BMC Bioinformatics

SOFTWARE Open Access

A parallel method for enumerating amino acid compositions and masses of all theoretical peptides

Alexey V Nefedov and Rovshan G Sadygov*

Abstract

Background: Enumeration of all theoretically possible amino acid compositions is an important problem in several proteomics workflows, including peptide mass fingerprinting, mass defect labeling, mass defect filtering, and de novo peptide sequencing. Because of the high computational complexity of this task, reported methods for peptide enumeration were restricted to cover limited mass ranges (below 2 kDa). In addition, implementation details of these methods as well as their computational performance have not been provided. The increasing availability of parallel (multi-core) computers in all fields of research makes the development of parallel methods for peptide enumeration a timely topic.

Results: We describe a parallel method for enumerating all amino acid compositions up to a given length. We present recursive procedures which are at the core of the method, and show that a single task of enumeration of all peptide compositions can be divided into smaller subtasks that can be executed in parallel. The computational complexity of the subtasks is compared with the computational complexity of the whole task. Pseudocodes of processes (a master and workers) that are used to execute the enumerating procedure in parallel are given. We present computational times for our method executed on a computer cluster with 12 Intel Xeon X5650 CPUs (72 cores) running Windows HPC Server. Our method has been implemented as a 32- and 64-bit Windows application using Microsoft Visual C++ and the Message Passing Interface. It is available for download at https://ispace.utmb.edu/users/rgsadygo/Proteomics/ParallelMethod.

Conclusion: We describe implementation of a parallel method for generating mass distributions of all theoretically possible amino acid compositions.

Background

Mass spectrometry (MS) plays a crucial role in modern proteomics as a key method for protein identification and quantification. MS provides accurate mass and abundance measurements of intact and fragmented peptide ions, which are then processed by specialized algorithms and transformed into peptide and protein identities. Thus, efficiency of many MS-based proteomics workflows depends on how well we understand – and can utilize – the properties of peptide masses and peptide mass distribution.

It has been observed that peptide masses have a nonuniform, clustered distribution, which is explained by the fact that peptides are made of twenty amino acids with specific masses. This distribution consists of repeating peaks separated by approximately 1 Da, which become taller and wider as the mass increases. Consecutive peaks are separated by low populated regions (quiet zones) and gaps (forbidden zones)-that is, the mass ranges for which there exist no possible sequences of amino acids. Nonuniformity (peaks, gaps) and discrete nature of the mass distribution of peptides are important for two major problems in MS-based proteomics: peptide identification and *de novo* sequencing.

The knowledge of the mass distribution of a particular type of peptide (for example, non-modified tryptic peptides) can be used to facilitate peptide identification in a

* Correspondence: rovshan.sadygov@utmb.edu
Department of Biochemistry and Molecular Biology, Sealy Center for Molecular Medicine, University of Texas Medical Branch, 301 University Blvd, Galveston, TX 77555, USA

BioMed Central

© 2011 Nefedov and Sadygov; licensee BioMed Central Ltd. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

number of ways. Forbidden zones allow us to filter out MS signals corresponding to non-target species (non-peptide contaminants or modified peptides) early on, before doing any complicated processing of MS data. Dodds and coworkers [1] showed that this results in exponential improvements in statistical significance and discrimination of protein identification based on peptide mass fingerprinting on the Mascot platform. Nonoverlapping or partially overlapping peaks in the mass distributions of different types of peptides allow recognition of these types based solely on precursor masses. For example, Spengler and Hester [2] showed that accurate masses (with accuracy of 0.1 or even 1 ppm) allow phosphorylated and nonmodified peptides to be distinguished. Lehmann and coworkers [3] and Jones and coworkers [4] showed that this is possible for glycopeptides and lipids. In addition, there have been many suggestions for label tags shifting the mass of labeled peptides to quiet or forbidden zones in order to allow easier identification and quantification of these peptides [5].

The major drawback of peptide identification algorithms based on database search is their inability to identify peptides that are not present in the reference database. *De novo* sequencing algorithms are designed to restore peptide compositions from MS data without the use of peptide databases. These algorithms employ several strategies for MS data analysis s [6], one of which is based on the fact that for a given mass there exist only a finite (though sometimes very large) number of amino acid sequences (or amino acid compositions) that can assume that mass, and that these sequences (compositions) can be explicitly enumerated. The use of the masses of fragment ions can further reduce the number of admissible compositions. Several reports have shown the feasibility of this strategy, especially for high accuracy data provided by modern Fourier transform mass spectrometers [7-9].

Proteomics applications mentioned above rely on specific properties of the peptide mass distributions that can only be obtained by enumerating all theoretically possible peptides. Moreover, in many circumstances it is impossible to generate these distributions once and for all, as many parameters can vary from experiment to experiment (peptide modifications, enzymatic specificity, number of missed cleavages, etc.) Thus, it is desirable to be able to generate peptide mass distributions (or some parts of these distributions) "to order" and, therefore, to be able to generate them fast.

Several works focusing on different MS-based proteomics applications employed enumeration of all theoretically possible peptides [8,10-13]. Because of the high computational complexity of the task, enumeration of peptides was done for the mass range below 2 kDa, which limited applicability of the obtained results. Also, even for this mass range long computational times and extensive computational capabilities were often required. Olson and others [8] mentioned the use of a parallel method for peptide enumeration, but details of its implementation as well as its computational performance were not reported.

In a recent paper [14] we described the mass distribution of all theoretically possibly tryptic peptides made of 20 amino acids, up to the mass of 3 kDa. The paper provided detailed characterization of forbidden zones and amino acid compositions of peptides from the quiet zones. We showed how forbidden zones shrink over the mass range, where they completely disappear and how they depend on the measured mass accuracy. We found that peptide sequence compositions in the quiet zones are less diverse than those in the peaks of the distribution, and that forbidden zones may be extended by eliminating certain types of unrealistic compositions. We also characterized symmetry of mass peaks and the accuracy of the Mann's equations [13] for the mass peak position and width. Our study was made possible by advancing computational techniques for the enumeration of amino acid compositions.

In this paper, we describe in detail a parallel method for enumerating all amino acid compositions up to a given length. First, we present a pseudocode for recursive procedures which are the core of this method. We then show how a single task of enumerating all peptide compositions can be divided into smaller subtasks that can be executed in parallel. We also show how the computational complexity of these subtasks compares with the computational complexity of the primary task. Finally, we provide pseudocode of processes (a master and workers) that are used to execute the enumerating procedure in parallel. To the best of our knowledge this is the first description of a computational method for a complete and unbiased enumeration of all theoretically possible peptides. We present computational times for our method, implemented by using Microsoft Visual C++ and the Message Passing Interface (MPI), and executed on a computer cluster with 12 Intel Xeon X5650 CPUs running Windows HPC Server 2008. The mass and length limits are input parameters of the program.

Implementation

Peptide compositions

Any peptide composition is represented by a numerical vector $(n_1, n_2,..., n_{20})$, whose i-th component is equal to the number of times the i-th amino acid occurs in the peptide. For example, sequence $a_1a_{20}a_1a_1$ has composition (3, 0,..., 0, 1). In some cases, it is convenient to consider peptides as sequences composed of less or more than 20 letters (tryptic peptides without missed

cleavages, post-translationaly modified peptides, etc.). For this reason, let us adopt a more general notation: assume we have an alphabet of N characters and composition vectors (n_1, n_2,..., n_N). The length of a composition is defined as $L = n_1 + n_2 + ... + n_N$. If m_i is the monoisotopic mass associated with the i-th letter, then the monoisotopic mass of a composition is defined as $m = n_1m_1 + n_2m_2 + ... + n_Nm_N$ (the monoisotopic mass of H_2O and a proton may be added to this mass if necessary.)

For a single sequence of letters we have a uniquely defined composition, while for a single composition of length L we have

$$\frac{L!}{n_1!n_2!\ldots n_N!} \tag{1}$$

corresponding sequences, given by the multinomial coefficient. Note that all these sequences will have the same mass, which explains the convenience of enumerating peptide compositions instead of peptide sequences in order to obtain all theoretically possible peptide masses.

The number of compositions of length L is equal to the number of ways to choose L elements from a set of N elements if repetitions are allowed:

$$\binom{N+L-1}{L} = \frac{(N+L-1)!}{L!(N-1)!}. \tag{2}$$

The number of compositions of all lengths not greater than L (including one composition of length 0) is equal to

$$\sum_{k=0}^{L}\binom{N+k-1}{k} = \binom{N+L}{N} \tag{3}$$

which follows from the equation

$$\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}.$$

The latter is based on the recurrence relation

$$\binom{r-1}{k} + \binom{r-1}{k-1} = \binom{r}{k},$$

and can be found in the book by Graham and others [15]. The number of sequences of all lengths not greater than L is equal to

$$\frac{N(N^L-1)}{N-1}.$$

Table 1 shows the number of compositions and sequences for peptides comprised of 20 amino acids. Note how the number of sequences exceeds the number of compositions as the length of peptides grows.

Enumerating peptide compositions

Figure 1 shows the pseudocode of a basic recursive procedure for enumerating all compositions of length not greater than L for an alphabet of N letters. Array c holds current composition (n_1, n_2,..., n_N) and is indexed from one. Procedure Mass returns the mass of the input composition c. For a given length L, procedure GenBasic should be called with parameters (L, *start* = 1). Note that the depth of recursion for this procedure is equal to N - 1. The number of compositions enumerated by this procedure is given by equation (3).

Procedure GenBasic begins enumeration with composition (0, 0,..., 0) and first generate all compositions with n_N ranging from 0 to L. It then sets n_{N-1} to 1, and generates all compositions with n_N ranging from 0 to L - 1, and so on. The last composition in this generation process is (L, 0,..., 0). Essentially, the compositions are generated like N-digit numbers, in ascending order, with requirement that the sum of the "digits" must not be greater than L. For instance, for N = 3 and L = 2 the procedure generates all compositions up to length 2 in the following order: (0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 1, 0), (0, 1, 1), (0, 2, 0), (1, 0, 0), (1, 0, 1), (1, 1, 0), (2, 0, 0).

Several changes to procedure GenBasic will make it faster. First, if L is equal to zero on line 3 then there is

Table 1 Number of compositions and sequences comprised of 20 letters, of length not greater than *L*, for *L* ranging from 3 to 10, and their ratios (rounded)

Length of Peptides (*L*)	Number of Compositions (*A*)	Number of Sequences (*B*)	Ratio *B*/*A*
3	1,770	8,420	5
4	10,625	168,420	16
5	53,129	3,368,420	63
6	230,229	67,368,420	293
7	888,029	1,347,368,420	1,517
8	3,108,104	26,947,368,420	8,670
9	10,015,004	538,947,368,420	53,814
10	30,045,014	10,778,947,368,420	358,760

```
1   procedure GenBasic(L, start)
2     if start < N then
3       for i ← 0 to L
4         c[start] ← i
5         GenBasic(L – i, start + 1)
6     else
7       for i ← 0 to L
8         c[N] ← i
9         m ← Mass(c)
10        print m
```

Figure 1 Pseudocode for recursive procedure GenBasic which enumerates all compositions of length not greater than *L* and prints their masses.

no need to make assignment on line 4 and call GenBasic on line 5, since it is already known that the rest of the composition will contain zeros only. Second, we can calculate the mass of a composition as soon as its component n_i becomes known, and then pass this mass to the next call of the generating procedure. By doing this, we avoid the need to recalculate the mass of the part of the composition that has not been changed.

Figure 2 shows the pseudocode of procedure Gen which is a faster version of procedure GenBasic. It generates a histogram of peptide compositions' masses, instead of printing them, which is more suitable for its further use. The histogram, stored in global array *massHist*, contains the number of compositions falling into the mass bins of width 0.001 Da. Procedure Round returns the rounded integer value of its argument. Note that since procedure Gen calculates the mass of compositions "on the fly", we do not need to store compositions in array *c*, so lines 10 and 16 may be removed. We assume that array *aam* of size *N* stores masses $m_1, m_2,..., m_N$. Procedure Gen should be called with parameters (*L*, *start* = 1, m_0 = 0).

Enumerating peptide compositions in parallel

The task of enumerating all compositions ($n_1, n_2,..., n_N$) can be split into smaller independent subtasks or jobs that can be executed in parallel. Indeed, a single call to procedure Gen with parameters (*L*, 1, 0) is equivalent to *L*+1 calls with parameters (*L*, 2, 0), (*L* - 1, 2, *aam*[1]),..., (0, 2, *aam*[1]**L*), while n_1 is set to 0, 1,..., *L*,

```
1   procedure Gen(L, start, m0)
2     comment: global array massHist has been initialized with zero values
3     if start < N then
4       if L = 0 then
5         k ← Round(m0*1000)
6         massHist[k] ← massHist[k] + 1
7       else
8         m ← m0 – aam[start]
9         for i ← 0 to L
10          c[start] ← i
11          m ← m + aam[start]
12          Gen(L – i, start + 1, m)
13    else
14      m ← m0 – aam[start]
15      for i ← 0 to L
16        c[N] ← i
17        m ← m + aam[N]
18        k ← Round(m*1000)
19        massHist[k] ← massHist[k] + 1
```

Figure 2 Pseudocode for recursive procedure Gen, a faster version of GenBasic. The procedure generates the mass histogram of all compositions of length not greater than *L*.

correspondingly (Figure 3). As before, we assume that array *aam* stores masses m_1, m_2,..., m_N of the used amino acids. Certainly, we will have to combine mass histograms produced by each call of procedure Gen, which can be done knowing parameters of each job, described by a triplet (L, *start*, m_0).

To illustrate this idea, consider again our example with N = 3 and L = 2. The primary task is to enumerate the following compositions: (0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 1, 0), (0, 1, 1), (0, 2, 0), (1, 0, 0), (1, 0, 1), (1, 1, 0), (2, 0, 0). This can be accomplished by independent enumeration of three subsets of compositions: (i) (0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 1, 0), (0, 1, 1), (0, 2, 0); (ii) (1, 0, 0), (1, 0, 1), (1, 1, 0); and (iii) (2, 0, 0). Compositions (i) can be enumerated by setting n_1 = 0 and calling Gen with parameters (L = 2, *start* = 2, m_0 = 0); compositions (ii) can be enumerated by setting n_1 = 1 and calling Gen with parameters (L = 1, *start* = 2, $m_0 = m_1$); and single composition (iii) is enumerated by setting n_1 = 2 and calling Gen with parameters (L = 0, *start* = 2, $m_0 = 2m_1$).

How can we create a list or table of jobs given the initial job described by parameters (L, 1, 0)? First, job (L, 1, 0) is replaced by L+1 jobs (L, 2, 0), (L - 1, 2, *aam*[1]),..., (0, 2, *aam*[1]**L*) (Figure 3). If, for a given L, job (L, 2, 0) is executed in acceptable time, we do not need to do anything else, and the table of jobs has been initialized. Otherwise, we can split job (L, 2, 0) into L+1 jobs with *start* = 3, and similarly split other jobs with *start* = 2. Thus, for all jobs with *start* = 2 there is certain $L_{\max,2}$ such that if the first parameter of the job is larger than $L_{\max,\,2}$ then this job should be split into jobs with *start* = 3. When this is done, we move to the jobs with *start* = 3 and process them in a similar manner: all jobs that have first parameter larger than $L_{\max,3}$ should be split into jobs with *start* = 4. We continue this until each job in the job table can be executed in acceptable time (see additional notes on this in the Discussion section).

When the table of jobs has been initialized, the jobs from the table can be assigned to computation processes. In this context, it is convenient to think about a master process, which does these assignments (Figure 4), and worker processes (Figure 5), which execute the assigned jobs and return results back to the master. The master then combines partial mass histograms computed by workers into a single final mass histogram. There may be different strategies utilized in assigning the jobs. For example, larger jobs (with larger L) may be assigned prior to smaller jobs (with smaller L). In our experiments, which are presented in the Discussion section, there were no particular strategy in job assignments (jobs were assigned in the order in which they had been inserted into the job table).

The data exchange between the master and workers (Figure 4, lines 12, 16; Figure 5, lines 2, 5, 6) can be organized by using functions MPI_Send and MPI_Receive from any library implementing MPI [16]. In our implementation, we used Microsoft Visual C++ and MPI library from Microsoft HPC SDK Pack.

Results and Discussion

It is worthwhile to make several additional comments on procedure Gen, presented on Figure 2. Various practical considerations may suggest using an upper limit on the mass of peptide compositions that one wants to enumerate. In this case, a significant improvement in computation speed may be achieved by canceling the enumeration of compositions whose mass exceeds a given limit. If array *aam* contains mass values in ascending order, we can return from function Gen as soon as the current mass (m_0 in line 5, m in lines 11 and 17) exceeds the threshold. To illustrate a possible gain in speed that may be achieved by using a maximum mass limit, consider enumeration of compositions corresponding to all tryptic peptides up to the length of 30. It takes 1 hour 20 minutes to complete the full enumeration of such compositions, while with the mass

start = 1	*start* = 2		*start* = 3
Gen(L, 1, 0) →	$n_1 = 0$, Gen(L, 2, 0)	→	$n_1 = 0, n_2 = 0$, Gen(L, 3, 0)
	$n_1 = 1$, Gen($L - 1$, 2, *aam*[1])		$n_1 = 0, n_2 = 1$, Gen($L - 1$, 3, *aam*[2])
	$n_1 = 2$, Gen($L - 2$, 2, *aam*[1]*2)		$n_1 = 0, n_2 = 2$, Gen($L - 2$, 3, *aam*[2]*2)
	...		...
	$n_1 = L$, Gen(0, 2, *aam*[1]**L*)		$n_1 = 0, n_2 = L$, Gen(0, 3, *aam*[2]**L*)

Figure 3 Creating multiple jobs from a single job of enumerating all compositions. A single call to procedure Gen with parameters (L, 1, 0) is equivalent to L+1 calls with parameters (L, 2, 0), (L - 1, 2, aam [1]),..., (0, 2, aam [1]**L*), while n_1 is set to 0, 1,..., L, correspondingly. Any job with *start* = 2 can be further expanded into L+1 jobs with *start* = 3, as shown for Gen(L, 2, 0).

```
1   procedure CreateMassHistMaster(L)
2       create numOfWorkers work processes
3       jobs ← CreateJobs(L)
4       numOfJobs ← number of created jobs
5       numOfBusyWorkers ← 0
6
7       while numOfJobs > 0 or numOfBusyWorkers > 0
8           comment: find free process and assign next job
9           p ← 1
10          while p < numOfProcs and numOfJobs > 0
11              if process p is free then
12                  assign next unassigned job from jobs to process p
13                  numOfJobs ← numOfJobs – 1
14                  numOfBusyWorkers ← numOfBusyWorkers + 1
15              p ← p + 1
16          wait for massHist from any worker
17          update massHistGlobal using massHist
18          numOfBusyWorkers ← numOfBusyWorkers – 1
19
20      tell work processes to terminate
21      return massHistGlobal
```

Figure 4 Pseudocode for procedure CreateMassHistMaster.

limit of 3 kDa (heavier peptides are rarely identified in MS experiments) it takes only 11 minutes, as about 87% of the compositions can be skipped (Tables 2 and 3).

There may be other modifications to this procedure, depending on the intended use of the generated mass distribution. For example, the maximum number of occurrences of each amino acid in a peptide may be made limited by a threshold based on the amino acid and the length and/or mass of the peptide. This would make the generated mass distribution more realistic and may increase the lengths of forbidden zones [14]. Instead of counting the number of peptide compositions, one can count the number of peptide sequences using equation (1). In this case, efficient computation of factorials "on the fly" can be implemented similar to the computation of peptide masses. If we are interested in enzyme-specific peptides, the procedure can be modified to allow a given number of missed cleavages. The

```
1   procedure CreateMassHistWorker(L)
2       wait for job from the master
3       while job is not to terminate
4           massHist ← Gen(job.L, job.start, job.m)
5           send massHist to the master
6           wait for job from the master
7
8       terminate
```

Figure 5 Pseudocode for procedure CreateMassHistWorker.

Table 2 Computation times for enumerating all tryptic compositions up to the length of 30, for different sets of jobs and number of work processes, with and without the maximum mass limit

Task	Number of Workers	Job Table					Computation Time	
		Number of Jobs	*start*	$L_{max,2}$	$L_{max,3}$	$L_{max,4}$	massMax = 3 kDa	no massMax
L = 30	1	1	1	-	-	-	6 h 03 min	35 h 11 min
	5	30	2	-	-	-	2 h 12 min	14 h 52 min
	30	30	2	-	-	-	1 h 39 min	13 h 32 min
	30	255	≤ 3	20	-	-	28 min	5 h 02 min
	71	255	≤ 3	20	-	-	27 min	4 h 57 min
	71	679	≤ 5	20	24	28	11 min	1 h 20 min

Computations were done using 71 work processes executed on a cluster with 12 Intel Xeon X5650 CPUs running Windows HPC Server 2008.

number of amino acids (N) and their monoisotopic masses may vary depending on specific proteases used in sample preparation, possible post-translational or chemical modifications, and other factors. The resolution of the mass histogram (0.001 Da) may be changed as well, without significantly impairing computational speed.

An important question is how the job (L, *start* + 1, 0) compares with the job (L, *start*, 0) in terms of computational complexity. Let us denote the number of compositions enumerated by the first procedure by $C(L, start + 1)$, and the number of compositions enumerated by the second procedure by $C(L, start)$. Using equation (3) we have:

$$C(L,\ start) = \binom{N - start + 1 + L}{N - start + 1}.$$

Hence,

$$\frac{C(L,\ start)}{C(L,\ start + 1)} = 1 + \frac{L}{N - start + 1}.$$

Table 3 Computation times for enumerating all tryptic compositions with different maximum lengths, with and without maximum mass limit

L	Computation Time	
	maxMass = 3 kDa	no mass limit
25	19 min	29 min
30	11 min	1 h 20 min
35	8 min	5 h 38 min
40	8 min	38 h 28 min
45	14 min	> 96 h
50	29 min	-

Parameters of the job table were: *start* ≤ 7, $L_{max,2}$ = 20, $L_{max,3}$ = 24, $L_{max,4}$ = 28, $L_{max,5}$ = 34, $L_{max,6}$ = 40. Computations were done using 71 work processes executed on a cluster with 12 Intel Xeon X5650 CPUs running Windows HPC Server 2008.

For example, if N = 20, L = 40, and *start* = 1, then $C(40, 1)/C(40, 2)$ = 3, which means that we get a three-fold decrease in computation time by replacing one call Gen(40, 1, 0) by 41 calls to Gen with *start* = 2, executed in parallel. Similarly, we get

$$\frac{C(L,\ start)}{C(L - 1,\ start)} = 1 + \frac{N - start + 1}{L}.$$

Thus, if N = 20, L = 40, and *start* = 2, then $C(40, 2)/C(39, 2) \approx 1.5$, which means that Gen(39, 2, 0) will be about 1.5 times faster than Gen(40, 2, 0).

Initialization of a job table requires the maximum value of parameter *start*, as well as parameters $L_{max,2}$, $L_{max,3}$, etc., to be specified. These can be determined empirically based on the available computational resources and the number of processes that can be executed in parallel. For example, we found that for enumerating tryptic peptide compositions of masses up to 3 kDa by using 72 processes running on 12 Intel Xeon X5650 CPUs the following parameters would give good performance: *start* ≤ 7, $L_{max,2}$ = 20, $L_{max,3}$ = 24, $L_{max,4}$ = 28, $L_{max,5}$ = 34, $L_{max,6}$ = 40. The tuning of these parameters is important to ensure good performance, as they directly affect the computation time (Table 2).

It should be noted that a job table may have jobs with the same parameters L and *start*, differing only in M. For example, consider the case illustrated in Figure 3. Splitting job (L, 2, 0) into L+1 jobs with *start* = 3 will give us, among others, job (L-1, 3, *aam*[2]). On the other hand, splitting job (L-1, 2, *aam*[1]) into L jobs with *start* = 3 gives us job (L-1, 3, *aam*[1]). It is clear that execution of these two jobs can be done in one call to function Gen, which should be modified to be able to accept two input masses m_0^1, m_0^2 instead of m_0, and to work with two variables m^1, m^2 instead of m. In a similar manner, execution of more than two jobs may be done in one call to function Gen. This approach will

lead to a significant speed-up in computations (it has not been implemented in our code).

In fact, a job table may have jobs with all three parameters L, *start* and M being equal. Consider, for example, a primary job with $L = 40$, $start = 0$, and $m_0 = 0$. Assume that array *aam* holds amino acid masses in ascending order. Then the first five masses stored in this array will correspond to glycine (G), alanine (A), serine (S), proline (P) and valine (V), and we can denote the first five elements of a composition by n_G, n_A, n_S, n_P, n_V. Assume that the job splitting algorithm (see subsection 2.3) yields the following two jobs:

$$n_G = 2, n_A = 0, n_S = 0, n_P = 0, n_V = 1, start = 6, L = 37,$$
$$n_G = 0, n_A = 3, n_S = 0, n_P = 0, n_V = 0, start = 6, L = 37.$$

Then these two jobs will have the same $m_0 = 213.111$ Da, since tripeptides GGV and AAA are isomeric. If a job table is generated using parameters $start \leq 7$, $L_{\max,2} = 20$, $L_{\max,3} = 24$, $L_{\max,4} = 28$, $L_{\max,5} = 34$, $L_{\max,6} = 40$, then for $L = 40$ about 2% of all jobs will be duplicates; for $L = 50$ – about 29%, and for $L = 60$ – about 47%. In the case when we are only interested in the mass distribution of peptide compositions, there is no need to execute duplicate jobs. If certain job occurs k times, it is enough to execute it once and then multiply the resulting histogram by k before adding it to the final histogram. However, if we would like to get every peptide composition, then we cannot remove duplicate jobs.

In the end of this section, we present Table 3 which shows computation times for enumeration of tryptic compositions for a range of lengths between 25 and 55, with and without the use of a maximum mass limit. The numbers in the second column may seem counterintuitive at first, since, for example, it takes 19 min to generate the distribution for $L = 25$ and 11 min for $L = 35$. The explanation, however, lies in using the maximum mass limit of 3 kDa. The longest job for the task with $L = 25$ was $L = 24$, $start = 2$, $m_0 = 0$, and it executed for 19 min. The longest job for the task with $L = 30$ was $L = 24$, $start = 2$, $m_0 = 285$, and it executed for 8 min. The difference in 11 min comes from the fact that more compositions were canceled out in the second case because of the mass limit that was used.

We would like to note, in addition to Table 3, that enumeration of all tryptic peptides having the mass no greater than 3 kDa (the length of these peptides does not exceed 51) took 32 minutes.

Conclusions

In this paper, we presented a detailed description of a parallel method for enumerating all theoretically possible amino acid compositions and discussed different aspects of its implementation. Enumeration of all amino acid compositions is important in several proteomics workflows, including peptide mass fingerprinting, mass defect labeling, mass defect filtering, and de novo peptide sequencing. Given the fact that multi-core computers and computer clusters are becoming increasingly available, it is natural to address this computationally expensive task using a parallelization approach.

We believe that by reducing computational times from hours to minutes, the applicability of the enumeration of all amino acid compositions in various proteomics studies may be significantly improved and extended. We have used the method described in this work to characterize forbidden and quiet zones in the mass distribution of tryptic peptides [14]. In the next step, we plan to apply this method to enhance the accuracy of protein identification in real mass spectrometry data. Our method has been implemented as a 32- and 64-bit Windows application using Microsoft Visual C++ and MPI. It is freely available for download at https://ispace.utmb.edu/users/rgsadygo/Proteomics/ParallelMethod.

Availability and Requirements

- **Project Name:** PepComp
- **Project home page:** https://ispace.utmb.edu/users/rgsadygo/Proteomics/ParallelMethod
- **Operating System**: MS Windows
- **Other Requirements**: Message Passing Interface, multi-core CPU
- **Programming Language**: Visual Studio C++
- **License**: No license needed

Acknowledgements

This work was supported in part by HHSN272200800048C NIAID Clinical Proteomics Center (Allan R. Brasier, UTMB) and NIH-NLBIHHSN268201000037C NHLBI Proteomics Center for Airway Inflammation (Alex Kurosky, UTMB).

Authors' contributions

RGS conceived the project. AVN conducted the analysis. AVN and RGS wrote the paper. All authors contributed to the underlying ideas of the method and the analysis. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Received: 27 July 2011 Accepted: 7 November 2011
Published: 7 November 2011

References

1. Dodds ED, An HJ, Hagerman PJ, Lebrilla CB: **Enhanced peptide mass fingerprinting through high mass accuracy: Exclusion of non-peptide signals based on residual mass.** *J Proteome Res* 2006, **5**:1195-1203.
2. Spengler B, Hester A: **Mass-Based Classification (MBC) of Peptides: Highly Accurate Precursor Ion Mass Values Can Be Used to Directly Recognize Peptide Phosphorylation.** *Journal of the American Society for Mass Spectrometry* 2008, **19**:1808-1812.
3. Lehmann WD, Bohne A, von der Lieth CW: **The information encrypted in accurate peptide masses-improved protein identification and assistance**

in glycopeptide identification and characterization. *Journal of Mass Spectrometry* 2000, **35**:1335-1341.
4. Jones JJ, Stump MJ, Fleming RC, Lay JO, Wilkins CL: **Strategies and data analysis techniques for lipid and phospholipid chemistry elucidation by intact cell MALDI-FTMS.** *Journal of the American Society for Mass Spectrometry* 2004, **15**:1665-1674.
5. Hall MP, Ashrafi S, Obegi I, Petesch R, Peterson JN, Schneider LV: **'Mass defect' tags for biomolecular mass spectrometry.** *Journal of Mass Spectrometry* 2003, **38**:809-816.
6. Lu B, Chen T: **Algorithms for de novo peptide sequencing using tandem mass spectrometry.** *Drug Discovery Today: BIOSILICO* 2004, **2**:85-90.
7. Spengler B: **De novo sequencing, peptide composition analysis, and composition-based sequencing: A new strategy employing accurate mass determination by Fourier transform ion cyclotron resonance mass spectrometry.** *Journal of the American Society for Mass Spectrometry* 2004, **15**:703-714.
8. Olson MT, Epstein JA, Yergey AL: **De novo peptide sequencing using exhaustive enumeration of peptide composition.** *J Am Soc Mass Spectrom* 2006, **17**:1041-1049.
9. Spengler B: **Accurate mass as a bioinformatic parameter in data-to-knowledge conversion: Fourier transform ion cyclotron resonance mass spectrometry for peptide de novo sequencing.** *Eur J Mass Spectrom (Chichester, Eng)* 2007, **13**:83-87.
10. Zubarev RA, Hakansson P, Sundqvist B: **Accuracy requirements for peptide characterization by monoisotopic molecular mass measurements.** *Analytical Chemistry* 1996, **68**:4060-4063.
11. Demirev PA, Zubarev RA: **Probing combinatorial library diversity by mass spectrometry.** *Analytical Chemistry* 1997, **69**:2893-2900.
12. Fenyo D, Qin J, Chait BT: **Protein identification using mass spectrometric information.** *Electrophoresis* 1998, **19**:998-1005.
13. Mann M: *Useful Tables of Possible and Probable Peptide Masses* Atlanta, GA; 1995.
14. Nefedov AV, Mitra I, Brasier AR, Sadygov RG: **Examining troughs in the mass distribution of all theoretically possible tryptic peptides.** *J Proteome Res* 2011, **10**:4150-4157.
15. Graham RL, Knuth DE, Patashnik O: *Concrete mathematics: a foundation for computer science.* 2 edition. Reading, Mass: Addison-Wesley; 1994.
16. Pacheco P: *Parallel Programming with MPI.* 1 edition. San Francisco: Morgan Kaufman; 1996.

doi:10.1186/1471-2105-12-432
Cite this article as: Nefedov and Sadygov: **A parallel method for enumerating amino acid compositions and masses of all theoretical peptides.** *BMC Bioinformatics* 2011 **12**:432.

Submit your next manuscript to BioMed Central and take full advantage of:

- **Convenient online submission**
- **Thorough peer review**
- **No space constraints or color figure charges**
- **Immediate publication on acceptance**
- **Inclusion in PubMed, CAS, Scopus and Google Scholar**
- **Research which is freely available for redistribution**

Submit your manuscript at
www.biomedcentral.com/submit

BioMed Central