Research Article

Preston Hamlin, W. John Thrasher, Walid Keyrouz and Michael Mascagni*

Geometry entrapment in Walk-on-Subdomains

https://doi.org/10.1515/mcma-2019-2052
Received August 3, 2019; revised October 13, 2019; accepted October 18, 2019

Abstract: One method of computing the electrostatic energy of a biomolecule in a solution uses a continuum representation of the solution via the Poisson–Boltzmann equation. This can be solved in many ways, and we consider a Monte Carlo method of our design that combines the Walk-on-Spheres and Walk-on-Subdomains algorithms. In the course of examining the Monte Carlo implementation of this method, an issue was discovered in the Walk-on-Subdomains portion of the algorithm which caused the algorithm to sometimes take an abnormally long time to complete. As the problem occurs when a walker repeatedly oscillates between two subdomains, it is something that could cause a large increase in runtime for any method that used a similar algorithm. This issue is described in detail and a potential solution is examined.

Keywords: Monte Carlo, walk on subdomains, Brownian motion, Poisson–Boltzmann, walk entrapment

MSC 2010: 65C05, 65N75

1 Introduction

When dealing with a solution containing biomolecules, one common approach describes the system as a continuum of various dielectric constants, rather than using detailed ionic interactions at the atomic or subatomic level [3, 5]. In such a system, the electrostatic potential, $u(x)$, satisfies Poisson's equation inside the molecule, while the potential in the solvent is distributed according to the Boltzmann law. This leads to describing the distribution of the electrostatic potentials using the Poisson–Boltzmann Equation (PBE) [7]:

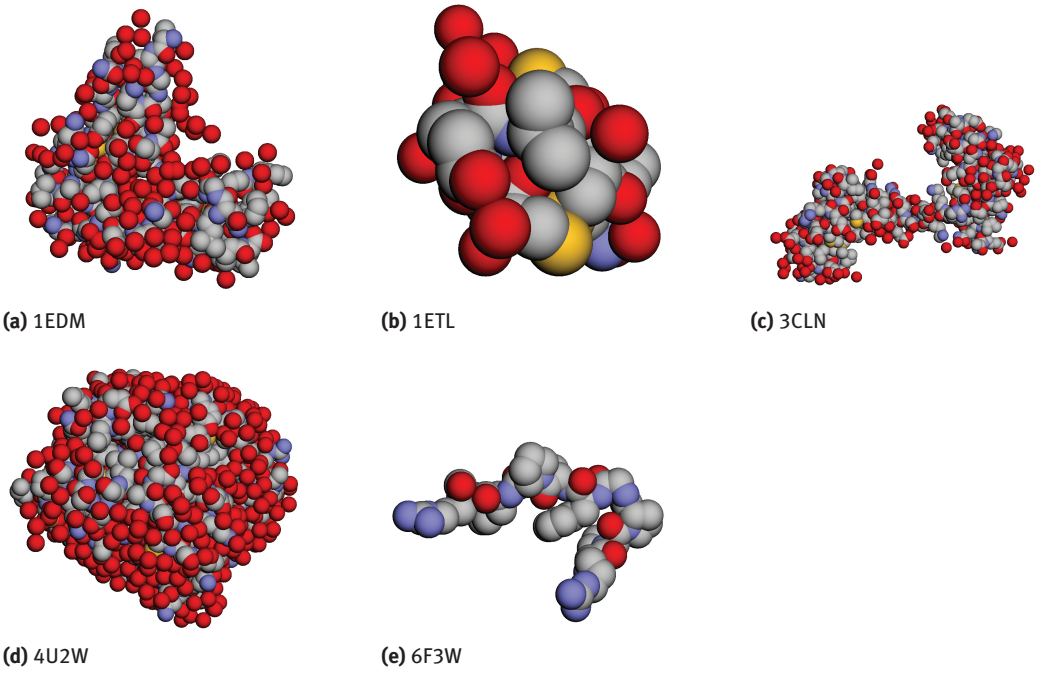
$$\Delta\psi(r) = \kappa^2 \sinh(\psi(r)). \tag{1.1}$$

For low-potential systems, this equation may be linearized to yield the Linearized Poisson–Boltzmann Equation (LPBE):

$$\Delta\psi(r) = \kappa^2\psi(r). \tag{1.2}$$

Both equations (1.1) and (1.2) rely upon κ, the inverse Debye length and the associated Debye–Hückel theory. The volume and surface of the biomolecule are formed from the union of the constituent atoms. A solution of equation (1.2) models the molecule as a union of perfect spheres. Examples of this type of representation are seen in Figure 1. Using this model, an estimate of the electrostatic potential can be constructed using a series of Markov processes started from each atomic center. The state space of each process

***Corresponding author: Michael Mascagni,** Department of Computer Science, Florida State University, Tallahassee, FL 32306-4530; and National Institute of Standards & Technology, ITL, Gaithersburg, MD 20899-8910, USA, e-mail: mascagni@fsu.edu. https://orcid.org/0000-0003-3058-4580
Preston Hamlin, Department of Computer Science, Florida State University, Tallahassee, FL 32306-4530; and National Institute of Standards & Technology, ITL, Gaithersburg, MD 20899-8970, USA, e-mail: hamlin@cs.fsu.edu. https://orcid.org/0000-0002-5030-2467
W. John Thrasher, Department of Computer Science, Florida State University, Tallahassee, FL 32306-453, USA, e-mail: wjt1321@my.fsu.edu. https://orcid.org/0000-0002-5487-5545
Walid Keyrouz, National Institute of Standards & Technology, ITL, Gaithersburg, MD 20899-8970, USA, e-mail: walid.keyrouz@nist.gov. https://orcid.org/0000-0003-3807-813X



(a) 1EDM **(b)** 1ETL **(c)** 3CLN

(d) 4U2W **(e)** 6F3W

Figure 1: Rendering of five sample input geometries.

is its current location in 3-dimensional Cartesian space and is herein referred to as a walker. Walkers traverse the interior and exterior of the model, with the Coulombic potential at a particular walker location tallied whenever the walker interfaces with the model's boundary. The partial Coulombic potentials contributed by each atom's spawned walkers are then accumulated to yield the free energy of solvation. Despite making several assumptions and approximations, this model has proven successful in predicting biomolecule properties [5, 7, 8].

1.1 Walk on spheres

The Walk-on-Spheres (WOS) family of Monte Carlo algorithms gain their name from the stepping mechanism they use. Rather than taking discrete fixed-size steps or moving along a fine grid in 3-dimensional space, WOS constructs a sphere centered at the walker's current location with a radius of the distance between the walker's current location and the nearest point of intersection with the geometry. The walker's next location is then sampled from this sphere. This is repeated until the walker intersects with the boundary of the region of interest. This method has been shown to have the same intersection probabilities as a Brownian motion while performing better than traditional traversal mechanisms [9].

1.2 Walk on subdomains

The Walk-on-Subdomains (WOSD) family of algorithms is closely related to the WOS algorithms. However, instead of walkers stepping through the entire domain until an exit point is found, the domain is divided into subdomains and each step of the walk samples to the exit point of the subdomain. Any method that approximates first-passage may be used to sample to the exit point, including using the WOS method inside the subdomain [8]. If a walker's first-passage probability can be calculated exactly, it has been shown to be more efficient to use that probability to directly sample to the subdomain boundary [4].

In the algorithm used here, walkers are spawned within a subdomain and sample to its surface directly. From there, walkers repeatedly sample to the surface of any alternate subdomain which they are also within.

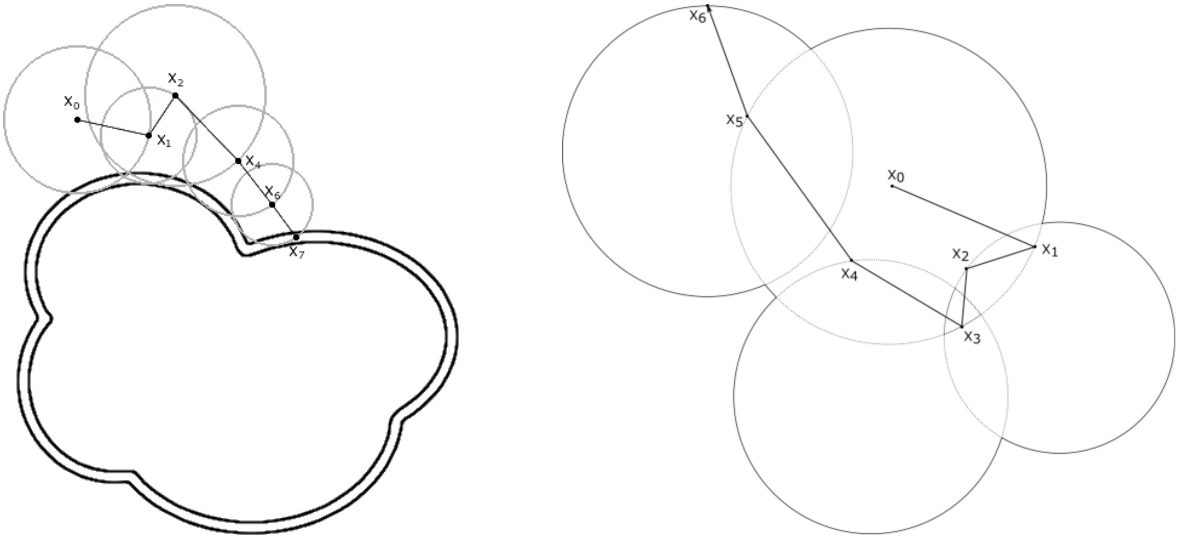


Figure 2: A run of WOS on the left demonstrates its capacity for expedient termination. A run of WOSD on the right demonstrates its traversal of subdomains.

This continues until the walker reaches the true surface of the union of subdomains. Analogous to how WOS will potentially intersect the surface each iteration, WOSD will always cross into a different subdomain each iteration that does not reach the true surface of the domain.

2 Algorithm structure

The algorithm used to estimate the LPBE is a combination of WOS and WOSD. Traversal of the interior of a biomolecule is done via WOSD, while exterior traversal is done with WOS. We will discuss certain aspects of this algorithm, but will not discuss the entire algorithm in detail, as this has been done previously [5, 7, 8].

2.1 Number of walks

Due to the potentially complex nature of the input geometry, it is possible that some regions are of greater interest or are more active than others [6]. Having a fixed number of walkers spawning from each location could cause some regions to be over- or under-represented. To account for this, the variance of the partial computations at each spawn location is examined. A high variance indicates that the current collection of walks differ substantially, and that more samples are required.

To determine an initial variance contribution estimate for each spawn location, a small number of walks are launched at each sphere center. If the calculated variance for walks spawned in a given sphere is too high, then more walks are scheduled to start at that location. This leads to each subdomain having a similar contribution to the final variance of the walk. This method of "variance balancing" has been shown to lead to a more efficient algorithm [6].

2.2 Internal walk

Initially, a walker is spawned at an atomic center, then the walker samples isotropically to the surface of the initial sphere. From this point, whenever a walker is in the interior of the biomolecule, it proceeds using the WOSD algorithm. If a walker resides on a sphere's surface and is not inside any other spheres, it must be on

one of the external portions composing the molecule's surface. If not, the walker will select a different sphere that it is also within. Then, the walker performs a sample to the surface of this target sphere using a Poisson kernel. This process is repeated until the walker finds itself on the surface of the molecule.

2.3 Boundary conditions

When the walker reaches the surface, be it from the interior or exterior, the Coulombic potential is computed. These potentials are summed at the end of the algorithm to yield the electrostatic free energy of solvation. Upon reaching the surface, the probability p_i of moving to the interior of the biomolecule and the probability p_e of moving to the exterior of the biomolecule are dependent upon the dielectric permeability of the two regions, ϵ_i and ϵ_e:

$$p_i = 1 - p_e = \frac{\epsilon_i}{\epsilon_i + \epsilon_e}.$$

2.4 External walk

After the walker exits the molecule, the external portion of the walk is performed using a Walk-on-Spheres algorithm. At each step of the walk, there is a probability that the walker is terminated, based in part on its distance to the boundary. Additionally, as the distance from the surface can grow exponentially, a walker is treated as having "gone to infinity" after reaching a sufficient distance and is terminated immediately.

3 Entrapment

When running this algorithm, it was noticed that a small percentage of runs were much longer and, as a result, took far more time. Upon investigating, it was determined that some walkers inside the biomolecule had an abnormally high walk length when compared to the average; these walkers oscillated between two joined spheres until they became entrapped at their cusp. In most cases, a stuck walker will escape after a few minutes. However, in some cases a walker may be stuck for several days. As can be seen in Figure 3, which shows the exact number of sample runs that finished with various run times, this can cause a few runs to take much longer than most, even discounting those that take several days. The exact number of walkers that finished at various walk lengths during one sample run that encountered entrapment for a few minutes

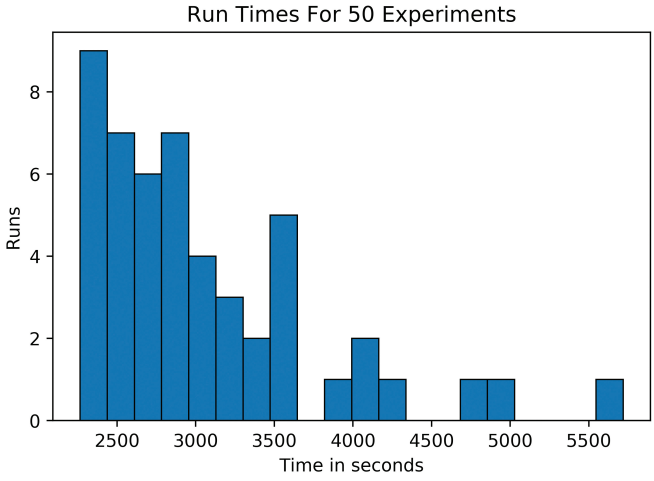


Figure 3: Run time counts for sample runs with the molecule 4U2W.

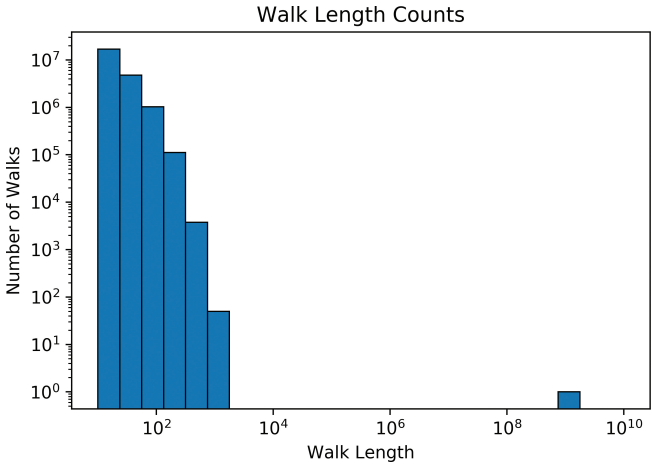


Figure 4: Walk length counts for a sample run that encountered walker entrapment with the molecule 1ETL.

can be seen in Figure 4. The extreme cases appear to be relatively rare, with 1.6 % of the 500 runs examined in this paper getting stuck for more than 24 hours.

One of the benefits of Monte Carlo methods such as this algorithm is that parallelization can be relatively easy to implement, while providing large benefits to efficiency. One way this is done is by running each of independent walks on separate processes. However, when a small number of walks dominate the overall run-time, such as when entrapment occurs, this potential improvement is greatly reduced. Thus, this issue needs to be explored before creating a parallel version of this algorithm in order to maximize the benefit of parallelization.

3.1 Sampling and subdomain connectivity

Walker entrapment is caused by a combination of the sampling method and subdomain connectivity. For sample input geometries representing various biomolecules, this model often has low connectivity, which means that points which are overlapped by multiple spheres are usually overlapped by only two spheres. This leads to cycles where a walker repeatedly jumps between the same two spheres. The Poisson kernel which is used for sampling from one surface to another uses a probability distribution which strongly favors the nearby portions of the target surface. Repeated sampling between two particular spheres results in an iterative minimization of the jump distances. If the oscillation continues, eventually the walker lies in the set of points which represent the minimal distance to either sphere, namely the cusp formed by the intersection of the two surfaces.

This issue is not restricted to this estimator, as it could extend to any instance of the WOSD algorithm. If the connectivity of the subdomains is sufficiently low, walkers will be pigeon-holed into a small set of traversal patterns. Sampling methods which introduce a preference for spatially proximate sections of a subdomain such as the Poisson kernel will favor cycles between two subdomains and could eventually lead to walkers becoming stuck. Any sampling mechanic which restricts or weights portions of a target subdomain over other portions of the same subdomain has the potential to increase the likelihood of cyclic patterns. As the subdomain connectivity and sampling method are often intrinsic to the problem model, any solution to walker entrapment must come from adjustments to other aspects of the algorithm.

3.2 Potential solutions

Two general strategies of solving this problem were considered: freeing entrapped walkers or terminating the problematic walks. It seems possible to determine when a walker has become entrapped and then freeing the

walker somehow. This would likely be done by forcing the walker into a new location. This would allow the walk to continue, but would require the computational overhead of determining when a walker is entrapped, which would likely require determining if the walker was cycling between two atoms repeatedly. Additionally, the potential bias created by moving the stuck walker via a method other than the Poisson kernel would have to be examined.

Alternatively, walks could be terminated and then restarted after certain cutoff conditions were met. One natural method to consider for a cutoff condition is to terminate walkers after they have completed a specified number of steps in the interior walk and then switch to a new walker or restart the previous walker. For this to work, a reasonable cutoff point needs to be determined. Terminating walkers that were stuck in a cycle for an extended period was also considered.

After consideration, the decision was made to explore terminating the walkers after they complete a specific number of steps. This method was chosen over the others in part because of its simplicity. Any method that required detecting whether a walker was in a cycle would require more computational overhead than a method that just requires the number of steps a particular internal walker had taken. Cycle detection requires keeping track of previous walker locations in addition to the number of steps.

3.3 First passage under restart

Recent research into generalizing the attributes of First Passage Time (FPT) processes examined the properties of First Passage Under Restart (FPUR), which is the process of stopping a FPT algorithm in the middle and then restarting it. It has been shown that
(1) all FPT processes share certain characteristics under restart,
(2) restarting FPT processes can sometimes lead to an improvement in run time, and
(3) certain restart strategies will behave similarly no matter the process in question.
Using the fact that the overall completion time of a process under restart can be calculated explicitly given the distributions of completion times and restart rates, restart strategies were examined and it was determined that the optimal average completion time using what is known as a sharp restart will always be the optimal average completion time of a first-passage algorithm. In this case, a sharp restart refers to restarting the algorithm after the process has taken a specific number of time steps [10].

Although the WOSD algorithm is not specifically concerned with first-passage location instead of first-passage time, it is still a given that the optimal sharp restart rate will lead to the optimal overall completion time. Given that, we chose to initially examine using sharp restart to avoid the walker entrapment problem, restarting based on the number of steps the interior WOSD process had taken.

4 Numerical experiments

4.1 Methods

We added a sharp restart condition to the previously used algorithm and examined its impact on the final result. One concern was whether to retain any accumulated statistics when restarting, but the decision was made to discard any statistics accumulated in a restarted walk. It seems that in most situations discarding previously accumulated statistics would lead to less potential for error, since restarted walks could begin at the same atomic center.

We ran a series of tests on the same machine while keeping all parameters identical. We ran these tests using both no restart and a set of restart points, on a variety of molecules. These molecules (shown in Figure 1) are identified by the Protein Data Bank [1] IDs 1EDM, 1ETL, 3CLN, 4U2W, and 6F3W. Combinations of molecule and termination point were run 100 times; the exception was the 4U2W molecule experiments, which were run 50 times only, because these experiments took significantly longer. When a run with no

restart point went beyond 24 hours, it was considered "stuck" and was stopped and discarded. Although it was assumed that we would run the experiments using the exact same set of termination points for each molecule, this was not possible in practice. For each molecule, there was a point beyond which decreasing the restart point would cause the run to either take an extremely long time to finish or potentially never finish at all. To understand why, consider the degenerate case of a molecule with a single atom at its center, which does not touch the boundary. If each run is restarted after one step, no run from that atom will ever successfully complete given that a single step will only reach the edge of this atom which, by definition, is not on the boundary. In other molecule configurations, there may be similar points for which a molecule will never, or at least very rarely, reach the boundary before a specific step. So, once a point that appeared to not finish was found, each molecule was explored by attempting additional runs around that same point until the minimum viable restart point for each molecule was found.

4.2 Results

The results of these experiments can be seen in Tables 1–5. As these results are averages, it is important to provide the variance. When looking at the average result of the calculation for electrostatic free energy, the variance was universally very small at $< 0.1\,\%$ of the average. For both average walk length and run time, the

Restart point	Average result	Error	Average walk length	Time (in seconds)
None	−13.7129	0.0568	51.9212	78.3364
100,000,000	−13.7110	0.0567	20.9138	77.0880
10,000,000	−13.7096	0.0561	14.3072	79.5120
1,000,000	−13.7098	0.0558	14.1270	76.8071
100,000	−13.7111	0.0559	14.1596	77.3607
50,000	−13.7142	0.0558	14.1472	77.3041
10,000	−13.7058	0.0558	14.1364	74.7281
5,000	−13.7086	0.0562	14.1352	74.5378
1,000	−13.7198	0.0560	14.1346	75.9846
500	−13.7064	0.0565	14.1343	75.0565
100	−13.7144	0.0552	14.0657	74.4466
50	−13.7276	0.0549	13.9831	75.3199
25	−13.8333	0.0552	11.4818	70.2528
10	−13.7090	0.0454	7.3197	56.4636

Table 1: Results with varying restart points: 1EDM.

Restart point	Average result	Average error	Average walk length	Time (in seconds)
None	−0.8868	0.0006	36.5192	189.7964
100,000,000	−0.8868	0.0006	19.6515	185.4263
10,000,000	−0.8868	0.0006	18.5510	179.0251
1,000,000	−0.8868	0.0006	18.5375	176.5825
100,000	−0.8867	0.0006	18.5372	175.3536
50,000	−0.8867	0.0006	18.5372	181.9247
10,000	−0.8867	0.0006	18.5372	177.6882
5,000	−0.8867	0.0006	18.5372	174.2456
1,000	−0.8867	0.0006	18.5369	172.7220
500	−0.8867	0.0006	18.5254	186.0867
100	−0.8880	0.0006	17.3952	178.5118
50	−0.8937	0.0006	15.4920	170.6816
25	−0.9100	0.0006	12.9868	156.7272
14	−0.8812	0.0005	10.1429	154.6872

Table 2: Results with varying restart points: 1ETL.

Restart point	Average result	Average error	Average walk length	Time (in seconds)
None	−30.7328	0.0965	56.0754	96.6112
100,000,000	−30.7300	0.0965	21.0292	99.0851
10,000,000	−30.7318	0.0964	15.3287	95.3058
1,000,000	−30.7285	0.0975	15.1058	93.1678
100,000	−30.7270	0.0966	15.1053	94.1206
50,000	−30.7468	0.0964	15.1050	93.1590
10,000	−30.7384	0.0962	15.1048	93.0327
5,000	−30.7408	0.0969	15.1048	92.9857
1,000	−30.7477	0.0966	15.1039	93.1639
500	−30.7359	0.0964	15.0970	93.9057
100	−30.7414	0.0966	14.7247	94.0392
50	−30.7778	0.0959	13.7335	91.8218
25	−30.9709	0.0939	11.7728	87.0892
11	−31.0929	0.0803	7.9566	67.3179

Table 3: Results with varying restart points: 3CLN.

Restart point	Average result	Average error	Average walk length	Time (in seconds)
None	−10.8576	0.0084	81.6771	3066.4531
100,000,000	−10.8570	0.0084	35.6938	3155.2219
10,000,000	−10.8590	0.0084	33.9225	3364.1489
1,000,000	−10.8594	0.0083	33.8885	3287.5920
100,000	−10.8582	0.0084	33.8891	3105.3100
50,000	−10.8568	0.0084	33.8892	3124.3066
10,000	−10.8589	0.0084	33.8892	3094.1496
5,000	−10.8577	0.0084	33.8891	3172.9268
1,000	−10.8593	0.0086	33.8431	3152.0747
500	−10.8594	0.0084	33.6403	3318.5277
100	−10.9623	0.0081	27.3894	3059.3463
50	−11.2143	0.0078	22.3026	2535.3186
25	−12.0719	0.0078	16.9585	2320.4548
19	−11.2443	0.0070	14.6984	2015.4529

Table 4: Results with varying restart points: 4U2W.

Restart point	Average result	Average error	Average walk length	Time (in seconds)
None	−1.08611	0.000658	27.146	102.5798
100,000,000	−1.08612	0.000655	20.51535	95.87435
10,000,000	−1.08611	0.000655	20.4744	95.03875
1,000,000	−1.08611	0.000655	20.47435	95.22887
100,000	−1.08614	0.000655	20.47436	94.90337
50,000	−1.08614	0.000656	20.47436	94.90442
10,000	−1.08611	0.000655	20.47435	94.91144
5,000	−1.08611	0.000655	20.47435	94.90833
1,000	−1.08611	0.000655	20.47419	95.08391
500	−1.08612	0.000655	20.46283	95.43036
100	−1.08689	0.000658	19.91288	98.87905
50	−1.09377	0.000649	18.13865	88.56578
25	−1.13828	0.000676	14.77466	73.45757
16	−0.95423	0.000556	11.32117	122.0429

Table 5: Results with varying restart points: 6F3W.

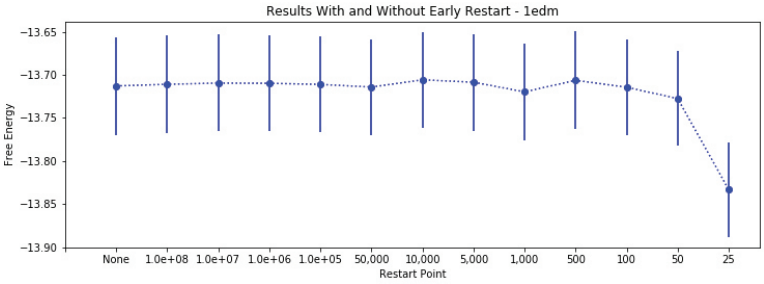
variance was larger with no restart and larger restart points. The variance of the average walk length ranged from < 0.01 % of the average at smaller restart points and up to > 1400 % of the average with no restart. Even at its smallest, the variance for run time was much larger, with most values lying within 80 % and 300 % of the average. For a given run of the algorithm, a single standard deviation is used as the error, as calculated by summing the variance of the walks from each atom and taking the square root of that number. For each termination point used, the mean error was then calculated.

As can be seen in the tables, using a restart point did not always lead to an improvement in run time. Decreasing the restart point often led to a better run time, but this was not universal. None of the molecules showed a monotonic decrease in run time as the restart point was decreased. In fact, molecule 4U2W showed an increase in run time that was not alleviated until the restart point was reduced to 100 steps. It should be noted, however, that this does not take into account the runs that were discarded as they took more than 24 hours, so the potential speed increase could be higher than what is shown in these results.

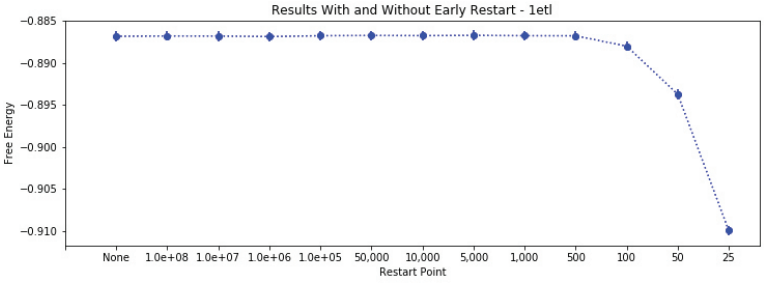
More important than raw run time, is whether this scheme removes the possibility of walker entrapment while still providing the same results. An initial qualitative view of the results certainly makes that look possible; all results up to a certain point appear to not fluctuate much and remain well within the error bounds of the original results as can be seen in Figure 5. In this figure, the error bounds are the average error of all runs at a particular termination point, as previously discussed. This figure also shows the large difference between estimates with no restart and estimates with some of the smaller restart points. In three of the cases, the difference is large enough that it makes the error bounds around the results difficult to see. However, a more quantitative method of examining whether the populations of results are significantly different is to run a series of ANOVA tests on the results [2]. These tests were completed following two different schemes. In the first set of ANOVA results, shown in Table 6, the tests included all restart points up to a specific value. So, the first ANOVA test on a particular molecule was to determine whether it appeared all results were from the same statistical distribution. The second test included all results except the runs restarted after 25 steps. Each subsequent ANOVA test included one fewer restart point, removing the results that used the lowest restart point included in the previous ANOVA test. The second set of ANOVA results, seen in Table 7, compared each individual restart point solely to the results with no restart. Using a significance value of $p < 0.05$, it can be seen that under both schemes, each molecule did not see a significant difference in results when using larger restart points, but as the restart point was reduced, there came a point in each molecule where the results were significantly different. Interestingly, this point was not identical for all molecules and did not seem to be directly linked to molecule size; this indicates that a molecule's geometry likely contributes to the first restart point that significantly differs from the original results.

Smallest included restart point	1EDM	1ETL	3CLN	4U2W	6F3W
25	< .001*	< .001*	< .001*	< .001*	< .001*
50	< .001*	< .001*	.048*	< .001*	< .001*
100	.946	< .001*	.841	< .001*	< .001*
500	.934	.850	.794	.728	.999
1,000	.932	.790	.710	.730	.999
5,000	.993	.695	.818	.733	.999
10,000	.985	.740	.794	.651	.999
50,000	.996	.687	.730	.567	.999
100,000	.997	.768	.992	.628	.998
1,000,000	.983	.798	.989	.497	.999
10,000,000	.936	.886	.978	.471	.992
100,000,000	.832	.640	.841	.734	.934

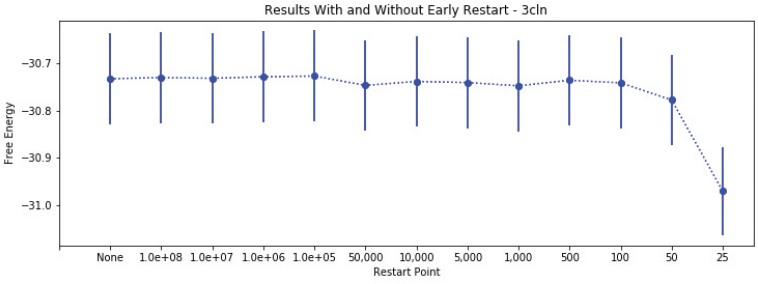
Table 6: ANOVA including multiple restart points – p value (* $p < 0.05$).



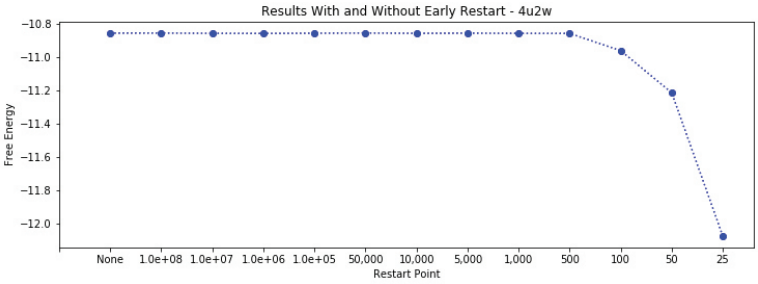
(a) 1EDM



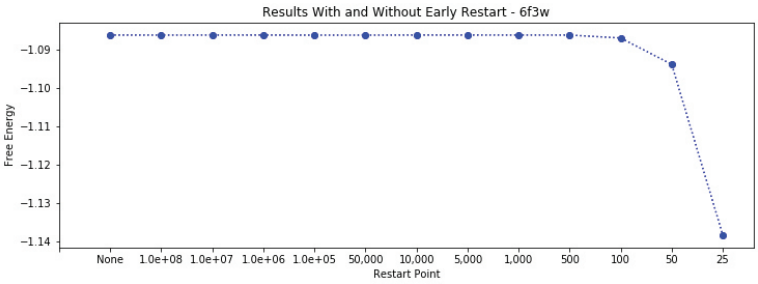
(b) 1ETL



(c) 3CLN



(d) 4U2W



(e) 6F3W

Figure 5: Results for different molecules with various restart points.

Included restart point	1EDM	1ETL	3CLN	4U2W	6F3W
25	< .001*	< .001*	< .001*	< .001*	< .001*
50	.083	< .001*	.002*	< .001*	< .001*
100	.853	< .001*	.541	< .001*	< .001*
500	.461	.483	.818	.313	.951
1,000	.432	.455	.266	.348	.929
5,000	.630	.214	.551	.945	.952
10,000	.441	.385	.681	.492	.965
50,000	.892	.284	.333	.652	.835
100,000	.841	.429	.670	.692	.808
1,000,000	.740	.672	.751	.367	.978
10,000,000	.718	.726	.943	.419	.967
100,000,000	.832	.640	.841	.734	.934

Table 7: ANOVA including single restart point – p value (* $p < 0.05$).

5 Summary and future work

Initial testing of using a sharp restart rate to solve the problem of walker entrapment during the WOSD algorithm provided positive results. It seems clear that using this method properly can, with a suitably chosen restart point, eliminate the issue without causing additional bias to the results and, in some cases, may even cause an overall speed improvement when walker entrapment is eliminated. However, there remain several avenues to explore in future work.

5.1 Choosing a restart point

As the potential exists for bias to occur if a restart point is chosen that is too small, an examination of an appropriate choice for restart point should be completed. It may be that a uniform, large restart point may be used successfully in most cases, but this is potentially not the most efficient method. As the inflection point after which there is significant bias varies between molecules, it seems possible that this point is related to the molecule geometry in some way. Potential variables to examine include the molecule size, the number of atoms in the molecule, the degree of connectivity between atoms in the molecule, and other measures of the shape of the molecule. If a reasonable relationship between the molecule's geometry and the lowest viable restart point can be found, then a method of automating the choice of restart point could potentially be implemented.

Another avenue worth exploring is an examination of the efficiency of using this method to improve the algorithm and its relation to the chosen restart point. Although using this method is likely to decrease run times over many executions, choosing the wrong restart rate can actually cause the run time of a single execution to increase when entrapment is not an issue. For at least one molecule, 4U2W, it seems every viable restart point causes an increase in run time. The effect on run time also seems as if it could potentially be related to the geometry of the molecule.

5.2 Other remedies

Other methods of avoiding the walker entrapment problem can also be examined. For example, a different sampling method that does not predispose walkers to become stuck can replace the one currently being used, if such a sampling method can be found. Additionally, the previously proposed method of attempting to detect cycles and moving the walker to a slightly different point on the atom when it appears to be stuck may be a solution. Yet another alternative is to simply replace the WOSD portion of the algorithm; the Walk-on-Boundary method [11, 12] may be a viable replacement in some cases.

5.3 Parallelization

As eliminating walker entrapment will allow for a reduced maximum run-time for each portion of the algorithm, this will cause the lengths of the interior walks to normalize. This should allow for more opportunities to consistently increase the efficiency of the algorithm using parallelization. With these improvements, it should be possible to create a consistent parallel implementation of this algorithm on either multi-processor systems or GPUs. This result may also be generalizable to other first-passage Monte Carlo methods.

5.4 Generalizations

The problem of walker entrapment may be found in other algorithms that use WOSD or similar methods. Once possible solutions for this problem in the LPBE algorithm have been examined more closely, these should be examined for general applicability to the WOSD algorithm or other first-passage Monte Carlo methods. Additionally, since using a sharp restart rate has been shown to improve the run time of first-passage time algorithms, it seems reasonable to assume that it can be used to improve the efficiency of other first-passage location algorithms, even if they do not suffer from problem of walker entrapment.

References

[1] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov and P. E. Bourne, The protein data bank, *Nucleic Acids Res.* **28** (2000), 235–242.
[2] R. N. Cardinal and M. R. F. Aitken, *ANOVA for the Behavioral Sciences Researcher*, Psychology Press, London, 2013.
[3] M. O. Fenley, M. Mascagni, J. McClain, A. R. J. Silalahi and N. A. Simonov, Using correlated Monte Carlo sampling for efficiently solving the linearized Poisson–Boltzmann equation over a broad range of salt concentration, *J. Chem. Theory Comput.* **6** (2009), 300–314.
[4] J. A. Given, J. B. Hubbard and J. F. Douglas, A first-passage algorithm for the hydrodynamic friction and diffusion-limited reaction rate of macromolecules, *J. Chem. Phys.* **106** (1997), 3761–3771.
[5] C.-O. Hwang, M. Mascagni and N. A. Simonov, Monte Carlo methods for the linearized Poisson–Boltzmann equation, in: *Advances in Numerical Analysis*, Nova Science, Hauppauge (2004).
[6] T. Mackoy, R. C. Harris, J. Johnson, M. Mascagni and M. O. Fenley, Numerical optimization of a walk-on-spheres solver for the linear Poisson–Boltzmann equation, *Commun. Comput. Phys.* **13** (2013), 195–206.
[7] M. Mascagni and N. A. Simonov, Monte Carlo method for calculating the electrostatic energy of a molecule, in: *Computational Science—ICCS 2003. Part I*, Lecture Notes in Comput. Sci. 2657, Springer, Berlin (2003), 63–72.
[8] M. Mascagni and N. A. Simonov, Monte Carlo methods for calculating some physical properties of large molecules, *SIAM J. Sci. Comput.* **26** (2004), no. 1, 339–357.
[9] M. E. Muller, Some continuous Monte Carlo methods for the Dirichlet problem, *Ann. Math. Statist.* **27** (1956), 569–589.
[10] A. Pal and S. Reuveni, First passage under restart, *Phys. Rev. Lett.* **118** (2017), Article ID 030603.
[11] K. K. Sabelfeld, *Monte Carlo Methods in Boundary Value Problems*, Springer, Berlin, 1991.
[12] K. K. Sabelfeld and N. A. Simonov, *Stochastic Methods for Boundary Value Problems. Numerics for High-dimensional PDEs and Applications*, De Gruyter, Berlin, 2016.