Bowser, M. E. Cudkovic, M. Novelli, W. Lat-tazi, A. Gambotto, and B. W. Day. Proteomic data mining challenges in identification of disease-specific biomarkers from variable resolution mass spectra. In *SIAM Bioinformatics Workshop*, 2004.
- [9] Life Science Grid. - <http://forge.gridforum.org/projects/lsg-rg>.
- [10] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge Univ Press, 1997.
- [11] N. Jeffries. Algorithms for alignment of mass spectrometry proteomic data. *Bioinformatics*, 21(14):3066–3073, 2005.
- [12] J. W. H. Wong, G. Cagney, and H. M. Cartwright. Specalign - processing and alignment of mass spectra datasets. *Bioinformatics*, 21(9):2088–2090, 2005.
- [13] B. Wu, T. Abbott, D. Fishman, W. McMurray, G. Mor, K. Stone, D. Ward, K. Williams, and H. Zhao. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics*, 1(19):1636–43, September 2003.
- [14] Y. Yasui, D. McLerran, BL. Adam, M. Winget, M. Thornquist, and Z. Feng. An automated peak identification/calibration procedure for high-dimensional protein measures from mass spectrometers. *Journal of Biomedicine and Biotechnology*, 1(4):242–248, 2003.
- [15] H. Zhuge. China's E-Science Knowledge Grid Environment. *IEEE Intelligent Systems*, 19(1):13–17, 2004.