FAS -Feature Architecture Similarity-

Julian Dosch, Holger Bergmann

January 10, 2019

1 FAS scoring

The FAS score is a combination of 2 different scores the Multiplicity score (new & classic) and the Positional Score. Additionally, there are two other scores the get calculated: The Clan score and the Length score.

1.1 Multiplicity Score (Classic)

$MS(S,O) = \sum_{i=1}^{N^S} w_i * \frac{N_i^S * N_i^O}{max(N_i^S, N_i^O)^2}$

Let S be the seed protein and O be the (potential) ortholog. The Multiplicity Score iterates over all features in the seed architecture. For each of these features it multiplies the number of occurrences (N) in the seed and the ortholog architecture. This is then divided by the squared maximum between the number of occurrences (N) in the seed and the ortholog architecture. Finally, the individual score is then multiplied by its weight (w). The sum of all weighted individual feature scores is the Multiplicity Score.

1.2 Multiplicity Score (New)

$MS(S,O) = \sum_{i=1}^{N^S} w_i * min(\frac{N_i^S * N_i^O}{(N_i^S)^2}, 1)$

Let S be the seed protein and O be the (potential) ortholog. The Multiplicity Score iterates over all features in the seed architecture. For each of these features it multiplies the number of occurrences (N) in the seed and the ortholog architecture. This is then divided by the squared number of occurrences (N) in the seed architecture. If the result of this calculation is greater than 1 it will be set to 1. Finally, the individual score is then multiplied by its weight (w). The sum of all weighted individual feature scores is the Multiplicity Score.

1.3 Positional Score

$PS(S,O) = \sum_{i=1}^{N^S} \frac{w_i}{N_i^S} \sum_{j=1}^{N_i^S} (1 - \min_{1 \leq l \leq N_i^O} |P_{i,j}^S - P_{i,l}^O|)$

Let S be the seed protein and O be the (potential) ortholog. The Positional Score iterates over all features in the seed architecture and over all occurrences of each of these features. For each occurrence, it finds the occurrence of that feature in the ortholog with the closest relative position in the sequence (P). The sum of the scores of all occurrences of a feature gives the Positional Score for the feature and the sum of the weighted scores for the features gives the full Positional Score between seed and ortholog.

1.4 complete FAS Score

$FAS(S,O) = \alpha * MS + \beta * PS$

The FAS score is a combination of the Multiplicity Score and the Positional

Score. Alpha and Beta are the weights for MS and PS, respectively. By default these weights are 0.7 for MS and 0.3 for PS.

1.5 Length Score (additional)

$LS(S,O) = \sum_{i=1}^{N^S} \frac{w_i}{N_i^S} \sum_{j=1}^{N_i^S} \frac{\min(L_{i,j}^S, L_{i,l}^O)}{\max(L_{i,j}^S, L_{i,l}^O)}$

The Length Score is an additional measure FAS can calculate. It iterates over all features in the seed architecture and over all occurrences of each of these features. For each occurrence, it calculates the length (L) difference between the occurrence in the seed and the positionally closest occurrence in the ortholog (see PS).

1.6 Weighting Features

The weighting schemes of FAS define how much each feature contributes to the score. There are several options for the weighting. The first is a uniform weighting where all feature are weighted equally. The second options needs a reference proteome. Here, FAS counts the occurrences of all features in the reference. The weighting is than calculated as follows:

$w_i = \frac{p_i}{\sum_{x=1}^{n} p_x}, with\ p_i = \frac{\sum_{l=1}^{n} f(o_l)}{f(o_i)}$

where o_i is the number of occurrences of feature i and n is the sum of all features in the currently scored path. This way, features that occur less often will have a higher weight than features that occur frequently. The function f(x) can take different forms depending on the options set by the user. The linear weighting where $f(x) = x$ is not always optimal as the difference between the most common features and the rarest can be quite extreme, with common features like low complexity regions numbering in the 1000s or even

10000s while some rare features may only occur once. This results in very common features having very low weights and becoming nearly obsolete for the scoring. To counteract this, FAS can use several different correction functions. By default, the counts are evened out by applying an natural logarithm rounded up to the an integer where $f(x) = ceil(ln(x))$

As a last way of defining the weight, the user can influence the weight of specific features directly by giving it a constraints file. FAS then ensures that the features will always have at least the weight set there.

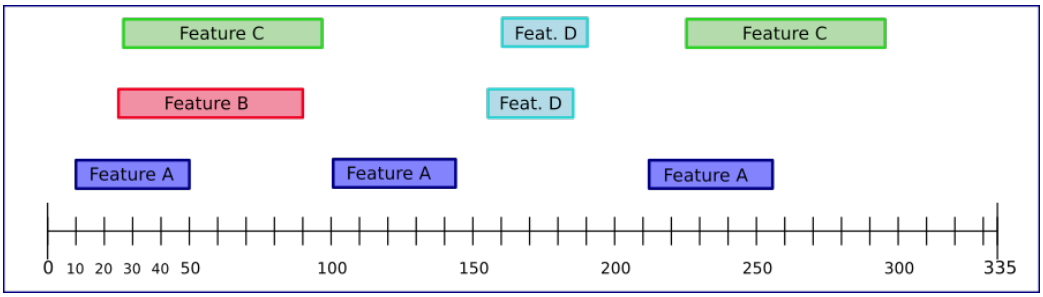
2 Resolving overlapping feature architectures

When annotating features onto a protein sequence, it can happen that several features occupy the same space in the sequence. This can for example happen in Pfam, where several features of the same clan are annotated onto the same space. This results with a redundancy of information which might not always be of interest.

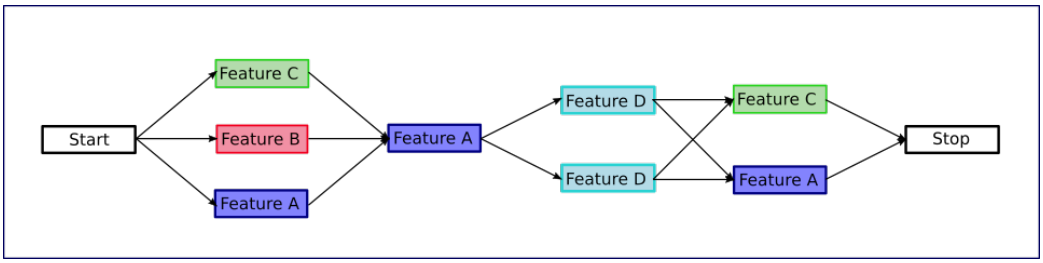
2.1 Resolving Overlaps

By default, FAS resolves feature architectures that have overlapping features from Pfam and Smart by searching for the best overlap-free path. This is normally done by extensively calculating the FAS score for all paths and picking the best scoring.

2.1.1 Example



The picture above shows an example of a feature architecture for a protein with the length 335 amino acids. It has four different features: A to D. Features A, C and D have multiple instances while B only has one. There regions with overlapping features: The first where instances of features A,B and C overlap, the second where two instances of feature D overlap and the third with instances of feature A and C overlapping. This architecture can be represented as a directed, acyclic graph where each instance is represented as a vertex together with an additional start and stop vertex. This graph can be seen below:



There are 12 possible paths:

CAD_1C, CAD_1A, CAD_2C, CAD_2A,

BAD_1C, BAD_1A, BAD_2C, BAD_2A,

AAD_1C, AAD_1A, AAD_2C, AAD_2A

2.2 Building the Graph

The example before showed that FAS represents all possible paths of a feature architecture as a graph. This graph is build in two main steps: FAS starts by creating a list for each feature that contains all non-overlapping features that have a higher start position. Additionally, the Stop node is added to each list and the Start node gets list containing all features and the Stop node as well as an empty list for the Stop node. The features are ordered by their starting position with the Start node preceding all features and the Stop node being in the last position. For the example architecture above this would look like this:

Start:(A1,B1,C1,A2,D1,D2,A3,C2,Stop)

A1:(A2,D1,D2,A3,C2,Stop)

B1:(A2,D1,D2,A3,C2,Stop)

C1:(A2,D1,D2,A3,C2,Stop)

A2:(D1,D2,A3,C2,Stop)

D1:(A3,C2,Stop)

D2:(A3,C2,Stop)

A3:(Stop)

C2:(Stop)

Stop:()

This is basically a graph that contains all paths including paths which 'skip' features in between. Since FAS should only evaluate 'full' paths all path that skip non-overlapping features have to be removed. This is done in the second step:

Here FAS goes through the graph backwards starting from the Stop node. At each step FAS goes through the list corresponding to the node. At the same time, it creates a second list of nodes which contains all nodes that can be reached directly or indirectly from the current node. This reached list can then be used for be used to remove edges that skip features. This process looks like this:

1. Step: Stop:() # The Stop node has no edges so nothing happens here, the reachable list stays empty

Stop:(),()

2. Step: C2:(Stop) # This node has an edge to the Stop node, the reachable list now also includes this node

C2:(Stop),(Stop)

3. Step: A3:(Stop) # Like Step 2

A3:(Stop),(Stop)

4. Step: D2:(A3,C2,Stop) # This node has multiple edges so there are more sub-steps, first we look at A3: since it has the Stop node in its reachable list this node can be removed from the edges and added to the reachable list of D2

D2:(A3,C2),(A3,Stop) # In the next sub-step we look at C2, it also has the Stop node in its reachable list, however that edge has already been removed, nothing else changes

D2:(A3,C2),(A3,Stop,C2)

5. Step: D1:(A3,C2,Stop) # Similar to step 4

D1:(A3,C2),(A3,Stop)

D1:(A3,C2),(A3,Stop,C2)

6. Step: A2:(D1,D2,A3,C2,Stop) # We start with D1, since it has A3, C2 and Stop in its reachable list those edges can be removed

A2:(D1,D2), (D1,A3,C2,Stop) # We look at D2 next, nothing else changes

A2:(D1,D2), (D1,A3,C2,Stop, D2)

7. Step: C1:(A2,D1,D2,A3,C2,Stop) # We start with A2, since it has all other nodes in its reachable list those edges can be removed

C1:(A2),(A2,D1,D2,A3,C2,Stop)

8. Step: B1:(A2,D1,D2,A3,C2,Stop) # Similar to step 7

B1:(A2),(A2,D1,D2,A3,C2,Stop)

9. Step: B1:(A2,D1,D2,A3,C2,Stop) # Similar to step 7

A1:(A2),(A2,D1,D2,A3,C2,Stop)

10. Step: Start:(A1,B1,C1,A2,D1,D2,A3,C2,Stop)

Start:(A1,B1,C1), (A1,A2,D1,D2,A3,C2,Stop)

Start:(A1,B1,C1), (A1,A2,D1,D2,A3,C2,Stop,B1)

Start:(A1,B1,C1), (A1,A2,D1,D2,A3,C2,Stop,B1,C1)

After this process FAS has the final graph from the example:

Start:(A1,B1,C1)

A1:(A2)

B1:(A2)

C1:(A2)

A2:(D1,D2)

D1:(A3,C2)

D2:(A3,C2)

A3:(Stop)

C2:(Stop)

Stop:()

2.3 Priority Mode

The exhaustive approach for choosing the best path can become problematic for larger proteins with many features annotated where FAS would have to go through millions of paths. Since this would take a long time, FAS makes use of priority mode, a greedy strategy. It is activated when a given threshold of number of features or number of paths is reached. In this mode FAS goes through the feature architecture n times, where n is the number of different features in the architecture. In each run, one feature gets priority. This means that, each time the algorithm reaches a fork in the architecture where features overlap, it will choose the feature with priority if possible. If there are multiple possibilities to choose from, FAS will choose the best scoring partial path in a greedy approach. Likewise, if the feature with priority is not present in the fork, FAS will choose the best scoring partial path from all options. This mode makes sure, that each feature was at least considered once in one of the complete paths checked. As this method is a greedy approach which only looks at a subset of paths, it is not always assured that the chosen path is the best path overall.

2.3.1 Example

The example graph from before would normally resolved extensively by evaluating all paths as the number of possibilities is not very high. However, here it will be used to show how the priority mode operates. First, FAS make a list of all features (A-D). Starting with feature A as priority, FAS will then go through the graph. As the first junction has the priority feature A it will pick this path. Next it will pick feature A again as there is no alternative. The second overlap region does not have the priority feature so FAS will make a greedy decision based on the FAS scores of the two possible partial paths. At the next junction will again pick feature A. This means that either AAD1 A or AAD2 A will be evaluated based on which had the better partial score at the second junction. The runs with features B and C as priority will go similar. For feature D, it will take the greedy approach at junction 1. When it reaches the second junction it has the choice between the two instances of the priority feature D. This will also be resolved by greedily picking the best of the two possible options for the priority features. Finally, FAS will make a pure greedy run without any priority feature.

All, in all the priority mode will evaluate only 5 full paths, one for each feature type, instead of the 12 from the exhaustive run.

2.3.2 Priority Check

There are 3 checks in place that decide wether priority mode will be applied: The first two are based on the number of features in the architecture and the cardinality of the architecture graph. The second check is a time-limit which applies when the first two thresholds are deactivated. The full decision tree

of FAS can be found in the picture below: