CS 267 HW 0 - Parallel Applications

Michael Jun

February 7, 2013

1 Personal Bio

I am currently a third year undergraduate student at University of California, Berkeley, pursuing a dual degree in electrical engineering and computer science and engineering mathematics and statistics. I have been working on research in knot theory under Professor Mariel Vazquez[1] and Professor Koya Shimokawa[2]. My focus has been on lattice knot, which is defined to be a self-avoiding curve in the lattice. I am working on exhaustive enumerations of all wanted lattice knots that is contained in a box with given dimensions. This project becomes very computationally expensive when the box grows, and I hope to learn to use parallel computing to efficiently enumerate all knots. I am very interested in computational mathematics such as my project. I hope to not only learn how to better parallelize problems but learn of other interesting problems to tackle.

2 Parallel Applications

2.1 Introduction

I picked the parallel application problems from my research area to discuss in this report. Before I state the problem that we try to solve, I must provide some background information such as definitions and terminologies.

Consider the simple cubic lattice Z^3 . We define a *step* to be a line segment joined by two lattice points of length 1. A lattice point may also be referred to as *vertex* and a line segment as *edge*. Note that a step is either parallel to the x, y, or the z-axis, calling each *x-step*, *y-step*, or *z-step* respectively. Now we define a *lattice knot* to be a polygon constructed by these steps as shown in Figure 1, and the number of steps required to construct the polygon is the step number. Also, we

[1]Department of Mathematics, San Francisco State University

[2]Department of Mathematics, Saitama University

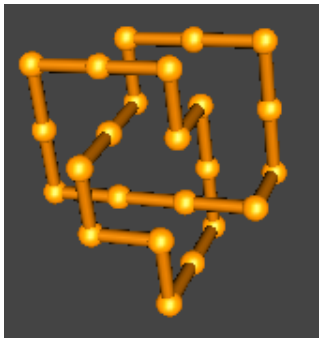

Figure 1: Example of a lattice knot

consider the minimum number of crossings a knot can have after being untangled to identify its *knot type*. We are mainly interested in minimizing the step number required to construct target knot types. Finally, we define a slab of width i, or an *i-slab*, to be the space

$$S_i := \{(x,y,z) \in Z^3 \mid 0 \leq z \leq i, i \in N\}.$$

Now that the definitions are given, we state the problem and motivation: We often find DNA and protein foldings occurring within organisms, and they happen in very confined spaces [2]. Motivated by this, we simulate them by looking at lattice knots confined to the 1-slab. More specifically, we look at a certain property of these knots. Given a target knot type, what is the minimum step number (MSN) required to construct this knot in the 1-slab?

We tackle this problem with two different methods: one is providing an upper bound by the Monte Carlo method, and another is an exhaustive enumeration within the 1-slab and determining the MSN.

2.2 Monte Carlo Method

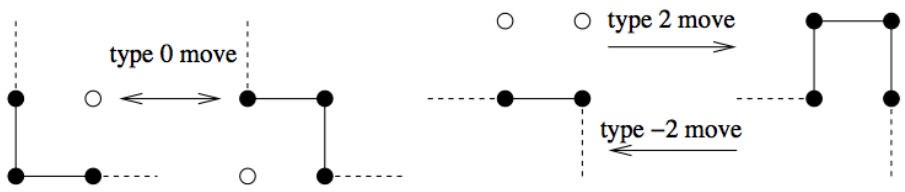

Figure 2: BFACF move type

Here we introduce the dynamical Monte Carlo method - the BFACF algorithm which [1] and [3] use to provide an upper bound for the MSN of knots up to 10 crossings. Figure 2 shows the different BFACF moves, type 0, 2, or -2 moves. As shown in the figure, type 0 does not change the step number whereas type 2 increases the number of steps by 2 and -2 move decreases the number of steps by 2. The interesting property of the BFACF algorithm is that when applied to any edges of the polygon, it will not affect its knot type. The algorithm then uses this to our advantage. We do the following algorithm:

We first generate a polygon with the target knot type. Each initial polygon is then placed in a wedge shaped region with enough room to grow by the BFACF Algorithm as shown in Figure 3. As the polygon changes with many iterations of the BFACF algorithm (applied to an edge at random) to fit the narrower slabs, we inductively reduce the size of the region. The right most vertical line in the figure represents a plane that bounds the right side. It moves closer to the left as the knot fits the smaller slabs, preventing the polygon to leave the smaller slab once it enters it. We repeat this process until the knot fits in the 1-slab.

Finally, once it is in the 1-slab we do more iterations of the BFACF algorithm, prioritizing the 0 and -2 move to try and reduce its size. We then effectively have an approximated upper bound on the MSN of the target knot type.

Figure 3: Polygon in a wedge

Our team wrote a parallel application to undergo millions of iterations of the BFACF for knots up to 10 crossings. In order to achieve a reasonable runtime, parallelizing the code was a necessity. This approach, by nature, is very easy to parallelize. Since we are dealing with randomly applying the BFACF moves on the polygon, each polygon is completely independent of another. Thus to reach millions of iterations, we just use distributed memory, each loaded with one polygon, and undergoes our algorithm independently. This application can also scale very well with the number of nodes. Also since this is an algorithm using randomization, each processor should do equal amount of work. This application worked great except it did not provide everything we wanted to know about the problem; it only gives us numerical results, not theoretical.

2.3 Minimum Step Number Algorithm

In this section, we will discuss another approach to the problem; we introduce the Minimum Step Number (MSN) algorithm that undergoes an exhaustive enumeration to put a theoretical number of the MSN of target knot types.

Let n be the total step number of a polygon. Then for a target knot type, we have an upper bound on m by the Monte Carlo simulation. The theoretical proof is simply enumerating all polygons of step number less than m, starting at $n = m-2$ (step number must be even), and see if we find the target knot type at a smaller step number. To do the enumeration, we perform the MSN Algorithm:

- Given target n, in [1] and [3], we see that the polygons in the 1-slab with n steps fit in the $\frac{n+4}{8} \times \frac{n+4}{8} \times 1$ box.
- **Step 0**: We look at the projection, so consider the $\frac{n+4}{8} \times 1$ rectangle. We enumerate all connected graphs that fit in this rectangle. This includes enumerating all unweighted graphs such as paths, trees, and graphs with cycle.
- **Step 1**: For each rectangle R identified in Step 0, enumerate all weighted planar graphs in R with total multiplicities at most $\frac{2n}{3}$.
- **Step 2**: For each weighted graph G obtained from Step 1, realize all possible polygons, KG, with G as its weighted projection.

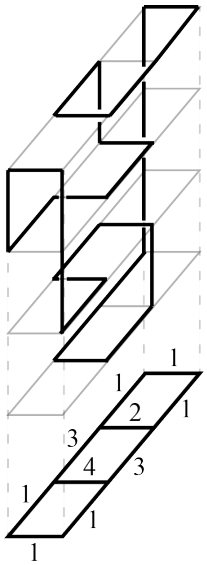

Figure 4: Polygon in a 1-slab and its projection

Figure 4 explains the relationship between the polygon and its weighted projections. The MSN Algorithm essentially works backwards, enumerating all projections, then adding weights to each projections, and then realizing the weighted projections into actual polygons.

Note that at each successive steps, we enumerate more things onto the set obtained from previous step. The parallel programming we are using at the moment is a series of map reduce. First we enumerate all the projections. Then, similar to the Monte Carlo method, we send equal number of graphs to each nodes using the independence of each graph. We then collect the weighted graphs, then distribute them to the nodes and finally collect the output polygons. However, there is a fundamental problem with this method of parallelization at the moment.

Unlike the Monte Carlo method which parallelized very nicely, our method of simple map reduce with no communication has limits. First, if we have more nodes than projections or weighted projections, we are not using the super computer to the fullest. Also the computation time of each step of the MSN Algorithm varies based on the graph. Especially the final step varies significantly based on the weighted projection. Thus we come to many situations in which some nodes are done generating polygons and just waiting for the other nodes to finish. It is important to note that map reducing Monte Carlo method and enumerations are very similar but they are fundamentally different parallelization problem.

3 Conclusion

Finally, I am taking this course to improve the parallelization - learn to divide up the problem better and communicate better to maximize the usage of resources available to us.

References

[1] Ishihara K, Scharein R, Diao Y, Arsuaga J, Vazquez M, Shimokawa K, Bounds for the minimum step number of knots confined to slabs in the simple cubic lattice. IOP Publishing, J. Phys. A: Math. Theor. 45(2012) 065003

[2] Diao Y 1993 J. Minimal knotted polygons on the cubic lattice, Knot Theory Ramifications 241325

[3] Scharein R, Ishihara K, Arsuaga J, Diao Y, Shimokawa K and Vazquez M, Bounds for the minimum step number of knots in the simple cubic lattice. 2009 J. Phys A: Math. Theor. 42 475006