Aligator v1.1 Description and Installation
Updated on 09/04/2017

I. Introduction

To help overcome tedious, manual chemical protein synthesis designs, we developed the "Automated Ligator", or Aligator program. Using individual FASTA text files as input, Aligator creates a rank-ordered list of potential ligation strategies based on scoring algorithms that evaluate junction sites, segment solubility and length, and number of ligations (see section IV for details). The overall schematic for Aligator is shown in Fig. 1 below, as well as Fig. 2 in the Aligator manuscript (https://www.ncbi.nlm.nih.gov/pubmed/28651912). The input for Aligator is individual FASTA text files within a single folder. The script first finds all the Cys and Ala junction sites within a protein, and then a list of viable segments is created. Viable segments have C-terminal residues that would not result in "forbidden" thioesters (Asp, Glu, Asn, Pro, or Gln). Asp and Glu can undergo thioester migration to the side chain[1], and Pro thioesters have extremely slow native chemical ligation kinetics.[2] Asn, Gln, and Asp cannot be prepared via the hydrazide method[3], which is our technique of choice. These thioester restrictions can be modified by the user via the "Custom Thioester Input" Excel file, which is an optional file that can be edited and placed into the FASTA text files folder (See section III). Viable segments must also be at least 10 residues in length, but no longer than a user-defined number of residues.

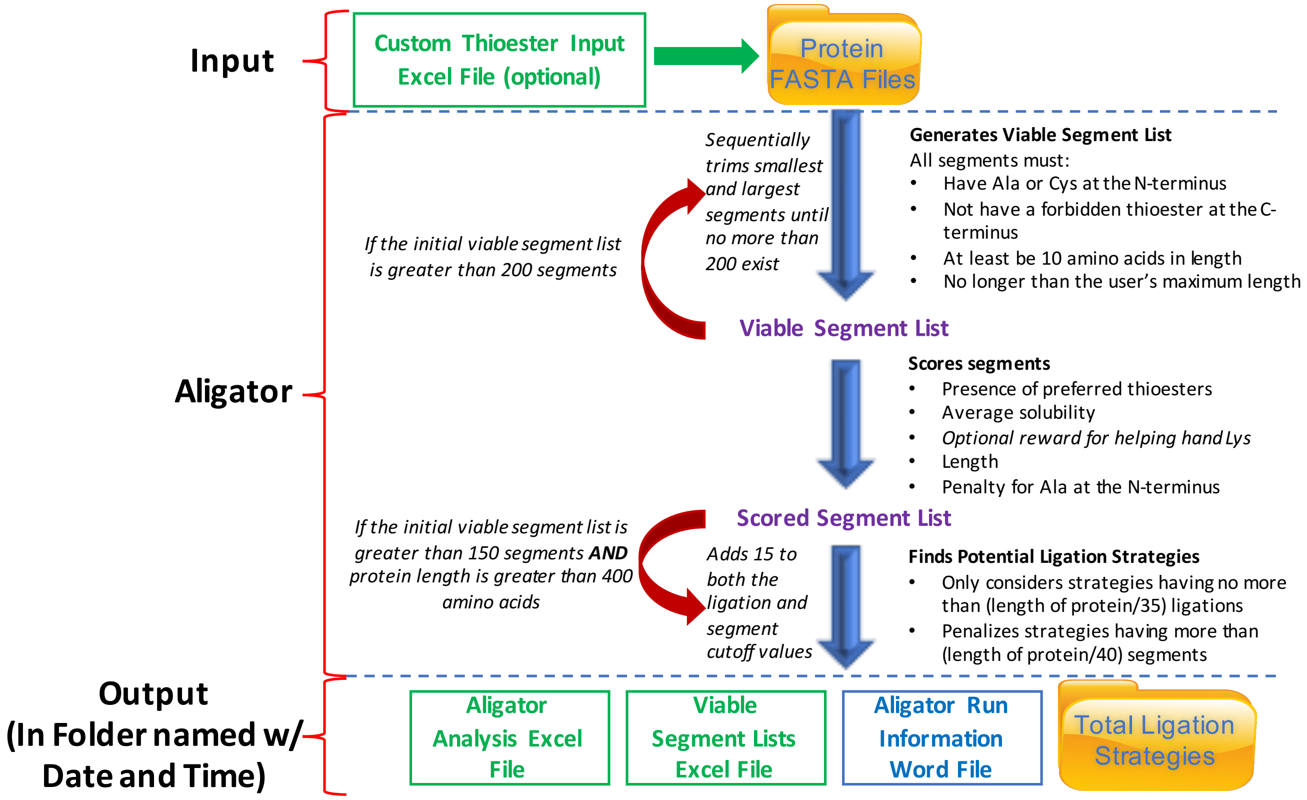
Input
Custom Thioester Input Excel File (optional)
Protein FASTA Files
Aligator
Sequentially trims smallest and largest segments until no more than 200 exist
If the initial viable segment list is greater than 200 segments
Generates Viable Segment List
All segments must:
Have Ala or Cys at the N-terminus
Not have a forbidden thioester at the C-terminus
At least be 10 amino acids in length
No longer than the user's maximum length
Viable Segment List
Scores segments
Presence of preferred thioesters
Average solubility
Optional reward for helping hand Lys
Length
Penalty for Ala at the N-terminus
Scored Segment List
If the initial viable segment list is greater than 150 segments AND protein length is greater than 400 amino acids
Adds 15 to both the ligation and segment cutoff values
Finds Potential Ligation Strategies
Only considers strategies having no more than (length of protein/35) ligations
Penalizes strategies having more than (length of protein/40) segments
Output (In Folder named w/ Date and Time)
Aligator Analysis Excel File
Viable Segment Lists Excel File
Aligator Run Information Word File
Total Ligation Strategies

Figure 1: Detailed schematic diagram describing how Aligator generates predictions of optimal ligation strategies for a protein based on the FASTA text file. The text to the right of the blue arrows describes the main components by which Aligator predicts optimal ligation strategies. The red arrows correspond to functions within Aligator's default "restriction mode", which helps to overcome computational costs for large proteins. Restriction mode can be turned off by the user in order to enter Aligator's "safe mode."

Once the viable segment list is prepared, Aligator evaluates each of the segments through four different scoring functions (see section IV). The first function scores the segments based on the presence of a preferred thioester (+2) or an acceptable thioester (0), and these thioester characterizations may be modified through the “Custom Thioester Input.xlsx” file (see section III). Another function scores the segment based on its average solubility, and the score ranges between 0 and -3. This solubility function also contains an optional “helping hand” lysine linker reward component, if activated by the user. A third function assigns each segment with a score based on length, with the highest score being set at +2 and the lowest score depending on the user’s maximum segment length. The final function penalizes any segments containing Ala as an N-terminal ligation junction with a score of -2 (vs. no score for Cys).

After scoring all of the viable segments, Aligator finds all potential ligation strategies for the protein of interest. While possible assemblies are being compiled, Aligator applies a final scoring component: a penalty for strategies that have more than (length of the protein / 40) total segments (see section IV for details). In other words, strategies that have an average segment length <40 residues are penalized for each additional ligation over this threshold. This function gives a score of -2 for each “excessive” ligation in these assemblies.

Compiling all the possible ligation strategies represents the most computationally expensive component of Aligator. To reduce the number of strategies that Aligator computes, the script only considers strategies that have no more than (length of the protein / 35) total ligations. 35 was chosen based on the fact that the final scoring function considers an average segment length of 40 amino acids to be “ideal” (see section IV for more discussion on this “ideal” segment length). As a result, strategies having > (length of the protein / 35) total ligations would be scored very poorly, thus allowing such strategies to be disregarded.

As described in the Aligator manuscript (https://www.ncbi.nlm.nih.gov/pubmed/28651912), we used Aligator to analyze past chemical protein synthesis projects completed in our lab (TNFα, GroES, and DapA), as well as the proteins making up the 30S, 50S, and Accessory/Translation Factors of the *E. coli* ribosome (Table S1 in manuscript). For some of the largest proteins in our *E. coli* ribosomal test set (S1, EF-G, and IF2), we discovered that our early Aligator script was much too computationally expensive for widespread use. We therefore developed a “restriction mode” to help reduce the computational costs associated with larger proteins. As Figure 1 illustrates (red arrows), the first function of restriction mode is triggered if the number of segments is larger than 200 and will sequentially trim the smallest and largest segments in the viable segment list until ≤ 200 segments remain. When this mode is triggered, Aligator will alert users to the new cutoffs for the minimum and maximum segment lengths. A second component of restriction mode is activated if the number of segments is >150 and the length of the protein is >400 residues. This component will add 15 to both the ligation and segment cutoffs used when finding potential strategies. Specifically, the script will only consider strategies that have ≤ (length of the protein / 50) ligations and will penalize strategies that have > (length of the protein / 55) segments. These requirements and restrictions were chosen based on trial-and-error when running S1, EF-G, and IF2. The current cutoffs allowed Aligator to produce viable ligation strategies for all but one protein in our data set.

The one protein that did not generate a viable ligation strategy due to Aligator’s restriction mode was IF2. This was caused by a required (unique) ligation segment being cut during the segment trimming process. To account for situations where restriction mode eliminates all viable strategies, users can turn off restriction mode. In this case, the script will analyze the protein using the normal ligation and segment cutoffs (35 and 40) without trimming any sequences from the viable segment list. To limit the computational cost and avoid memory errors, Aligator in this situation enters “safe mode”. In safe mode, Aligator will stop finding strategies when a certain threshold is reached. If the operating system is Mac OS X, then the script will stop after a certain amount of memory is consumed (100 MB per processor), whereas the script will stop after 10 minutes of searching for other operating systems.

All output data is stored within a folder named by the date and time in which Aligator was initialized. The main output file of Aligator is the “Aligator Analysis” Excel file, which shows the top 1000 ligation strategies for each protein. The “best” strategies are those that have the highest score, which is compiled by adding the overall score for each individual segment while considering any penalties imposed for strategies containing too many segments. Aligator also shows all of the potential strategies for a protein within the “All Strategies” text file, which is located in the “Total Ligation Strategies” folder. All of the segments used to find potential ligation assemblies for every protein in the directory are stored in the “Viable Segment Lists” Excel file. Finally, the “Aligator Run Information” Word document shows the user inputs, protein ligation information, and total run time for each protein run through Aligator.

II. Installing Aligator

A. Executable

Aligator is now available as an executable for both Mac and Windows operating systems! This removes the need for users to create script directories and install Python packages on their computer.

NOTE: The Aligator executable for Mac may only work on Mac OS X versions 10.8 and above. The executable for Windows may only work on Windows versions 7 and above. If you cannot run the executable, please follow parts B or C below to manually install Aligator on your computer.

1. Download the Aligator executable file from Github

1.1: Go to the "Releases" section on the Aligator Github repository: https://github.com/kay-lab/Aligator/releases.

1.2: At the bottom of the release for Version 1.1, download either the "Aligator_Mac_Executable" or "Aligator_Windows_Executable" compressed folders, depending on your operating system.

1.3: After opening the folder, place the executable file and the "Custom Thioester Input" Excel file in any location on your computer.

NOTE: Depending on your computer's settings, the Aligator executable may not run when you first try to run it. You will need to modify your computer's settings to allow executables from the internet to be opened. With any executables downloaded from the internet, it is also recommended to run them through a virus protection software to ensure there is no malicious content.

B. Mac OS X

The installation processes are done using the Terminal application, which can be found in the Utilities section of the Applications folder.

Mac OS X (versions 10.8 and above) comes with Python 2.7 (http://www.python.org) pre-installed. Type "python" in the terminal to confirm the presence of Python version 2.7.x (the value of x is not important). Otherwise, download and install Python version 2.7.x (https://www.python.org/downloads). Aligator will not work with Python 3.0 or higher. After checking the version of Python, you can exit Python by entering "exit()" into Terminal.

1. Prepare a script directory

1.1: On your desktop, prepare a new folder. This folder will be your "script directory," which will store Aligator and allow the script to be called within Terminal.

1.2: To run Aligator in any location on your Mac, your operating system needs to know where the script is located. Your "bash profile" tells the operating system where to look when commands are entered into Terminal, so you will need to export your script directory's path into your bash profile.

- Enter "cd" into the Terminal window.
- Enter "nano .bash_profile".
- Export the path of your script folder into your bash profile. This can be done by typing into a blank line "export PATH=$PATH:" followed by dragging your script folder into the Terminal window. This will paste your folder's path into the line.

(example: export PATH=$PATH:/Users/nszabo/Desktop/scripts)

- Press “ctrl” + “o”, followed by pressing “enter”, to write out and save your modified bash profile. Press “ctrl” + “x” to exit the bash profile.
- Restart Terminal to have the changes take effect.

1.3: Any scripts that you place into this script directory can now be called within Terminal by simply entering the entire name of the script into the Terminal window.

2. Download “Aligator.py” and the “Custom Thioester Input.xlsx” Excel file

2.1: Go to the following Github repository to find the source code and the Custom Thioester Input file for Aligator: https://github.com/kay-lab/Aligator.

2.2: Click the “Clone or download” button. Choose the “Download ZIP” option to download “Aligator.py” and all the accompanying files to your computer.

2.3: Place the “Aligator.py” script and the “Custom Thioester Input.xlsx” Excel file into your script directory. As noted in section III, the Excel file will need to be placed into the folder containing your FASTA text files in order to enable custom thioester characterizations. The remainder of the accompanying files in the ZIP folder do not need to be stored in the script directory, as these just contain more information about what Aligator does.

3. Install the openpyxl and joblib Python libraries

3.1: Enter “cd” into the Terminal window.

3.2: Enter “sudo easy_install openpyxl”. You will then be prompted to enter a password, which is the one linked to your Mac user account.

- If your account does not have administrator privileges, then this command will not install openpyxl. If you know the username and password of an account with administrative privileges, you will need to log in to your Mac with this account to install openpyxl.

3.3: Enter “sudo easy_install joblib”. You will again be prompted to enter your password.

- Again, you will not be able to install joblib unless you are logged into an account that has administrative privileges.

C. Windows

The following instructions describe how to install Cygwin, a collection of tools that provides functionality similar to Linux on Windows. Since Linux and Mac OS are similar to each other, Cygwin is similar in feeling to the Mac OS X’s Terminal application.

While this tutorial was written for using Cygwin to run Aligator, you should not need to install Cygwin to run Aligator. Alternatively, you can install Python 2.7.x (the value of x does not matter), install the openpyxl and joblib Python libraries into Python 2.7’s package library, and download “Aligator.py” from the following Github repository: https://github.com/kay-lab/Aligator. To run Aligator through this method, you will need to make sure that you place the “Aligator.py” script into the folder containing your FASTA text files (see section III), and then run Aligator through the IDLE shell that comes with Python 2.7.

1. Download and install Cygwin

1.1: Download and run the Cygwin installation executable file, which can be found on Cygwin's website: https://cygwin.com/install.html. You will need to install either the 32- or 64-bit version of the program, depending on the bit version of your Windows operating system.

1.2: Open the installation executable file. Follow the on-screen Cygwin installation instructions, as well as the following instructions, to correctly install Cygwin with the necessary packages to run Aligator:

- When the installation executable asks you to choose a download source, choose "install from Internet".
- For the "Select Root Install Directory" page, you can place Cygwin anywhere that you wish, as well as make it available for all users or just for your user account. Leaving the default root directory will work.
- For the "Select Local Package Directory" page, you can select any location to store the installation files. The default location suggested by the executable will work fine.
- For the "Choose a download site" page, you can select any site to install Cygwin from. When preparing this tutorial, the "http://cygwin.mirror.constant.com" site was selected and worked.
- When the "Select packages" page appears, type "Python" into the Search box. You will see a category called "Python" appear in the list of packages. Click on the word "Default" listed to the right of the Python package, and the word will change to "Install".
- Next, search for "nano" in the search box. Next to the word "All" that appears in the results, click on the word "Default" to change it to "Install". Nano is a text editor that will help with setting up your script directory later.
- Click the "Next" button in the bottom right corner of the page.
- Install the suggested dependency files. Ignore any warning messages telling you to install packages from Win32, as these are not important for Aligator functionality.
- Allow Cygwin to install on your computer. This can take a long time (1-2 hours). If you need to pause the installation, you can close the installation executable and simply restart the executable at a later time. All of your selections will still remain when you restart, so you just need to click "Next" until the installation picks up where it left off.
- Once Cygwin is finished installing, you may see a few post-installation errors. These can be ignored, as they do not affect the functionality of Aligator. Finish the executable installation instructions, and Cygwin will be ready to use!

2. Check for the correct version of Python in Cygwin

2.1: Launch Cygwin and enter "python" into the command line. This will launch Python, and you will see the version next to the word "Python". Make sure that version 2.7.x (the value of x does not matter) is what launches, as Aligator is written for Python 2.7. If a later version of Python launches, you will need to go back to the Cygwin installation executable to uninstall this version of Python and make sure that 2.7.x gets installed.

2.2: Enter "exit()" into the command line to close Python after you have checked the version.

3. Prepare a script directory

3.1: On your desktop, prepare a new folder. This folder will be your "script directory," which will store Aligator and allow the script to be called within Cygwin.

3.2: To run Aligator in any location on your computer, your operating system needs to know where the script is located. Your "bashrc" tells the operating system where to look when commands are entered into Cygwin, so you will need to export your script directory's path into your bashrc.

- Enter "cd" into the Cygwin command line.
- Enter "nano –w ~/.bashrc".
- Scroll down to the bottom of the file that appears in the Cygwin window. The cursor needs to appear in the first blank line at the end of the file (a line that does not start with "#").
- Export the path of your script folder into bashrc. This can be done by typing into the blank line "export PATH=$PATH:" followed by dragging your script folder into the terminal window. This will paste the folder's path into the Cygwin window.

  (example: export PATH=$PATH:/Users/nszabo/Desktop/scripts)
- Press "ctrl" + "o", followed by pressing "enter", to write out and save your modified bashrc. Press "ctrl" + "x" to exit the bashrc.
- Restart Cygwin to have the changes take effect.

3.3: Any scripts that you place into this script directory can now be called within Cygwin by simply entering the entire name of the script into the Cygwin window.

4. Download "Aligator.py" and the "Custom Thioester Input.xlsx" Excel file

4.1: Go to the following Github repository to find the source code and the Custom Thioester Input file for Aligator: https://github.com/kay-lab/Aligator.

4.2: Click the "Clone or download" button. Choose the "Download ZIP" option to download "Aligator.py" and all the accompanying files to your computer.

4.3: Place the "Aligator.py" script and the "Custom Thioester Input.xlsx" Excel file into your script directory. As noted in section III, the Excel file will need to be placed into the folder containing your FASTA text files in order to enable custom thioester characterizations. The remainder of the accompanying files in the ZIP folder do not need to be stored in the script directory, as these just contain more information about what Aligator does.

5. Install the openpyxl and joblib Python libraries

5.1: Enter "cd" into the Cygwin command line.

5.2: Enter "easy_install-2.7 openpyxl". This will install the openpyxl Python package into your Python package library.

5.3: Enter "easy_install-2.7 joblib". This will install the joblib Python package into your Python package library.

III. Using Aligator

A. Download separate FASTA files for each protein of interest. Save each FASTA file as separate .txt files, and place all of the .txt files into one folder.
   - NOTE: You may name the files however you wish, but they must have the extension ".txt". In addition, if you place "fasta" or ".fasta" right before the ".txt" extension (e.g., Insulin fasta.txt or Insulin.fasta.txt), then Aligator will trim "fasta" out of the file name when writing the protein name into the output files. However, if "fasta" is written in uppercase, then Aligator will not trim it.

B. If using the Aligator executable, simply place the executable file into the folder containing your FASTA files. Double-click on the executable to launch Aligator (go to part E for further instructions). If you are not using the Aligator executable, follows parts C and D below.

C. Using the Terminal (Mac) or Cygwin (Windows) application, change the working directory to the folder containing the FASTA .txt files:
   - Type "cd " into the Terminal or Cygwin window (note the space following cd), followed by dragging your folder into the Terminal/Cygwin window. This pastes the path of your folder into the command line.
   - Press "enter".

D. Launch Aligator by simply entering "Aligator.py" into the Terminal/Cygwin window.

E. The script will first explain the default thioester characterizations built into Aligator (based mainly on Fmoc hydrazide SPPS; see section IV for more details). The user will then have the option to use the default thioester characterizations or to customize them using the "Custom Thioester Input.xlsx" Excel file. If you would like to customize the thioesters, edit the Excel file to classify all 20 possible thioesters (follow the directions listed in the heading of column B). Save your edited Excel file into the same folder containing the FASTA .txt files (**DO NOT CHANGE THE NAME OF THE EXCEL FILE**), and then close out of the Excel file.
   - Enter "yes" to keep the default thioester characterizations ("Custom Thioester Input.xlsx" does not need to be in the FASTA text file folder, in this case).
   - Enter "no" to customize thioester characterizations ("Custom Thioester Input.xlsx" file needs to be in the FASTA text file folder).

F. Aligator will next ask you to enter the maximum length of peptide segments that can be considered in making strategy predictions. Since the minimum length of segments is set to 10 residues, the number you enter needs to be larger than this minimum value.
   - Enter the maximum segment length allowed (using numbers only).

G. The script will explain restriction mode and will ask if you would like to turn this mode on.
   - Enter "yes" to turn on restriction mode.
   - Enter "no" to leave restriction mode off and enter safe mode (not recommended if you're analyzing a protein for the first time).

H. The script will then explain the helping hand Lys linker reward function and ask if you would like to turn on the reward function.
   - Enter "yes" to turn on the helping hand reward.
   - Enter "no" to leave the helping hand reward function off.

I. Before making strategy predictions, Aligator will show you the answers to the 4 questions that were asked in order to ensure that all of these parameters are set correctly.
   - If all of the parameters are correct, enter “yes” to continue to the strategy predictions.
   - Enter “no” to go back and re-enter all of the parameters.

J. Aligator will then start, and you will see the status of the analysis in the program window as each FASTA file is analyzed. If you have large proteins, any restriction mode component that is triggered by one of your FASTA files will be shown to you in the window. Additional details on these restrictions are included in the “Aligator Run Information” Word document.

K. When Aligator is finished, you will see the run time (in seconds). In your FASTA file folder, you will see an output folder named with the date and time in which Aligator was launched. Within this folder, you will see four components: the “Total Ligation Strategies Text Files” folder, the “Aligator Run Information” Word document, the “Aligator Analysis for <FolderName>” Excel file (contains top 1000 strategies for each protein), and the “Viable Segment Lists for <FolderName>” Excel file.

IV. Interpreting the Ligation Strategy Scores

As described above, Aligator predicts optimal ligation strategies by using 5 different scoring functions. As shown in Table 1, the "Aligator Analysis" Excel file will show you the total strategy score for the top 1000 hits, along with the total score from each individual function. The following will describe how each of the functions scores potential ligation strategies, as well as provide rationale for assigning scores in each function. Each scoring function was designed to give scores that are on the same scale, so that certain functions do not dominate the overall score.

Table 1: Example of data that is shown within the "Aligator Analysis" Excel output file. In an actual Excel output file, the peptide segments for each row's proposed synthesis strategy would be shown to the right of the final column in this table.

Total Strategy Score	Thioester Total Score	Solubility Total Score	Segment Length Total Score	Total Ala Junction Site Penalty	Total Penalty for # of Ligations
6.06827347	8	-0.53172653	6.6	-8	0
5.72720917	6	-0.27279083	8	-8	0
4.88043015	6	-0.31956985	7.2	-8	0

A. Thioester Score

1. Description of Scoring Function

Except for the segments corresponding to the C-terminus of the full-length protein, the thioester scoring function assigns segments a score of 2 or 0, depending on the C-terminal residue that would need to be converted to a thioester for native chemical ligation. While forbidden thioesters are not technically included in this scoring function, as these segments will be filtered out before being scored, it is worth mentioning again that Aligator does not allow segments with poor thioesters to be available for creating ligation assemblies. Table 2 shows all 20 possible thioesters and their assigned scores, based on the default Aligator settings (for thioesters prepared using the hydrazide method). The user can customize the thioester characterizations via the "Custom Thioester Input.xlsx" Excel file, as discussed in section III.

Table 2: List of potential thioesters and their scores (using the default Aligator thioester settings).

Category	Thioesters	Score Received
Preferred Thioesters	Ala, Arg, Cys, His, Gly, Met, Phe, Ser, Trp, Tyr	+2
Accepted Thioesters	Ile, Leu, Lys, Thr, Val	0
Forbidden Thioesters	Asn, Asp, Gln, Glu, Pro	Not allowed

2. Rationale

2.1: Preferred vs. accepted thioester list – Preferred thioesters were selected primarily for their enhanced ligation rates compared to the rates of accepted thioesters.[2] Lys was not selected as a preferred thioester due to reported lactamization.[4]

2.2: Forbidden thioester list – Asp and Glu have been shown to undergo thioester migration to the side chain.[1] Pro thioesters have extremely slow ligation kinetics.[2] Asp, Asn, and Gln thioesters cannot be easily prepared via the hydrazide method[3], which is our method of choice.

B. Solubility Score

1. Description of Scoring Function

Aligator first calculates a solubility value for each segment by simply scoring the segment for the presence of positive residues (+1 for each occurrence) or problematic residues (-1 for each occurrence). Table 3 lists the residues that are categorized as positively charged or problematic. The script then uses this solubility value to then determine an average "per residue" solubility value for the segment, based on Equation 1.

Table 3: List of residues considered to be positively-charged or problematic, and the corresponding value given to a segment when Aligator finds one in the sequence.

Category	Amino Acids	Score Received
Positively-charged	His, Lys, Arg	+1
Problematic	Asp, Glu, Val, Ile, Leu	-1

$$\overline{\boldsymbol{Solub}} = \frac{\boldsymbol{Solub}}{\boldsymbol{n}}$$

Equation 1: Calculation of average solubility, where $\overline{Solub}$ is equal to the average solubility score, *Solub* represents the solubility score calculated for a segment, and *n* equals the number of residues in a segment.

The average solubility score is used to assign a segment's final solubility score, which can be anywhere between 0 and -3. Table 4 shows the equations used, depending on the value of the average solubility score, to assign the final solubility score (see rationale section for details on the values used for the constants).

Table 4: Equations used to determine the final solubility score, depending on the average solubility value calculated by Equation 1. $\overline{Solub}$ corresponds to the segment's average solubility score, $\overline{SolubTestSet}$ is equal to the mean average solubility score for all viable segments in our ribosomal test data set (Table S1 in manuscript), $OneStdDev$, $TwoStdDev$, and $3StdDev$ refer to one, two, and three standard deviations away from the mean average solubility score for viable segments in the test data set, and FinalSolub is the final solubility score given to a segment.

$\overline{\boldsymbol{Solub}}$ **Score Range Needed**	**Final Score Equation**
$\overline{Solub} \geq \overline{SolubTestSet}$	FinalSolub = 0
$\overline{SolubTestSet} > \overline{Solub} \geq OneStdDev$	FinalSolub = $(-1)x(\frac{\overline{Solub}-\overline{SolubTestSet}}{OneStdDev-\overline{SolubTestSet}})$
$OneStdDev > \overline{Solub} \geq TwoStdDev$	FinalSolub =$\left((-1)x\left(\frac{\overline{Solub}-OneStdDev}{TwoStdDev-OneStdDev}\right)\right)-1$
$TwoStdDev > \overline{Solub} \geq 3StdDev$	FinalSolub = $\left((-1)x\left(\frac{\overline{Solub}-TwoStdDev}{3StdDev-TwoStdDev}\right)\right)-2$
$3StdDev > \overline{Solub}$	FinalSolub = -3

If the user decides to use the helping hand Lys linker reward function, then the presence of a Lys in a segment will result in the final solubility score being divided by 2. For example, if a segment receives a final solubility score of -2 from Table 4 and has a Lys in its sequence, then the final solubility score becomes -1 for the segment.

Each segment's average solubility score (calculated via Equation 1) and final solubility score (including any helping hand reward) are recorded in the "Viable Segment Lists" Excel file.

2. Rationale

2.1: Calculations of initial solubility values (Table 3) – The scoring system used to calculate an initial solubility value of a segment was based off of our experiences with working on peptides of varying solubility.[5-8] Generally, we observe that segments rich in positively-charged residues have increased solubility. On the other hand, peptides that have a lot of Asp, Glu, Val, Ile, or Leu (DEVIL) residues have been known to cause segment insolubility.

2.2: Calculation for average solubility score (Equation 1) – In order to obtain unique solubility scores for each segment, Aligator needs to calculate an average solubility score for all segments. This score is basically the mean solubility value per residue within a segment. Without this calculation, the solubility values of individual segments are heavily dependent on length, so obtaining the average solubility score normalizes this value.

2.3: Calculations for final solubility score (Table 4) – Instead of having the average solubility score being the final score given to a segment, we wanted to get an idea of what the "mean" average solubility value is for a random peptide segment in the viable segments list for segments in the *E. coli* ribosomal data set (Table S1 in manuscript). For example, our scoring function may result in most segments having negative average solubility scores, meaning that the negative values would not always correspond to poorly soluble segments, thus requiring readjustment of the zero point for the solubility score.

To get a sense of the mean average solubility score for segments, the average solubility scores were examined for all segments within the proteins of the *E. coli* ribosomal 30S subunit, 50S subunit, and Accessory/Translation Factors (Table S1 in manuscript). All of these segments satisfied the requirements of the initial viable segment list (between 10 and 80 residues long without the default forbidden thioesters). Figure 2 below shows the box-and-whisker plots obtained for segments within each of the 3 protein classes, as well as the total.

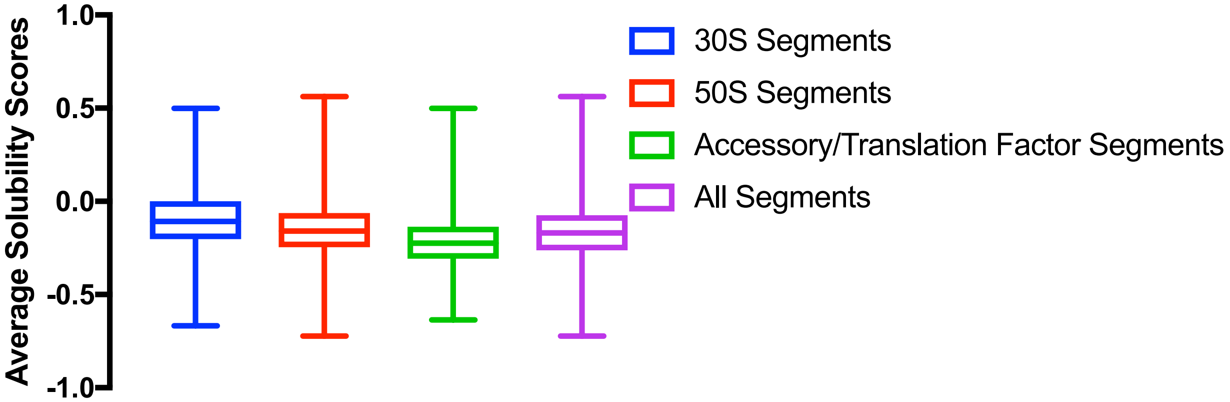

Figure 2: Box-and-whisker plots for the average solubility scores calculated on viable segments within the 3 *E. coli* ribosomal protein classes, as well as the total (see Table S1 in manuscript for list of proteins used). The following statistical data about the average solubility scores was calculated for each of the protein classes: *30S Subunit*: Mean =-0.1041, Standard deviation = 0.1565; *50S Subunit*: Mean = -0.1430, Standard deviation = 0.1585; *Accessory & Translation Factors*: Mean = -0.2203, Standard deviation = 0.1246; *Total*: Mean = -0.1581, Standard deviation = 0.1547

Figure 2 illustrates that the mean of the average solubility scores for all 3 of the ribosomal protein classes was slightly negative, meaning that setting the mean average solubility score at 0 for the final solubility scoring function is inappropriate. To set the true mean, we thus used the mean average solubility score for all segments in all three protein classes combined ("Total" in Figure 2). The 3 standard deviation from the mean values in Table 4 were calculated by using the standard deviation for all the segments. For example, the $OneStdDev$ value is equal to the standard deviation subtracted from the mean average solubility value. Table 5 shows the values used in Aligator to define these four values.

Table 5: Calculated values for each of the variables used in the final solubility segment calculation, which is described in Table 4.

Variable	Calculated Value
$\overline{SolubTestSet}$	-0.1581
$OneStdDev$	-0.3128
$TwoStdDev$	-0.4675
$3StdDev$	-0.6222

2.4: Helping hand reward – The recent development of the traceless helping hand lysine linker has helped to increase the solubility of problematic peptide segments.[7] Since the helping hand can dramatically increase solubility of previously insoluble segments, we include an optional function that rewards segments containing a Lys residue. If at least one Lys is observed within a segment, the helping hand reward function simply divides the segment's final solubility value (Table 4) by 2, representing a 50% decrease in the solubility penalty when the helping hand is incorporated into the segment.

C. *Segment Length Score*

1. **Description of Scoring Function**

Aligator employs a simple function for scoring segments on the basis of length. The program has 40 amino acids set as the "ideal" segment length, which results in the maximal score of +2 given to segments 40 amino acids in length. For segments that are smaller or larger than 40 residues, a penalty of 0.1 is given for each amino acid above or below this ideal. For example, a segment that is 57 residues long will have a length score of 0.3, whereas a segment 23 residues long will also get a score of 0.3.

2. **Rationale**

In our experiences with chemical protein synthesis of large proteins, an ideal length for each of the segments is approximately 40 amino acids. While segments between 20 and 60 amino acids are typically routine for solid-phase peptide synthesis, segments below 20 amino acids are usually not desired, as having many small segments requires more ligations to obtain the final protein. Additional ligations result in reduced yields due to the need for more RP-HPLC purifications. Segments above 60 residues can be synthesized, but these are much more likely to suffer from poor yield (e.g., increased level of contaminants and on-resin aggregation) and purity (e.g., reduced resolution using RP-HPLC purification).

D. *Alanine Junction Penalty*

1. **Description of Scoring Function**

Except for segments that represent the N-terminus of the desired protein, any segments that have an Ala at the N-terminus receive a penalty of -2. Segments containing a Cys at the N-terminus do not receive a penalty.

2. **Rationale**

While both Cys and Ala sites can be used as native chemical ligation junctions, Ala junction sites require an additional desulfurization step to convert the temporary Cys back to Ala after ligation has occurred. Additionally, native Cys in the protein must be protected when desulfurization steps are required. These additional steps, and potential yield losses from them, justify penalizing segments containing Ala junction sites compared to segments having Cys.[9,10] Aligator's source code can also be modified to use other ligation junctions (e.g., thiolated residues).

E. *Excessive Number of Ligations Penalty*

1. Description of Scoring Function

When a ligation strategy contains more than the ideal number of segments, this function penalizes the strategy's score by giving a -2 for each segment that is over the threshold segment number. For example, if the threshold segment number for a protein was calculated to be 10, then a ligation strategy containing 12 segments would be penalized with -4. Equation 2 shows the calculation of the threshold number of ligations for assigning penalties. As discussed in section I, Aligator uses an equation similar to Equation 2 for discarding ligation strategies that contain too many ligations. Instead of 40 being the denominator for this function, 35 is the number placed in the denominator, and the threshold number refers to ligation number, not segment number.

$$Thresh = \frac{n}{40}$$

Equation 2: The equation used to calculate the threshold segment number for assigning penalties in ligation strategies. $Thresh$ corresponds to the threshold segment number, and *n* equals the number of amino acids in the protein of interest.

2. Rationale

As discussed in the segment length scoring function section, the ideal length for peptide segments used in large chemical protein syntheses is 40 residues. As a result, Equation 2 uses 40 residues to calculate the penalty threshold segment number for ligation strategies. In other words, assemblies that have an average segment length that is less than 40 residues would be penalized by this function (-2 for every additional ligation).

It is important to note that, if a protein contains more than 150 segments in the viable segment list and is longer than 400 residues, Equation 2 changes so that the denominator is 55 instead of 40. In this case, the number is being scaled up to match the increase in the threshold ligation number used to calculate the number of ligations that can be used in a potential strategy (usually 35, but increases to 50 when the restriction mode gets activated; see Figure 1). The smaller number is always used to calculate the ligation number cutoff for strategies that will be considered in Aligator, whereas the larger number is always used to assess which strategies have an acceptable, but non-ideal, number of segments in order to penalize such strategies.

V. Troubleshooting:

If running a particular protein with Aligator causes a “Killed: 9” error to occur, then the memory being used by Aligator is too large, and Python kills Aligator to make sure Python does not crash. In this case, run the protein in safe mode (restriction mode turned off) to get an idea of the ligation strategies available for your protein.

References

1. Villain M, Gaertner H, Botti P. Native chemical ligation with aspartic and glutamic acids as C-terminal residues: Scope and limitations. *Eur J Org Chem.* 2003(17):3267-3272.
2. Hackeng TM, Griffin JH, Dawson PE. Protein synthesis by native chemical ligation: Expanded scope by using straightforward methodology. *P Natl Acad Sci USA.* 1999;96(18):10068-10073.
3. Fang GM, Wang JX, Liu L. Convergent Chemical Synthesis of Proteins by Ligation of Peptide Hydrazides. *Angew Chem Int Edit.* 2012;51(41):10347-10350.
4. Siman P, Karthikeyan SV, Nikolov M, Fischle W, Brik A. Convergent chemical synthesis of histone H2B protein for the site-specific ubiquitination at Lys34. *Angew Chem Int Ed Engl.* 2013;52(31):8059-8063.
5. Clinton TR, Weinstock MT, Jacobsen MT, et al. Design and characterization of ebolavirus GP prehairpin intermediate mimics as drug targets. *Protein Sci.* 2015;24(4):446-463.
6. Weinstock MT, Jacobsen MT, Kay MS. Synthesis and folding of a mirror-image enzyme reveals ambidextrous chaperone activity. *P Natl Acad Sci USA.* 2014;111(32):11679-11684.
7. Jacobsen MT, Petersen ME, Ye X, et al. A Helping Hand to Overcome Solubility Challenges in Chemical Protein Synthesis. *J Am Chem Soc.* 2016;138(36):11775-11782.
8. Petersen ME, Jacobsen MT, Kay MS. Synthesis of tumor necrosis factor a for use as a mirror-image phage display target. *Org Biomol Chem.* 2016;14(23):5298-5303.
9. Yan LZ, Dawson PE. Synthesis of peptides and proteins without cysteine residues by native chemical ligation combined with desulfurization. *J Am Chem Soc.* 2001;123(4):526-533.
10. Wan Q, Danishefsky SJ. Free-radical-based, specific desulfurization of cysteine: A powerful advance in the synthesis of polypeptides and glycopolypeptides. *Angew Chem Int Edit.* 2007;46(48):9248-9252.