Development of a Scalable and Efficient Computational Framework for the Structure Prediction of G Protein-Coupled Receptors

Christina Lin
SURF 2014

ABSTRACT

G protein-coupled receptors (GPCRs) are a family of membrane proteins that regulate cell function by controlling response to extracellular signals. They are essential to cellular metabolism, cell growth, immune defense, and many more processes, and as such may be important to inducing drug reception.[1] Understanding these proteins begins with identification and analysis of distinct low-energy conformations via the SuperBiHelix method.

The original SuperBiHelix method runs on a fixed 12 processors. Its performance can be improved when the program is rewritten to run on any number of processors. With a modular reconstruction of the algorithm into small, independent units, the SuperBiHelix method makes more efficient use of computational resources and produces complete results within a shorter timeframe.

TARDIS is the computational framework that enables this more efficient and flexible SuperBiHelix. It handles the division of SuperBiHelix into small units, the distribution of these units to processors, and the bookkeeping that allows a single job to be efficiently completed on multiple processors. TARDIS is composed of self-contained modules and can be applied to raise the performance of virutally any embarrassingly parallel program.

INTRODUCTION/BACKGROUND

Introduction to GPCRs

Cell membranes are embedded with proteins that allow cells to interact with their surroundings safely. One such protein is the G protein-coupled receptor (GPCR). Through interaction with G proteins, they transmit signals from the outside to the inside of the cell. These signals are responsible for many physiological processes, including neurotransmission, cellular metabolism, secretion, cell growth, immune defense, and differentiation.[2]

GPCRs are composed of seven transmembrane alpha-helices whose configurations are altered in the activation process. Whether in response to extracellular signals of due to their inherent conformational flexibility, GPCRs have the tendency to take the form of various conformations. The functional characterization of these conformations is essential to understanding the effect of ligands and mutations on cells.

The SuperBiHelix method

The SuperBiHelix method facilitates a complete sampling of the conformational space at modest computational cost. It targets both inactive and active states, which are overlooked in homology-based computational methods, and covers the complete range that cannot be covered with *de novo* or knowledge-based methods. Not only is the SuperBiHelix method able to efficiently

sample rotation angles, but it also begins to identify other GPCR conformations and is applicable to other helical membrane proteins.

The method is based on a spherical coordinate system with six parameters for each transmembrane helix. *x* and *y* represent the helix's position on the hydrophobic plane. *h* specifies the hydrophobic center reside's intersection with the z=0 plane. Theta(θ) is the angle from the helical axis. Phi(φ) is the angle from the x or y axis. Eta(η) is the rotation about the x axis, allowing of representation of bent helices. Since a GPCR has seven transmembrane helices, each conformation is represented by 42 numbers.

The sampling and analysis process can be divided into three stages, summarized in Figure 1.

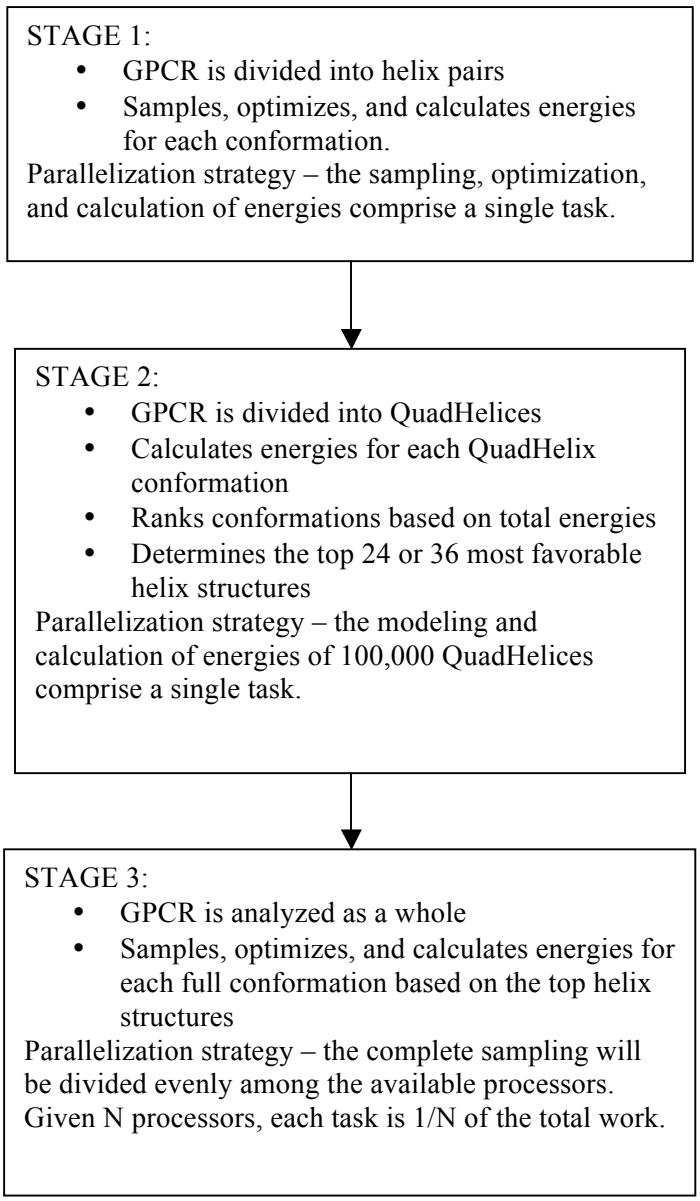


Figure 1. Flowchart of SuperBiHelix method stages.

STAGE 1: A single conformation acts as the starting template. The loop regions are ignored, as it is assumed that optimization is not affected by the loops. The SuperBiHelix method organizes the GPCR bundle into 12 nearest-neighbor helix pairs, as shown in Figure 2. The conformational space of each SuperBiHelix couple is first sampled. This is done with the SCREAM method using a DREIDING force field.

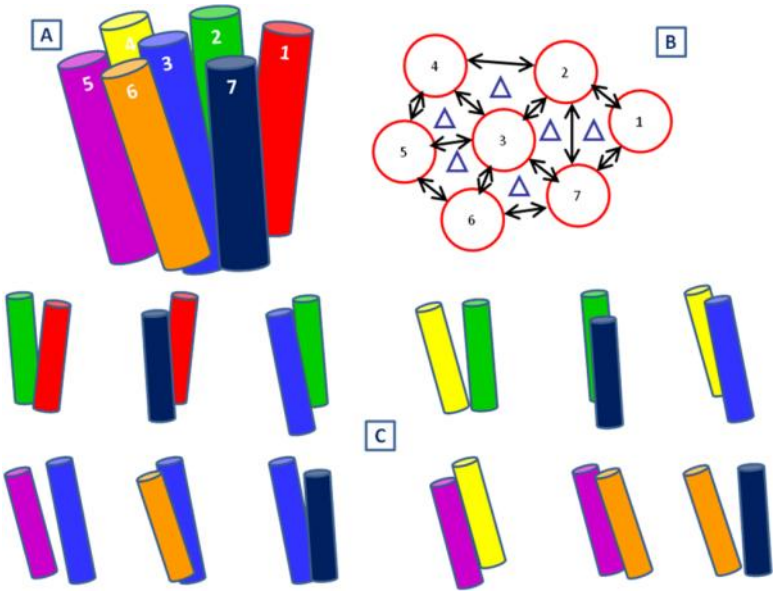


Figure 2. Division of full GPCR into bihelix pairs

Next, the energies can be estimated. For each helix pair, there are three energy components: the interhelical energy and one intrahelical energy value associated with each member of the pair.

STAGE 2: To determine which individual helices are candidates for the most favorable full GPCR conformations, the seven-helix bundle is grouped into three QuadHelices, as seen in Figure 3. Energy components for each QuadHelix are calculated and ranked. By tallying the appearance of particular helix conformations in the lowest-energy QuadHelix structures, the best conformations for each helix are chosen.

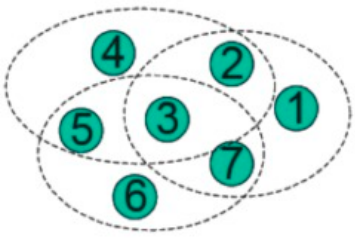


Figure 3. Division of full GPCR into QuadHelices

STAGE 3: With the most favorable conformations for each helix chosen, the final step is to use them to construct full seven-helix bundles. Their energies are calculated, and the lowest-energy structures are ranked and printed.

The original code utilizes 12 processors, assigning each processor to run the sampling for one SuperBiHelix for Stage 1. Only three of the original 12 processors are used in Stage 2, with each processor running the analysis for one QuadHelix. In Stage 3, the sampling of structures is divided evenly among up to 12 processors.

METHODS

Parallelization by task

To parallelize the SuperBiHelix method, the method itself must first be broken into distinct tasks. There are two requirements for the division of tasks. First, a task must be able to be run on a single, arbitrary processor independently, without communication with other parallel tasks. Second, the task should be an ideal size: small enough to require parallelization, but large enough for the overhead of bookkeeping to be negligible. The minimum runtime per task for the computational framework developed in this project is 2-3 minutes.

For Stage 1, the task is the complete sampling, sidechain optimization, and energy estimation of a single bihelix conformation. Thus, each task is characterized by eight unique numbers: eta, theta, phi, and HPM for each helix. For Stage 2, the task is the energy calculations of 100,000 QuadHelix conformations, which takes around five minutes, as determined experimentally. For Stage 3, the tasks are created based on the number of processors, by dividing the work among the available processors.

Task distribution

The distribution framework keeps track of the processors allocated by the PBS queuing system in a list, called *proclist*. In its primary execution loop, it distributes tasks, enumerated in a *tasklist*, to each free processor.

The task distribution is controlled by a simple loop through the *tasklist* and *proclist*, based on the outline below:

```
while len(tasklist) > 0:
        for proc in proclist:
                if proc.is_free() and len(tasklist) > 0:
                        task = tasklist.pop()
                        proc.run_task(task)
```

Development of TARDIS

Three main Python scripts hold together the infrastructure of the task distribution for the new SuperBiHelix; collectively, they are called TARDIS and have been developed into a task distribution library that can used for adapting any embarrassingly parallel job.

Given a specific number of allocated nodes, TARDIS catalogs the processors in *proclist* by creating an ID and working directory for each. Then, the initialization specific to the program, in this case the SuperBiHelix method, is carried out, and the *tasklist* is created. Once both lists have been populated, the program enters the task distribution loop described above.

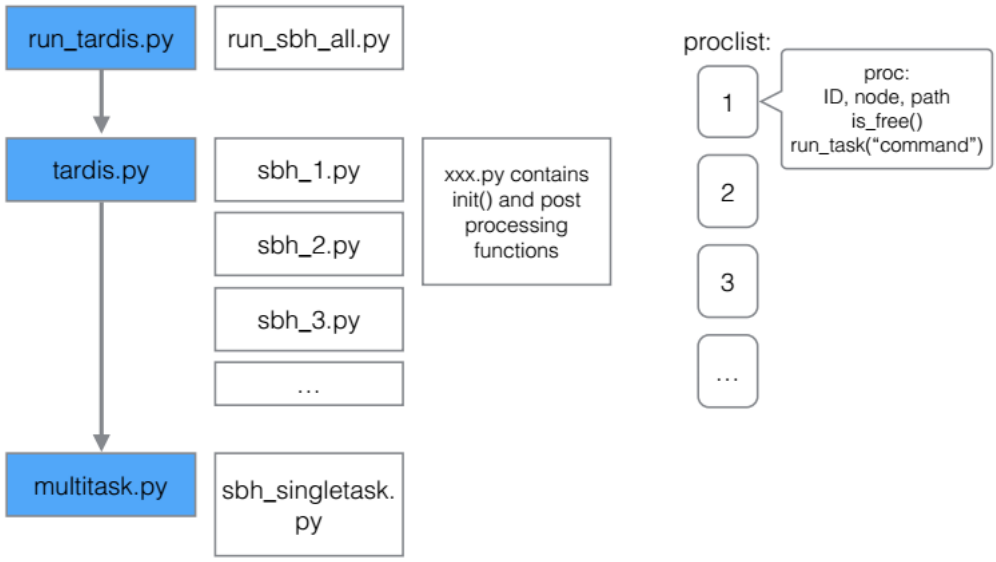


Figure 4. Diagram of TARDIS program architecture.

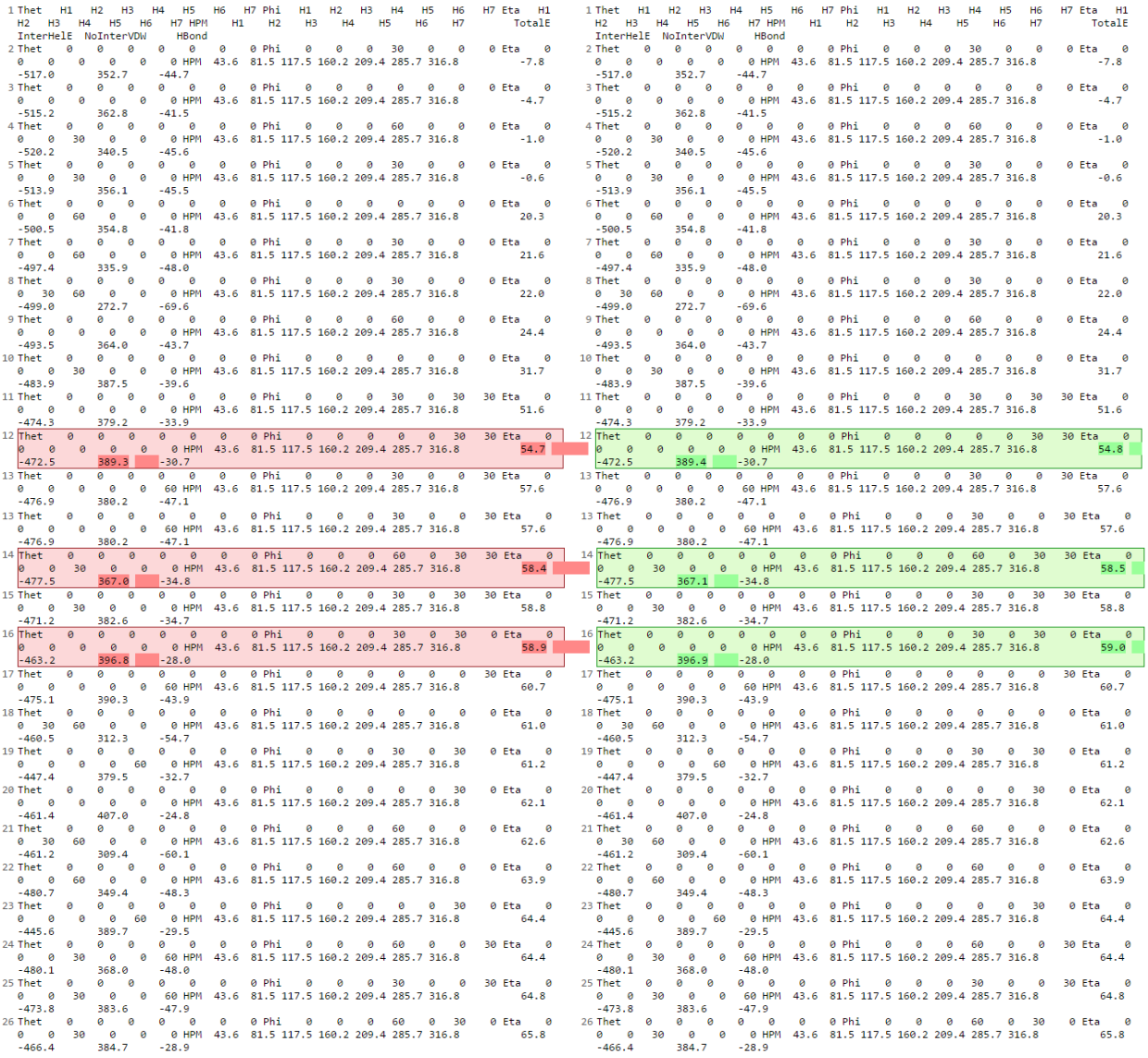
The TARDIS code is completely separate from the SuperBiHelix method and is by itself a library applicable to any embarrassingly parallel program. The SuperBiHelix-specific files in Figure 4 can be replaced by snippets of code from a parallel algorithm; the resulting program is a flexible version of that algorithm that runs on any number of processors.

Thus, the reconstruction of SuperBiHelix is applicable to any similarly parallel program. If an existing parallel job can be divided into modules according to the TARDIS model, that program can be easily modified, using the TARDIS library, to run on any number of processors.

RESULTS AND DISCUSSION

To verify that the new program produces the expected results, identical to the original program, the intermediate and output files were compared. Although there is some discrepancy in some of the numerical values, which leads to a shift by one position in some rankings, the top structures chosen by each program is the same, as shown in Table 1.

Table 1. Top 25 structures selected for beta2 files. Lines 12, 14, and 16 exhibit small discrepancies that do not affect the ranking of the structures.



Timing Results

The newly adapted SuperBiHelix method has been tested on two jobs of different size, each running on 12, 24, and 48 processors.

Table 2. Timing results of TARDIS-based and original codes.

# of tasks	# of procs	TARDIS wall time (hr)	TARDIS wall time (s)	Approx. original wall time (hr)	Original wall time (s)	Speed-up
1728	12	2:17:00	8220	2:53:00	10380	1.26x
1728	24	1:08:08	4088	n/a	n/a	2.54x
1728	48	0:36:04	2164	n/a	n/a	4.80x
8748	12	12:59:02	46742	13:41:00	49260	1.05x

8748	24	6:08:39	22119	n/a	n/a	2.23x
8748	48	3:18:33	11913	n/a	n/a	4.13x

On 12 processors, the new parallelized version of SuperBiHelix runs faster than the original version. This can be attributed to either the flexible distribution of tasks in Stage 1, which ensures equal usage of the 12 processors, or the parallelization in place for Stages 2 and 3.

As expected, when SuperBiHelix is run on twice as many processors, it is complete in half the time. The speed-up is close to linear.

CONCLUSION

The improved, parallel SuperBiHelix method is flexible and efficient. It takes the existing methodology for completely sampling a GPCR and adapts the program to make better use of resources. The runtime can be reduced to just a fraction of the previous time

The parallelized SuperBiHelix method can be run on however many processors are available, making full use of computational resources. When 10 processors are free, rather than wait until two more can be allocated, the process can begin immediately. If 100 processors are free, the job can be split among them and completed in a fraction of the time it would take 12 processors. Because processors are becoming cheaper and increasingly available, a scalable program is serviceable for years to come compared.

Furthermore, the TARDIS adaption can be extended to other parallel programs, including CombiHelix, the analysis step run after SuperBiHelix. Large computational methods with repeated, independent units are prevalent in areas of chemistry, biology, and other research. The TARDIS model transforms such embarrassingly parallel methods into fully flexible and scalable programs that are able to adapt to the computational resources available.

ACKNOWLEDGMENTS

Thank you to Dr. Abrol for guidance and mentorship throughout the entire project. Additionally, I would like to thank Sidney R. and Nancy M. Petersen, as well as the Caltech SFP Office, for their invaluable support.

REFERENCES

[1] Bray JK, Abrol R, Goddard WA, 3rd, et al. SuperBiHelix method for predicting the pleiotropic ensemble of G-protein-coupled receptor conformations. Proc Natl Acad Sci U S A (2014) 111(1):E72-8.

[2] Abrol R, Bray JK, Goddard WA 3rd. BiHelix: Towards de novo Structure Prediction of an Ensemble of G-Protein Coupled Receptor Conformations (2012). Prot Struc Func Bioinf, 80(2):505-18.