Description of TQrouting's Hardware and Runs during the CVRPLib BKS Challenge

Giorgi Tadumadze,* Mikhail Somov, Katsiaryna Tsarova, and Michael Perelshtein

Terra Quantum AG, Kornhausstrasse 25, 9000 St. Gallen, Switzerland

E-mail: gt@terraquantum.swiss

Overview

This document describes the computational setup, execution strategy, and runtime usage of team TQrouting during the CVRPLib BKS Challenge. TQrouting achieved **27 out of 100 final BKSs** and, at peak time during the competition, held up to **70 BKSs** simultaneously.

Due to the iterative and adaptive execution strategy of our runs—combining multiple cold starts, several warm starts with parameter adjustments, and checkpoint-based restarts—it is not possible to provide detailed per-instance machine times for each achieved BKS, some of which may result from a cold-start run followed by several sequential warm-start runs. However, we provide detailed documentation of the strategy for our runs, the planned and actual runtimes for the cold-start runs, and the hardware resources used.

Overall Strategy

Initial Strategy

Our initial strategy was based on two main approaches: (1) repeated independent long cold-start runs using various parameter configurations to ultimately generate new high-quality solutions, ideally from distinct solution regions; and (2) warm-start runs with high-quality incumbent solutions with a focus on marginal improvements.

The original plan was to dedicate the first phase of the challenge to cold-start runs with gradually increasing runtime for each subsequent run, and to focus on warm-start runs with high-quality initial solutions in the second phase. The increasing runtime strategy for each subsequent cold-start run would more likely improve previous BKSs and, by the end of each run, regain (or increase) the number of BKSs, even if we temporarily lost some during intermediate competition days. In the second phase, the focus had to shift to improving the high-quality solutions via warm starts and aggressive intensification to further refine the identified BKSs and secure them through the end of the challenge.

Major Strategic Adjustment

A key turning point occurred by the end of the first week during the second cold-start run, which was designed as a 5-day cold start. However, the run was interrupted due to an unexpected hardware issue on the GKE cluster approximately 27 hours before its planned completion. This interruption occurred exactly during the final phase, when new BKSs were being found and systematically improved. As a result, approximately four days of critical computation (in terms of solution maturation) during the early critical phase were effectively lost. This translated into losing not only significant score by not updating new BKSs until the end of the next run, but also the chance to secure strategically important instances, for which the final BKSs were found in the first phase of the competition. This forced us to shift our initial strategy and focus more on larger/harder instances.

Hardware Infrastructure

We ran our experiments on Google Kubernetes Engine (GKE) using `c2d-highcpu-16` virtual machines (VMs), each with an AMD EPYC (Milan) processor (2.1 GHz), 16 vCPUs, and 32 GB RAM. At the beginning of the competition, we started with a cluster of 15 nodes. On each of these VMs (apart from the last one), we ran 7 instances in parallel (each on a single physical core), leaving the remaining cores for additional jobs such as storing intermediate checkpoints to monitor convergence behavior and upload new BKSs during the run.

After our strategy adjustment, instead of using the existing resources exclusively for warm-start runs in the second phase of the challenge, we added another cluster with the same configuration for warm-start runs and used the first cluster for cold-start runs until the end of the challenge. To reduce the computational costs and optimize resource utilization, we reduced both clusters to 14 nodes after 15 days. Particularly, we gave up solving two instances (`XL-n1094-k157` and `XL-n1794-k163`), whose BKSs were very likely to be either global optima or such local optima, which were unlikely to be improved. Our decision was based on the observation on the results of previous runs, in which we were consistently finding solutions with identical objective values in the early search phase, but could not improve them further.

In parallel with the cluster runs, we conducted multiple warm-start runs locally on two MacBooks equipped with Apple M2 Pro chips (maximum frequency 3.5 GHz), 12-core CPUs, and 16 GB RAM. These local runs targeted promising instances and were used opportunistically throughout the competition. We ran multiple warm-start runs with different aggressive parameter configurations to guide the algorithm out of current local optima. Unlike the cluster runs, the local runs utilized multiple threads, significantly accelerating the search.

Chronological Summary of Runs

Cold-Start Runs on the GKE Cluster

Run	Period	Planned Duration	Actual Duration	Nodes / Instances	Notes
1	∼12–14 Jan	2 days	2 days	15 / 100	Ended with 68 BKSs (peak 70).
2	14–18 Jan	5 days	∼3.9 days	15 / 100	Early termination due to unexpected cluster issue during the final intensification phase. In the last 15 hours before termination, we regained lost BKSs and increased the number of BKSs from 22 to 29.
3	19–23 Jan	4 days	4 days	15 / 100	Recovered the lead with 40 BKSs from 16 BKS at the lowest time.
4	23–27 Jan	7 days	∼4 days	15 / 100	Terminated after premature stagnation detected; algorithm stuck in local optima. Only two BKS improvements were obtained, while the run ended with 26 BKSs.
5	27 Jan–3 Feb	7 days	7 days	14 / 98	Two instances dropped (likely optimal). The exploration phase produced no new BKS for five days, reducing the count to 15. Parallel warm starts partially compensated this drop. The final two-day intensification phase raised the count to 19 and generated strong incumbents for later improvements.
6	3 Feb–11 Feb	8 days	8 days	14 / 98	Final cold-start run until the last competition day.

Warm-Start Runs

In addition to the cold-start runs summarized above, we conducted a large number of warm-start experiments throughout the challenge. These runs used previously discovered high-quality solutions as incumbent solutions and focused on improving them through intensified local search and modified diversification strategies. Because the warm-start experiments were executed adaptively and opportunistically, it is not possible to provide exact runtimes for individual runs or for specific improvements.

Warm-start runs were carried out on two different types of infrastructure. In the second half of the challenge, we operated an additional compute cluster dedicated exclusively to warm-start runs. On this cluster, the algorithm was repeatedly restarted using incumbent solutions obtained from earlier cold- or warm-start runs. The objective was to refine these solutions through repeated restarts with different algorithmic configurations and diversification mechanisms, allowing the solver to explore alternative neighborhoods. In many cases, improvements were obtained through a sequence of several shorter runs rather than a single long execution. When the search stagnated, the algorithm was restarted with modified parameters controlling diversification intensity, neighborhood sizes, or other internal mechanisms. This adaptive strategy allowed us to repeatedly re-enter promising regions of the search space from different directions.

In parallel with the warm-start runs on the cluster, we also executed additional experiments locally on two MacBooks. These runs typically targeted instances for which our current solutions had objective values equal or very close to the best-known values reported on the leaderboard. The local runs were conducted in a more exploratory and experimental manner. For selected instances, we tested aggressive diversification strategies, large neighborhood configurations, and intensified local search phases to probe the surrounding search space of promising incumbent solutions. Consequently, the execution pattern of these runs was intentionally flexible and sometimes irregular, reflecting their exploratory nature. Many runs were restarted frequently with modified parameters whenever convergence behavior

indicated that further improvements might require a different search strategy.

An important difference between these local experiments and the main cluster runs is that in the local runs, we ran the algorithm on multiple threads, whereas the main cluster runs were limited to a single physical thread per instance. As a result, local warm-start experiments were often able to explore neighborhoods and intensification phases significantly faster. Several BKS improvements were obtained through these targeted warm-start campaigns, often after multiple iterative restarts.

Because these runs were highly adaptive, restarted frequently, and executed across different machines and configurations, precise accounting of their individual runtimes was not recorded. However, they played an important complementary role to the long-running cold-start cluster computations by allowing focused improvements on promising instances.

Role of the Single-Thread Constraint

Due to limited computational resources during the challenge, the cluster runs were configured so that each instance ran on a single physical core. The primary reason for this configuration was the need to process the entire benchmark set in parallel. Each virtual machine in the cluster provided 16 vCPUs corresponding to 8 physical cores with simultaneous multi-threading. During preliminary experiments we evaluated different configurations and found that running more threads than the number of physical cores per VM caused resource contention and reduced performance. With 100 instances in the dataset (later reduced to 98 after excluding two instances) and a cluster consisting of 15 nodes initially and 14 nodes later (each with 8 physical cores), allocating one physical core per instance was the only configuration that allowed us to maintain full coverage of the dataset during the long-running cold-start campaigns. This ensured that all instances were continuously explored throughout the challenge, maximizing the probability of discovering improvements across the full benchmark set.

In contrast, the runs on the local machines allowed greater flexibility in resource allocation. For these runs, we were able to assign multiple CPU cores to each instance. As a result, we could consistently observe a substantial speedup of the search phase, and improvement cycles during local runs were often significantly faster than during the single-threaded cluster runs.

In general, the architecture of TQrouting benefits strongly from multiple threads, effectively leveraging parallelism. Several core components of the solver—including the decomposition framework and the local search engine—benefit from multi-threading. In practice, we have observed that the algorithm scales very efficiently with the number of available CPU cores. Across multiple existing benchmark datasets and different problem variants, TQrouting's runtime performance improves almost proportionally to the number of threads assigned to an instance. Similar effects were observed during the challenge itself. When running the multi-threaded solver on local machines, we were able to explore neighborhoods more aggressively and complete improvement cycles significantly faster than in the single-threaded cluster configuration. These observations are consistent with our earlier benchmark experiments on classical VRP datasets, where the solver demonstrated strong scalability when additional computational threads were available.

In summary, the single-thread configuration used for the main cluster runs was a practical decision driven by the need to cover all benchmark instances simultaneously with limited hardware resources. However, both prior benchmarking experiments and the local warm-start runs conducted during the challenge confirm that TQrouting's architecture benefits strongly from multi-threading. When additional threads are available, the solver can efficiently utilize them to accelerate the search and accelerate exploration of promising regions of the solution space.

Summary

During the 30-day CVRPLib BKS Challenge, our computational strategy combined long-running cold-start cluster computations with targeted warm-start improvement runs. The main cluster infrastructure executed repeated large-scale cold-start runs across all instances in parallel, while additional warm-start runs on a secondary cluster and local machines refined promising solutions.

Despite infrastructure interruptions and limited computational resources that constrained multi-threading on the main cluster, this combination of large-scale exploration and adaptive improvement runs proved effective. By the end of the competition, the solver achieved **27 best-known solutions** on the final leaderboard, including several of the largest instances in the benchmark set.