

# GreenMalloc: Allocator Optimisation for Industrial Workloads

Aidan Dakhama<sup>1</sup> William B. Langdon<sup>2</sup> Héctor Menéndez<sup>1</sup> Karine Even-Mendoza<sup>1</sup>  
<sup>1</sup>King's College London, <sup>2</sup>University College London

## Introduction

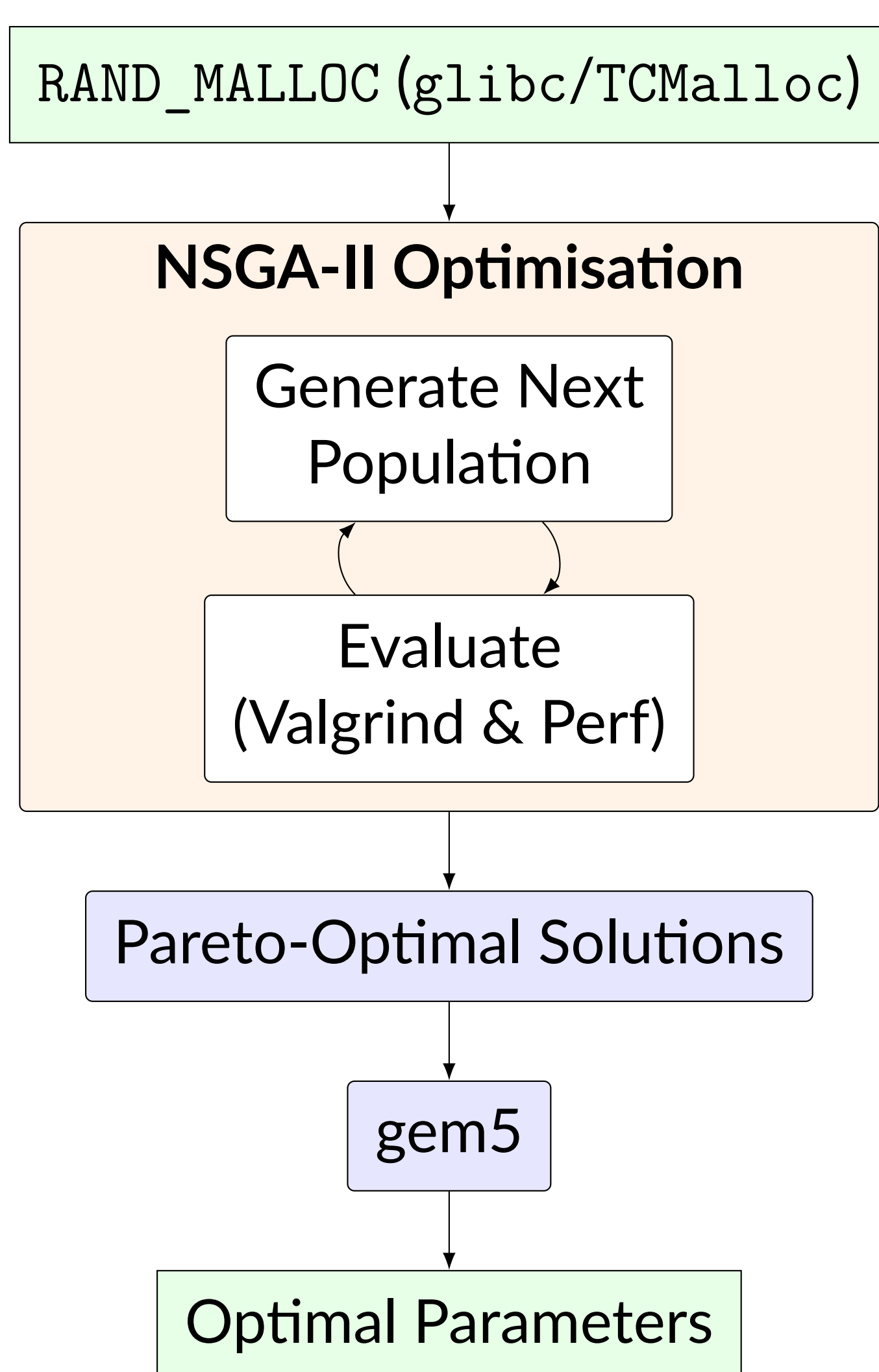
We present GreenMalloc, a multi-objective framework for automatically tuning C++ heap memory allocators (glibc & TCMalloc). Optimising allocators for large-scale, industrially relevant software like gem5 is challenging, as their slow runtimes and complex memory behaviour make direct benchmarking impractical. We address this by using NSGA-II and RAND\_MALLOC for lightweight exploration of allocator parameters. We then transfer the best configurations to gem5 for validation.

## Allocator Optimisation

Efficient memory management is challenging: allocator choice and configuration strongly affect performance. Widely used allocators like GLIBC\_MALLOC and TCMALLOC are hard to tune, so systems often use defaults, wasting memory, energy, and performance.

This is worsened in situations where the target system takes a long time to run, and therefore, a long time to benchmark and improve configurations.

## Overview of Our Approach



We begin with RAND\_MALLOC, a **lightweight benchmark** emulating gem5's memory behaviour. A multi-objective genetic algorithm (**NSGA-II**) optimises **allocator parameters** for glibc and TCMalloc. In this loop, each configuration (individual) is **evaluated** using **Valgrind & Perf** to measure **peak heap usage** and **instruction count**. The algorithm iterates for 500 generations to find the best trade-offs. The resulting **Pareto-optimal configurations** are then validated on gem5 to identify the best-performing parameters.

## Gem5: Our Case Study



### Why Gem5?

**Large:** 1.34 million lines of code, reflecting real system complexity

**Slow-running:** Simulations take substantial time

**Unusual memory behaviour:** Complex allocator interactions and workload-dependent patterns

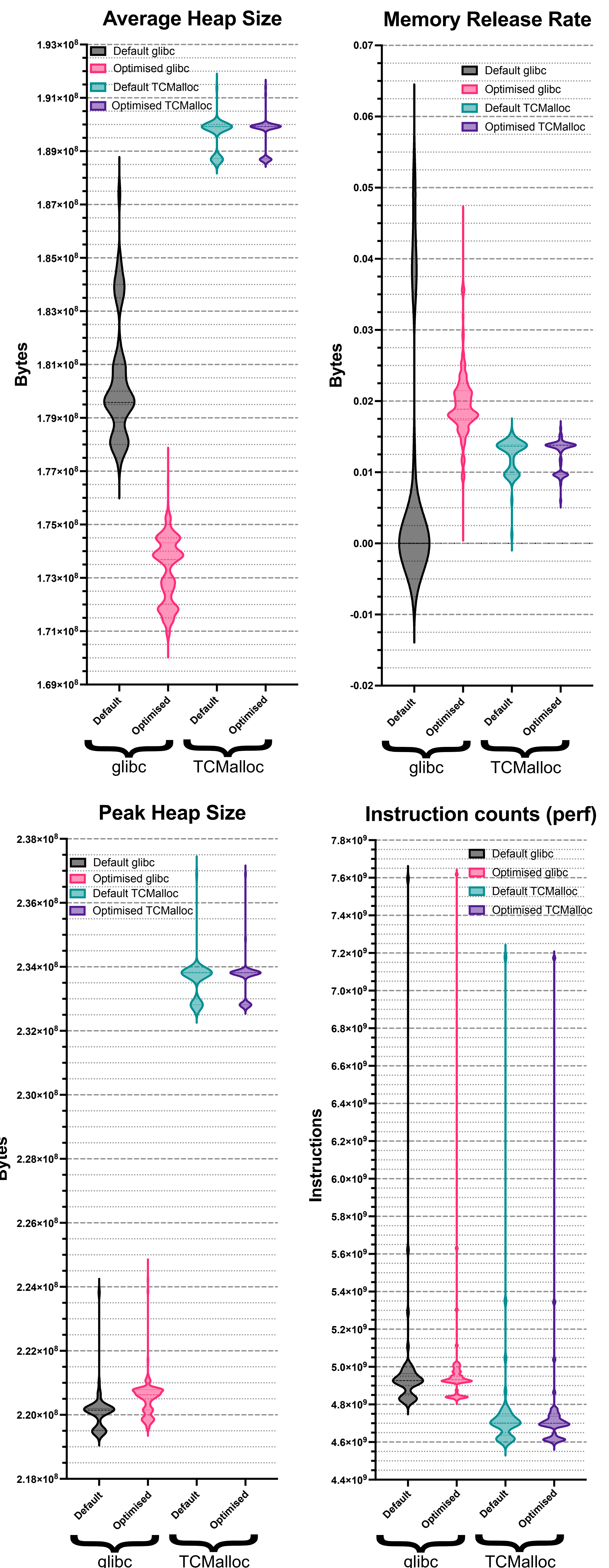
**Industrially relevant:** Used for system development, hardware-software co-design, and computer architecture research in both academia and industry

**Developer(s)** Community contributors  
**Initial release** August 2011; 13 years ago  
**Stable release** v24.1.0.3 / April 11, 2025; 44 days ago  
**Repository** [github.com/gem5/gem5](https://github.com/gem5/gem5)  
**Written in** C++, Python  
**Operating system** Linux, Unix-like  
**License** BSD 3-Clause



Source: Wikipedia (gem5 page, accessed 25/05/2025)

## Results



Allocator trade-offs differ: glibc allows gradual tuning with small changes in instructions and heap, while TCMalloc operates near optimal boundaries, offering higher peak-heap potential but requiring steeper trade-offs. For glibc, tuning **reduces average memory** (180M to 173M) and **instructions** while **improving memory release**. tcmalloc shows stable averages but **better consistency** in memory use and execution. Peak memory remains largely unchanged.

## Conclusions

We introduced GreenMalloc, a search-based framework for optimising memory allocator parameters via lightweight benchmarking. On gem5, it **reduced average heap usage by up to 4.1%** and **instruction counts by up to 0.25%** across glibc and TCMalloc, demonstrating practical gains in efficiency. This approach can extend to other complex software, including full-system simulation, VMs, and interpreters.