Benefit of the two-step approach (R1, R2): As R1 suggested, we examined 2-OPT's robustness as measured by the 90% quantile of the (log) regret over replications (see figure). By this measure, 2-OPT improves more significantly in each problem over the best competing method. 2-OPT is also more robust than competing methods when looking at the mean regret (or 90% quantile) across *problems*. 2-OPT is better than or comparable to the best other method in nearly all problems and iterations. In contrast, consider EI. By mean performance, EI performs as well as 2-OPT on several problems (logistiic regression, SVM, ATO, and 2d Camel) but underperforms in others (Ackley, Cosine, Levy) and sometimes severely so (Branin, Hartmann6 — note that performance is on a log scale). Other methods behave similarly, performing as well as 2-OPT in some problems but underperforming significantly in others.

Computational Cost (R1, R2, R3): The figure below shows the time required for acquisition function optimization on a single core using AWS instances. Time for other problems is similar, with higher-dimensional problems requiring more time. 2-OPT's computation time is comparable to KG, about 10 times slower than EI, and about 10 times faster than GLASSES. (Code from Lam et al. 2016 is unavailable.) We'll report computation time in the final version.

While 2-OPT's computation per iteration can be substantial, the multiple (1000) restarts of SGD used by 2-OPT can be trivially parallelized with a linear speedup. Parallelizing all starts gives < 2 seconds of wall-clock time per iteration for all problems. Moreover, for objective functions that require several hours per evaluation on a multi-core machine, being able to find a good solution with fewer time-consuming objective function evaluations often merits the additional overhead required to optimize a more sophisticated acquisition function. Since 2-OPT has more robust query efficiency than other methods, and is as fast as KG and GLASSES, we feel that 2-OPT is of significant practical value in an important range of problems: those objective function evaluation are costly enough to make improving query efficiency over EI and other faster myopic methods worth the additional computational cost.

More experimental settings (R2, R3): We ran experiments for Rosenbrock (see figure) and will add Michalwicz and Robot pushing to the final version. Goldstein-Price is available in Table 1.

"Authors compare with related [non-myopic] methods only on a set of synthetic problems extracted from another reference." (R2) At submission we lacked code for GLASSES and Lam et al. 2016. We recently obtained code for GLASSES and will include comparisons in the final version. We're awaiting email replies from Lam et al. If we obtain code in time we'll also include those comparisons in the final version.

"The only contribution seems to be the optimization of the acquisition function which is done using stochastic gradient algorithms" (R2) Our primary contribution is, indeed, an efficient method for optimizing the 2-step optimal acquisition function. Our secondary contribution is to show that this is practical (i.e., fast enough to use in practice) and provides more robust performance than both existing widely-used myopic acquisition functions and previously-proposed non-myopic acquisition functions. While it is true that stochastic gradient ascent is a standard approach, the challenge in applying it in our setting is in creating an efficient stochatsic gradient estimator and proving it is unbiased.

Other Comments:

"variance on the optimization traces for EI and LCB" (R1): We think this may be because EI and LCB explore less than other methods, and whether they fall into sub-optimal local optima depends strongly on the problem. Ackley has more local optima perhaps explaining why performance is less smooth. Define Q (R1,R2): This was a typo. Q is $2 - \mathrm{OPT}(X_1)$. Code is not provided (R2): We will include our github repo in the final version. Batch evaluations (R2): Our method generalizes naturally to batch evaluations and our code supports them but we did not include batch experiments. We'll include them in the appendix in the final version.