Reduce, Reuse, Recalculate: Improvement of the Carbon Condensation Capabilities of MeOXC using K*

Analysis Supplementing the Mission of a Circular Carbon Economy

Sara Liszeski

Abstract

Human society releases massive amounts of CO_2 into the atmosphere every second. In order to combat this damage, researchers are looking to capture and resuse carbon in purposeful ways. MeOXC is an enzyme in *Methy-lorbrum extroquens* that has a role in catalyzing C1-C1 condensation of carbon molecules. Nattermann. et. al. mutated this enzyme to improve its efficacy, resulting in the enzyme MeOXC4. This project seeks to validate and expand upon these results. The PDB for 2JI8 was used as a model and was processed in a limited manner by the OSPREY software package running the K* algorithm [4], [5]. With a 72 hour time limit, 217 viable sequences were found, with a top K* score of 35.994817. Among the top 20 sequences found, the average similarity to MeOXC4 in its 8 mutable positions was 12%.Positions 556 and 483 had clear improvements with valine and tyrosine substitutions, respectively.

1 Introduction

At any moment in time, there are a few topics that gather a large share of the public's attention for science. In the 1960s, this was the moon landing and in 1980, it was genetic modification of food; in the 21st century, this topic has become climate change and environmental concerns. One major concern had by climate scientists is the level of carbon in the form of CO_2 being constantly released by industrial plants and other forms of human energy consumption. In order to combat high levels of atmospheric carbon, society can either reduce the amount of carbon being produced or they can remove the existing carbon.

The latter method is the focus of several environmental scientists, including Nattermann et. al. This group intends to use the C_1 compounds in industry and other areas. However, this cannot occur with just C1; these carbons must be condensed into multi-carbon compounds to produce useful materials. This process is called carbon fixation and is incredibly complex. Nattermann et. al. focus on linear pathways that catalyze direct C1-C1 condensation in order to avoid altering the host organism's metabolism in a fatal or harmful manner.

In nature, several microorganisms can successfully and efficiently condense carbon in this manner. These organisms are methylotrophic and formatrophic [**?**]. However, these organisms cannot perform at the efficiency needed to implement carbon condensation in the industrial setting. Therefore, chemical engineers, biological scientists, and other science professionals are hard at work in places like the Max-Planck Institute for Terrestrial Microbiology to alter these organisms to be successful at this task.

Natterman et. al. specifically examined MeOXC, a variant of oxalyl-CoA decarboxylase from Methylorbrum extroquens. It binds with thiamine diphosphate (ThDP or TPP) and adenosine Di-phosphate (ADP) to catalyze the decarboxylation of oxalyl-CoA. Nattermann et. al. focused on altering these enzymes to mimic the activity of the efficient (yet currently un-analyzable due to lack of structural information) HACL enzymes, which catalyze the transformation of formyl-CoA with formaldehyde to produce glycolyl-CoA. This activity is referred to as Glycolyl-Coa Synthase (GCS) Nattermann et. al. sought to enable *Methylorubrum extorquens* to produce glycolyl-CoA at physiologically accurate leevels of formaldehyde concentrations. Using structure-guided enzyme engineering, the team created a series of 3600 clones. These clones contained mutations at I48, E134, Y135, A415, Y497, S568, E567, and I571– sites which were chosen for their close position to the α-carbanion or enamine intermediate. These sites were also close to the binding position of TPP, making them important in catalysis. They identified the final variant, MeOXC4, as one carrying four substitions: E135G, A415C, Y497F, and S568G.

This project was completed in order to validate these results computationally, rather than through wet lab analysis. This necessitates a computational approach for choosing replacement amino acid rotamers in the eight target positions. Then, once conformations are created, their energy must be analyzed in order to determine which yields the lowest-energy structure, signifying optimality of binding. Therefore, the overall structure of the project was the following

1. Obtain structures of the MeOXC/ formyl-CoA complex
2. Compute the K* scores for mutations of the eight residues
3. Compare the results and find the optimum conformation

The end goal was to obtain several structures and compare them to the 4 mutations present in MeOXC4 to verify improvement or optimality of the active site.

2 Methods

2.1 The K* algorithm

The K* algorithm, developed by Georgiev et. al. and Lilien et. al in [5], [2], is a means of analyzing the strength of protein-ligand binding in a complex. The K* score is made to be an approximation of the true binding constant $K_a = \frac{1}{K_D}$. It utilizes several partition functions:

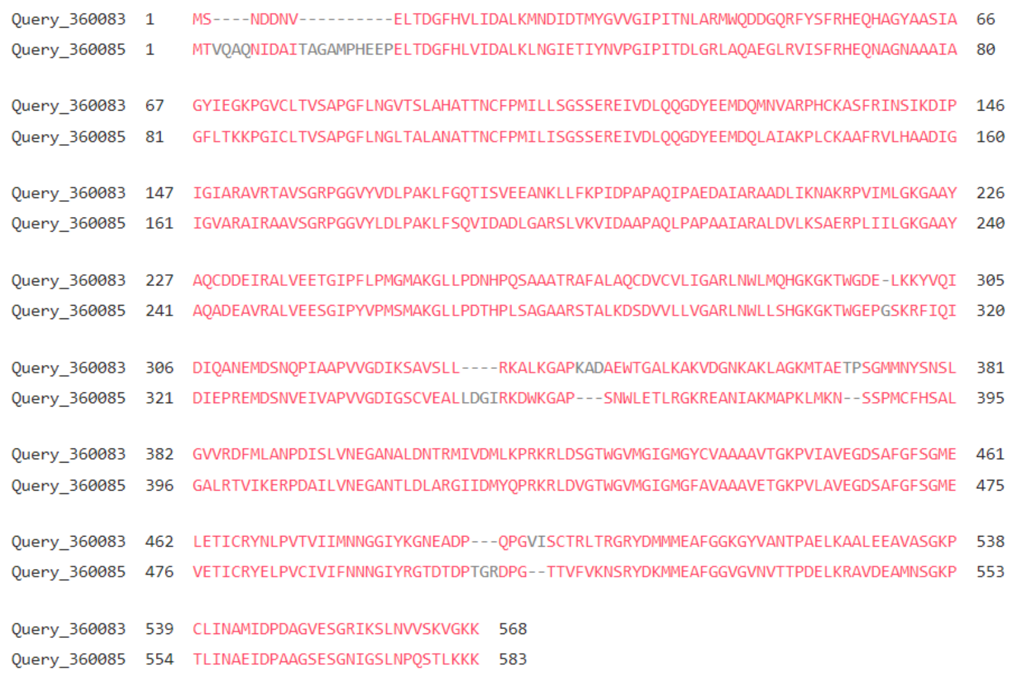
$$q_{pL} = \sum_{b \in B} exp(-E_b/RT) \tag{1}$$

$$q_p = \sum_{f \in F} exp(-E_f/RT) \tag{2}$$

$$q_l = \sum_{l \in L} exp(-E_l/RT) \tag{3}$$

$$K* = \frac{q_{pL}}{q_p q_L} \tag{4}$$

where B, F, and L signify the energies of ensembles based on rotamers for the protein-ligand complex [1]. P is the free protein, and L is the free ligand. The partition functions aid in the calculation of the binding constant, as explained in [1]. There are several different strategies for shrinking the search space of the K* algorithm; here, we employ a manual modifed residue type filter. The K* algorithm also uses a minimized dead-end elimination strategy, ensuring that all conformations not in the range of $(1 - \epsilon)$ will be pruned. Thus, in an ideal situation, after pruning to the resistor protocol's default 0.683 epsilon value, the K* algorithm could be run again with smaller and smaller ϵ values until the GMEC is found. Georgiev et. al. found that the optimum method of running K* involves using the minDEE/A* algorithm as a


```
Query_360083  1    MS----NDDNV----------ELTDGFHVLIDALKMNDIDTMYGVVGIPITNLARMWQDDGQRFYSFRHEQHAGYAASIA  66
Query_360085  1    MTVQAQNIDAITAGAMPHEEPELTDGFHLVIDALKLNGIETIYNVPGIPITDLGRLAQAEGLRVISFRHEQNAGNAAAIA  80

Query_360083  67   GYIEGKPGVCLTVSAPGFLNGVTSLAHATTNCFPMILLSGSSEREIVDLQQGDYEEMDQMNVARPHCKASFRINSIKDIP  146
Query_360085  81   GFLTKKPGICLTVSAPGFLNGLTALANATTNCFPMILISGSSEREIVDLQQGDYEEMDQLAIAKPLCKAAFRVLHAADIG  160

Query_360083  147  IGIARAVRTAVSGRPGGVYVDLPAKLFGQTISVEEANKLLFKPIDPAPAQIPAEDAIARAADLIKNAKRPVIMLGKGAAY  226
Query_360085  161  IGVARAIRAAVSGRPGGVYLDLPAKLFSQVIDADLGARSLVKVIDAAPAQLPAPAAIARALDVLKSAERPLIILGKGAAY  240

Query_360083  227  AQCDDEIRALVEETGIPFLPMGMAKGLLPDNHPQSAAATRAFALAQCDVCVLIGARLNWLMQHGKGKTWGDE-LKKYVQI  305
Query_360085  241  AQADEAVRALVEESGIPYVPMSMAKGLLPDTHPLSAGAARSTALKDSDVVLLVGARLNWLLSHGKGKTWGEPGSKRFIQI  320

Query_360083  306  DIQANEMDSNQPIAAPVVGDIKSAVSLL----RKALKGAPKADAEWTGALKAKVDGNKAKLAGKMTAETPSGMMNYSNSL  381
Query_360085  321  DIEPREMDSNVEIVAPVVGDIGSCVEALLDGIRKDWKGAP---SNWLETLRGKREANIAKMAPKLMKN--SSPMCFHSAL  395

Query_360083  382  GVVRDFMLANPDISLVNEGANALDNTRMIVDMLKPRKRLDSGTWGVMGIGMGYCVAAAAVTGKPVIAVEGDSAFGFSGME  461
Query_360085  396  GALRTVIKERPDAILVNEGANTLDLARGIIDMYQPRKRLDVGTWGVMGIGMGFAVAAAVETGKPVLAVEGDSAFGFSGME  475

Query_360083  462  LETICRYNLPVTVIIMNNGGIYKGNEADP---QPGVISCTRLTRGRYDMMMEAFGGKGYVANTPAELKAALEEAVASGKP  538
Query_360085  476  VETICRYELPVCIVIFNNNGIYRGTDTDPTGRDPG--TTVFVKNSRYDKMMEAFGGVGVNVTTPDELKRAVDEAMNSGKP  553

Query_360083  539  CLINAMIDPDAGVESGRIKSLNVVSKVGKK  568
Query_360085  554  TLINAEIDPAAGSESGNIGSLNPQSTLKKK  583
```

Figure 1: BLAST alignment of the sequences of 2JI8 (Oxalobacter formigenes) and 7AYG (Methylorubrum extorquens AM1)

pre-processing step. To complete the K* rankings, the software package OSPREY (Open Source Protein Redesign for You) was used [4], which uses the minDEE/A*/K* algorithm. The RESISTOR protocol was used as a general guideline for the procedure [6].

2.2 Issues and Approximation-Based Solutions

Several issues arose with using OSPREY, mostly due to inexperience and skill issues, combined with lack of available information. To begin, there was no PDB including Formyl-CoA bound to MeOXC in Methylorubrum extorquens AM1. Because creating such a file would be a significant endeavor involving docking algorithms, the simplist approximation was gotten using an existing PDB file with an OXC bound to Formyl-CoA from another organism, Oxalobacter formigenes. BLAST was used to compare the sequences of PDB 7AYG, a non-bound MeOXC molecule, with PDB 2JI8, the bound OXC mentioned above. The two FASTA sequences had an identity match of 356/569 (63%) and a positive match of 429/569 (75%), with a 15/569 (2%) gap. The two sequences were similar, as shown in Figure 1, and identical as far as amino acid identity in the spots that were identified as targets by [**?**], so 2JI8 was chosen as an approximation protein. Because of slight differences in the PDB files of the two molecules 7AYG and 2JI8, the positions of the amino acids mutated in the algorithm differ. For the purposes of analysis, it was simpler to use the altered numbers in the results and discussion section. The conversion may be found in Figure 2.

The next steps of RESISTOR involved separating the protein from the ligand. However, several issues arose here; RESISTOR and canonical K* is design for two-body systems, whereas the MeOXC protein involves several separate bodies, including the protein itself, ADP, TPP, and Mg a magneisum ion. Be-

7AYG Position	2JI8 Position	MeOXC4 Mutation
48	34	
134	120	
135	121	GLY
415	401	CYS
497	483	PHE
567	552	
568	553	GLY
571	556	

Figure 2: Translation between 7AYG and 2JI8 residue numbering

Position	Starting Amino Acid	Provided Mutations
34	ILE	LEU, VAL
120	TYR	ALA, PHE
121	GLU	PHE, GLY
401	ALA	CYS, SER
483	TYR	PHE, GLY
552	GLU	GLN
553	SER	ALA, GLY
556	TYR	VAL

Figure 3: Permitted Mutations per Position for K*

cause the additional molecules caused issues in this naive instance of OSPREY, they were removed with acknowledgement that their absence will cause lack of precision. To run the K* algorithm in its true form, all mutable residue positions should be allowed to mutate to a wide range of amino acids. However, this was infeasible due to the time constraint present after the task of creating the input files. Therefore, since this project is focused on reverse engineering solutions, the given substitutions E135G, A415C, Y497F, and S568G were provided as allowable amino acid substitutions, as well as similar amino acids. For the positions that did not mutate for MeOXC4, amino acids similar to the ones present in MeOXC were provided.

3 Results

3.1 Preliminary Results:

Due to the runtime of the solution, a full enumeration of all sequences permitted by the above mutations was not allowed; instead, preliminary results will consist of analysis of the first 1800 of 2916 sequences. The algorithm's run was cut off after 82 hours. These sequences were processed using the pandas code in the supplemental informatin section. All sequences that produced an unstable configuration were dis-

Position	Starting Amino Acid	Identity Count
34	ILE	7 LEU, 6 VAL, 7 ILE
120	TYR	9 PHE, 9 TYR, 2 ALA
121	GLU	12 PHE, 6 GLU, 2 GLY
401	ALA	20 ALA
483	TYR	20 TYR
552	GLU	20 GLN
553	SER	12 ALA, 8 SER
556	TYR	20 VAL

Figure 4: Mutation Identities per Position of the Top 20 Sequences by K* score

carded. Around 15 percent (217/1480) of the sequences yielded stable results. Of these sequences, the average identity match to Nattermann's results, quantified as the percentage of the eight mutable acids that matched the eight in the MeOXC4 enzyme, was 0.1822. In the allowed runtime, the algorithm prodec 2 sequences that matched 7 of the 8 desired identities, 37 sequences with 6 of the 8 identities, and 114 that matched 5 of the 8 identities. Of these 153 sequences, only two were stable enough to produce a result; both had 5 of 8 matches and were in the bottom 25% of K* scores. In the top 20 sequences examined, weighted by K* score, the average match was 12% – less than one match per sequence.
The top K* score was 35.994817, with the top 20 within a range of 35.968883-35.994817, and an overall range of 35.483094-35.994817. The top three sequences were

1. L34 F120 F121 A401 Y483 Q552 A553 V556
2. V34 F120 F121 A401 Y483 Q552 A553 V556
3. I34 F120 F121 A401 Y483 Q552 A553 V556

We can also use the results of the estimated partition functions yielded by OSPREY. OSPREY yields partition scores for the free protein, free ligand, and the protein-ligand complex. According to Donald, K* is an approximation of the binding constant

$$K_A = \frac{1}{K_D} \tag{5}$$

Therefore, by taking the reciprocal of the K* values, we have a K_D that we can compare to the data in the paper. K_D is the dissociation constant of an enzyme. The larger the K_D value, the weaker the binding and the worse the catalytic efficiency of the enzyme. This value is directly calculable as the reciprocal of the K* score. The values all lay within the range of 0.027782-0.028182. The top 20 scores all had K_D values of 0.027802-0.027782. Position 401 had 0 stable sequences with the CYS sustitution.

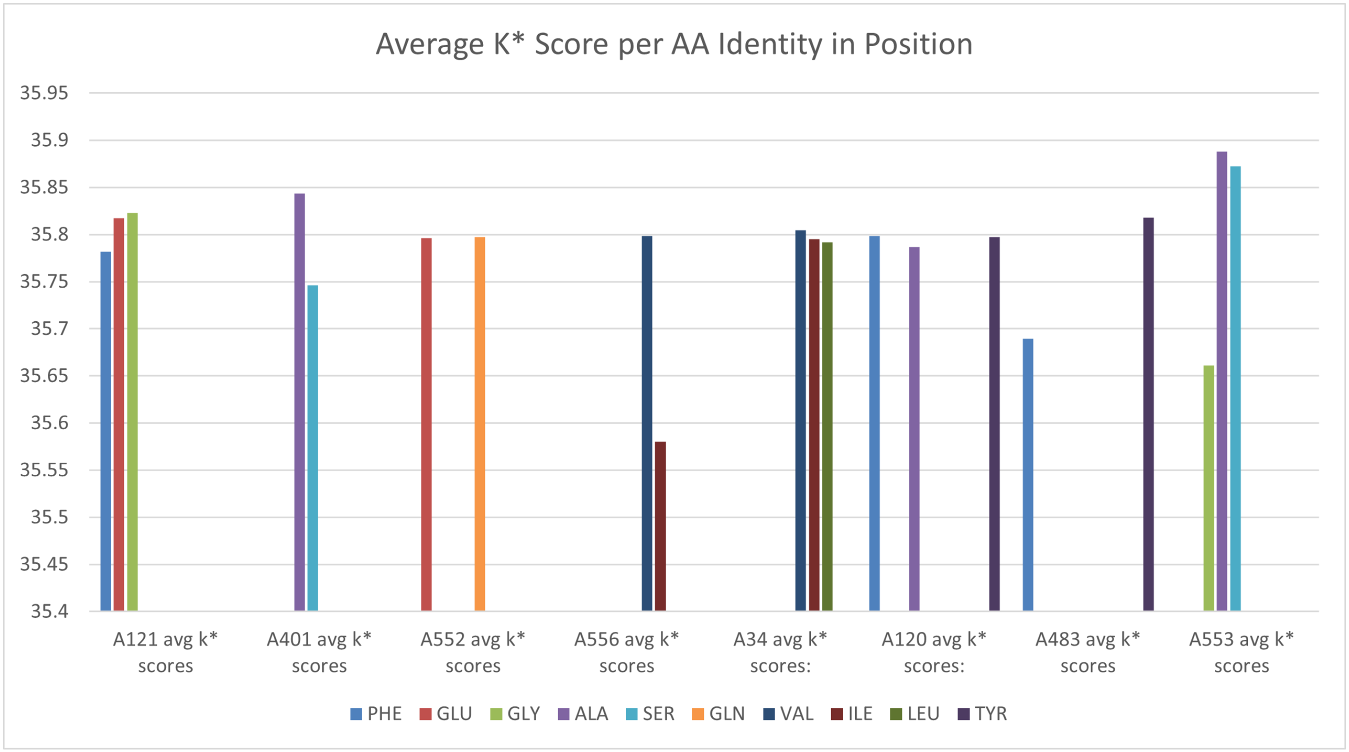


Figure 5: Average K* score for each viable mutation in each mutable position. Each average is the sum of the K* scores of all sequences with the given amino acid in position i, regardless of the mutations in all other positions.

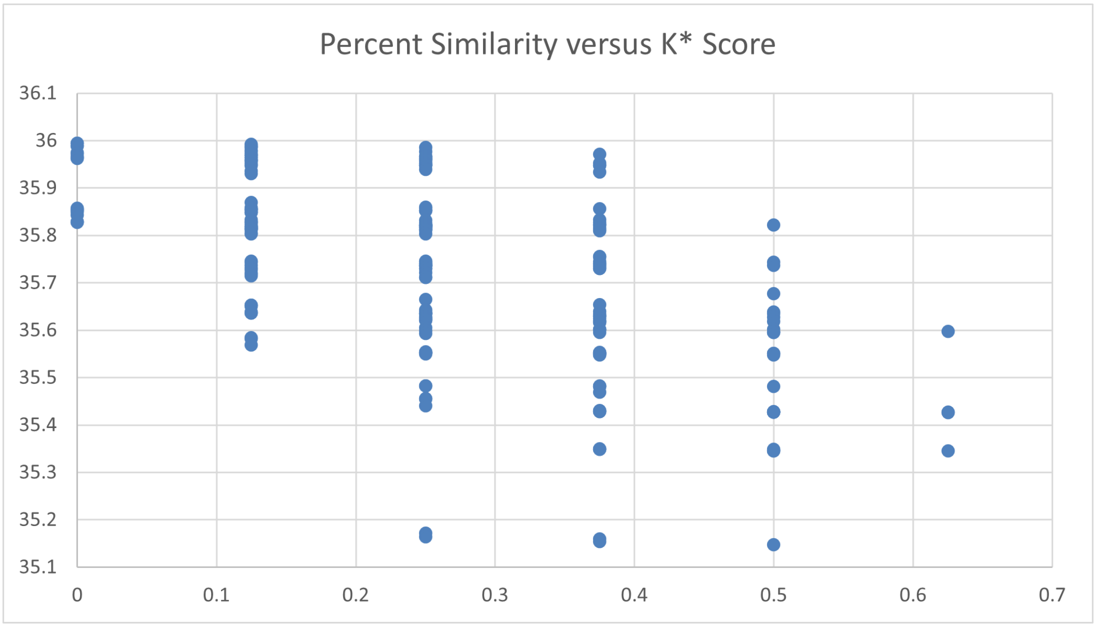


Figure 6: K* Score and Percent Similarity to MeOXC4

4 Discussion

The K* scores measured by the algorithm represents the stability that a given mutation will grant to a protein or protein complex. Since we are focused on binding in this experiment, a higher K* score means a better binding ,since the bound state will be more stable. We do not use a negative design here, so we only focus on mutating our sequence toward a goal, rather than away from an undesirable function. The results yielded by this run of the K* algorithm are interesting but not conclusive. To begin, they were severely limited by the assumptions described in the methods sections. The energies calculated lack consideration of the TPP and ADP molecules and the magnesium atom, both present in the active site under consideration and essential to the catalyzation power of the enzyme. Their absence causes spatial consideration issues, along with the discarding of several vital interactions. Additionally, due to time, each mutable residue was only allowed to mutate to up to three other amino acid types, depending on specification. This assumption severely limits the applications of K* for this experiment, as all possible mutations are not being enumerated.

However, with these allowances aside, the sequences produced did not closely match the results found in the wet lab mutations of Nattermann. In fact, the top 20 K* scores belong to sequences with on average less than one residue matching the identity of MeOXC4. The maximum matching was 0.625%, or five residues, with a K* score of 35.597 and a ranking of 207 overall. This is obviously undesirable and proves gaps in either this project's configuration file OR the efficacy of MeOXC4. It is possible that thousands of sequences exist that are better than those produced by the lab, meaning that MeOXC4's sequence did not make the epsilon cutoff of this run of K*. Almost 90 percent of all generated sequences were called unstable; this number is quite high and should be lowered via improved planning for the algorithm's search space. It is also possible that the configuration is limited or incorrect, making inaccurate energy functions based upon a severely simplified model.

Additionally, since there are gaps in the PDB file, it is possible that these sequences are in fact energetically more favorable than MeOXC4 and thus cause it to be locally dominated and therefore pruned by K*. However, these sequences might not be biologically plausible due to clashes that were not accounted for in this instance of the energy function. Therefore, without a more accurate model, it is difficult to speak on the true meaning of these results. If this version of the OXC enzyme appeared in real life, the enumerated sequences would be ranked highly among possible active sites; however, this is unlikely.

This lack of matching may also be due to the fact that the wet lab's search space was severely limited by mutational probability, which this K* run did not take into consideration, and thus the experimenters did not screen enough clones to reach a point where the more sequences with better K* were found. Due to time constraints, no additional runs could be done with greater flexibility of positions near the binding site, meaning that the modeled molecule had more clashes than the real molecule might have had.

The data is semi-conclusive on the best offered amino acid for several positions, namely positions 483 and 556 as evidence by these mutations' average K* scores in Figure 5. In position 483, the wild type residue TYR was the sole 483 amino acid present in the top 78 sequences. In the enzyme MeOXC4, this position is occupied by PHE, which did not confer extra stability in the sequences examined. No mutation occurred for 2JI8 position 556 in MeOXC4, but PHE appears to confer some stability. This could be due to the fact that PHE is nonpolar, compared to tyrosine's polarity. Both are uncharged and aromatic, so if polarity is set aside, the difference might be due to phenylalanines smaller size in its lack of an OH group. The best 3 sequences all had F120 F121 A401 Y483 Q552 A553 V556 as their last 7 residues. As these results do not include all possible enumerations, these mutations cannot be said to be the optimum sequence, but they are an option that can be improved upon in future iterations and tested in a wet lab.

5 Conclusion

The sequences with the top K* values offer new options for researchers wishing to investigate better amino acid sequences for the MeOXC enzyme. Though they are not polished, they provide insight into the trends of well-performing sequences for this active site. With more work, the results of the K* algorithm could offer valuable insight and options for future work on the catalytic efficiency of MeOXC.

5.1 Future Work

The options for future work on this project are bountiful. First, more possible rotamers may be specified for each position considered in Natterman, with considerations of probabilities of mutations for replicability. Second, a better input model can be generated, with an additional third body, a cofactor, containing ADP, TPP, and the magnesium ion. This would create different spatial and charge parameters for the algorithm, making the results more accurate. Third, more positions around the active side can be made flexible, potentially allowing conformations said to cause clashes or have excess energy values in this run of K*. Additionally, K* can be run with a higher ϵ value, enumerating fewer sequences.

5.2 Lessons Learned

During this project, I learned several lessons. To begin, I learned that I need to limit my ambitions to a more reasonable level when they include learning to use a complex software system. I believed that the formatting would be an insignificant amount of my time whereas analysis and multiple runs would make up the bulk of the project. In this instance, I was incorrect. I learned how difficult it is to get up to speed on file formatting specifications for software packages. Specifically, I learned that fitting a pre-specified protocol, such a resistor, to a new problem, such as a 3-body problem, can be problematic and involve creative ways of re-examining the problem. I learned how sensitive yaml files can be, but how important it is to create them correctly so that specifications are as accurate as possible. I additionally learned that while methods may be called fast in theory, it is often difficult for a student's schedule. Specially, I naively did not expect the K* algorithm to take as long as it did. I now have a better understanding of the timescale of the research performed on computational methods. I also used the pandas data processing package for the first time and gained a better understanding of Excel's capabilities to help me comprehend trends in data.

References

[1] Donald, Bruce R. Algorithms in Structural Molecular Biology. The MIT Press, 2011.

[2] Georgiev, Ivelin et al. "The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles." Journal of computational chemistry vol. 29,10 (2008): 1527-42. doi:10.1002/jcc.20909

[3] Guerin N;Feichtner A;Stefan E;Kaserer T;Donald BR; "Resistor: An Algorithm for Predicting Resistance Mutations via Pareto Optimization over Multistate Protein Design and Mutational Signatures." Cell Systems, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/36265469/.

[4] Hallen, Mark A et al. "OSPREY 3.0: Open-source protein redesign for you, with powerful new features." Journal of computational chemistry vol. 39,30 (2018): 2494-2507. doi:10.1002/jcc.25522

[5] Lilien, Ryan H et al. "A novel ensemble-based scoring and search algorithm for protein redesign and its application to modify the substrate specificity of the gramicidin synthetase a phenylalanine adenylation enzyme." Journal of computational biology : a journal of computational molecular cell biology vol. 12,6 (2005): 740-61. doi:10.1089/cmb.2005.12.740

[6] Guerin N;Feichtner A;Stefan E;Kaserer T;Donald BR; "Resistor: An Algorithm for Predicting Resistance Mutations via Pareto Optimization over Multistate Protein Design and Mutational Signatures." Cell Systems, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/36265469/.

6 Supplemental Information

The following code was used to process the K* data:

```python
import pandas as pd
import openpyxl
df = pd.DataFrame()
with open('resistor.txt') as f:
    lines = [line for line in f.readlines()]

split_lines = []
for line in range(28, len(lines)): #lines 1-28 are pre-processing data
    x = lines[line].split()
    split_lines.append(x)

#arrays for data
seq_nums = []
A34 = []
A120 =[]
A121=[]
A401=[]
A483=[]
A552 = []
A553 = []
A556= []
kstar = []
```

```python
numconfs = []
protz = []
ligz = []

for line in split_lines:
    if line[32] != "NaN":
        seq_nums.append(line[1])
        A34.append(line[2].split("=")[1])
        A120.append(line[3].split("=")[1])
        A121.append(line[4].split("=")[1])
        A401.append(line[5].split("=")[1])
        A483.append(line[6].split("=")[1])
        A552.append(line[7].split("=")[1])
        A553.append(line[8].split("=")[1])
        A556.append(line[9].split("=")[1])
        kstar.append(line[11])
        numconfs.append(line[30])
        protz.append(line[16])
        ligz.append([line[26], line[27]])

df["sequences numbers"] = pd.Series(seq_nums)
df["A34"] = pd.Series(A34)
df["A120"] = pd.Series(A120)
df["A121"] = pd.Series(A121)
df["A401"] = pd.Series(A401)
df["A483"] = pd.Series(A483)
df["A552"] = pd.Series(A552)
df["A553"] = pd.Series(A553)
df["A556"] = pd.Series(A556)
df["kstar"] = pd.Series(kstar)
df["numconfs"] = pd.Series(numconfs)
df["protein z scores"] = pd.Series(protz)
df["ligand z scores"] = pd.Series(ligz)

df.to_excel("kstar_results.xlsx")
```

The .yaml file, frcmod file, and other information used to parameterize and run the K* program may be found at https://github.com/saraliszeski/CBB-590