Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

UNFOLDING DIFFUSIVE AND REFINEMENT PHASES OF HETEROGENEOUS PERFORMANCE-AWARE RE-PARTITIONING FOR DISTRIBUTED TRAFFIC SIMULATION

Anibal Siguenza-Torres[1,2], Alexander Wieder[2], Stefano Bortoli[2], Margherita Grossi[2], Wentong Cai[3], and Alois Knoll[1]

[1]Department of Informatics, Technical University of Munich, Munich, GERMANY
[2]Intelligent Cloud Technologies Laboratory, Huawei Munich Research Center, Munich, GERMANY
[3]College of Computing and Data Science, Nanyang Technological University, SINGAPORE

ABSTRACT

This work presents substantial improvements to Enhance, a recent approach for graph partitioning in large-scale distributed microscopic traffic simulations, particularly in challenging load-balancing scenarios within heterogeneous computing environments. With a thorough analysis of the diffusive and refinement phases of the Enhance algorithm, we identified orthogonal opportunities for optimizations that markedly improved the quality of the generated partitionings. We validated these improvements using synthetic scenarios, achieving up to a 46.5% reduction in estimated runtime compared to the original algorithm and providing sound reasoning and intuitions to explain the nature and magnitude of the improvements. Finally, we show experimentally that the performance gains observed in the synthetic scenario partially translate into performance gains in the real system.

1 INTRODUCTION

Agent-based microscopic traffic modeling can provide an in-depth analysis of vehicle dynamics by individually modeling each vehicle in the road network. This approach enables urban planners to make informed decisions regarding infrastructure and policy (Zehe et al. 2015; Zehe et al. 2017; Meng et al. 2023). However, simulating microscopic traffic at the city level is computationally demanding, and therefore, distributed computing is used to scale up the simulations (Siguenza-Torres et al. 2024; Xu et al. 2017).

Deploying a simulation on a distributed system naturally raises the challenge of how to distribute the load. A common approach is to divide the computational load by spatially segmenting the road network into smaller sub-regions. Each sub-region and the agents therein are then processed by a dedicated process while exchanging relevant information with the other processes (Potuzak 2022). The performance of such distributed simulations is heavily influenced by how the road network is partitioned into sub-regions (Potuzak 2022; Xu et al. 2014). In the case of time-stepped simulations, the overall pace matches the slowest process in the simulation. Consequently, effective load balancing ensures that the workload is evenly distributed among all the processes, preventing bottlenecks and improving performance. The synchronization between different processes also introduces extra overhead, which can further slow down the simulation. Consequently, good partitioning requires balancing the computational load and minimizing communication costs. This challenge can be framed as a graph partitioning problem, with node weights representing computational load and edge weights reflecting communication overheads (Potuzak 2022). While the problem is NP-hard, several heuristic approaches exist to address it (Valejo et al. 2020).

An extra challenge is provided by the fact that real-world runtime environments are far from the ideal homogeneous distributed cluster. Examples are cloud deployments, where, even within instances of the same type, significant performance variability is observed (Uta et al. 2020). This reality justifies the community's interest in developing methods for executing distributed workloads that effectively cope with resource heterogeneity (Musoles et al. 2019; Xu et al. 2015). In heterogeneous environments, load balancing is influenced by the varying node performance, where the simulation is deployed; this additional level of complexity must be handled explicitly by *heterogeneous performance-aware* partitioning algorithms.

Furthermore, traffic simulations often exhibit variable load patterns (Xu et al. 2014), necessitating runtime partition adjustments for optimal performance. The standard approach is to monitor the workload and re-partition when significant changes occur. However, this can be problematic as partitioning algorithms might produce vastly different partitions even with minor input changes, leading to high costs of redistributing the simulation state (Xu et al. 2014; Schloegel et al. 2001). Re-partitioning algorithms, which modify existing partitions rather than create new ones from scratch, can mitigate this issue (Schloegel et al. 2001).

The recently introduced Enhance algorithm addresses the challenges of distributed traffic simulation by refining initial partitionings using predictive cost models to guide re-partitioning decisions (Siguenza-Torres et al. 2024). Designed primarily for heterogeneous runtime environments, Enhance improves simulation performance by adapting partitions to the computational capabilities of each node. However, a key limitation of Enhance is its tendency to significantly increase communication costs, especially at higher partition counts, which can offset the benefits of improved load balancing. To overcome this, we introduce Enhance++, an enhanced version that generates higher-quality partitionings through several key modifications: incorporating **internal heavy-edge matching** to improve coarsening quality (Sections 4.1 and 5.1), adopting an **edge-centric heuristic** for the diffusive phase (Sections 4.2 and 5.1), and utilizing the **global cost model** as guiding function in the refinement phase (Sections 4.3 and 5.2).

The remainder of this paper is organized as follows. In Section 2, we summarize the state of the art. In Section 3, we summarized the most essential concepts of Enhance. In Section 4, we described the improvements done to the algorithm. In Section 5, we evaluate the proposed improvements over synthetic experiments. In Section 6, we evaluate the improved algorithm on the real system, i.e., distributed CityMoS. We make some final remarks and conclusions in Section 7.

2 RELATED WORK

The partitioning problem is commonly modeled as a weighted graph, where vertices represent computational tasks and edges represent data dependencies. Vertex weights encode workload units, while edge weights reflect data transfer costs. The objective is to find a partitioning P such that the sum of vertex weights is balanced, and edge cut (i.e., the sum of the adjacent edges with vertices in different partitions) is minimized (Karypis and Kumar 1998b). This ensures a balanced workload and reduced communication overhead, improving simulation performance.

Graph partitioning is NP-hard (Valejo et al. 2020), so heuristic methods are typically used. Notable examples include spectral clustering, which is based on the eigenvalues and eigenvectors of the graph Laplacian to identify substructures within the graph (Nascimento and de Carvalho 2011), genetic algorithms (Kim et al. 2011), and the widely adopted *multilevel approach* (Valejo et al. 2020). The latter is among the most effective and extensively used strategies. It involves three main phases: coarsening the graph, performing an initial partition on the smaller graph, and then refining it during uncoarsening (Karypis and Kumar 1998b). Coarsening reduces the graph size by collapsing vertices while preserving its structure, enabling efficient initial partitioning. METIS (Karypis and Kumar 1998b) is one of the most popular multilevel partitioning tools.

The cities' road networks can be naturally represented as graphs, for which the graph partitionings have been commonly employed in distributed simulations (Potuzak 2022). For the case of agent-based microscopic traffic simulations, the computational cost is often modeled as the number of agents on the road, and the communication cost as the number of migrating agents (Xu et al. 2017; Xu et al. 2014; Potuzak 2021). The work on Potuzak (2021) explores genetic algorithms and geographical information for road network partitioning, while a graph-growing technique is used in (Xu et al. 2017).

In Xu et al. (2014), the authors present a dynamic re-partitioning approach using METIS to adapt to changing simulation loads. This scratch-remap method generates new partitionings from the current simulation state and then maps them to the most similar partitions in the previous configuration, aiming to minimize the cost of redistribution. An alternative to scratch-remap partitions is diffusive re-partitioning algorithms. These algorithms take an imbalanced partitioning and move vertices across partitions to balance them (Schloegel et al. 1997; Schloegel et al. 2001). Therefore, the modified partitioning is more similar to the original one, which helps to reduce the re-partition cost for dynamic re-partitioning processes. Diffusive re-partitioning can be combined with the multilevel approach (Schloegel et al. 1997). Multilevel diffusion algorithms often have two phases: a diffusive phase focusing on re-balancing and a refinement phase in which balance and the edge cut are considered (Schloegel et al. 2001).

Previous works often assume homogeneous resources. There has been interest in accounting for heterogeneity in the partitioning problem. For instance, in (Xu and Ammar 2004), heterogeneous communication resources are considered in the partitioning process, using benchmarks to model communication costs. Also, streaming partitioning algorithms (Stanton and Kliot 2012) have been extended to account for resource heterogeneity. In Xu et al. (2015), a heterogeneous performance-aware streaming partitioning algorithm is developed using a physical graph to model a runtime environment. In Zheng et al. (2016), an architecture-aware partitioning algorithm (ARGO) was proposed to exploit modern high-speed networks, prioritizing inter-node communication over intra-node communication. In Musoles et al. (2019), the authors extended the streaming approach to hypergraphs, using calibrated communication cost models and multiple iteration streams to improve partitioning.

3 CORE PRINCIPLES OF ENHANCE

This section presents Enhance's formalism, algorithms, and core principles, which are necessary to understand the improvements proposed in this work. Readers are encouraged to read the original study (Siguenza-Torres et al. 2024) for a deeper understanding of Enhance.

We first establish the graph formalism for this work. Let $G = (V,E)$ denote an undirected graph with vertices V and edges E. Each vertex $v \in V$ corresponds to one road in the road network, and an edge $e = \{v_1, v_2\} \in E$ exists if and only if the roads represented by v_1 and v_2 are connected. We define the vertex *partitioning* of G as disjoint subsets (i.e., *partitions*) of vertices $P = \{V_1,\ldots,V_N\}$, where N denotes the number of partitions.

We call an edge $e = \{v_1, v_2\}$ a *cut edge* if the endpoints are in different partitions (i.e., $v_1 \in V_i \wedge v_2 \in V_j \wedge i \neq j$). We denote the endpoints of a cut edge as *neighbor vertices*. For convenience, we let $E_c(P)$ denote the set of cut edges for a given partitioning P: $E_c(P) = \{e \in E | e \text{ is cut edge}\}$.

For a partitioning P and a vertex $v \in V_i$, we let the partitioning $P^{v\to j}, 1 \leq j \leq N$, denote the partitioning with vertex v re-assigned to partition V_j (and removed from its original partition V_i):

$$P^{v\to j} = \{V_i \setminus v, V_j \cup \{v\}\} \cup \bigcup_{h, h\neq i, h\neq j} \{V_h\}. \tag{1}$$

Also, we define a function to obtain the partition index of a vertex for a partitioning $Part(v) = i$, for $v \in V_i$.

3.1 Cost Models

Enhance leverages the concept of *Cost Models*, functions that predict computation and communication costs using relevant performance features. Vertices hold features related to computational cost, while edges hold those related to communication. This approach effectively *decouples* the graph weights from cost estimation, breaking a common assumption in the literature (Hendrickson and Kolda 2000). Consequently, each vertex v is attached to a feature vector f_v; each edge e is attached to a feature vector f_e.

To predict the computational cost of a particular partition V_i, the features are combined by a specified aggregation function, which results in the partition feature $f_i^v = Agg_v(f_v | v \in V_i)$. The objective of the computational cost model is to take the aggregated featured vector and predict the computation cost measured in wall-clock time $\text{CompCost}_i(f_i^v)$.

One of the advantages of adding the cost model layer is that under the same features, we can predict the computational effort for different runtime environments. With heterogeneous nodes, the separation of features and cost allows to predict the computational effort for each node (or node type) simply by changing the $CompCost_i$ to match the node where the partition V_i is deployed.

Similarly, we define a communication cost model that transforms edge features into the communication cost. Only cut edges contribute to the communication cost; therefore, the aggregation acts under these edges $f^e_{glob} = Agg_e(e|e \in E_c(P))$. The communication cost model is defined as $\mathrm{CommCost}\left(f^e_{glob}\right)$. Combining these models, the Total Predicted Cost (TPC) is defined as the function that, given a partitioning P, will predict the total cost in wall-clock time:

$$\mathrm{TPC}(P) = \max_{1 \le i \le N} \mathrm{CompCost}_i\,(f^v_i) + \mathrm{CommCost}\left(f^e_{glob}\right). \tag{2}$$

We use $\mathrm{TPC}(P)$ as the objective function to minimize during the partitioning process. Given sufficiently accurate cost models, this approach aims to directly reduce the total wall-clock time of the simulation.

3.2 Heuristic for Finding Modification Actions

The core component of Enhance is the heuristic deciding how to modify the current partitioning P shown in Algorithm 1. For a given vertex v, the algorithm finds the partition (considering the neighbor partitions and $Part(v)$ as options) to which v should be assigned to minimize *cost*. The cost is computed by the function *ComputeCost*, and different cost models can be used, as elaborated in Section 3.3. The neighboring partitions are evaluated in (deterministically seeded) random order. The final output is the action that minimizes the cost according to the provided model.

All vertices are processed in random order during an *iteration*, inspired by streaming partitioning from Xu et al. (2015), Stanton and Kliot (2012), the heuristic is applied only to neighboring vertices. Multiple iterations are performed, improving the partitioning until a specified termination criterion, as in Musoles et al. (2019). In Enhance, this criterion follows a diminishing returns approach: If an iteration yields no further improvement in partition quality, the stream terminates.

3.3 Diffusion and Refinement

Like prior multilevel diffusive re-partitioning algorithms, Enhance works in two phases: the diffusive phase and the refinement phase (Schloegel et al. 2001; Schloegel et al. 1997). In the diffusive phase, the emphasis is on moving vertices across partitions to balance the total computation cost even if it results in a higher communication cost (Schloegel et al. 1997). During the refinement phase, the partitioning is modified to consider computation and communication costs. The diffusive phase in Enhance is performed by making a stream using the local (i.e., only considering the subset I of all partitions) Predicted Computational Cost (PPC) as the $CostModel$ in Algorithm 1, defined as:

$$\mathrm{PCC}(P,I) = \max_{i \in I}\left(\mathrm{CompCost}_i\,(f^v_i)\right). \tag{3}$$

When communication costs are ignored, the vertices can be freely moved from heavily loaded partitions to those with lighter loads. Additionally, focusing on *local cost* rather than global cost allows for adjustments that yield immediate local benefits, even if they do not directly optimize the overall cost. This approach enables incremental improvements within the system as local adjustments progressively accumulate to produce global cost reductions over time.

The partitioning resulting from the diffusive phase is further improved during the refinement phase, aiming to reduce the total costs. This implies striking a trade-off between the computation and the communication costs, as a single action can impact these two aspects differently. For instance, moving a vertex may decrease computation but increase communication costs. To account for the total cost during the refinement phase, the *local TPC* is used as a cost model:

$$\mathrm{TPC}(P,I) = \mathrm{PCC}(P,I) + \mathrm{CommCost}\left(f^e_{glob}\right). \tag{4}$$

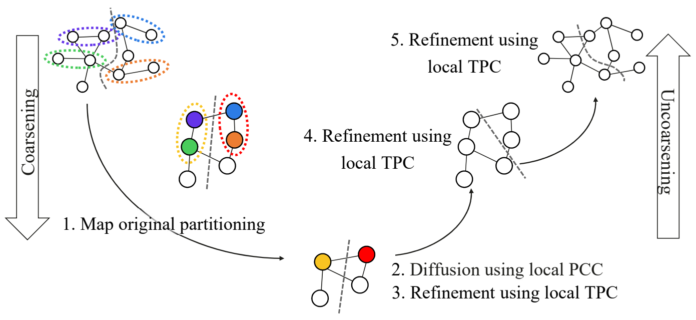
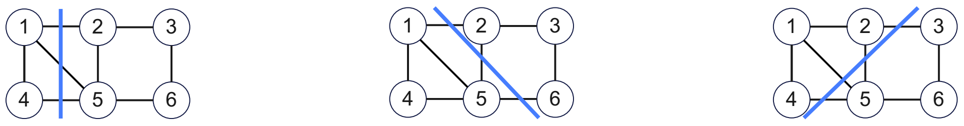

Coarsening
Uncoarsening
1. Map original partitioning
2. Diffusion using local PCC
3. Refinement using local TPC
4. Refinement using local TPC
5. Refinement using local TPC

Figure 1: Summary of the overall Enhance algorithm.



(a) Initial sub-optimal partitioning. (b) Balanced Optimal Partitioning. (c) Balanced Sub-optimal partitioning.

Figure 2: Example graph and partitionings.

Figure 1 illustrates the overall workflow of the Enhance algorithm. The process begins with a multilevel coarsening of the original graph, where the graph is progressively reduced by collapsing groups of vertices into single aggregated vertices. As depicted in the figure, the vertices enclosed within the colored ellipses are merged to form a single vertex at the next coarser level marked with the same color. Afterward, the initial partitioning is mapped to the coarsest level. The diffusive phase takes place only on this level of the graph. After that, the refinement is applied from the coarsest level up to the original graph.

4 ENHANCING ENHANCE

A key limitation of Enhance is that the communication cost of the generated partitioning is higher than the original, which limits its usability in cases with a large N. The main contribution of this work is to enhance Enhance by finding orthogonal optimizations for the diffusive and refinement phases, which exhibit unique behaviors and needs. These are explained in detail in the Sections 4.2 and 4.3. Based on these, we propose improvements for each of these phases and validate them with an experimental evaluation.

4.1 Internal Coarsening

The original Enhance algorithm uses Heavy Edge Matching (HEM) (Karypis and Kumar 1998a) for graph coarsening, a greedy method that prioritizes collapsing the heaviest edges to reduce edge cuts during coarse-level partitioning (Karypis and Kumar 1998b). This matching could occur even between vertices from different partitions, thus altering the original partitioning when mapped into a coarsest level. In this work, we evaluate the effect of *Internal Coarsening* (IC), which restricts matching to vertices within the same partition, thereby preserving the initial structure throughout the coarsening process (Schloegel et al. 1997). We assess the impact of this modification in Section 5.1.

4.2 Diffusion Nature Analysis

Analysis of the diffusion process revealed that it may result in an increased edge cut (i.e., communication cost) of the resulting graph. This outcome was not unexpected, as communication cost is not considered in the diffusion process. However, the implementation of the diffusion process in Enhance may also increase the edge cut *unnecessarily*, posing additional challenges to the subsequent refinement phases.

As an example, consider the graph in Figure 2. For the sake of simplicity, we assume one unit of processing load per vertex and one unit of communication load per edge: $\forall v \in V: f_v = 1$ and $\forall e \in E: f_e = 1$. In this case, the computational cost is $\text{CompCost}_i = |V_i|$ and the communication cost is $\text{CommCost} = |E_c|$. Consider the initial partitioning shown in Figure 2a with computational costs of $\text{CompCost}_l = 2$ and $\text{CompCost}_r = 4$ for the left and right partition, respectively. In this initial partitioning, only two actions lower the computational cost: moving vertex 2 to the left partition and moving vertex 5 to the left partition. Since communication cost is not considered during the diffusion phase, either of these actions results in a perfectly balanced partitioning, depicted in Figures 2c and 2b, respectively. Crucially, when applying the vertex-centric diffusion process from Enhance (Algorithm 1) with vertices traversed in random order, both actions have the same probability (1/2) of being applied, depending on which vertex is explored first. However, one action (moving vertex 2 to the left partition, as depicted in 2c) results in a partitioning that has a larger edge cut, yielding higher total cost, and hence, posing a potentially higher burden on the subsequent refinement phase to *correct* this sub-optimal decision.

However, performing the diffusion process in an edge-centric rather than vertex-centric manner reduces this effect. The edge-centric algorithm is shown in Algorithm 2. It resembles the Kernighan–Lin (KL) heuristic, guided by cost models (Kernighan and Lin 1970). For each cut edge v, v', three choices are evaluated: 1) keep both vertices in their original partitions, 2) move v' into $Part(v)$, and 3) move v into $Part(v')$. When exploring modifications in an edge-centric rather than vertex-centric way, the likelihood of actions increasing the edge cut is reduced.

Algorithm 1 Vertex centric heuristic for v.

```
1: ChooseAction($G, P, v, ComputeCost$):
2: $I \leftarrow \{P(v') | v = v' \vee v' \in \text{adj}(v)\}$
3: $minCost \leftarrow \infty$
4: for each $k$ in I in random order do
5:     $cost \leftarrow ComputeCost(P^{v \to k})$
6:     if $cost < minCost$ then
7:         $minCost \leftarrow cost$
8:         $index \leftarrow k$
9:     end if
10: end for
11: Output: $P^{v \to index}$
```

Algorithm 2 Edge centric heuristic for cut edge e.

```
1: ChooseAction($G, P, e, ComputeCost$):
2: $\{v, v'\} \leftarrow e$
3: $i \leftarrow Part(v)$, $j \leftarrow Part(v')$, $minCost \leftarrow \infty$
4: for each $choice$ in $\{P, P^{v \to j}, P^{v' \to i}\}$ in random order do
5:     $cost \leftarrow ComputeCost(choice)$
6:     if $cost < minCost$ then
7:         $minCost \leftarrow cost$
8:         $selectedP \leftarrow choice$
9:     end if
10: end for
11: Output: $selectedP$
```

For illustration, again consider the graph in Figure 2a with three cut-edges: $\{1,2\}$, $\{1,5\}$, and $\{4,5\}$ in the initial partitioning. When evaluating these cut edges in random order, each has the same probability of being first processed, and the possible actions that reduce the imbalance of the partitioning (moving 2 and moving 5 to the left partition) are identical to the vertex-centric diffusion. However, the overall probability of moving vertex 5 is higher than moving vertex 2. While each cut-edge has the same probability of being evaluated first, vertex 5 is contained in two cut-edges ($\{1,5\}$ and $\{4,5\}$), and exploring either of these first results in vertex 5 being moved to the left partition. On the other hand, vertex 2 is only moved to the left partition if the cut-edge $\{1,2\}$ is explored first. Hence, vertex 5 has a probability of $2/3$ being moved, while vertex 2 has a probability of $1/3$ being moved.

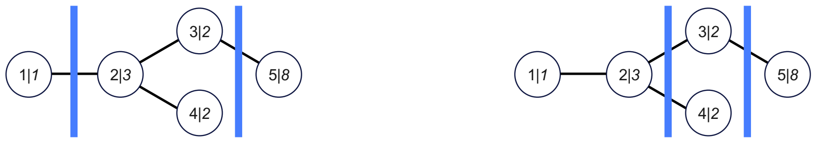
At a high level, the effect of the edge-centric diffusion illustrated above relies on moving vertices adjacent to more cut edges with higher probability than moving vertices adjacent to fewer cut edges. We evaluate the impact of the edge-centric heuristic in Section 5.1 using the following iteration variants.

- **Random Vertex Traversal (RVT)**: All the vertices are traversed in random order using the original vertex-centric heuristic. This is the iterator that was used in the original Enhance algorithm.
- **Random Edge Traversal (RET)**: All the edges are traversed in random order. This iterator allows us to compare the vertex and edge heuristics.

- **Random Initial Cut Edge Traversal (RICE)**: Edges are also traversed in random order, but only cut edges *at the beginning* of the streaming (excluding edges becoming cut edges *during* the streaming) are traversed. This confines partitioning modifications to the initial frontier in each streaming.

4.3 Refinement Nature Analysis

A limitation of the original Enhance refinement phase is its reliance on a local total cost model (Equation 4), which can lead to decisions that inadvertently increase the TPC. While locality is beneficial during diffusion, enabling gradual global improvements through local gains, it may backfire during refinement, where computation and communication costs interact more intricately. For illustration, consider the graph depicted in Figure 3a as an example, where the first number in the vertex label corresponds to the index and the second number directly corresponds to the units of work. Evidently, the simulation bottleneck is the partition on the right, resulting in $max(\text{CompCost}_i) = 8$. Additionally, each cut edge represents a unit of communication cost, thus $\text{CommCost} = 2$. Consequently, the partitioning has a total cost of $\text{TPC} = 10$.



(a) Initial Optimal partitioning. (b) Sub-Optimal partitioning.

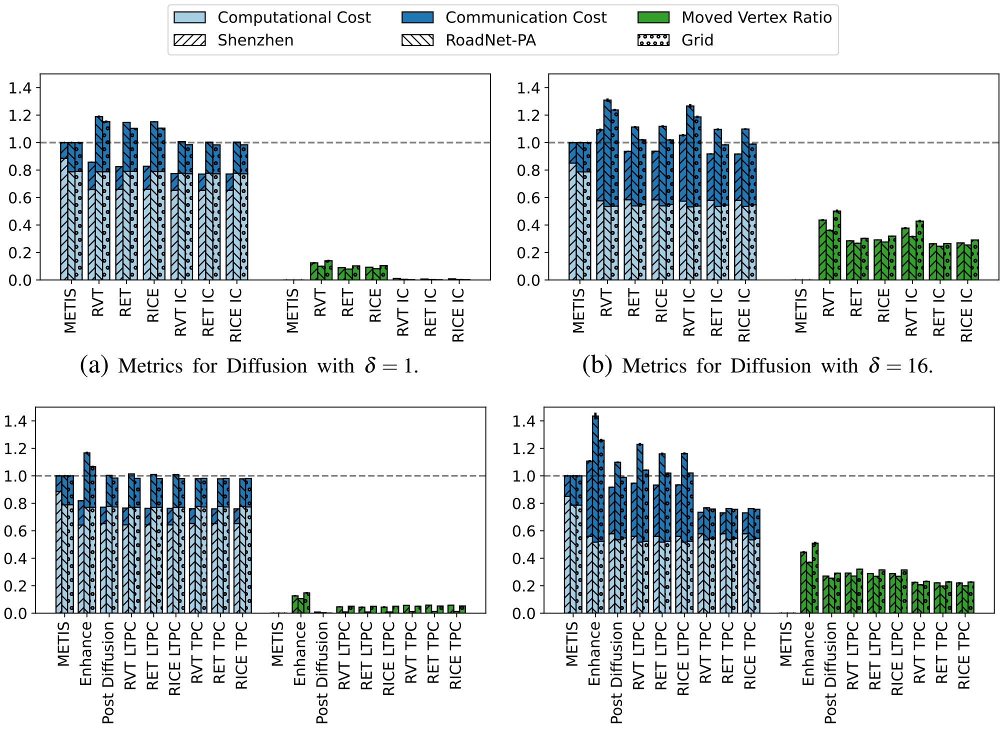
Figure 3: Example graph with three partitions.

Applying a vertex-centric heuristic with local cost (i.e., see 4) on vertex two would move the vertex to the left partition (i.e., $I = \{\text{left}, \text{center}\}$), resulting in a *decrease* of local cost, but an *increase* of global cost. This occurs because the local cost heuristic ignores the computation cost for non-neighboring partitions, such as the right partition. Consequently, it would assume a $\text{TPC}(P,I) = 7 + 2 = 9$ for the partitioning depicted in Figure 3a, and a local cost of $\text{TPC}(P,I) = 4 + 3 = 7$ for the partitioning in Figure 3b. Globally, however, the total TPC increases to 11 because the right partition remains the computational bottleneck, and the additional communication cost resulting from this greedy decision further increases the overall cost. In contrast, using the global TPC model (see 2) ensures all decisions align with minimizing the actual objective, leading to higher-quality partitions. Section 5.2 presents experimental evidence of the improvements this change brings.

5 SYNTHETIC EVALUATIONS

Before evaluating the partitioning in actual simulations, we first evaluated the behavior of the diffusive and refinement phases with synthetic scenarios. We assign simple numerical workloads to the nodes and edges of different graphs and apply METIS and Enhance to generate and refine partitions. This study enables us to evaluate the impact of the improvements to Enhance under controlled conditions and excluding the impact of potential limitations of the cost model used. Additionally, the synthetic evaluation allows considering (hypothetical) homogeneous and heterogeneous hardware configurations without the need for actual access to such deployments. We conducted the evaluation with the following graphs: **Shenzhen**: Simplified road network of Shenzhen, generated with the tool (Meng et al. 2022) consisting of 85,204 vertices and 117,788 edges. Each vertex is a road, and each edge connects roads. This graph is used to test partitioning large-scale traffic simulations. **RoadNet-PA**: Road network of Pennsylvania with 1,088,092 vertices and 1,541,898 edges (Leskovec et al. 2009). Each vertex corresponds to an intersection or endpoint, and each edge corresponds to a road connecting them. **Grid**: This is a uniform grid road network with 90 by 90 intersections and bidirectional roads with 3 lanes each generated using SUMO's `netgenerate`. The graph representation contains $159,120$ vertices and $254,160$ edges.

We assume one unit of processing load per vertex and one unit of communication load per edge for these graphs (i.e, $\forall v \in V: \ f_v = 1$ and $\forall e \in E: \ f_e = 1$). The computational cost is defined as the load



(a) Metrics for Diffusion with $\delta = 1$. (b) Metrics for Diffusion with $\delta = 16$.

(c) Metrics for Refinement with $\delta = 1$. (d) Metrics for Refinement with $\delta = 16$.

Figure 4: Normalized cost breakdown for the diffusion (a) and (b), as well as refinement (c) and (d) phases for a homogeneous (left-hand side) and heterogeneous (right-hand side) case.

divided by a performance constant: $CompCost_i = |V_i|/C_i$, where C_i reflects the relative speed of node i. A higher C_i indicates a faster machine, reducing cost, while a lower C_i corresponds to slower performance and higher cost. Communication cost is scaled as $CommCost(f_e) = \beta f_e$, with $\beta = 0.01$ for Shenzhen and Grid, and 0.1 for RoadNet-PA. These values result in communication costs between 10–20% of the total cost for $N = 256$ (with this partition size the original Enhance struggled to outperform the original partitioning). We simulate heterogeneity via δ, the number of unique C_i values. For $\delta = 1$, all nodes are equal ($C_i = 1$). For $\delta > 1$, C_i values are linearly spaced in $[1,2]$, e.g., for $\delta = 4$: $C_1 = 1$, $C_2 = 1.\bar{3}$, $C_3 = 1.\bar{6}$, $C_4 = 2$. This models a heterogeneous cluster with varying node speeds.

5.1 Diffusive Phase Analysis

Figure 4 summarizes the average results of applying *Enhance* with 30 random seeds to re-partition the same initial METIS partitionings. On the left-hand side of the figure, we present the breakdown of computation and communication costs. Each group of columns shows the result for the three different graphs in a particular configuration. These costs are normalized by the total initial cost of the METIS partitioning, which is also shown in the first group of columns as a reference. The right-hand side displays the vertex migration ratio, with 0 indicating no changes and 1 meaning all vertices were reassigned.

The top row shows results after the diffusion phase for homogeneous ($\delta = 1$) and heterogeneous ($\delta = 16$) settings. In this phase, we tested the three aforementioned iterators in Section 4.2: RVT, RET, and RICE as well as the effect of using Internal Coarsening (IC), as described in Section 4.1. As Section 3.3 explains, only computational cost is considered during the diffusive phase. Thus, the main objective here is to reduce computational cost. This effect is evident in the figures, particularly in the heterogeneous case, where migrating work from slower to faster nodes yields a significant reduction in this component. For instance, in the RoadNet-PA graph, the computational cost drops from 0.79 to 0.54 across all the iterators.

Table 1: Geometrical mean evolution of the metrics with the introduced improvements.

Normalized TPC	Moved Vertices Ratio	Description
1.0	0.0	Original METIS baseline
1.126	0.234	**Enhance**: Local RVT diffusion - Local RVT refinement
1.034	0.181	Local RICE diffusion - Local RVT refinement
0.987	0.091	Internal coarsening - Local RICE diffusion - Local RVT refinement
0.824	0.085	**Enhance++**: Internal coarsening - Local RICE diffusion - Global RVT refinement

Nevertheless, the communication cost can increase significantly during this phase. For example, in the case of the RVT iterator used in the original Enhance, the communication cost increases from 0.21 to 0.77. This is significantly reduced with the edge-centric iterators, as discussed in Section 4.2. By operating on edges instead of vertices, these iterators increase the likelihood of moving highly connected vertices together without directly considering communication cost. This results in a lower communication overhead; for instance, both RET and RICE achieve a communication cost of 0.57. This pattern holds consistently across all three graphs evaluated in this study.

IC further reduces communication costs and vertex movement, particularly in homogeneous cases where METIS already produced balanced loads. For instance, in Shenzhen, RICE moves 0.09 of vertices, while RICE IC moves only 0.008, with similar computational costs ($\sim$0.65). By preserving the original partitioning structure during coarsening, the diffusion phase only needs to relocate a small fraction of vertices to achieve balance.

In conclusion, both the edge-centric iterators and the IC strategy improve the overall quality of the diffusion phase. No statistically significant difference was observed between the RET with IC and RICE with IC variants; therefore, we proceed with only the RICE variant in subsequent experiments.

5.2 Refinement Phase Analysis

The second row of Figure 4 shows the refinement phase results, applied after RICE diffusion and using IC. These figures include the initial METIS partitioning as a reference in the first group of columns, the original end-to-end Enhance results in the second group, and the post-diffusion state in the third group.

As explained in Section 3.3, the objective of the refinement phase is to directly minimize the TPC by balancing communication and computation costs. Originally, the Enhance algorithm employed the Local TPC (LTPC), as defined in 4. However, this model does not consistently lead to improved re-partition quality. For example, in the RoadNet-PA graph with $\delta = 16$, the TPC increases after refinement from 1.10 (post-diffusion) to 1.22, 1.16, and 1.16 using the RVT, RET, and RICE LTPC iterators, respectively. This is contrary to the goal of the refinement phase. We address this by using the global TPC (Equation 2) during refinement, leading to significant improvements. In the same example, TPC drops from 1.10 to 0.76 far outperforming the original Enhance (TPC = 1.43), a 46.5% reduction.

The improvements achieved by the enhanced version of Enhance are consistent across all configurations. However, in homogeneous cases like RoadNet-PA and Grid, gains over METIS are small due to its already effective balancing. Since no significant differences were found among TPC iterator variants, we use the RVT TPC variant in subsequent experiments.

5.3 Summary of Enhance++

This section presents an end-to-end evaluation of each improvement made to the original Enhance algorithm. As in the previous section, we generate partitions of size $N = 256$ using the three synthetic graphs under two heterogeneity settings: $\delta = 1$ (homogeneous) and $\delta = 16$ (heterogeneous), with 30 different random seeds for each configuration. We report the geometric mean of normalized TPC and moved vertex ratio, commonly used in benchmarking normalized metrics. As shown in Table 1, at this partition size, the original Enhance algorithm increased the total predicted cost reducing the performance. Each improvement from Sections 5.1 and 5.2 consistently reduced TPC from 1.126 to 0.824 and the moved vertex ratio from 0.234 to 0.085. We refer to the improved version as **Enhance++**.

6 CITYMOS EVALUATIONS

To evaluate the impact of the improvements in a real distributed simulation, we used Distributed CityMoS (Siguenza-Torres et al. 2024) on a heterogeneous cluster composed of two machine types, *fast* and *slow*. Each fast machine is equipped with two Intel(R) Xeon(R) E5-2697 v4 processors with 18C/36T, and each slow machine is equipped with two Intel(R) Xeon(R) E5-2680 v3 processors with 12C/24T. All machines are connected by a 10 Gbps network.

We consider homogeneous and heterogeneous setups:

- Homogeneous: only slow machines, one process per CPU, 16 simulator threads per CPU;
- Heterogeneous $\delta = 4$: to emulate a higher degree of hardware heterogeneity, we vary the number of threads available to each node in the simulation. Specifically, the simulation assigns resources in a round-robin fashion: a slow CPU with 16 threads, a fast CPU with 16 threads, a slow CPU with 8 threads, and a fast CPU with 8 threads.

The cost models used for the algorithm were the same as in (Siguenza-Torres et al. 2024). The computational cost is modeled as a linear combination of *Vehicle Count* and *Average Active Lanes* with coefficients calibrated for each combination of machine type and number of simulator threads. Communication cost is similarly modeled as a linear function of the *Number of Migrations* and a calibrated coefficient.

We use a grid road network to run the simulations, also used in Section 5. The traffic scenario consists of a single traffic wave with $350,000$ concurrent agents at peak, each with a randomly selected origin and destination. Before partitioning experiments, we ran the simulation once to generate vertex and edge feature vectors (e.g., average number of agents per road, average number of migrations), which were used to derive weights for METIS.

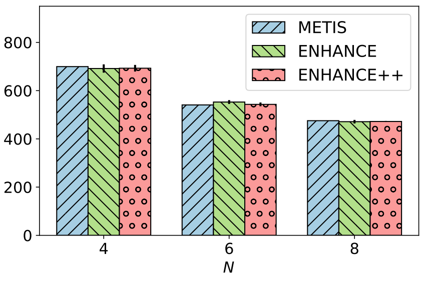
We generated 10 re-partitionings using different random seeds for both Enhance and Enhance++ and evaluated their impact by running CityMoS with the original METIS partitioning as well as with the newly generated re-partitionings. Figure 5 reports the average wall-clock time of the simulation under two deployment configurations.

In the homogeneous configuration, the performance remained consistent across all partitionings, demonstrating that Enhance++ does not degrade performance when the initial METIS partitioning is already well-balanced. In contrast, in the heterogeneous configuration, both Enhance and Enhance++ resulted in noticeable improvements over the baseline METIS partitioning.

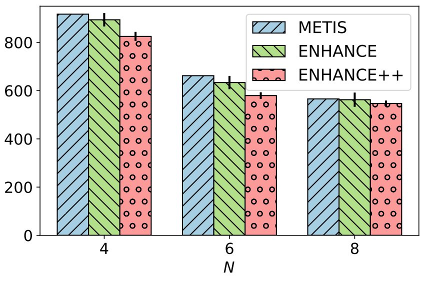
The largest improvement was observed for $N = 6$: the total simulation time was reduced by 4.3% using Enhance and by 12.5% using Enhance++. In the case of $N = 4$, the reductions were 2.5% and 10.0%, respectively. The aforementioned results are statistically significant with $p = 0.05$. For $N = 8$, the improvements in wall-clock time were not statistically significant, although Enhance++ still produced lower and more consistent wall-clock time, as reflected by its reduced standard deviation. These results suggest that while METIS performs adequately under homogeneous conditions, re-partitioning with Enhance++ is more effective in heterogeneous environments.

7 CONCLUSIONS

Multilevel graph partitioning remains a foundational and widely used technique in large-scale systems. A key finding of this work is that even when enhanced with domain-specific cost models, the effectiveness of such approaches is highly sensitive to what might appear as minor implementation choices. Crucial factors such as the nature of the cost model (e.g., local versus global, TPC versus PCC) and the processing perspective (vertex versus edge centric) significantly impact the final outcome. Our study reveals that these factors influence the diffusive and refinement phases in fundamentally different ways, an insight that enabled us to design orthogonal, phase-specific optimizations and ultimately develop Enhance++.



(a) Homogeneous Infrastructure.



(b) Heterogeneous ($\delta = 4$) Infrastructure.

Figure 5: Average total cost of running CityMoS.

Through comprehensive experiments on both synthetic graphs and a distributed traffic simulator, we demonstrated that Enhance++ consistently produces higher-quality partitionings. Although graph partitioning is an inherently hard problem, this work illustrates that substantial gains are still achievable through empirically driven enhancements. Enhance++ advances the state of heuristic partitioning by revealing and exploiting subtle yet impactful design choices, paving the way for more effective and practical distributed systems.

REFERENCES

Hendrickson, B., and T. G. Kolda. 2000. "Graph Partitioning Models for Parallel Computing". *Parallel Computing* 26(12):1519–1534.

Karypis, G., and V. Kumar. 1998a. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". *SIAM Journal on Scientific Computing* 20(1):359–392.

Karypis, G., and V. Kumar. 1998b, Nov. "Multilevel Algorithms for Multi-Constraint Graph Partitioning". In *SC '98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*, 7$^{\text{th}}$–13$^{\text{th}}$ Nov, Orlando, FL, USA, 28–28.

Kernighan, B. W., and S. Lin. 1970. "An Efficient Heuristic Procedure for Partitioning Graphs". *The Bell System Technical Journal* 49(2):291–307.

Kim, J., I. Hwang, Y.-H. Kim, and B.-R. Moon. 2011. "Genetic Approaches for Graph Partitioning: a Survey". In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, 12$^{\text{th}}$–16$^{\text{th}}$ Jul, Dublin, Ireland, 473–480.

Leskovec, J., K. J. Lang, A. Dasgupta, and M. W. Mahoney. 2009. "Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters". *Internet Mathematics* 6(1):29–123.

Meng, Z., X. Du, P. Sottovia, D. Foroni, C. Axenie, A. Wieder, *et al*. 2022, May. "Topology-Preserving Simplification of OpenStreetMap Network Data for Large-scale Simulation in SUMO". In *SUMO User Conference 2022 (SUMO 2022)*, 9$^{\text{th}}$–11$^{\text{th}}$ May 2022, Virtual Event.

Meng, Z., A. Siguenza-Torres, M. Gao, M. Grossi, A. Wieder, X. Du, *et al*. 2023. "Towards Discrete-Event, Aggregating, and Relational Control Interfaces for Traffic Simulation". In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '23, 21$^{\text{th}}$–23$^{\text{th}}$ June, Orlando, FL, USA, 12–22.

Musoles, C. F., D. Coca, and P. Richmond. 2019. "HyperPRAW: Architecture-Aware Hypergraph Restreaming Partition to Improve Performance of Parallel Applications Running on High Performance Computing Systems". In *Proceedings of the 48th International Conference on Parallel Processing*, 5$^{\text{th}}$–8$^{\text{th}}$ Aug, Kyoto, Japan, 1–10.

Nascimento, M. C., and A. C. de Carvalho. 2011. "Spectral Methods for Graph Clustering – A Survey". *European Journal of Operational Research* 211(2):221–231.

Potuzak, T. 2021. "Improved Road Traffic Network Division based on Genetic Algorithm and Graph Coarsening". In *2021 14th International Conference on Human System Interaction (HSI)*, 8$^{\text{th}}$–10$^{\text{th}}$ Jul, Gdansk, Poland, 1–8.

Potuzak, T. 2022. "Current Trends in Road Traffic Network Division for Distributed or Parallel Road Traffic Simulation". In *2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 26$^{\text{th}}$–28$^{\text{th}}$ Sep, Ales, France, 77–86.

Schloegel, K., G. Karypis, and V. Kumar. 1997. "Multilevel Diffusion Schemes for Repartitioning of Adaptive Meshes". *Journal of Parallel and Distributed Computing* 47(2):109–124.

Schloegel, K., G. Karypis, and V. Kumar. 2001. "Wavefront Diffusion and LMSR: Algorithms for Dynamic Repartitioning of Adaptive Meshes". *IEEE Transactions on Parallel and Distributed Systems* 12(5):451–466.

Siguenza-Torres, A., A. Wieder, Z. Meng, S. Narvaez Rivas, M. Gao, M. Grossi, *et al*. 2024, jun. “ENHANCE: Multilevel Heterogeneous Performance-Aware Re-Partitioning Algorithm For Microscopic Vehicle Traffic Simulation”. *ACM Transactions on Modeling and Computer Simulation*.

Stanton, I., and G. Kliot. 2012. “Streaming Graph Partitioning for Large Distributed Graphs”. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 12th–16th, Beijing, China, 1222–1230.

Uta, A., A. Custura, D. Duplyakin, I. Jimenez, J. Rellermeyer, C. Maltzahn, *et al*. 2020. “Is Big Data Performance Reproducible in Modern Cloud Networks?”. In *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, 25th–27th Feb, Santa Clara, CA, USA, 513–528.

Valejo, A., V. Ferreira, R. Fabbri, M. C. F. d. Oliveira, and A. d. A. Lopes. 2020, apr. “A Critical Survey of the Multilevel Method in Complex Networks”. *ACM Computing Surveys* 53(2).

Xu, D., and M. Ammar. 2004. “BencHMAP: Benchmark-Based, Hardware and Model-Aware Partitioning for Parallel and Distributed Network Simulation”. In *The IEEE Computer Society’s 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings.*, 8th Oct, Volendam, Netherlands, 455–463.

Xu, N., B. Cui, L. Chen, Z. Huang, and Y. Shao. 2015. “Heterogeneous Environment Aware Streaming Graph Partitioning”. *IEEE Transactions on Knowledge and Data Engineering* 27(6):1560–1572.

Xu, Y., W. Cai, H. Aydt, and M. Lees. 2014. “Efficient Graph-Based Dynamic Load-Balancing for Parallel Large-Scale Agent-Based Traffic Simulation”. In *Proceedings of the Winter Simulation Conference 2014*, 3483–3494 https://doi.org/10.1109/WSC.2014.7020180.

Xu, Y., W. Cai, D. Eckhoff, S. Nair, and A. Knoll. 2017. “A Graph Partitioning Algorithm for Parallel Agent-Based Road Traffic Simulation”. In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 24th–26th May, Singapore, Republic of Singapore, 209–219.

Zehe, D., A. Knoll, W. Cai, and H. Aydt. 2015. “SEMSim Cloud Service: Large-scale Urban Systems Simulation in the Cloud”. *Simulation Modelling Practice and Theory* 58:157–171.

Zehe, D., S. Nair, A. Knoll, and D. Eckhoff. 2017. “Towards CityMoS: A Coupled City-Scale Mobility Simulation Framework”. *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication* 2017:6th–7th Apr, Erlangen, Germany, 26–28.

Zheng, A., A. Labrinidis, P. K. Chrysanthis, and J. Lange. 2016. “Argo: Architecture-Aware Graph Partitioning”. In *2016 IEEE International Conference on Big Data (Big Data)*, 5th–8th Dec, Washington, DC, USA, 284–293.

AUTHOR BIOGRAPHIES

ANIBAL SIGUENZA-TORRES is a Ph.D. student at the Technical University of Munich and Huawei Munich Research Center. His work is centered on distributed large-scale microscopic traffic simulations. His email address is anibal.siguenza@tum.de.

ALEXANDER WIEDER is a research engineer at the Huawei Munich Research Center. His research interests include distributed systems, vehicular traffic simulation, and performance optimization. His email address is alexander.wieder@huawei.com.

STEFANO BORTOLI is a principal research engineer and chief architect at Huawei Munich Research Center. His research interests are traffic optimization, traffic simulations, and in general high-performance simulations. His email address is stefano.bortoli@huawei.com.

MARGHERITA GROSSI is a principal research engineer in applied AI and ML at the Huawei Munich Research Center. Her expertise includes AI-driven solutions in telecommunications and transportation. Her email address is margherita.grossi@huawei.com.

WENTONG CAI is a professor in the College of Computing and Data Science (CCDS) at Nanyang Technological University (NTU), Singapore. He is a senior member of the IEEE and a member of the ACM. He is currently the Editor-in-Chief of the ACM Transactions on Modeling and Computer Simulation (TOMACS) and an editorial board member of the Journal of Simulation (JOS). His research interests are in the areas of Modeling and Simulation, and Parallel and Distributed Computing. His email address is aswtcai@ntu.edu.sg.

ALOIS KNOLL is a professor of Computer Science at the Technical University of Munich (TUM). He received his diploma (M.Sc.) degree in Electrical/Communications Engineering from the University of Stuttgart and his Ph.D. degree in Computer Science from the Technical University of Berlin. His email address is knoll@mytum.de.