Chapter 12
Future work

Macroscopic-cycle quotient: Now that the α-dependence of the macroscopic-cycle quotient's constant upon α has been found empirically, one would next like to explain that dependence analytically.

Non-asymptotic algorithm correctness: The detailed-balance correctness proof of the swap-and-reverse algorithm shows that there is a non-zero transition probability between all pairs of permutations. However, those non-zero transition probabilities can be quite small. As the number of Metropolis steps goes to infinity, asymptotically all permutations can be reached; more interesting is the question of which permutations are actually reachable in reasonable simulation time. One answers this question empirically simply by running simulations; perhaps this suffices. For the system discussed in this paper, as well as for other systems studied using MCMC methods, it would be useful to have a non-asymptotic correctness theory.

Winding numbers of all parities: Ideally, one would have an algorithm to permit odd winding numbers, as discussed in section 5.4.

Bose-gas Hamiltonian: Sampling from the true Bose-gas distribution using the random-cycle model requires three changes. First, one needs to conduct simulations using the Bose-gas interaction (equation (2.1.1)) rather than the cycle-weight interaction (equation (2.1.3)). The interaction term V is a CPU-intensive Brownian-bridge computation [BU07]; unpublished work of Ueltschi and Betz shows that it may be approximated in the weak-interaction case by a simpler Riemann integral. Precomputed tables and interpolation may make use of this integral feasible.

Second, point positions must be allowed to vary on the continuum. This entails a second type of Metropolis step, in addition to that shown in section 5.1. Namely, one picks a point and moves it to a new position nearby, using the detailed-balance condition to choose the acceptance probability.

Third, software efficiency requires a hierarchical partitioning of Λ. The Metropolis step of section 5.1 relies on picking $\pi(\mathbf{y})$ near to $\pi(\mathbf{x})$. For points on the lattice, this is easy: each site has six nearest neighbors. For freely placed points, though, one must remember which sites are close to which. The most naive implementation involves computing the distances between all $N(N-1)/2$ pairs of points; the $O(N^2)$ computation time is overwhelming. Instead, the lattice may be partitioned into subcubes. Distances need to be computed only between each given point $\mathbf{x}$ and those in $\mathbf{x}$'s subcube and the 26 nearest-neighbor subcubes.

The second and third points simply require a software effort. Implementing them will be worthwhile only if the interaction terms can be simplified to the point that they are computationally feasible. This is a mathematical effort.