Published in final edited form as:

Curr Protoc Bioinformatics. ; 55: 13.29.1–13.29.23. doi:10.1002/cpbi.15.

Tempest: Accelerated MS/MS database search software for heterogeneous computing platforms

Mark E. Adamo¹ and Scott A. Gerber^{1,2,3}

¹Norris Cotton Cancer Center, Geisel School at Dartmouth, Lebanon, NH 03756

²Department of Genetics, Geisel School at Dartmouth, Lebanon, NH 03756

³Department of Biochemistry, Geisel School at Dartmouth, Lebanon, NH 03756

Abstract

MS/MS database search algorithms derive a set of candidate peptide sequences from in-silico digest of a protein sequence database, and compute theoretical fragmentation patterns to match these candidates against observed MS/MS spectra. The original Tempest publication described these operations mapped to a CPU-GPU model, in which the CPU generates peptide candidates that are asynchronously sent to a discrete GPU to be scored against experimental spectra in parallel (Milloy et al., 2012). The current version of Tempest expands this model, incorporating OpenCL to offer seamless parallelization across multicore CPUs, GPUs, integrated graphics chips, and general-purpose coprocessors. Three protocols describe how to configure and run a Tempest search, including discussion of how to leverage Tempest's unique feature set to produce optimal results.

Keywords

proteomics; mass spectrometry; peptide identification; database search; parallel computing; GPGPU; OpenCL; neutral loss

Introduction

Tempest is a software program for MS/MS peptide identification by database search. At the time of its original publication (Milloy et al., 2012), Tempest followed the model of existing database search programs SEQUEST (Eng et al., 1994) and MacroSEQUEST (Faherty and Gerber, 2010), and incorporated NVIDIA's CUDA API to parallelize computationally intensive peptide-spectrum scoring operations using a graphics processing unit (GPU). Recent development on Tempest has focused on updating the software in two aspects: first, re-implementing the algorithm to utilize OpenCL; and second, introducing features that bring Tempest toward parity with modern search engines such as Comet (Eng et al., 2013) and take advantage of the processing power to which Tempest provides access.

Internet Resources: <https://github.com/mhoopmann/mstoolkit>. MSToolkit file parsing library.
<http://www.uniprot.org/UniProt> proteome database for FASTA file retrieval
<http://proteomics.dartmouth.edu/k/software/tempest> Point-of-contact for Tempest files and code.

MS/MS database search algorithms derive a set of candidate peptide sequences from in-silico digest of a protein sequence database, and compute theoretical fragmentation patterns to match these candidates against observed MS/MS spectra. The original Tempest manuscript describes these operations mapped to a CPU-GPU model, in which the CPU generates peptide candidates that are asynchronously sent to a discrete GPU to be scored against experimental spectra in parallel (Milloy et al., 2012). The current version of Tempest expands this model, incorporating OpenCL to offer seamless parallelization across multicore CPUs, GPUs, integrated graphics chips, and general-purpose coprocessors.

Parameters for a Tempest search are specified in two text files passed to the program at the command line. Fields in the system configuration file, described in Basic Protocol 1, determine which OpenCL devices will be used for scoring and define constraints on host and device memory usage. A search method file, described in Basic Protocol 2, specifies locations of the protein database and MS/MS peak data, and also contains parameters governing search resolution, post-translational modifications, fragmentation patterns, and spectral processing.

Basic Protocol 3 describes how to run a Tempest search from the command line.

Basic Protocol 1: Setting up a system configuration

The system configuration file contains settings pertaining to the allocation of system resources and the specific OpenCL hardware to utilize. Typically, users will set up a single system configuration file to be used with all searches on a given computer platform. The system configuration file consists of a list of parameters, each of which must be entered on a separate line. A valid parameter line contains the parameter identifier followed by one or more values, all separated by whitespace (spaces and/or tabs). Comments are added using the ‘>’ symbol, which signals to Tempest to stop parsing the line. Comments may be entire lines or inline comments at the end of lines.

Necessary Resources

A computer running Linux, with OpenCL-compatible hardware available and corresponding OpenCL drivers installed. Currently, Tempest runs exclusively as a command-line executable. Windows and Mac versions are forthcoming. Links to source code, downloads, and supporting resources can be found at <http://proteomics.dartmouth.edu/k/software/tempest>.

The system configuration file can be created or modified with any basic text editor. It is advisable to reference the **tempest.config** file included with the software to ensure proper formatting.

Device Setup

1. Run Tempest with the command line option **--device-info** to show a map of all installed OpenCL platforms and available devices on the host.

Devices are specified with a platform and device ID, displayed next to each device as ([platform ID]:[device ID]). Tempest can distribute scoring operations across multiple devices, provided that the devices share the same platform.

2. **platform_id, device_id:** Set a platform ID, followed by any number of device IDs. Tempest will exit with an error if the selected platform or devices are not available on the system.

```
platform_id 0
device_id 0
device_id 1
```

Selecting devices 0 and 1 on platform 0 (devices 0:0 and 0:1)

Memory Constraints

- 3 **max_host_mem:** Specify maximum host memory usage. Tempest scales candidate buffer allocations in order to target a user-defined memory footprint in host RAM. The batch size of candidates submitted for scoring increases as a function of **max_host_mem**. Larger values reduce the number of kernel launches required, and can allow for increased scoring parallelization depending on device capability. For best performance, set **max_host_mem** to approach, but not exceed, available system RAM.

```
max_host_mem 12 GB
```

- 4 **max_device_mem:** Specify the maximum amount of memory to be allocated for spectrum data on each selected OpenCL device. Tempest will build binned spectra into device memory until allocated memory is exhausted, at which point the spectrum accessed least recently will be swapped out, freeing memory for another spectrum to be built. Increasing **max_device_mem** reduces the number of build and transform operations required over the course of a search. For devices with dedicated memory (e.g. discrete GPUs), **max_device_mem** can be set to the global memory size reported by **tempest --device-info**. For devices that share host memory (e.g. CPUs, integrated graphics chips), memory constraints should be balanced such that the sum of **max_host_mem** and **max_device_mem** does not exceed available system RAM. A **max_device_mem** value of 0 will reserve the minimum amount of memory to

store 1 binned spectrum at a time, which may be a preferable mode of operation for devices that share host memory.

```
max_device_mem 4 GB
```

- 5 **parallel_reduce**: Specify whether scoring results are folded into the top-scoring PSM list using a parallel or sequential algorithm. CPU devices perform best with **parallel_reduce** set to 0, forcing sequential operation. For highly parallel devices such as GPUs, **parallel_reduce** should be kept at 1 (default).

```
parallel_reduce 1
```

Basic Protocol 2: Setting up a search method

A search method file contains parameters relating to attributes of the experimental data and the desired search constraints (precursor and fragment tolerances, fixed/variable modifications, enzyme properties, etc.). Whereas the system configuration file from Basic Protocol 1 is designed to apply to all searches on a given computer platform, a user might maintain a collection of method files tailored to a variety of search types. Included in the Tempest software package are a number of pre-written method files configured for common enzymes, post-translational modifications, and resolution settings.

The search method file described in this protocol is a plain-text file that can be created and/or edited with any text editor. Entry of parameters in this file follows the same layout as the system configuration file described in Basic Protocol 1.

Necessary Resources

A search method file can be created or modified with any basic text editor. It is advisable to reference one of the pre-written methods in the Tempest software package for guidance on formatting and recommended settings.

Input Files

1. **spectra_file**: Provide an absolute path to a spectra file. Compatible formats are mzXML, mzML, and ms2 (ms2, bms2, cms2).

```
spectra_file /home/user/tempest/f04747.mzXML
```

2. **database_file**: Provide an absolute path to a database (FASTA) file. The database should be in standard FASTA format; FASTA files provided by UniProt (<http://www.uniprot.org/>) may be used as compatible examples. Reference lines are parsed up to the first whitespace on the line, and are subsequently truncated to 64 characters. No line in the FASTA file should exceed 1024 characters.

```
database_file /home/user/tempest/fasta/human.fasta
```

Alternatively, these file paths can be specified at the command line. The ‘-i’ (spectra input file) and ‘-f’ (FASTA database) command line arguments override paths in the method file if paths are specified in both places.

Digest Definition

- 3 **digest_definition:** Define enzyme properties for in-silico digest. The digest_definition line provides (1.) sites at which the enzyme cuts, (2.) sites adjacent to which the enzyme will not cut, and (3.) a flag indicating whether the enzyme cuts N-terminally or C-terminally to the cut site (values of 0 and 1, respectively)

```
digest_definition KR P 1
```

Standard digest definition for trypsin. Sequences are cleaved C-terminally (1) to K and R residues (KR), except when the cleavage point is adjacent to a P residue (P).

Enter at least one character for all three sub-fields. A dash (-) can be used when no sites are applicable.

```
digest_definition D - 0
```

Digest definition for endoproteinase Asp-N. Sequences are cleaved N-terminally (0) to D residues (D), with no restrictions on adjacent residues (-).

- 4 **enzyme_specificity:** Set the number of enzymatic termini required by a candidate sequence. Valid entries are 0 (non-specific) 1 (partially specific), and 2 (fully specific). When enzyme_specificity is set to 0, the digest_definition line is disregarded and any enzyme cut sites specified there will not be counted as missed cleavages.

```
enzyme_specificity 2
```

- 5 **digest_missed_cleavages:** Specify a number of missed cleavages. Tempest will not consider peptides exceeding this number of internal cut sites.

```
digest_missed_cleavages 3
```

- 6 **digest_length:** Specify a lower and upper bound for valid candidate sequence length. Bounds are given as [low, high), with the lower bound inclusive and upper bound non-inclusive.

digest_length 5 64

- 7 **digest_mass_range:** Specify a lower and upper bound for valid candidate sequences. Bounds are given as [low, high), with the lower bound inclusive and upper bound non-inclusive.

digest_mass_range 400 5000

- 8 (Optional) **decoy_search:** Enable or disable generation of decoy peptides for target-decoy search. With **decoy_search** enabled, Tempest generates a decoy peptide from each digested peptide by reversing the internal sequence while fixing the specific enzymatic cleavage residue in place. Decoy sequences are scored and reported alongside target sequences and are flagged as decoys in the search results.

decoy_search 1

Target-decoy search enabled.

Target sequence R.FPTPILK.V
Decoy sequence R.LIPTPFK.V

Target sequence and corresponding decoy sequence in a trypsin digest. The tryptic cleavage residue (K) remains fixed while the internal sequence is reversed. This approach ensures that the decoy sequence is a valid tryptic peptide with the same mass as the target sequence.

Target sequence Q.DGVIVSN.D
Decoy sequence Q.DNSVIVG.D

Target sequence and corresponding decoy sequence in an Asp-N digest.

Modifications

- 9 (Optional) **fixed_modification:** Specify any fixed (static) modifications. Fixed modifications are defined using one line for each distinct modification mass, with all applicable residues entered as a contiguous string. Applying a fixed modification to a residue changes the base mass that Tempest uses for all occurrences of the residue.

fixed_modification C 57.021460

Cysteine carbamidomethylation (mass delta +57.021460 Da) as a fixed modification.

Modifications to N- and C- termini can be specified using the special strings 'peptide-nterm', 'peptide-cterm', 'protein-nterm', and 'protein-cterm'.

```
fixed_modification peptide-nterm 229.16293200
```

N-terminal TMT 6-plex label (mass delta +229.16293200 Da) as a fixed modification.

- 10 (Optional) **variable_modification**: Specify any variable modifications. Variable modifications are defined in a similar style to static modifications, with the addition of a symbol used to denote the modification in Tempest output. For each peptide, Tempest will consider all valid combinations of variably modified/unmodified residues. Any number of variable modifications may be given, with a limit of 5 distinct modifications on a single residue.

```
variable_modification STY # 79.966330
variable_modification M * 15.994910
```

Variable modifications for S, T, or Y phosphorylation (mass delta +79.966330) and methionine oxidation (mass delta +15.994910 Da), respectively. Phosphorylation will be reported in the output with the '#' symbol, and methionine oxidation with the '' symbol, as specified here.*

- 11 **variable_modifications_max**: Specify a maximum number of variable modifications per peptide.

```
variable_modifications_max 4
```

Neutral losses

- 12 (Optional) **neutral_loss**: Tempest can generate and score user-defined neutral loss events as independent theoretical fragments. Neutral losses can be applied to unmodified residues, termini, or specific modification symbols. Fragments containing one or more neutral loss sites will generate one additional neutral loss fragment for each distinct **neutral_loss** definition.

```
neutral_loss # 97.97689509 1.0
```

Definition for neutral loss of phosphate. When Tempest encounters a phosphorylated residue (using the variable modification symbol '#' – see step 9) in a theoretical fragment ion, it will score an additional theoretical fragment ion with mass delta -97.97689509. Score contributions from these theoretical fragments will be multiplied by 1.0, giving them equal magnitude to the primary ion series from which they are derived.

Neutral losses of the unmodified peptide N- and C-terminus can be specified using the special strings 'peptide-nterm' and 'peptide-cterm', respectively. This can also be used to apply generic neutral losses to all N- or C-terminal fragments, such as loss of H₂O and NH₃.

```
neutral_loss peptide-nterm 18.0105647 0.2
neutral_loss peptide-cterm 18.0105647 0.2
neutral_loss peptide-nterm 17.02654911 0.2
neutral_loss peptide-cterm 17.02654911 0.2
```

Neutral losses of H₂O and NH₃ applied to both the N- and C-terminus. Since Tempest does not calculate internal fragments, these neutral losses will be computed for every theoretical fragment, agnostic of sequence.

```
neutral_loss STED 18.0105647 0.2
neutral_loss RKNQ 17.02654911 0.2
```

*Neutral losses of H₂O and NH₃ defined with sequence dependence. Neutral loss fragments will be computed for only those fragments that contain the specified residues. This is a more sophisticated representation of these neutral loss events compared to the previous example using **nterm** and **cterm**.*

Ion series

- 13** **[a|b|c|x|y|z]_ions:** Indicate which ion series Tempest should consider by specifying a weighting for each applicable ion series. These parameters should be set to reflect the fragmentation pattern of the ion activation method used on the instrument (e.g. CID, ETD) (Brodelt, 2015). Score contributions from each ion series will be multiplied by the weighting factor. Each of these parameters defaults to 0.0 (off) if omitted.


```
a_ions 0.0
b_ions 1.0
c_ions 0.0
x_ions 0.0
y_ions 1.0
z_ions 0.0
```

Ion series definition for collision-induced dissociation (CID) data. B and Y ions are weighted equally. A, C, X, and Z ions are weighted such that they do not contribute to the score, and will be excluded from scoring.

MSⁿ levels

- 14 **ms_level:** Restrict fragment ion scoring to scans from selected MSⁿ levels. Currently, MS¹, MS², or MS³ scans can be selected. Repeated **ms_level** lines can be entered to search scans at multiple MSⁿ levels. If this parameter is omitted, only MS² (MS/MS) scans will be searched.

```
ms_level 2
```

Search tolerances

- 15 **precursor_tolerance:** Specify precursor tolerance. Spectra are scored against candidate peptides with masses within the specified tolerance of the spectrum precursor mass. Values may be given in Daltons (Da, amu) or ppm (ppm).

```
precursor_tolerance 1 Da
```

Wide-window search with 1 Da precursor tolerance.

Precursor_tolerance 20 ppm

Narrow-window search with 20 ppm precursor tolerance.

- 16 **fragment_bin_width, fragment_bin_offset:** Set parameters governing MS/MS spectrum binning. Bin width is given in m/z. Bin offset is given as a proportion of the bin width (0.0 to 1.0). Bins are shifted to the right by the bin offset value, which is used in higher bin-width (unit resolution) searches to account for peak periodicity. Bin offset has little impact in high-resolution searches, which may use a bin width of 0.02 m/z or less.

```
fragment_bin_width 1.0005
fragment_bin_offset 0.68
```

Standard binning settings for unit-resolution MS/MS data.

```
fragment_bin_width 0.02
fragment_bin_offset 0.0
```

Standard binning settings for high-resolution MS/MS data.

Spectrum preprocessing

- 17 **normalization_regions:** Set a number of MS/MS normalization regions. Tempest evenly partitions MS/MS spectra into n normalization regions by m/z, with peaks in each region linearly scaled such that the top peak in the region has an intensity value of 1.0. Increasing the number of regions (default 10) may increase identification rate at the risk of introducing periodicity in the output score distribution.

```
normalization_regions 10
```

- 18 (Optional) **msms_remove_precursor:** Set this flag to 1 in order to discard any MS/MS peaks within $\pm$ **fragment_bin_width** m/z of the precursor mass.

```
msms_remove_precursor 1
```

- 19 (Optional) **msms_remove_mz_range:** Define an MS/MS region by providing an m/z minimum and maximum (inclusive). Peaks within the bounds are discarded from all MS/MS spectra. If needed, repeat this parameter to define multiple regions. In isobarically-labeled (TMT, iTRAQ) samples, this parameter can be used to eliminate scoring interference from reporter ions.

```
msms_remove_mz_range 126.0 131.8
```

- 20 (Optional) **intensity_threshold**: Set a peak intensity threshold as a proportion of base peak intensity (0.0 to 1.0). Peaks falling below this threshold are discarded from each spectrum.

```
intensity_threshold 0.0025
```

- 21 (Optional) **flanking_intensity**: Each bin propagates a proportion (0.0 to 1.0) of its intensity into the two bins directly flanking it. This value is 0.0 if the **flanking_intensity** parameter is omitted. Nonzero values are useful for searches with small MS/MS bin widths (e.g. **fragment_bin_width**=0.02).

```
flanking_intensity 0.5
```

- 22 (Optional) **xcorr_transform_width**: Tempest will optionally apply a SEQUEST-like fast cross-correlation transformation to the binned spectra prior to scoring (Eng et al., 2008). For each bin, averaged intensity from $\pm n$ surrounding bins is subtracted from the bin intensity. Since the value is given in bins, this parameter must be manually scaled alongside **fragment_bin_width**. Due to the sparseness of high-resolution binned spectra, this transformation has little effect for small **fragment_tolerance** values and may be omitted ($n=0$, default) (Milloy et al., 2012).

```
xcorr_transform_width 75
```

Standard xcorr transform width for unit-resolution MS/MS data.

Output options

- 23 **num_internal_psms**, **num_output_psms**: Tempest tracks and reports a user-specified maximum number of PSMs for each spectrum. By default, 10 PSMs are tracked and reported per spectrum. A minimum of 2 PSMs must be tracked internally (**num_internal_psms**) in order to calculate delta score. Increasing the number of internal PSMs enables deeper backbone delta score calculations for peptides with many variable modification sites, at the cost of runtime performance. A minimum of 1 PSM per peptide must be reported (**num_output_psms** ≥ 1), and Tempest cannot report more PSMs than it tracks internally.

```
num_internal_psms 10
num_output_psms 10
```

Basic Protocol 3: Running a Tempest Search

Necessary Resources

The same system prerequisites listed in Protocol 1 apply to this protocol. To run a search, Tempest requires a protein database file, a spectra file, a configuration file (see Basic Protocol 1) and a method file (see Basic Protocol 2). The protein database file must be in valid FASTA format. Tempest uses MSToolkit (<https://github.com/mhooopmann/mstoolkit>) to parse spectra files, so any format supported by MSToolkit may be used. Supported formats include mzXML, mzML, and MS2.

From a command prompt:

1. Locate the path to the Tempest executable, named **tempest**. The executable may be called via its absolute or relative path, or added to the PATH environment variable and called by name.
2. Use the **-c** argument to provide an absolute or relative path to a system configuration file (see Basic Protocol 1). If this parameter is omitted, Tempest will use **tempest.config** in the current working directory.
3. Use the **-m** argument to provide an absolute or relative path to a search method file (see Basic Protocol 2). If this parameter is omitted, Tempest will use **tempest.method** in the current working directory.
4. Use the **-o** argument to provide an output file path prefix. Tempest will write its two output files to [prefix].tsv and [prefix].log. If this parameter is omitted, the prefix will default to the spectra file filename.
5. If no path to a spectra input file was given in the search method file, or if you wish to override the path given in the search method file, use the **-i** argument to provide an absolute or relative path to a spectra input file.
6. If no path to a FASTA database was given in the search method file, or if you wish to override the path given in the search method file, use the **-f** argument to provide an absolute or relative path to a FASTA file.
7. (Optional) To search a subset of spectra in the input file, use the **-s** argument in the format **-s x:y**, where *x* is the lower bound (inclusive) of scan numbers to be searched and *y* is the upper bound (non-inclusive) of scan numbers to be searched.

Guidelines for Understanding Results

Log file (.log)

Following successful completion of a search, Tempest records metadata about the search in a plain-text log file. The log file includes information about search runtime, statistics from spectral processing and database digest, and a full dump of the search method file, system configuration file, and command-line parameters that were used in the search.

PSMs (.tsv)

Tempest outputs a list of peptide-spectrum matches (PSMs) in TSV (tab-separated values) format. For each spectrum, Tempest will report up to **n** top-ranked PSMs, with *n* specified in the **num_output_psms** parameter. Spectra are reported by scan number, and a spectrum will be omitted from the output if it generates no peptide matches with a score greater than 0.

A Tempest output file reports the following data for each PSM:

- **Scan:** The scan number of the spectrum, recorded from the spectra input file.
- **PrecursorMass:** The mass of the parent scan, recorded from the spectra input file.
- **Charge:** The charge of the parent scan, recorded from spectra input file.
- **Candidates:** The number of peptide candidates scored against the spectrum over the course of the search.
- **Rank:** The rank of the PSM, ranging from rank **1** (top match) to rank **n**, with **n** specified in the **num_output_psms** parameter.
- **Sequence:** The full sequence of the peptide match, including flanking residues and modifications. In each sequence, peptide N- and C- termini are indicated by lower-case 'n' and 'c' characters, respectively. Variable modification symbols are printed directly following the modified residue or terminus.
- **Ref:** The protein reference corresponding to the matched peptide sequence. For sequences that match to multiple proteins, Tempest reports the reference occurring closest to the beginning of the database file.
- **Decoy:** Values are **1** or **0**; a value of **1** indicates that the reported peptide sequence is an internal decoy derived from the protein reference in **Ref**.
- **Occurrences:** The number of times that the peptide sequence occurs in the database. A value of **1** indicates that the sequence unambiguously maps to the protein reported in **Ref**, and occurs only once in that protein sequence. Decoy sequences are not counted as occurrences.
- **SequenceMass:** The theoretical precursor mass of the peptide sequence.
- **ppm:** Mass error for the PSM, computed as ppm deviation of **PrecursorMass** from **SequenceMass**.
- **Score:** PSM score, computed from normalized intensities at matched fragment m/z bins. Higher values indicate a better match. PSMs with scores ≤ 0 are not reported.
- **DeltaScore:** Difference between the PSM score and the score of the next most highly ranked PSM, as a proportion of the higher score. Reports a value of **1.0** when no PSM of a lower rank was recorded.
- **DeltaScore_backbone:** Difference between the PSM score and the score of the next most highly ranked PSM with a different sequence backbone (sequence sans

modification symbols), as a proportion of the higher score. Reports a value of **1.0** when no lower-ranked PSM with a different sequence backbone was recorded.

Commentary

Background Information

In the landscape of currently available MS/MS database search algorithms, Tempest's distinguishing features include hardware acceleration via OpenCL, constant memory size (independent of binning resolution and database size), and a candidate-centric scoring schema that enables digestion and scoring to be conducted in parallel. The core mechanisms of spectral preprocessing and fragment matching mostly adhere to the SEQUEST/Comet model, with a number of optimizations and additional options that are discussed in the Critical Parameters section. The major steps of the Tempest algorithm are as follows:

- 1. Spectrum loading**

As spectra are read from the input file, Tempest converts the m/z value for each peak into a bin index based on the user-specified **fragment_bin_width**. Peaks are then stored in host memory as bin:intensity pairs. With this sparse representation, the size of peak data in host memory is unaffected by the **fragment_bin_width**.

- 2. Spectrum preprocessing**

Each spectrum undergoes three preprocessing steps:

- 1. Thresholding**

Peaks that fall below a user-defined noise threshold are discarded. The noise threshold is given as a proportion of base peak intensity in the **intensity_threshold** parameter. The precursor peak, as well as peaks in user-defined m/z ranges (**msms_remove_mz_range**) are removed in this step if these options are enabled.

- 2. Scaling**

The spectrum is divided equally into a user-defined number of normalization regions (default 10, specified in **normalization_regions** parameter). Peak intensities in each region are scaled such that the highest peak in the region has an intensity value of 1.0.

- 3. Normalization**

Peak intensities are stored as the square root of their scaled value, multiplied by 50.

- 3. Spectrum indexing**

Spectra are binned by precursor mass using the user-specified **precursor_tolerance**. Each spectrum is allocated its own memory buffer for storing theoretical peptide candidates.

4. Database digest

Peptides are derived from the FASTA database using enzyme rules specified in the method file (cut sites, specificity, missed cleavages, length range, mass range). If a sequence contains residues subject to variable modification (**variable_modification** parameter lines), Tempest will recursively generate peptides with all possible modification patterns up to a maximum of **variable_modifications_max** modifications. After a peptide sequence is generated, its mass is compared to the precursor masses of the experimental spectra. The peptide is appended to the candidate buffer for each spectrum with a matching precursor mass (within the m/z or ppm tolerance specified in **precursor_tolerance**). If a target-decoy search is selected, Tempest will also append a corresponding decoy peptide to the candidate buffer at this stage. Once a spectrum's candidate buffer is filled, the candidate peptides are submitted to the device to be scored.

5. Scoring

Once a spectrum's candidate buffer is submitted to a device, the following scoring operations are enqueued for asynchronous execution on the device:

1. Build

Sparse spectrum data, previously stored as bin:intensity pairs, is expanded into device memory as an array of intensity values for random access by bin index. Spectra are expanded into unoccupied memory where possible, and the *build* operation is skipped if the spectrum is already built in memory. Spectra accessed most frequently are kept in memory and reused to reduce the number of *build* and *transform* operations required over the course of a search.

2. Transform

For each bin in the previously built spectrum, an intensity average is computed from $\pm n$ surrounding bins, where n is the user-specified **xcorr_transform_width**. This intensity average is deducted from the bin intensity. This transformation, primarily useful for unit-resolution spectra, is an implementation of the SEQUEST fast cross correlation algorithm (Eng et al., 2008).

3. Score

Tempest generates theoretical fragment m/z values for all selected ion series at all charge states less than the spectrum precursor charge. Tempest then indexes into matching m/z bins in the processed spectrum, and incorporates matching spectral intensities into the peptide score calculation. If a peptide fragment contains user-specified neutral loss sites (amino acid, modification, or terminus), Tempest will consider an additional neutral loss fragment for each distinct neutral loss mass.

4. Reduce

Candidates are sorted by score and incorporated into the list of the top n PSMs for the given spectrum, with n specified in **num_internal_psms**. As these device kernels are executed, database digest continues in parallel. Once the digest completes and all enqueued scoring operations have been processed, results are reported and the search is complete.

Critical Parameters

During the development of Tempest, we observed that we could increase identification rates by modulating important elements of the algorithm. The parameters **normalization_windows** and **xcorr_transform_width** parameterize two steps of spectrum preprocessing that are not currently user-modifiable in Comet or SEQUEST. In the spectral normalization phase, a spectrum is divided into a fixed number of normalization regions (historically 10), with the lower and upper bounds set to the two observed peaks with the lowest and highest m/z , respectively. Using this normalization algorithm, spectra containing a larger m/z range will undergo coarser normalization (wider normalization windows, by m/z) than will spectra with a narrower m/z range. For samples containing larger digested peptides (average MS/MS range of ~ 1500 m/z), optimal **normalization_windows** values may range from 20 to 40.

In phosphorylation searches, we observe an additional increase in identification rates by considering neutral loss of phosphate with the **neutral_loss** parameter. In Tempest's neutral loss implementation, score contributions from neutral loss peaks are weighted by a user-specified factor and are not predicated on the existence of the primary peak. Therefore, we can match fragments that are only observed with the neutral loss mass, and (if desired) weigh these fragments as highly as the primary fragments. For one test sample containing phosphorylated TMT-labeled peptides, we observed a $>15\%$ increase in target-decoy identification rates simply by optimizing the **normalization_windows** and **neutral_loss** parameters.

Advanced Parameters

Whereas the system configuration parameters listed in Basic Protocol 1 are critical for achieving a baseline level of performance, the following optional parameters in the **tempest.config** file can be used to debug and/or tweak performance for specific hardware.

1. **profile**: Profile mode enables kernel runtime data to be reported upon completion of a Tempest search. This option, when set to 1 (profile mode on), can be used to identify execution bottlenecks and to optimize configuration options. Otherwise, **profile** should be set to 0 (default). Profile mode forces synchronous host-device execution, impacting runtime.

profile 0

2. **min_work_size:** The **min_work_size** parameter governs the minimum local workgroup size used by the OpenCL scoring kernel. This value must be a power of 2 (default 32). In cases of low processor utilization, this value may be increased in order to mitigate performance bottlenecks related to multithread memory accesses and register pressure. **min_work_size** can also be increased to work around errors pertaining to device resource usage (e.g. allocation of registers and local memory).

min_work_size 32

Troubleshooting

Running Tempest in **--device-info** mode displays all OpenCL platforms and devices currently visible to the system. If multiple OpenCL platform drivers are installed on a system, it may be necessary to manipulate environment variables in order for the desired platform(s) to be recognized.

So far, Tempest has been tested on a limited variety of hardware platforms but should be compatible with any standards-conforming implementation of OpenCL version 1.2 or newer. Each time an OpenCL library function is invoked, Tempest checks for driver-level errors and, in the case of an error, will exit with a message reporting the error type and the line of source code at which the error was detected.

To minimize errors caused by unexpected user input, create customized parameter files following the layout of the sample parameter files (**tempest.method** and **tempest.config**) included alongside the Tempest executable.

Performance Considerations

Tempest's device kernels currently force single-precision floating point operations. Since many consumer GPUs provide limited or unoptimized support for double-precision operations, Tempest can efficiently utilize a wide variety of devices including older and lower-end hardware. In many cases, devices marketed for scientific computing will yield a lower performance-to-price ratio compared to their consumer-grade counterparts.

Suggestions for Further Analysis

While the method files included with the Tempest software provide suggested settings for analyzing high- and low-resolution data, we hope that the discussion in this paper empowers users to find, through experimentation, settings that yield optimal peptide identification rates for their data. In particular, creative applications of the user-defined neutral loss settings might improve identification of peptides with complex fragmentation patterns. We are investigating whether this parameter can be used to increase identification rates in isobarically labeled (TMT, iTRAQ) samples by accounting for the fragmentation of the reporter.

We are also currently modifying Tempest to support chemical crosslinking analysis, enabling users to specify one or more crosslinkers with customizable chemistry. With these proposed modifications, pairs of crosslinked peptides could be identified and reported alongside standard non-crosslinked peptide matches.

Appendix A: tempest.config

All parameters that may be included in the .config file are listed here. The process of designing a .config file with these parameters is expanded upon in Basic Protocol 1.

All arguments listed for each parameter are required. Bracketed arguments are annotated with their associated datatypes and (if applicable) valid input ranges. Arguments with enumerated values separated by a vertical bar (‘|’) are restricted to one of the values given.

Each parameter line is annotated with a minimum and maximum number of times it can appear in the .config file. In cases where a parameter has 0 minimum occurrences, the parameter is optional and will revert to the listed default value when omitted.

platform [id]

id=(int): Use OpenCL platform with ID **id**.

Valid values range from 0 to $n-1$, where n is the number of available OpenCL platforms.

Minimum occurrences: 1

Maximum occurrences: 1

device [id]

id=(int): Use OpenCL device with ID **id**.

Valid values range from 0 to $n-1$, where n is the number of devices on the selected OpenCL platform.

Minimum occurrences: 1

Maximum occurrences: any

max_host_mem [mem] B|MB|GB

mem=(float): Reserve **mem** bytes/megabytes/gigabytes of host memory (approximate).

mem must be a non-negative value. With **mem**=0, Tempest will reserve the minimum amount of memory for 1 stored candidate per precursor bin.

Minimum occurrences: 1

Maximum occurrences: 1

max_device_mem [mem] B|MB|GB

mem=(float): Reserve **mem** bytes/megabytes/gigabytes of device memory (approximate) per device.

mem must be a non-negative value. With **mem**=0, Tempest will reserve the minimum amount of memory for 1 binned spectrum.

Minimum occurrences: 1

Maximum occurrences: 1

parallel_reduce 0|1

When set to 1, use a parallel algorithm to fold scores into the list of top-ranked PSMs (suitable for GPUs). When set to 0, perform this step using a sequential algorithm (suitable for CPUs).

Minimum occurrences: 1

Maximum occurrences: 1

Default value: **parallel_reduce 1**

profile 0|1

When set to 1, perform OpenCL kernel runtime profiling and report results upon search completion.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **profile 0**

min_work_size [n]

n=(int): use a minimum local work size of **n** for the scoring kernel.

n must be a power of 2. This parameter may be used to tweak performance for certain OpenCL devices, or to work around errors pertaining to device resource usage (e.g. allocation of registers and local memory).

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **min_work_size 32**

Appendix B: tempest.method

All parameters that may be included in a .method file are listed here. The process of designing a .method file with these parameters is expanded upon in Basic Protocol 2.

All arguments listed for each parameter are required. Bracketed arguments are annotated with their associated datatypes and (if applicable) valid input ranges. Arguments with enumerated values separated by a vertical bar (‘|’) are restricted to one of the values given.

Each parameter line is annotated with a minimum and maximum number of times it can appear in the .method file. In cases where a parameter has 0 minimum occurrences, the parameter is optional and will revert to the listed default value when omitted.

spectra_file [path]

Provide a path to the spectra input file to be searched. Supported filetypes are mzXML, mzML, and ms2. This parameter is optional if overridden by the **-i** command line option.

path=(string): Absolute filesystem path.

Minimum occurrences: 0

Maximum occurrences: 1

database_file [path]

Provide a path to the FASTA file to be used in the search. This parameter is optional if overridden by the **-f** command line option.

path=(string): Absolute filesystem path.

Minimum occurrences: 0

Maximum occurrences: 1

digest_definition [cleavage sites] [not before] [offset]

Enzyme definition governing in-silico digest of the protein database.

cleavage sites=(string): Contiguous string of all residues at which the enzyme cleaves. ‘-’ if no residues.

not before=(string): Contiguous string of all residues adjacent to which the enzyme does not cleave. ‘-’ if no residues.

offset=0: cleave before cleavage sites.

offset=1: cleave after cleavage sites.

Minimum occurrences: 1

Maximum occurrences: 1

digest_specificity 0|1|2

Require 0, 1, or 2 enzymatic termini per peptide in in-silico digest. Protein N- and C-termini, as well as cleavage sites specified in the **digest_definition** count as enzymatic peptide termini. With **digest_specificity 0**, the digest is fully nonspecific, overriding **digest_definition**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **digest_specificity 2**

digest_missed_cleavages [n]

Maximum number of missed cleavages allowed in a candidate sequence, based on **digest_definition**.

n=(int): Maximum allowed missed cleavages per peptide.

Value must be non-negative.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **digest_missed_cleavages 2**

digest_length [min] [max]

Range of allowed lengths for candidate sequences as [**min**, **max**).

min=(int): Minimum number of amino acid residues per peptide (inclusive).

max=(int): Maximum number of amino acid residues per peptide (non-inclusive). Values must be non-negative.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **digest_length 5 64**

digest_mass_range [min] [max]

Range of allowed masses for candidate sequences as [**min**, **max**).

min=(float): Minimum peptide mass (M+H) (inclusive).

max=(float): Maximum peptide mass (M+H) (non-inclusive).

Values must be non-negative.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **digest_mass_range 400.0 5000.0**

decoy_search 0|1

Conduct a target-decoy search. If set to 1, Tempest generates and scores a decoy sequence for each candidate peptide sequence. Decoys are generated by reversing the internal sequence while fixing the specific enzymatic cleavage residue in place.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **decoy_search 0**

fixed_modification [location] [mass delta]

Modification mass to be applied to all specified residues/termini.

location=(string): Mod mass is applied individually to each residue in contiguous string.

location=peptide-nterm: Mod mass is applied to peptide N-terminus.

location=peptide-cterm: Mod mass is applied to peptide C-terminus.

location=protein-nterm: Mod mass is applied to protein N-terminus.

location=protein-cterm: Mod mass is applied to protein C-terminus.

mass delta=(float): Mass delta in Daltons.

Minimum occurrences: 0

Maximum occurrences: any; modifications on repeated sites are additive.

variable_modification [location] [symbol] [mass delta]

When specified residues/termini appear in a peptide, consider the peptide both with and without the modification mass applied to the residue/terminus.

location=(string): Mod mass is applied individually to each residue in contiguous string.

location=peptide-nterm: Mod mass is applied to peptide N-terminus.

location=peptide-cterm: Mod mass is applied to peptide C-terminus.

location=protein-nterm: Mod mass is applied to protein N-terminus.

location=protein-cterm: Mod mass is applied to protein C-terminus.

Symbol=(char): Residues with this modification are annotated with the specified symbol when printing peptide sequences. Symbol must be printable ASCII and non-alphabetical.

mass delta=(float): Mass delta in Daltons.

Minimum occurrences: 0

Maximum occurrences: any, but up to five variable modifications per residue or terminus. Multiple variable modifications on a single residue are mutually exclusive rather than additive.

variable_modifications_max [n]

Tempest considers all possible variable modification patterns for each peptide, discarding those patterns that contain more than **n** variably-modified residues/termini.

n=(int): Maximum modifications allowed per peptide (inclusive).

minimum_occurrences: 0

maximum_occurrences: 1

Default value: **variable_modifications_max** 3

neutral_loss [location] [symbol] [mass delta] [weighting]

When the specified residues, modified residue symbols, or termini appear in a peptide fragment, generate an additional theoretical fragment minus **mass delta**. This neutral loss peak contributes to the peptide score after being multiplied by **weighting**.

location=(string): Apply neutral loss to fragments containing any residue in the string.

location=nterm: Apply neutral loss to fragments containing the peptide or protein N-terminus.

location=cterm: Apply neutral loss to fragments containing the peptide or protein C-terminus.

mass delta=(float): Mass delta in Daltons.

weighting=(float): Factor by which to multiply matched neutral loss peak intensities before adding to the score. A **weighting** of 1.0 weights the neutral loss peaks as highly as the primary fragment peaks.

Minimum occurrences: 0

Maximum occurrences: any

a_ions [weighting]

weighting=(float): When **weighting** is a non-zero value, incorporate score contributions from the theoretical A-ion series, scaled by a factor of **weighting**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **a_ions 0.0**

b_ions [weighting]

weighting=(float): When **weighting** is a non-zero value, incorporate score contributions from the theoretical B-ion series, scaled by a factor of **weighting**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **b_ions 0.0**

c_ions [weighting]

weighting=(float): When **weighting** is a non-zero value, incorporate score contributions from the theoretical C-ion series, scaled by a factor of **weighting**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **c_ions 0.0**

x_ions [weighting]

weighting=(float): When **weighting** is a non-zero value, incorporate score contributions from the theoretical X-ion series, scaled by a factor of **weighting**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **x_ions 0.0**

y_ions [weighting]

weighting=(float): When **weighting** is a non-zero value, incorporate score contributions from the theoretical Y-ion series, scaled by a factor of **weighting**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **y_ions 0.0**

z_ions [weighting]

weighting=(float): When **weighting** is a non-zero value, incorporate score contributions from the theoretical Z-ion series, scaled by a factor of **weighting**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **z_ions 0.0**

ms_level 1|2|3

Score candidate peptides against scans at MSⁿ level **1**, **2**, or **3**. Multiple MSⁿ levels can be included as repeat **ms_level** definition lines. If this parameter is omitted, only MS² scans will be scored.

Minimum occurrences: 0

Maximum occurrences: 3

Default value: **ms_level 2**

precursor_tolerance [tol] Da|ppm

Search candidate peptide sequences against spectra with precursor masses within **tol** units of the sequence mass.

tol=(float): tolerance (+/-) in either Da or ppm.

Minimum occurrences: 1

Maximum occurrences: 1

fragment_bin_width [width]

width=(float): MS/MS bin width in m/z.

Minimum occurrences: 1

Maximum occurrences: 1

fragment_bin_offset [offset]

offset=(float): shift MS/MS bins to the right by **offset** * **fragment_tolerance**. Value should be between 0.0 and 1.0.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **fragment_bin_offset 0.0**

normalization_regions [n]

n=(int): Number of normalization regions used in MS/MS spectrum preprocessing.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **normalization_regions 10**

intensity_threshold [f]

f=(float): In each MS/MS spectrum, discard peaks below an intensity threshold defined by the base peak intensity multiplied by **f**. Value should be between 0.0 and 1.0.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **intensity_threshold 0.0**

msms_remove_precursor 1|0

If set to **1**, remove MS/MS peaks matching the precursor mass (using a tolerance of **fragment_tolerance**).

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **msms_remove_precursor 1**

msms_remove_mz_range [min] [max]

Remove MS/MS peaks from the m/z range provided by [**min**, **max**].

min=(float): minimum bound of m/z range (inclusive).

max=(float): maximum bound of m/z range (inclusive).

Minimum occurrences: 0

Maximum occurrences: any

xcorr_transform_width [width]

Variable-width SEQUEST-style xcorr transformation for MS/MS spectra.

i=(int): For each MS/MS bin, averaged intensity from +-**i** surrounding bins is subtracted from the bin intensity. Value must be manually scaled alongside **fragment_tolerance**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **xcorr_transform_width 0**

flanking_intensity [f]

f=(float): Each MS/MS bin propagates its intensity, multiplied by **f**, into the two bins directly flanking it. Value should be between 0.0 and 1.0.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **flanking_intensity 0**

num_internal_psms [n]

n=(int): Track **n** top-scoring PSMs per spectrum internally. Value must be at least 2 in order to compute **DeltaScore**. Increase this value to compute deeper **DeltaScore_backbone** scores.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **num_internal_psms 10**

num_output_psms [n]

n=(int): Report up to **n** top-scoring PSMs per spectrum in the search results. Value cannot exceed **num_internal_psms**.

Minimum occurrences: 0

Maximum occurrences: 1

Default value: **num_output_psms 10**

Appendix C: command line options

The following is a copy of the help text ('tempest -h') describing command line arguments for the Tempest executable. Basic Protocol 3 includes an example of these arguments being applied.

USAGE tempest [options]

OPTIONS

-i <PATH>	spectra file (override method)
-f <PATH>	fasta database (override method)
-m <PATH>	method file (default: tempest.method)
-c <PATH>	config file (default: tempest.config)
-o <PATH>	output file (default: <spectra path>.tsv)
-d <N:N>	OpenCL platform:device to use (override config)
-s <N:N>	Scan range (min:max MS2 scan IDs)
-l <N>	print level (verbosity)
-h	help (display this dialog)
-v	version

MISC

--device-info	Print OpenCL device map.
---------------	--------------------------

Literature Cited

- Brodbeck JS. Ion Activation Methods for Peptides and Proteins. *Analytical Chemistry*. 2015; 88(1):30–51. [PubMed: 26630359]
- Eng JK, Fischer B, Grossmann J, MacCoss MJ. A Fast SEQUEST Cross Correlation Algorithm. *Journal of Proteome Research*. 2008; 7(10):4598–4602. [PubMed: 18774840]
- Eng JK, Jahan TA, Hoopmann MR. Comet: an open-source MS/MS sequence database search tool. *Proteomics*. 2013; 13(1):22–4. [PubMed: 23148064]
- Eng JK, McCormack AL, Yates JR. 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):976–989. [PubMed: 24226387]
- Faherty BK, Gerber SA. MacroSEQUEST: Efficient Candidate-Centric Searching and High-Resolution Correlation Analysis for Large-Scale Proteomics Data Sets. *Analytical Chemistry*. 2010; 82(16):6821–6829. [PubMed: 20684545]
- Milloy JM, Faherty BK, Gerber SA. Tempest: GPU-CPU Computing for High-Throughput Database Spectral Matching. *Journal of Proteome Research*. 2012; 11(7):3581–3591. [PubMed: 22640374]

```
File Edit View Search Terminal Help
GNU nano 2.5.3 File: tempest.config

> Tempest v2.0 system configuration file
> Text following a '>' character is commented out.
> Run 'tempest --device-info' to see available platforms and devices.

platform      0
device        0

max_host_mem   4 gb
max_device_mem 1 gb

parallel_reduce 1

profile        0
min_work_size  32

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Figure 1.
Editing a tempest configuration file.


```
File Edit View Search Terminal Help
[glpp@g03 tempest]$ ./tempest --device-info
Platform 0
  VERSION = OpenCL 1.1 CUDA 6.5.14
  NAME    = NVIDIA CUDA
  VENDOR  = NVIDIA Corporation
  Device 0 (0:0)
    NAME           = Tesla M2075
    VENDOR         = NVIDIA Corporation
    DEVICE_VERSION = OpenCL 1.1 CUDA
    DRIVER_VERSION = 340.29
    MAX_COMPUTE_UNITS = 14
    MAX_WORK_GROUP_SIZE = 1024
    MAX_CLOCK_FREQUENCY = 1147
    GLOBAL_MEM_SIZE = 6442123264
    MAX_MEM_ALLOC_SIZE = 1610530816
  Device 1 (0:1)
    NAME           = Tesla M2075
    VENDOR         = NVIDIA Corporation
    DEVICE_VERSION = OpenCL 1.1 CUDA
    DRIVER_VERSION = 340.29
    MAX_COMPUTE_UNITS = 14
    MAX_WORK_GROUP_SIZE = 1024
    MAX_CLOCK_FREQUENCY = 1147
    GLOBAL_MEM_SIZE = 6442123264
    MAX_MEM_ALLOC_SIZE = 1610530816
```

Figure 2.
Output of **tempest --device-info** on a system with two Nvidia Tesla M2075 GPUs.

```
File Edit View Search Terminal Help
GNU nano 2.5.3 File: tempest.method

> Tempest v2.0 method file (basic tryptic high-res search)
> Text following a '>' character is commented out.

spectra_file           /absolute/path/to/spectra.mzXML
database_file          /absolute/path/to/db.fasta

digest_definition      KR P 1
digest_specificity     2
digest_missed_cleavages 3
digest_length          5 64
digest_mass_range      400 5000

decoy_search           0

fixed_modification      C 57.021460
variable_modification   M * 15.994910
variable_modifications_max 4

a_ions                 0.0
b_ions                 1.0
c_ions                 0.0
x_ions                 0.0
y_ions                 1.0
z_ions                 0.0

precursor_tolerance     1 Da
fragment_bin_width      0.02
fragment_bin_offset     0.0
normalization_regions   20
msms_remove_precursor   1
intensity_threshold     0.0

xcorr_transform_width   0
flanking_intensity      0.5
num_internal_psms       10
num_output_psms         10

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit     ^R Read File ^\ Replace  ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figure 3.
Editing a search method file.

```
File Edit View Search Terminal Help
[glpp@g03 tempest]$ ls
config  Makefile  mstoolkit  results  spectra  tempest.config
fasta  methods  README.md  source  tempest  tempest.method
[glpp@g03 tempest]$ ./tempest -f fasta/human.fasta -i spectra/f04747.mzXML -o re
sults/tmt-phos -m methods/trypsin-TMT-phos-highres.method
» (0:0) Using Tesla M2075
» (0:1) Using Tesla M2075
» Preparing 13955 MS/MS spectra... 100%
» (0:0) Allocating 1023.70 MB of device memory for 2684 cached spectra.
» (0:1) Allocating 1023.70 MB of device memory for 2684 cached spectra.
» Digesting 20241 proteins... 100%
» Writing results to results/tmt-phos.tsv...
[glpp@g03 tempest]$
```

Figure 4.

Example of executing Tempest from the command line. In this example, the **tempest** executable is in the current working directory alongside folders named **fasta**, **spectra**, **methods**, and **results**. Tempest will read protein sequences from **fasta/human.fasta** and spectrum data from **spectra/f04747.mzXML**. Output will be written to **results/tmt-phos.tsv** and **results/tmt-phos.log**. Search method parameters are read from **methods/trypsin-TMT-phos-highres.method**. The system configuration is read from the default **tempest.config** in the working directory because the **-c** argument is omitted.

```
File Edit View Search Terminal Help
Spectra: 13882
Masses: 1398.780762 - 4976.008301 Da
Proteins: 20241
Peptides: 3558041
PSMs: 271108064
Runtime: 142s
Modifications:
C      + 57.021460      = 160.030645 Da
K@     + 229.162932     = 357.257895 Da
S#     + 79.966330     = 166.998358 Da
T#     + 79.966330     = 181.014008 Da
Y#     + 79.966330     = 243.029659 Da
M*     + 15.994910     = 147.035395 Da
peptide-nterm@ + 229.162933 Da
=====
:[]
```

Figure 5.
a portion of the log file readout from a search with phosphorylation, TMT 6-plex labeling, and methionine oxidation as variable modifications. Cysteine carbamidomethylation is applied as a static modification. The modification masses and corresponding symbols inform interpretation of the sequences printed in the search results.

File Edit View Search Terminal Help													
[gpp@03 tempest]\$ column -t results/tmt-phos-dynamic_20717.tsv													
Scan	PrecursorMass	Charge	Candidates	Rank	Sequence	Ref	Decoy	Occurrences	SequenceMass	ppm	Score	DeltaScore	DeltaScore_backbone
20717	1855.93364	3	9012	1	R.n@SLGSLQTPEAVTTRc.K	sp Q7Z2W4 ZCOHV_HUMAN	0	1	1855.9364	-1.4904	4.4652	0.0221	0.5327
20717	1855.93364	3	9012	2	R.n@SLGSLQTPEAVTTRc.K	sp Q7Z2W4 ZCOHV_HUMAN	0	1	1855.9364	-1.4904	4.3663	0.1171	0.5221
20717	1855.93364	3	9012	3	R.n@SLGSLQTPEAVTTRc.K	sp Q7Z2W4 ZCOHV_HUMAN	0	1	1855.9364	-1.4904	3.8551	0.2244	0.4587
20717	1855.93364	3	9012	4	R.n@SLGSLQTPEAVTTRc.K	sp Q7Z2W4 ZCOHV_HUMAN	0	1	1855.9364	-1.4904	2.99	0.3021	0.3021
20717	1855.93364	3	9012	5	K.nWT#GVKAIGPIQLYAMKc.Y	sp A8K714 CLCA1_HUMAN	1	0	1855.95398	-10.9616	2.0868	0.1168	0.1168
20717	1855.93364	3	9012	6	K.nSSKSST#PVPSKQSARc.W	sp Q9H8V3 ECT2_HUMAN	0	1	1855.94763	-7.5415	1.8431	0	0.07
20717	1855.93364	3	9012	7	K.nSSKSST#PVPSKQSARc.W	sp Q9H8V3 ECT2_HUMAN	0	1	1855.94763	-7.5415	1.8431	0.07	0.07
20717	1855.93364	3	9012	8	-nMEAAAVTVTRSA#RRRc.R	sp Q8NFI3 ENASE_HUMAN	0	1	1855.93201	0.8774	1.7141	0.0207	0.0207
20717	1855.93364	3	9012	9	K.n@SLGSLQTPEAVTTRc.K	sp Q9H8V3 ECT2_HUMAN	0	1	1855.94763	-7.5415	1.6786	0.0078	1
20717	1855.93364	3	9012	10	K.nSSKSST#PVPSKQSARc.W	sp Q9H8V3 ECT2_HUMAN	0	1	1855.94763	-7.5415	1.6656	1	1

Figure 6.

Example top-10 search results (ranked by Score) for one scan illustrating sequence strings with variable modifications. The top 4 results share a sequence backbone, i.e. different modification patterns mapped to the same sequence of amino acids. **DeltaScore** compares each **Score** to the next-ranked result, while **DeltaScore_backbone** compares each **Score** to the next-ranked result with a different peptide backbone. The top result for this scan has a high **DeltaScore_backbone** (**Rank 1** vs. **Rank 5**) but a low **DeltaScore** (**Rank 1** vs. **Rank 2**), indicating high confidence in the backbone sequence but lower confidence in the phosphosite localization. Since this search tracked a maximum of 10 PSMs per scan, the **Rank 10** result has a **DeltaScore** of 1.0 based on the absence of a lower-ranked result against which to compare.