We would like to thank the reviewer for their insightful comments. Below we address the main concerns:

To Reviewers 1 & 2: Comparison with heuristic methods:
Making a meaningful comparison with heuristic methods at this stage is a hard: While our algorithm is designed to stop when a near-optimal configuration is found provably, the method can continuously provide a candidate configuration, and it is not clear at which time point one should make the comparison (and what ε and δ parameters should be selected). Nevertheless, Figure 1 shows (for Minisat with $\gamma = 0.05$) that the best configuration is found much earlier than the time needed to prove that it is indeed (near-)optimal (19 vs. 101 CPU-days).

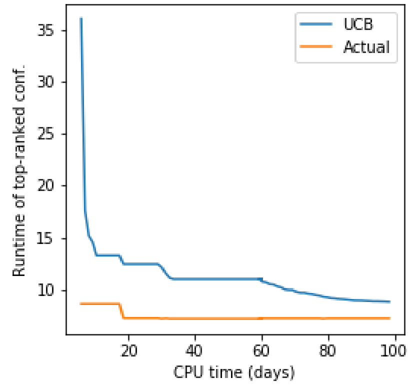

Figure 1: Actual expected runtime and the corresponding high probability bound maintained by ICAR for the top-ranked configuration (Minisat, $\gamma = 0.05$).

To Reviewer 1:

- More detailed high-level description of the algorithm: We will try our best to fit a more detailed high-level description of the algorithm in the main text.
- EPM: The accuracy of the EPM model has been analyzed thoroughly in [12]. Any controllable (additive or multiplicative) error of the simulator can easily be incorporated in the analysis. Of course, when the EPM model exhibits poor performance, the practical performance can degrade, but to guarantee the correctness of the algorithm, we only need a good accuracy of the EPM approximation near the optimum, although the runtime can be increased if additional variance of the runtime distribution is introduced by the simulator.
- sat vs unsat: The Minisat dataset contains both satisfiable and unsatisfiable instances.

To Reviewer 2:

- Presentation: For ease of reference, we will improve the presentation of the theoretical bounds of previous work as recommended. Nevertheless, the order of these bounds ($\frac{\mathrm{OPT}^{\gamma}_{\delta/2}}{\varepsilon^2\delta\gamma}$) is presented in discussions (i) and (iii) following Theorem 1, and also in Appendix B. We will also try our best to accommodate the suggested changes to the introduction.
- Pool of size $n = \log(\zeta)/\log(1-\gamma)$: every randomly selected configuration is in the top γ fraction of all the configurations with probability γ. Hence, the probability that no configuration out of n is top-γ is $(1-\gamma)^n = \zeta$ (see also the proof of Lemma 4).

To Reviewer 3:

Theoretical comparison to CAR: The bound for CAR [35, eq. 1] is much more complex because it is a problem-dependent bound which includes, e.g., the relative variance of the runtimes for each configuration and the gaps between the expected runtime of the best configuration and the other configurations. This bound in the worst case (in the minimax sense) simplifies to $\tilde{\mathcal{O}}\left(\frac{n\mathrm{OPT}_{\delta/2}}{\delta\varepsilon^2}\right)$ (where n is the number of configurations), which is essentially the same as the first (dominant) term of (1) in our paper without the factor F (and replacing $\mathrm{OPT}_{\delta/2}$ with $\mathrm{OPT}^{\gamma}_{\delta/2}$, using the number of instances $n_I = \log(K/\zeta)/\gamma$ ICAR uses). In fact, if one follows the more complicated derivation of [35], the first term in (1) can be replaced with a similar problem-dependent summation as in [35, eq. 1], however, only over the configurations not rejected by precheck (i.e., over at most $n_I F(38\mathrm{OPT}^{\gamma}_{\delta/2})$ configurations, and replacing n in the first term of [35, eq. 1] with the same quantity), while the rest of the terms remain the same (an added complication here is that the gaps and relative variances would become random variables in our case, with a tricky dependence introduced by the precheck mechanism). We omitted this analysis for simplicity and for the clarity of the presentation.

To Reviewer 4:

Spikes in Figure 1 (of the paper): Note that this figure shows the runtime of individual configurations, thus the fact that ICAR runs some configurations longer than CAR++ is not indicative of their overall performance (nevertheless, as shown in Table 1, the total runtime of CAR++ can be slightly smaller than that of ICAR when the runtime distribution is simple, and there is no real need to run our carefully designed precheck method).