ENHANCE: Multilevel Heterogeneous Performance-Aware Re-Partitioning Algorithm For Microscopic Vehicle Traffic Simulation

ANIBAL SIGUENZA-TORRES, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany
ALEXANDER WIEDER, Huawei Munich Research Center, Munich, Germany
ZHUOXIAO MENG, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany
SANTIAGO NARVAEZ RIVAS, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany
MINGYUE GAO, Huawei Munich Research Center, Munich, Germany
MARGHERITA GROSSI, Huawei Munich Research Center, Munich, Germany
XIAORUI DU, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany
STEFANO BORTOLI, Huawei Munich Research Center, Munich, Germany
WENTONG CAI, Nanyang Technological University, Singapore, Singapore
ALOIS KNOLL, Department of Informatics Informatics, Technical University of Munich, Garching, Germany

Driven by our work on a large-scale distributed microscopic road traffic simulator, we present ENHANCE, a novel re-partitioning approach that allows incorporating fine-grained simulator-specific cost models into the partitioning process to account for the actual performance characteristics of the simulator.

The use of explicit cost models enables partitioning for heterogeneous resources, which are a common occurrence in cloud deployments. Importantly, ENHANCE can be used in conjunction with other partitioning approaches by further *enhancing* partitions according to provided cost models. We demonstrate the benefits of our approach in an experimental evaluation showing performance improvements of up to 29% against METIS under heterogeneous conditions. Taking a

Authors' Contact Information: Anibal Siguenza-Torres, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany; e-mail: anibal.siguenza1@gmail.com; Alexander Wieder, Huawei Munich Research Center, Munich, Germany; e-mail: alexander.wieder@huawei.com; Zhuoxiao Meng, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany; e-mail: zhuoxiao.meng@huawei.com; Santiago Narvaez Rivas, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany; e-mail: narvaez@in.tum.de; Mingyue Gao, Huawei Munich Research Center, Munich, Germany; e-mail: mingyue.gao@huawei.com; Margherita Grossi, Huawei Munich Research Center, Munich, Germany; e-mail: margherita.grossi@huawei.com; Xiaorui Du, Huawei Munich Research Center, Munich, Germany and Informatics, Technical University of Munich Department of Informatics, Garching, Germany; e-mail: xiaorui.du@huawei.com; Stefano Bortoli, Huawei Munich Research Center, Munich, Germany; e-mail: stefano.bortoli@huawei.com; Wentong Cai, Nanyang Technological University, Singapore, Singapore, Singapore; e-mail: aswtcai@ntu.edu.sg; Alois Knoll, Department of Informatics Informatics, Technical University of Munich, Garching, Germany; e-mail: knoll@mytum.de.

ACM 1558-1195/2024/6-ART
https://doi.org/10.1145/3670401

different perspective, the partitioning produced by ENHANCE can provide similar performance as METIS, but using up to 20% fewer resources.

CCS Concepts: • **Computing methodologies** → **Distributed simulation**; **Agent / discrete models**; **Massively parallel and high-performance simulations**.

Additional Key Words and Phrases: Microscopic Traffic Simulation, Partitioning algorithm, Road network partitioning, Agent-Based Simulation

1 INTRODUCTION

Agent-based microscopic vehicle traffic simulations make possible to study with high level of detail the dynamics of the vehicle traffic. This enables city planners to take decisions on infrastructure and policies [28, 29]. At large scale, these simulations are compute intensive, and distributed computing is one way to scale up agent-based microscopic vehicle traffic simulations [15, 23, 26, 27] . The most common approach is to distribute the workload into different Logical Processes (LPs) by spatial decomposition of the road network into sub-regions. Each LP handles the agents within a sub-region while exchanging relevant information with the LPs working on neighboring sub-regions [18]. This process is called inter-process communication.

The partitioning, i.e., the division of the road network into sub-regions, may have significant impact on the simulation performance [16, 18, 23, 26]. In this work, we study the problem of partitioning the road network for high performance with a distributed, step-synchronized, microscopic traffic simulator. In this context, the performance is dictated by the slowest LP and the communication overhead. Hence, to achieve high performance, the partitions need to have a good load balancing while simultaneously minimizing communication cost. A common approach is to cast the problem into a graph partitioning problem, encoding the computational cost in the node weights and the communication cost in the edge weights of the road network graph [18]. This problem is NP-hard, but multiple heuristics have been presented.

For agent-based microscopic traffic simulations, computation and communication costs are commonly modeled as numbers of agents (that need to be processed or moved from one node to another) to compute the graph weights [5, 16, 23, 26]. Despite its prevalence, this approach bears two main limitations. First, this approach implies a simplistic cost model in which cost is directly proportional to the number of agents, which may not reflect actual performance characteristics of a given simulator. Second, this approach typically assumes homogeneous resources, which might not be the case in practice. For instance, cloud resources may exhibit heterogeneous performance characteristics both in terms of computation and communication [1, 2, 25]. Both of these limitations may lead to partitions hampering performance.

To make matters worse, microscopic traffic simulations often exhibit variable load patterns (just like traffic itself) [5, 26], which requires to adjust the partitions during run-time to yield high performance. The straightforward approach for coping with variable load is to monitor load, and whenever it significantly changes, create a new partitioning adapting to the new conditions. In general, however, this approach bears additional challenges for the partitioning technique because partitioning algorithms may produce very different partitions even when the input differs only marginally. In this case, the cost of re-distributing major parts of the simulation state among nodes may dominate any potential benefits from adapting to load [12, 19]. Avoiding this effect requires partitioning algorithms that can operate in an *incremental* fashion. That is, rather than producing partitions from scratch, an incremental partitioning algorithm also takes as input an existing partitioning and aims to derive a new one by modifying the provided partitioning [19].

In light of the aforementioned considerations, we believe that the state of the art approaches are not suitable for the effective load balancing in the execution of large-scale traffic simulation in heterogeneous performance environments. In fact, we argue that to achieve maximum performance under these conditions, the partitioning algorithm must account for the inherent heterogeneity of the run-time environment in which the simulation runs.

In this paper we propose a multilevel diffusive partitioning algorithm (ENHANCE) which employs a cost model directly derived from performance measurements. Given a partitioning, the cost model predicts the wall clock time the simulation will take. Starting from an initial partitioning, we improve upon it to generate a partitioning which minimizes the predicted total wall-clock time. To the best of our knowledge, this is the first heterogeneous performance-aware diffusive partitioning algorithm in the context of agent based microscopic traffic simulation.

The remainder of the paper is organized as follows. In Section 2, we provide a summary of the state of the art in road network and heterogeneous-aware partitioning. In Section 3, we explain our motivation and our contribution. In Section 4, we introduce our approach for creating cost models suitable for partitioning. In Section 5, we describe a partitioning algorithm consulting these cost models. In Section 6, we explain how the CityMoS cost models were created. In Section 7, we provide experimental evaluation results for distributed CityMoS. Finally, in Section 8, we highlight directions for future work and conclusions.

2 RELATED WORK

2.1 Partitioning Algorithms for Road Networks

Road networks can naturally be represented as graphs, for which the partitioning problem has been studied for decades in many different fields. In the context of parallel and distributed simulation, partitioning has been used to optimize the performance of the simulations. The computational workload can naturally be expressed as vertices and the data dependencies can be expressed as edges [4]. When the partitioning is done, the workload can be evenly distributed to be processed by the different LPs in parallel, while minimizing the edge cut diminishes the dependencies of the workload, and thus the overhead required to synchronize the LPs. This is a convenient and simple representation of the simulations, and it has been used in many different simulation fields. For instance, it is used in the Computational Fluid Dynamics (CFD) simulation for flows in unstructured meshes [3]. It is also used to improve the performance of parallel optimistic simulations based on the Time Warp synchronization protocol [21]. Additionally, it is employed on Spiking Neural Network (SNN) simulations [14], to mention a few.

One of the most widely used approaches to graph partitioning is the multilevel approach [16, 21, 22]. Such algorithms operate in three phases: graph coarsening, initial partitioning, and graph uncoarsening [21]. In the coarsening phase, the original graph is *coarsened* by collapsing vertices to produce a smaller graph with a similar structure. This process is repeated multiple times, creating multiple levels of coarsening. Then, an initial partitioning is computed for the coarsest level graph. The initial partitioning is then mapped to the next finer level and further improved using various heuristics, many of which are inspired by the Kernighan–Lin (KL) heuristic [9]. The resulting partitions are then projected onto the subsequent finer coarsening level, where the process is reiterated until partitions for the original graph are achieved. This method operates across multiple levels of coarsening, hence the term *multilevel* approach. One of the most commonly utilized multilevel partitioning program in the literature is METIS [8]. METIS is a multilevel partitioning algorithm which takes an undirected graph with weighted vertices and edges, and aims to produce balanced partitions while minimizing the edge cut between partitions. It uses the k-way heuristic to greedily decide to move vertices across boundaries to improve the edge-cut and balance until the local minimum is reached.

In the context of road network partitioning for agent-based vehicle traffic simulations, computational cost is often modeled as the number of agents on a given road, and communication cost is modeled as the number of *migrating* (i.e., moving from one road to another one) agents [5, 15, 23, 26, 27]. For cases where the number of agents is not known a priori, Wei et al. propose to use the origin-destination (OD) matrix of agents to predict their routes, and thus the quantities to be used as vertex and edge weights [23]. The weighted graph is then partitioned using METIS.

Potuzak presented methodologies for using genetic algorithms in a multilevel way and leveraging geographical information in addition to graph weights for road network partitioning [15–17]. Another notable approach presented by Xu et al. relies on graph-growing [27].

While all of the previous work described above aims to compute *static* partitions, Xu et al. present an algorithm for leveraging METIS also for dynamic re-partitioning in response to variable load [26]. To dynamically change parts of the road network between partitions, it is necessary to exchange the relevant information, which introduces a cost of repartitioning that depends on the amount of information to be exchanged. Their approach is a scratch-remap method, meaning they use the current state of the simulation to create a new METIS partitioning. The newly generated partitions are not guaranteed to be similar to the current partitioning, therefore they find the best match of the current partition to the new one in order to minimize the re-partition cost.

2.2 Heterogeneous Performance-Aware Partitioning Algorithms

The works above assume homogeneous resources. For the case of homogeneous computational resources, but heterogeneous communication resources, Xu and Ammar presented a static partitioning algorithm considering two different communication channels (e.g., shared memory and network) [24]. This setting is of particular interest for nodes with multiple sockets, where communication between sockets and between nodes can have very different performance characteristics. The approach considers communication cost models calibrated with benchmark scenarios.

Kiefer et al. explore the possibility of incorporating a more realistic *cost model* into the partitioning process by modifying the METIS algorithm in order to use a non-linear cost model [10]. This is based on the fact that the computational load may not scale linearly due to various factors, such as the contention of shared resources. Therefore, the standard assumption that the computation load is the sum of the weights of the vertices does not represent the actual computation cost when the simulations are run on the hardware. As a result, the load balancing is no longer done over the linear sum of vertex weights, but over a modified cost model based on the sum of weights plus a penalty factor.

Streaming partitioning algorithms have also been extended to account for resource heterogeneity. *Streaming partitioning* approaches (e.g., [20]) greedily iterate over all graph vertices in a given order (e.g., random, breadth first, depth first) and use heuristics to decide the partition for each vertex. Once the vertex is assigned to a partition it will not be moved any further. The work by Xu et al. extended this approach in order to generate heterogeneous performance-a ware partitioning algorithm, which requires, in addition to the graph to be partitioned, a *physical graph* that models the run-time environment including the nodes, their computational capacity, and the communication channels between them along with their performance characteristics [25]. The physical graph is then used by the heuristics to decide the partition for each vertex in a streaming fashion. Notably, the rather fine-grained modeling of resource heterogeneity assumed by this approach allows the direct use of job completion time as an optimization objective (assuming a bulk-synchronous parallel processing model).

Another interesting work was done by Zheng et al. where they developed an architecture-aware partitioning algorithm (ARGO) to exploit the bandwidth of modern high-speed networks [30]. They found that one limiting factor in communication performance was the contention due to the sharing of L2 caches between multi-cores nodes. This, along with the use of Remote Direct Memory Access (RDMA) technology, created a context in which inter-node communication under the network was prioritized over the communication within the same machine. Using the same streaming partitioning model as in [20], they found a heuristic able to offload the contention among the cores in the same node, and obtain a speedup of up to 12x. Fernandez Musoles et al. extended the approach to work with hyper-graphs and focus on communication, considering calibrated communication cost models for each pair of nodes during the streaming partitioning process [13]. Additionally, they used multiple iterations of streams until a stopping criterion is met.

3 BACKGROUND AND MOTIVATION

3.1 Formal Definitions

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 exactly 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 subsets of vertices $P = \{V_1, \ldots, V_{\mathcal{P}}\}$ with $V = \cup_{1 \leq i \leq \mathcal{P}} V_i$ and $i \neq j \Leftrightarrow V_i \cap V_j = \emptyset$, where $\mathcal{P}$ 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:

$$E_c(P) = \left\{ e = \{v_1, v_2\} \in E | v_1 \in V_i \wedge v_2 \in V_j \wedge i \neq j \right\}. \tag{1}$$

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

$$\forall P, v, i, j \text{ such that } v \in V_i, 1 \leq j \leq \mathcal{P}, i \neq j: \quad P^{v \to j} = \left\{ V_i \setminus v, V_j \cup \{v\} \right\} \cup \bigcup_{h, h \neq i, h \neq j} \{V_h\} \tag{2}$$

3.2 Distributed CityMoS

This research was sparked by our work with distributed CityMoS, a parallel agent-based microscopic traffic simulator used for mobility studies[29]. CityMoS employs shared-memory thread-based parallelism using OpenMP. Distributed CityMoS additionally scales out by distributing the workload among multiple LPs, each in charge of exactly one sub-region of the road network.

Communication and coordination among LPs are performed via MPI and the simulation is step-synchronized such that at any time all LPs are in the same logical simulation time. Explicit communication is required in two cases: agent *migration* and *remote-sensing*. Migration is the process of transferring an agent from one LP to another one as it crosses the boundary between the respective sub-regions. Remote-sensing is the process of exchanging information required to compute the local agent state that is depending on agents managed by another LP (i.e., letting agents *see* other nearby agents in a different sub-region). Distributed CityMoS employs latency hiding techniques to reduce the impact of communication on overall simulation performance.

3.3 Shortcomings of Established Partitioning Approaches

3.3.1 Fine-Grained Cost Models. State of the art partitioning algorithms rely on vertex and edge weights of a graph to directly encode the *cost* of processing a vertex or edge. Nevertheless, the implicit direct correspondence between *cost* and *weight* often does not allow to accurately encode an application's run-time behavior.

As an example, consider CityMoS where the computational cost for conducting a simulation –unsurprisingly– directly depends on the number of agents in a simulation. However, the computational cost also depends on the number of lanes occupied by the agents (i.e., how the agents are distributed across the road network). This dependency is depicted in Figure 1, which plots the computational cost when simulating the same number of agents while varying the number of average active lanes (with two different CPUs). While the reasons for this effect are implementation specific, this example illustrates the limitations of state of the art implicit cost models: the actual cost *depends* on the number of agents and number of average active lanes, but these components do not have to be *balanced*. In fact, it is well possible that the best overall balance (i.e., total computational cost) among nodes is achieved with a very uneven distribution of agents and active lanes among nodes. The direct correspondence between weights and cost, however, does not allow considering such behavior when partitioning.

3.3.2 Resource Heterogeneity. Another common assumption in the state of the art is the homogeneous processing capacity among the nodes processing the partitions. In reality, however, nodes may exhibit heterogeneous performance profiles for several reasons. Particularly in cloud deployments, customers commonly have limited means to control the concrete hardware, location and co-location with other workloads. In fact, prior studies have found significant performance heterogeneity in cloud deployments [1, 2, 25].

As an example, consider again Figure 1 showing a significant difference in computational cost with two different CPU models that may well be used within the same instance type (depending on the cloud provider). Neglecting such heterogeneity as part of the partitioning process may leave performance potentials unexploited.

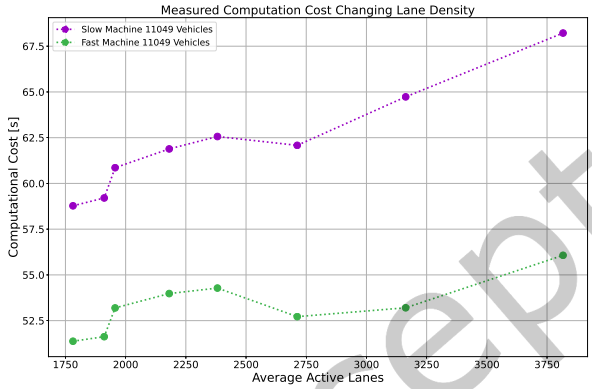


Fig. 1. Computational cost (wall-clock) when simulating the same number of agents while varying the average number of active lanes, measured with two different CPU models (*fast*: Intel(R) Xeon(R) CPU E5-2697 v4 with 36 threads, *slow*: Intel(R) Xeon(R) CPU E5-2680 v3 with 24 threads.)

3.4 Contributions

We identified three inherent limitations of prior partitioning approaches: lack of support for heterogeneous resources, lack of support for finer-grained application-specific cost models, and lack of suitability for dynamic re-partitioning. The key contribution of this work is the development of a partitioning approach that seeks to address aforementioned limitations to obtain higher performance in face of heterogeneous resources and simulators exhibiting performance behavior not captured by the (implicit) cost model assumed by prior state of the art. As for re-partitioning, this paper focuses only on the algorithm; the policy to trigger a re-partitioning and the required data for the algorithm is a challenge by itself, and thus is out of scope for the presented work.

The aforementioned approach for partitioning road networks was implemented for use with distributed CityMoS. Our prototype

(1) Supports fine-grained cost models by changing the vertex and edge weights for features and incorporating simulator specific cost models further explained in Section 4.1.
(2) Supports heterogeneous resources by allowing the use of per-partition cost models, explained in Section 4.2.
(3) Allows for re-partitioning by iteratively changing an existing partitioning further explained in Section 5.6.

We demonstrate the performance benefits of our approach over METIS in an experimental evaluation.

4 MODELING COST FOR PARTITIONING

4.1 Decoupling Cost and Weights

As elaborated in Section 3.3.1, the tight coupling between (vertex and edge) weights in the state of the art may impede capturing performance characteristics relevant for partitioning. We break this coupling by replacing graph weights with *features* that, in contrast to weights, do not directly correspond to the cost, but form the input to a cost model estimating the actual cost from the provided features.

As an example, consider the feature vector $f = (132, 1)$ for a given road, where the first element denotes the number of agents on that road and the second one denotes the number of lanes used on that road. While the number of agents has an obvious relation to the processing cost, the number of used lanes may also contribute to cost independently of the number of agents for implementation specific reasons (e.g., for updating usage statistics for each used lane).

Formally, we attach a feature vector f to each vertex and each edge in the graph to be partitioned:

$$\forall v \in V: \quad f_v = \left(f_{v,1}, \ldots, f_{v,\eta_v}\right) \in \mathcal{F}_v \tag{3}$$

$$\forall e \in E: \quad f_e = \left(f_{e,1}, \ldots, f_{e,\eta_e}\right) \in \mathcal{F}_e \tag{4}$$

Here, η_e and η_v denote the number of dimensions of the feature vectors for edges and vertices, respectively, and $\mathcal{F}_e$ and $\mathcal{F}_v$ denote the feature spaces for edges and vertices, respectively. Note that the features are defined for vertices and edges, but not for partitions (including potentially multiple vertices and edges). Before reasoning about the cost of partitions, we first define *aggregation functions* for feature vectors. For a set of feature vectors F, we define an aggregation function $\mathrm{Agg}(F): \mathcal{F}^* \mapsto \mathcal{F}$ that produces a single feature vector containing the features required to estimate the total cost when processing the respective elements (edges or vertices). The resulted feature vector belongs to the original feature vector space, and thus it maintains the dimensions of the input feature vectors.

As an example, consider again the feature vector $f = (132, 1)$ and additionally the vector $f' = (217, 3)$. Both features can be naturally aggregated by addition (the number of agents and number of lanes used results from the element-wise sum of these to vectors) to obtain the aggregated feature vector $f'' = (349, 4)$. The obtained vector f'' contains the information to estimate the computation cost to process two vertices with the feature vectors above. However, note that the aggregation function may use other operations for aggregation depending on the semantics of the features. For instance, consider the type of road being contained in the feature vector and the cost depending on the number of unique road types in a partition. In this case, the aggregation function may compute the unique elements in the set of road types for that partition.

Since vertices and edges typically represent distinct aspects in the graph to be partitioned (e.g., roads and connections between them), we require two aggregation functions for feature vectors, for vertices and edges, respectively:

$$\mathrm{Agg}_v(F_v): \quad \mathcal{F}_v^* \mapsto \mathcal{F}_v \tag{5}$$

$$\mathrm{Agg}_e(F_e): \quad \mathcal{F}_e^* \mapsto \mathcal{F}_e \tag{6}$$

Since aggregation functions can be used to compute a single feature vector from a set of feature vectors, we can define cost models operating on aggregated single feature vectors. We distinguish between *computation* and *communication* costs:

$$\mathrm{CompCost}(f_v): \quad \mathcal{F}_v \mapsto \mathcal{R}_{\geq 0} \tag{7}$$

$$\mathrm{CommCost}(f_e): \quad \mathcal{F}_e \mapsto \mathcal{R}_{\geq 0} \tag{8}$$

$\mathrm{CompCost}(f_v)$ estimates the cost (in terms of time) of processing a *computation load* characterized by the vertex feature vector f_v (resulting from aggregating the features of the vertices in a graph partition V_i). $\mathrm{CommCost}(f_e)$

estimates the total communication cost for a *communication load* characterized by the feature vector obtained by aggregating all cut edges in a partitioning P.

Concrete implementations of $\text{CompCost}(V_i)$ may aggregate the feature vectors of the vertices within V_i using $\text{Agg}_v(F_v)$ and then estimate the cost from the resulting feature vector. In contrast, concrete implementations of $\text{CommCost}(P)$ may only consider the set of cut edges $E_c(P)$ resulting from partitioning P (as other edges do not result in inter-process communication), aggregate them using $\text{Agg}_e(F_e)$ and estimate the total communication cost from the resulting feature vector. Note that, however, concrete implementations have to be tailored to the performance characteristics of a given simulator.

In the definition above, the cost model implicitly assumed homogeneous resources, since the node type is not considered. Next, we describe how to incorporate heterogeneity into the cost models.

4.2 Accounting for Resource Heterogeneity

The cost models as defined above, $\text{CompCost}(V_i)$ and $\text{CommCost}(P)$, estimate costs *regardless* of node types, implicitly assuming a single type (i.e., homogeneous resources). In practice, however, different nodes may exhibit different performance characteristics for a variety of reasons, many of which are beyond the control of customers in cloud deployments. Since we decoupled graph weights from cost, however, incorporating heterogeneous resources is straightforward: models estimating the cost from a feature vector can be established *for each node* rather than globally.

Formally, without loss of generality, we assume a *node* as a processing unit responsible for one partition; the approach can be generalized to cores, sockets, or other granularities. Let $N = \{n_1, \ldots, n_{\boldsymbol{N}}\}$ be the set of nodes, potentially each exhibiting different performance characteristics. For the sake of simplicity, but without loss of generality, we assume $\boldsymbol{N} = \mathcal{P}$ and that each node n_i is responsible for partition V_i. Rather than using a single cost model for computational cost, $\text{CompCost}(V_i)$, we use one cost model per node, $\text{CompCost}_i(V_i), n_i \in N$, reflecting the performance characteristics of the respective node n_i. For example, consider again Figure 1, which illustrates that two different nodes (with different CPUs) may have different processing costs for the same aggregate features (i.e., number of average active lanes and number of agents). Per-node cost models can be created by simulation specific profiling and model regression techniques. Analogously, resource heterogeneity can be incorporated for the communication cost model.

Next, we describe how to use the cost models to formulate an objective function driving the partitioning process

4.3 Objective Function for Partitioning

In general, the objective function is composed by two elements: the computation and the communication. To obtain the computation of the partition i, the feature vectors of a partition need to be aggregated via $f_i^v = \text{Agg}_v(\{f_v | v \in V_i\})$ which will be transformed to cost by the computational cost model of that partition. We can formulate one component of the objective function, namely, the computation cost per node:

$$\text{CompCost}_i\left(f_i^v\right). \tag{9}$$

The second component we consider in the objective function is the communication cost. Since only cut-edges contribute to inter-process communication, with the aggregated features of the edges $f_{glob}^e = \text{Agg}_e\left(\{f_e | e \in E_c(P)\}\right)$ we can estimate the total communication cost for a given partitioning P as:

$$\text{CommCost}\left(f_{glob}^e\right). \tag{10}$$

With both computation and communication costs, we can formulate the *total predicted cost* (TPC) of a partitioning P:

$$\mathrm{TPC}(P) = \max_{1 \leq i \leq \mathcal{P}} \mathrm{CompCost}_i \left(f_i^v\right) + \mathrm{CommCost}\left(f_{glob}^e\right). \tag{11}$$

We use $\mathrm{TPC}(P)$ as an objective function to minimize as part of the partitioning process. The formulation of $\mathrm{TPC}(P)$ above implies a model where computation and communication are never overlapped and where all communication is handled by a central instance. Note that this does *not* directly apply to distributed CityMoS. However, for the sake of simplicity and since the development of cost models *exactly* reflecting all implementation aspects of distributed CityMoS is beyond the scope of this work, we use the formulation of $\mathrm{TPC}(P)$ above as objective function to be minimized as part of partitioning. All results reported in this paper were obtained with this objective function.

For convenience, we also define $\mathrm{TPC}(P, I)$ a variant of $\mathrm{TPC}(P)$ that only considers partitions with indices I in the evaluation of the computational cost:

$$\mathrm{TPC}(P, I) = \max_{i \in I} \mathrm{CompCost}_i \left(f_i^v\right) + \mathrm{CommCost}\left(f_{glob}^e\right). \tag{12}$$

Next, we detail how we employ this objective function in an effective partitioning algorithm.

5 MULTILEVEL STREAM HETEROGENEOUS PERFORMANCE-AWARE PARTITIONING ALGORITHM

5.1 Overview

On a very high level, our partitioning approach, ENHANCE, works by taking an initial partitioning as an input, and modifying the partitioning until a termination criterion is satisfied. This approach bears multiple challenges: obtaining the initial partitioning, defining the modification actions to perform, deciding the actions to take, reducing the risk of getting stuck in local optima, efficiently coping with potentially large graphs, and coping with the high number of evaluations of the cost model when making decisions. In the following, we describe how we address each of these challenges.

First, we describe how to obtain an initial partitioning for our approach.

5.2 Obtaining an Initial Partitioning

As we point out above, ENHANCE takes an initial partitioning as input rather than computing a partitioning from scratch. This naturally leads to the question of how to obtain that, and in our case the answer is straightforward: we suggest the use of partitioning algorithms presented in prior work (see Section 2) that produce the best results for the given application. As such, ENHANCE does not directly *compete* with other partitioning algorithms, but rather *enhances* them by utilizing them and further improving their results. If ENHANCE was to guarantee optimality, the choice of an initial partitioning would be arbitrary, since convergence to an optimal partitioning would be guaranteed regardless of the choice. However, since ENHANCE does not guarantee optimality (which is inherent to approaches avoiding the inherent complexity of the underlying partitioning problem), the choice of the initial partitioning may have significant impact in two dimensions: first, starting from a *good* partitioning may require fewer iterations by ENHANCE to converge to a *better* partitioning (compared with a less performing initial partitioning). Second, ENHANCE may find *better* partitioning when starting from *good* initial partitioning that would not be found when using another initial partitioning. The choice to assume an initial partitioning to be used by ENHANCE was deliberate: it enables us to stand on the shoulders of giants by leveraging the plethora of prior work on partitioning by incorporating it, and further improving the result.

Next, we describe how we *modify* the partitions, starting from the initially provided one as well as during run-time.

5.3 Incremental Modifications of Partitions

ENHANCE incrementally modifies the partition through a sequence of *actions*, where each action re-assigns a vertex from one partition to another one. More formally, we let $\text{adj}(v)$ denote the set of vertices a given vertex v (i.e. $\text{adj}(v) = \{v' \in V | \exists e = \{v, v'\} \in E\}$,) and we let $\text{P}(v)$ denote the index of the partition containing v in partitioning P (i.e. $\text{P}(v) = i : \ 1 \leq i \leq \mathcal{P} \wedge v \in V_i$)

As mentioned above, given a neighbor vertex v (i.e. a vertex connected to a cut edge, see Section 3.1), we consider the actions of moving the vertex to each of its neighboring partitions (or not moving it). For each one of such actions we evaluate a provided cost model to determine the respective *local cost*, defined as the cost only considering the neighbor partitions while ignoring the others.

The use of the *local cost* rather than the global cost allows considering actions providing local benefit even though they may not necessarily improve upon the global cost (yet). The process somewhat resembles natural diffusive processes. As an example, consider gas diffusion: gas diffuses *locally* as long there is a local difference in concentration *regardless* of higher or lower concentrations further away. Importantly, this diffusive process ultimately strives to even out the concentrations globally. Similarly, considering local cost enables taking actions providing only local benefits, which eventually accumulate to global cost benefits.

The pseudocode is depicted in Algorithm 1. It takes as input the graph G, the current partitioning P, the current neighbor vertex v to operate on, and the cost model H. Different cost models H can be employed to take the decision, and we will clarify the cost model selection in detail in Section 5.5. The **ChooseAction** algorithm returns the index of the partition where v has to be assigned to in order to yield in minimal local cost. Note that this function may return the index of the partition v is already assigned to, in which case the action is actually to not re-assign v to a different partition. Next, we detail how this logic is employed by ENHANCE in a *streaming* fashion for re-partitioning.

Algorithm 1 Choosing an action for vertex v

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

5.4 Streaming Re-Partitioning

ENHANCE takes a greedy *streaming* approach to determine the sequence of actions to apply for re-partitioning. The key idea of streaming is to traverse the vertices of the graph in a given order (e.g., random), choose an action for each vertex, apply it, and repeat until a termination criterion T is met [20, 25]. Our prototype processes neighbor vertices (since other vertices do not have a neighborhood partition to assign them to) in random order, a common choice also taken, for example, in [25]. We leave the analysis of other ordering schemes with their impact on both convergence time and quality of partitioning found to future work. The pseudocode of this

approach is depicted in Algorithm 2. The termination criterion $T(G, P, H)$ is a binary predicate indicating if the current partitions should be refined in another pass over the neighbor vertices.

Algorithm 2 Streaming Re-Partitioning Algorithm

```
1:  Repartition(G, P, H, T):
2:  while not T(G, P, H) do
3:      for each vertex v in V in random order do
4:          if v is neighbor vertex then
5:              i ← ChooseAction(G, P, v, H)
6:              P ← P^{v→i}
7:          end if
8:      end for
9:  end while
10: Output: P
```

Each neighbor vertex is considered to be re-assigned to a different partition at most once in each iteration, similar to approaches taken in [20, 25]. Potentially multiple iterations are performed (until termination criterion is met) as in [13].

In Algorithms 1 and 2 we did not specify the cost model H used. In fact, ENHANCE currently employs two cost models, and we next describe the concrete cost models and how they are used.

5.5 Multiple Cost Models

ENHANCE greedily modifies partitions by sequentially performing *locally* beneficial actions and, as such, is potentially prone to reach locally optimal partitions that cannot be further modified (without tracing back or taking other actions) to reach the global optimum. This effect is inherent to approaches that do not guarantee optimality and cannot be completely ruled out without facing the hardness of the underlying partitioning problem. However, ENHANCE employs two techniques to reduce the risk of converging to *dead ends*. First, we use two different cost models, each encapsulating slightly different aspects of the performance characteristics. Second, we use *graph coarsening* which we detail in Section 5.6.

For CityMoS, we employ two cost models: the first is the objective function $\mathrm{TPC}(P, I)$ of equation 12 (described in Section 4.3) that accounts for both computation and communication cost. The second cost model, $\mathrm{PCC}(P, I)$ (*predicted computation cost*), is a *reduced* one that only accounts for the computation cost:

$$\mathrm{PCC}(P, I) = \max_{i \in I} \mathrm{CompCost}_i \left(f_i^v \right). \tag{13}$$

We use PCC in addition to TPC because *always* accounting for communication cost (included in TPC) may lead to avoidable local optima: as an example, consider a group of highly connected vertices on partition V_1 and a weakly utilized partition V_2. Re-assigning any single vertex from the group from V_1 to V_2 may not be beneficial because the computational cost on V_1 is only marginally reduced, and the additional communication cost incurred by splitting the group may result in higher overall cost (and hence, lower performance). Moving the whole group of vertices from V_1 to V_2 at once, however, may have resulted in lower overall cost, but it would not be done since vertices are processed individually. In this situation, using PCC instead of TPC may have allowed moving all or some of the vertices of the group from V_1 to V_2, avoiding this local optimum.

For each cost model, we use a *diminishing returns* termination criterion, that is, the iterative process stops as soon as one iteration did not yield cost improvements. Both cost models are used in different stages of algorithm, which we describe next.

5.6 Multilevel Re-Partitioning

The multilevel optimization is a widely used approach for partitioning and minimization [22]. The approach consists of reducing the complexity of a graph by coarsening the network into multiple levels. The re-partitioning process performed by ENHANCE, as described above, operates on the level of individual vertices that are re-assigned between partitions. In case the initial partitioning is *far away* (i.e., differs a lot) from a better-performing partitioning, this may require a potentially large number of individual modification actions. Intuitively, this could be reduced by taking fewer *bigger* steps, rather than many smaller ones. Following this intuition, we employ *graph coarsening* as a way to both increase the rate of convergence to *good* partitionings, as well as to reduce the risk of converging to local optima.

Formally, let **coarsen**(G^k) denote a *coarsening function* that, given a graph G^k with a *coarsening level* k, produces another graph G^{k+1} that it is smaller or equal size of G^k (in terms of vertices) and maintains a similar structure. For the coarsening level k, we assume that higher levels result in coarser (i.e., smaller) graphs, that is $i > j \Leftrightarrow |G^i| \leq |G^j|$. We further assume a function **mapPartitioning**(G, P, G^k) that produces a partitioning P' for a coarsened graph G^k given a partitioning P for the original graph G. For convenience, without loss of generality, we assume $G^1 = G$. The used **mapPartitioning** in our case is simply the statistical mode of the vertices that are collapsed from level G to G^k. Notice that the vertex v_i^k of level G^k is the result of collapsing one or more vertices from G. Therefore $P(v_i^k) = Mode(\{P(v_j)|v_j \in collapsedVertices\})$.

Algorithm 3 Multi-Level Partitioning using Graph Coarsening

1: **MultilevelRepartition**(G, P, k):
2: $(G^1, \ldots, G^k) = \big(G, \textbf{coarsen}(G), \ldots, \textbf{coarsen}(G^{k-1})\big)$
3: $P' \leftarrow$ **mapPartitioning**(G, P, G^k)
4: $P' \leftarrow$ **Repartition**(G^k, P', PCC, T)
5: **for** each coarsening level i from k down to 1 **do**
6: $\quad P' \leftarrow$ **Repartition**(G^i, P', TPC, T)
7: **end for**
8: **Output:** P'

The coarsening of the graph is performed using the Heavy Edge Matching (HEM) [7] with the number of migrations as weight for the edges. The HEM is an efficient method to match the heaviest edges together in a greedy fashion. This ensures that the edge cut on the coarsened graph is diminished by a significant quantity [8]. To maintain the original properties of the graph on the coarser representations of it, we are simply aggregating the vertex features of the collapsed vertices. The edge features are kept unchanged, unless more than one collapsed vertex has an edge to another vertex, then the edges features are also aggregated.

Algorithm 3 depicts the pseudocode for incremental multilevel re-partitioning. In line 2, we first compute the coarsened graphs for each level. Then the initial partitioning is mapped to the coarsest graph (line 3) obtained in the previous step. We then refine the coarsest graph using the cost model (PCC) considering only computation cost (line 4), and then refine the partitioning using the complete cost model (TPC) for each coarsening level, starting with the coarsest one (line 6). Note that in the final iteration ($i = 1$) the re-partitioning is performed on the original graph ($G^1 = G$) and the resulting partitioning P' is returned.

Note that the above the approach as described so far may require many (for each coarsening level and for each iteration $O(|E|)$) —potentially expensive— evaluations of the cost model. Next, we describe how we can evaluate the cost model *incrementally*.

5.7 Incremental Cost Model Evaluation

In ENHANCE, the cost models used are evaluated each time potential partitioning modification actions are assessed (see Algorithm 1). Although the cost models used throughout this work are not very complex, the cost for one evaluation of the cost model (including the aggregation function) can still be critical for overall performance of ENHANCE due to the high number of evaluations. To reduce the evaluation cost, we next describe how to perform the calculations in the cost models we used in an incremental fashion. The underlying intuition of computing the cost incrementally is that individual partitioning modification actions (i.e., re-assigning one vertex) often do not invalidate already computed cost components for the partitioning before that modification. For instance, consider a vertex v being re-assigned from partition 1 to 2. This action certainly affects the total computation cost on partitions 1 and 2, but does not affect the computation cost on other partitions, so they can be re-used.

Formally, we next describe how to efficiently *update* the result of the cost models PCC and TPC for a given partitioning P in response to a partitioning modification action (i.e., re-assigning a vertex) to obtain the result for the modified partitioning. Note that the following technique for incrementally evaluating the cost models assumes that the aggregation functions used, $\text{Agg}_v(F_v)$ and $\text{Agg}_e(F_e)$, aggregate by element-wise addition of the feature vectors. Other aggregation functions may require different approaches or cannot be evaluated in an incremental fashion at all.

From the previous example, recall that re-assigning a single vertex only affects the computation cost in the origin and destination partitions, while others are not affected. More precisely, for a partitioning P and vertex v with feature vector f_v being re-assigned from partition V_i to V_j, the aggregate feature vectors f_i^v and f_j^v can be *updated* rather than recomputed with $f_i^v \leftarrow f_i^v - f_v$ and $f_j^v \leftarrow f_j^v + f_v$. The feature vector f_v of the vertex being re-assigned is subtracted from the aggregate feature vector for the origin partition V_i and added to the aggregate feature vector for the destination partition V_j, while leaving other aggregate feature vectors unchanged. In both PCC and TPC, the computation cost for partitions other than i and j can be directly cached and re-used. For partitions i and j, the respective aggregate feature vectors can be efficiently updated as described above, and then the cost model has to be re-evaluated for the updated aggregate feature vectors.

The TPC cost model additionally accounts for the communication cost which depends on the partitioning cut edges. Similar to computation cost, the aggregate feature vector f_{glob}^e used for the communication cost can be incrementally updated in response to re-assigning a vertex v with feature vector f_v from partition V_i to V_j (i.e., $(v \rightarrow j)$). This action can affect an edge e in two ways: e can become a cut edge when it was not a cut edge before, or e was a cut edge and can become a non-cutting edge, otherwise e is not affected by the action. We formally define the sets of new cutting edges as $\text{NCE}(P, v, i) = E_c(P^{v \rightarrow j}) \setminus E_c(P)$, and new non-cutting edges as $\text{NNCE}(P, v, i) = E_c(P) \setminus E_c(P^{v \rightarrow j})$.

Note that both $\text{NCE}(P, v, i)$ and $\text{NNCE}(P, v, i)$ can be directly incrementally computed in a straightforward way: both sets can only contain edges adjacent to the vertex being re-assigned v, and each edge e adjacent to v can be easily checked to determine if e should be added to $\text{NCE}(P, v, i)$ or $\text{NNCE}(P, v, i)$ (or none of them).

Given $\text{NCE}(P, v, i)$ and $\text{NNCE}(P, v, i)$, the aggregate feature vector for the cut edges f_{glob}^e can be incrementally updated as follows:

$$f_{glob}^e \leftarrow f_{glob}^e + \text{Agg}_e(f_e | e \in \text{NCE}) - \text{Agg}_e(f_e | e \in \text{NNCE}). \tag{14}$$

With f_{glob}^e being updated, the actual communication cost has to be re-computed via $\text{CommCost}(f_{glob}^e)$.

The incremental update of the respective aggregate feature vectors may improve the performance of ENHANCE significantly. However, the concrete benefits depend on the structure of the underlying graph and the cost models used. Next, we present the results of an experimental evaluation of ENHANCE.

6 CITYMOS COST MODEL CREATION

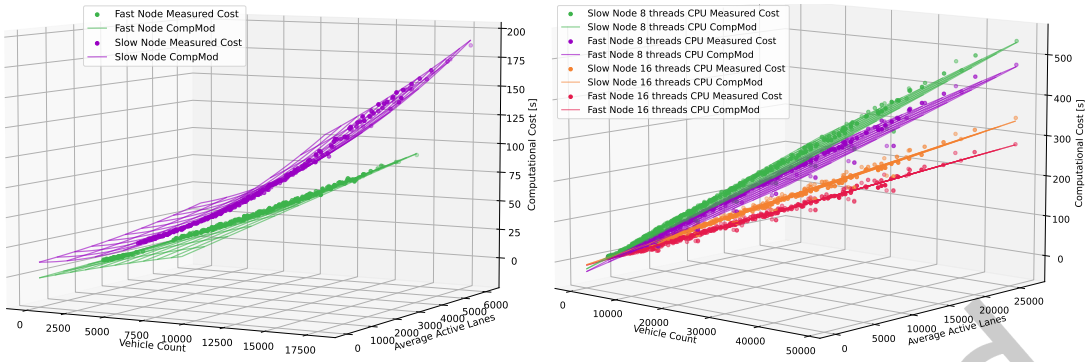
In order to use ENHANCE to improve the CityMoS performance, we first needed to create ad-hoc cost models to predict the *CompCost* and *CommCost* for the simulations. The experiments of section 7 were performed in two different run time environments, which have different performance characteristics, therefore a previous step to create the partitionings was to create the cost model for the given platforms. In this section, we describe how the cost model were obtained.

The first environment which we will refer to as the *Homogeneous Machine* (HM) consists of 16 Intel(R) Xeon(R) CPUs E7-8890 v4. Each CPU is in its own NUMA domain and has 24 cores (48 threads). The nodes are deployed into individual CPUs, and to emulate heterogeneity under this machine, we control the number of usable CPU cores on each CPU. We used two different deployment configurations under the HM, the first one is the **homogeneous deployment**. In this deployment all the CPUs have 8 usable cores, and as such exhibit homogeneous performance characteristics. In contrast, on the **heterogeneous deployment**, half of the CPUs has 16 usable cores and the other half has 8 usable cores. The CPUs with 16 usable cores will be denoted as *fast nodes*, and the ones with 8 usable cores as *slow nodes*.

The second run time environment is a *Natural Heterogeneous Cluster* (NHC) combining two kinds of machines. There are four fast machines which have two Intel(R) Xeon(R) CPU E5-2697 v4 with 18 cores (36 thread), each having their own NUMA domain. Similarly, there are five slow machines with two Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz with 12 cores (24 threads), also having their own NUMA domain. All the machines are connected by a 10 Gbps network. In the NHC, there are four deployment configurations:

- **Homogeneous Fast**: which deploys the simulation on the fast machines, mapping one node per CPU, and using 16 threads per CPU.
- **Homogeneous Slow**: which deploys the simulation on the slow machines, mapping one node per slow CPU and using 16 threads per CPU.
- **Natural Heterogeneous**: which deploys the nodes on the same amount of fast and slow CPUs and using 16 threads per CPU.
- **Degree Four Heterogeneous**: In this configuration, we emulated a four degree heterogeneity by deploying the nodes in a round robin fashion to a slow CPU with 16 threads, fast CPU with 16 threads, slow CPU with 8 threads and fast CPU with 8 threads.

To obtain the concrete computational costs model for distributed CityMoS, we generated multiple controlled calibration scenarios designed to capture the general performance behavior depending on the features we use. As mentioned in the Section 3.3.1, we empirically found that the two most significant features in order to characterize CityMoS's performance correspond to the vehicle count and the average amount of active lanes. For the case of the communication cost, the feature corresponds to the number of vehicle migrations between partitions. Therefore, our approach was to run scenarios which exhibit different traffic patterns (e.g., only traffic coherently flowing into a single direction, or fully random) and different distributions (e.g., spatially uniform or skewed) to ensure that trained cost model parameters generalize and do not depend on specific scenarios. We used low-overhead code instrumentation which consisted of timers to measure the wall-clock time used in the computation segments of the code, as well as the total cost. The wall clock time information is saved in a database together with the performance features to later be process in the cost model creation. The calibration scenarios were run under the different kind of nodes to later perform an offline analysis on the data. In our experience the predictions were consistent across the different kind of nodes i.e. same CPU and same amount of threads, therefore the calibration scenarios could be run once per node type. The amount of time needed to generate the fitting data was significant around 40 hours for the CPUs with 8 threads and 20 hours for the 16 threads, and the fitting of the models using data took around 20 seconds. The models were saved and could be instantly used in the partitioning process, so the actual calibration had to be performed only once. Nevertheless, we did not aim to minimize this time. The



(a) HM (b) NHC

Fig. 2. Computational cost data used for model calibration and resulting models for both run time environments. Each color indicates a specific type of node

problem to generate an accurate cost model by minimal sampling to reduce the total initial fitting phase is a challenge by itself and it is out of scope of this work.

For the case of the computation cost model two kind of model were used, linear and polynomial. We took the data from the calibration scenarios and perform polynomial regression with the given features. In the case of computation cost on the HM, we use polynomial model of second order as model for the fast and slow nodes resulting in determination coefficients of $R^2 = 0.98$ and $R^2 = 0.99$, respectively. Figure 2a shows the data used for calibration and the model for both kind of nodes. For the NHC, we use a linear model resulting correlation coefficients of $R^2 \approx 0.99$ for all the different kinds of nodes. Figure 2b shows the data used for calibration and the model for the four kind of nodes.

For modeling the communication cost, we followed a similar approach. With the cost data obtained from the calibration scenarios we estimate the communication cost by $\mathrm{TC} - max(\mathrm{CompCost}_i)$, which is a simplification of the actual communication model (which is not centralized, but pair-wise among nodes). However, the development of cost models more precisely resembling the performance characteristics of CityMoS is beyond the scope of this work. We used a linear model to capture the relation between number of migrations and communication cost, which yielded a determination coefficient of $R^2 = 0.94$.

7 EXPERIMENTAL EVALUATION

To assess the effectiveness of each component of ENHANCE, we conduct three distinct experiments, each designed with an increasing complexity. The first experiment is designed to assess ENHANCE's functionality without reliance on the accuracy of the proposed cost model. This is achieved by artificially assigning costs to the test graphs, essentially conducting the experiment in an *in vitro* manner, as detailed in the section 7.1. The second experiment is designed to test if ENHANCE is able to improve the TPC with features obtained from a real simulation and the cost model obtain from the runtime environment, as detailed in section 7.2. Finally, we evaluated the new partitionings *in-vivo* by running CityMoS with the obtained partitions to see if the performance was improved in section 7.3. In summary, the experimental evaluation of ENHANCE aims to answer three key questions:

- **Q1:** Is ENHANCE able to improve the partitioning according to the used cost model?
- **Q2:** Is ENHANCE able to improve end-to-end performance for CityMoS?
- **Q3:** Is ENHANCE able to improve end-to-end performance under heterogeneous resource environment?

Note that Q1 does not depend on the cost models to accurately reflect the performance characteristics of a simulation, but rather concerns the ability of ENHANCE's approach to improve partitioning according to the cost models. This is required, together with sufficiently accurate cost models, for ENHANCE to provide benefits in practice Q2. Also, we tested if our approach was flexible enough to incorporate different cost models coming from heterogeneous hardware, and use this information effectively in order to extract more performance Q3.

7.1 *In vitro* experimental evaluation

Prior to employing ENHANCE in actual simulations, we initially engage with an evaluation with synthetic scenarios. In essence, we allocate straightforward numerical workloads to the nodes and edges of graphs and apply METIS along with ENHANCE to generate and refine partitionings. This study allows us to evaluate if our re-partitioning result in lower TPC than the original partitioning under control conditions. Furthermore, it provides the flexibility to simulate heterogeneity without being constrained by available resources, allowing for the hypothetical performance of any number of hardware configurations. We evaluate ENHANCE from different domains and featuring different characteristics:

- **Shenzhen**: Road network of the Shenzhen metropolitan area, simplified with the tool [11]. Each vertex corresponds to a road, and each edge corresponds to a connection between the roads. The graph has 85, 204 vertices and 117, 788 edges in total. Partitioning this particular graph was a challenge stemming from our given research on a large-scale distributed traffic simulation, and ENHANCE's primarily aim is to efficiently partition this and similarly structured graphs.
- **RoadNet-PA**: Road network of Pennsylvania. Each vertex corresponds to an intersection or endpoint, and each edge corresponds to a road connecting them. The graph has 1, 088, 092 vertices and 1, 541, 898 edges [6]. Apart from encoding of roads and intersections in a different way, this graph is also approximately one order of magnitude larger than the Shenzhen graph.
- **Email-Enron**: Graph representing e-mail interaction. Vertices correspond to e-mail addresses, and edges correspond to communication between them. The graph has 36, 692 vertices and 183, 831 edges [6], as well as a higher ratio of edges to vertices than the previous two graphs. We include this graph in our evaluation to exercise ENHANCE's ability to efficiently partition graphs from other domains and featuring structure different from road networks.

We use a simple workload for each of these graphs by assuming one unit of processing load per vertex, and one unit of communication load per edge: $\forall v \in V: \ f_v = 1$ and $\forall e \in E: \ f_e = 1 | e \in E$. We use a simple additive feature aggregation, that is: $f_i^v = |V_i|$ and $f_{glob}^e = |E_c|$. The computational cost is simply computed from the computational load normalized by a constant C_i denoting the relative performance of node i: $CompCost_i = f_i^v / C_i$. Higher values for C_i indicate a more efficient (or faster) machine capable of handling the same workload at a lower cost, whereas a lower C_i indicates a less efficient (or slower) machine, which incurs a higher cost for processing the same amount of work. For achieving optimal workload distribution (i.e., without any imbalance), the workloads should be adjusted to align with the computational capacities of the nodes, based on the weighted average of their performance coefficients, i.e. $f_i^v = |V| \times C_i / \sum C_i$. This ensures that each node incurs an equal computational cost, eliminating bottlenecks where some nodes might otherwise have to wait for others. The optimal computational cost, serving as a lower bound for the TPC in the absence of communication overhead, is then $OptimalCompCost = |V| / \sum C_i$.

The communication cost is computed in a similar way, by scaling the communication load by a constant factor: $CommCost(f_e) = CommCoeff \times f_e$. The reason for scaling is to put both cost terms in a proportion that allows for efficient partitioning up to the maximum number of partitions for each graph (1024 for RoadNet-PA and 512 for the others). Without scaling, the communication cost together with the graph structure may otherwise render the underlying workload structurally unscalable (up to the maximum number of partitions considered):

the communication cost may far exceed the benefits of more computational resources available, and using *more resources* may result in *lower performance*. Note that this effect is unrelated to our work, but an inherent property of the workload. For the Shenzhen, RoadNet-PA, and Email-Enron graphs considered in this evaluation, we used the values of 0.005, 0.01, and 0.01 for $CommCoeff$, respectively.

We study the behavior of ENHANCE with different *heterogeneity degrees* (δ), which correspond to the number of unique values of C_i. The set of unique values of C_i are assigned to the nodes in a round robin fashion. The baseline case has $\delta = 1$ which degenerates to the homogeneous case having a unique value of $C = 1$. For $\delta > 1$ the coefficients are chosen to be the linear space between $[1, 2]$, for example the coefficients for a simulation with 4 nodes and $\delta = 2$ would correspond to $C_1 = C_3 = 1$ and $C_2 = C_4 = 2$, and the same nodes with $\delta = 4$ would have coefficients $C_1 = 1$, $C_2 = 1.\bar{3}$, $C_3 = 1.\bar{6}$, $C_4 = 2$. In this way, the heterogeneous environment may resemble scenarios where, for example, a cluster consisting of slow nodes is extended with additional faster nodes.

In each experiment, we used a partitioning obtained from METIS as input to ENHANCE. For each configuration, we apply ENHANCE five times with a unique random seed in each invocation. Note that in each invocation of ENHANCE, the same partitioning obtained from METIS is used as a start. The average TPC along with standard deviation are plotted in Figure 3. Each row of images corresponds to a graph, the first for the Shenzhen, the second for the RoadNet-PA and the final for the Email-Enron. The columns are the different heterogeneity degrees δ starting from left with the homogeneous case ($\delta = 1$), in the middle $\delta = 4$ and finally $\delta = 16$.

The result shown in Figure 3 illustrate that ENHANCE is indeed able to enhance the METIS-generated partitions and further reduce the TPC across a broad range of configurations. This effect is particularly pronounced for the Shenzhen graph, but there are also cases in which the *enhance*d partitions yield a TPC higher than the initial METIS-generated partitions. In practice, however, as a fallback mechanism, it is easily possibly to use the METIS-generated partitioning in case the partitioning produced by ENHANCE does not improve performance (in terms of TPC reduction). A partitioning of the Shenzhen graph using ENHANCE is depicted in Figure 5, where roads are color-coded by partition. Interestingly, we can observe that the relatively smooth boundaries between the partitions obtained by METIS are blurred when processed by ENHANCE in the sense that partitions may be locally *invaded* by other partitions. This effect is visible, for instance, on the boundary between the pink and blue partitions. Naturally, this behavior may increase the edge cut, resulting in higher communication cost (but still mostly lower total cost), as pointed out earlier. In extreme cases, this may result in disconnected components within the same partition.

In the case of homogeneous resources ($\delta = 1$), we expected METIS to perform well, since homogeneous resources are an (implicit) assumption made in the approach taken by METIS. In fact, for instance in figure 3a, the TPC with METIS is almost overlapping with the $OptimalCompCost$ up to 2^5 partitions. After this point, the TPC with METIS exceeds the optimal $OptimalCompCost$. ENHANCE takes partitionings produced by METIS and *enhance*s them according to the provided cost models. Recall from algorithm 4 (line 4) that ENHANCE does so by first aiming to balance the load, ignoring the communication cost. This results in a more balanced partitionings at the expense of an increase of communication cost in the first step. The later refinements in the line 6 move the vertices reducing the TPC, now considering the communication as well, which basically allows to refine the borders between the balanced partitions, and finding the local minimum of the TPC. In some cases, moving a vertex can result in fewer communication, and more imbalance or the other way around, but the criteria is which decision will minimize the TPC as a full. Note that we only allow the partition to be fine-tuned in a way that vertices in the edge cut can be moved to neighbor partitions, until the lowest TPC is reached. Hence, the size of the edge cut (or, more precisely, the number of vertices adjacent to edges in the edge cut) corresponds to the number of possible modification actions considered. Interestingly, we find that the partitionings generated with ENHANCE tent to result in higher communication cost than METIS.

In the case of heterogeneous resources ($\delta \in \{4, 16\}$), METIS does not consider the actual per-node processing performance, and hence, tends to overload slower nodes and under-utilize faster ones. As an example, for instance,

consider the results for the Shenzhen graph with a $\delta = 16$ shown in Figure 3c. In this environment, the METIS *CompCost* line is surpassing the *OptimalCompCost* because it is equally distributing the load among the node, without the consideration of the heterogeneity. Therefore, the performance is limited by the slowest nodes, and the faster nodes are not taken advantage to their full potential. This is reflected in the imbalance ratios: for instance, for sizes 2^5, 2^6 and 2^7, the imbalance ratios are 1.49, 1.67 and 1.52. In contrast, the *CompCost* with the enhanced partitionings are almost overlapping with the *OptimalCompCost* and exhibit lower imbalance (e.g., maximum imbalance ratio of 1.01 for the 2^9 nodes). Yet, the resulting partitionings tend to have a higher *CommCost* than the originals as in the homogeneous case. As a result, the partitioning for 2^9 nodes does actually not improve upon METIS overall (according to TPC).

As pointed out earlier, the Email-Enron graph has a much higher ratio of edges to vertices than the other graphs representing road networks. This property renders the workload more bound by communication, which is also reflected in the characteristics of the resulting partitionings. The results for the Email-Enron graph are depicted in Figures 3g, 3h and 3i. The strong impact of the communication cost for this graph, compared to the other two graphs considered, can be observed, for instance, in Figure 4b, the total cost is broken down into communication and computation cost. Here, the *CommCost* has roughly the same order of magnitude as the *CompCost* at 2^4, and for larger sizes, *CommCost* becomes the dominant component of the TPC. In contrast, the other two graphs have a smaller ratio of edges to vertices, and the communication cost only becomes the dominant factor for larger number of partitions. As an example, consider the cost breakdown for the Shenzhen graph, depicted in Figure 4a. Compared to the Email-Enron graph, the communication cost in the Shenzhen graph exceeds the computation cost only for higher numbers of partitions.

In this section, we evaluated ENHANCE's capacity of improving METIS-generated partitions, and our results indeed indicate that, over a range of different configurations, ENHANCE is capable of producing partitions that improve upon METIS according to a cost model only, without actually running our target application, CityMoS. In that sense, this evaluation was an *in vitro* evaluation that deliberately excludes any potential impact of model inaccuracy. Next, we investigate if the partitionings produced by ENHANCE also benefit actually measured runtime performance.

7.2 Partitioning for CityMoS

In this section, we evaluate the performance of CityMoS without actually running CityMoS by only studying the predicted cost according to the cost model (Q1). This is a similar evaluation than the one performed in section 7.1, but this time the cost models correspond to the CityMoS models obtained in Section 6 and features were obtained from a simulation. Therefore, we are evaluating if ENHANCE can minimize the objective function (i.e., the TPC) in a more complex and realistic case.

As a performance benchmark, we use a grid-like road network with 90 by 90 intersections and bidirectional roads with 3 lanes per road. We represent the road network as a graph with vertices representing roads and edges representing connections between roads. The resulting graph G consists of 159, 120 vertices and 254, 160 edges. The traffic in this scenario is composed of a single traffic wave with 350, 000 concurrent agents at peak, where each agent has a randomly selected origin and destination. Before running partitioning experiments, we ran the simulation once to generate the vertex and edge feature vectors (e.g., average number of agents per road, average number of migrations) which we also used to derive the weights for METIS.

Similar to the experiments described in the previous section, we used METIS to generate an initial partitioning and then applied ENHANCE to improve them. The results are summarized in Table 1 for all the runtime environments and all deployment configurations. Note that the number of fast machines for the NHC was limited to 4, therefore we consider are up to 8 (total) nodes in this setting. For each number of partitions, the results were obtained from 5 samples, all starting from the same METIS partitioning, but using different random seeds.

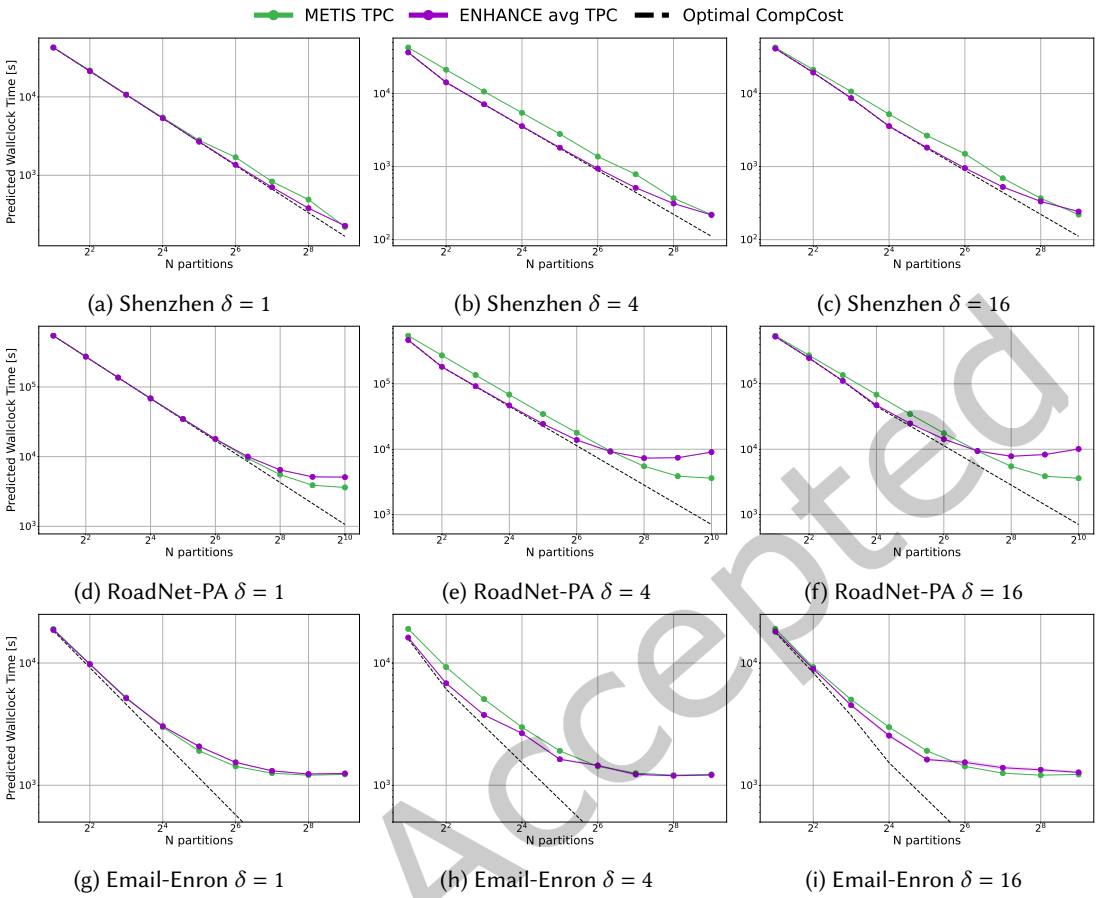
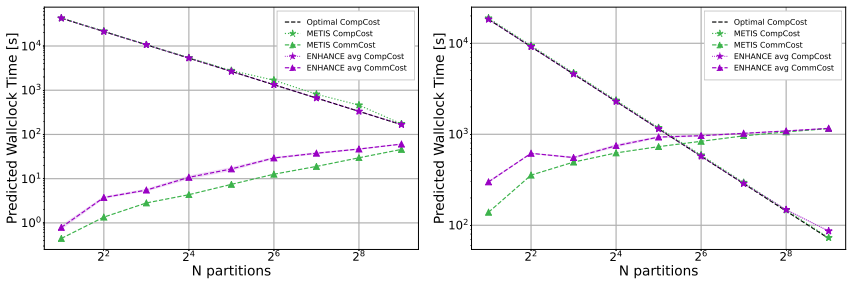


Fig. 3. Predicted performance of METIS and ENHANCE partitionings for the different graphs under different heterogeneity degrees.

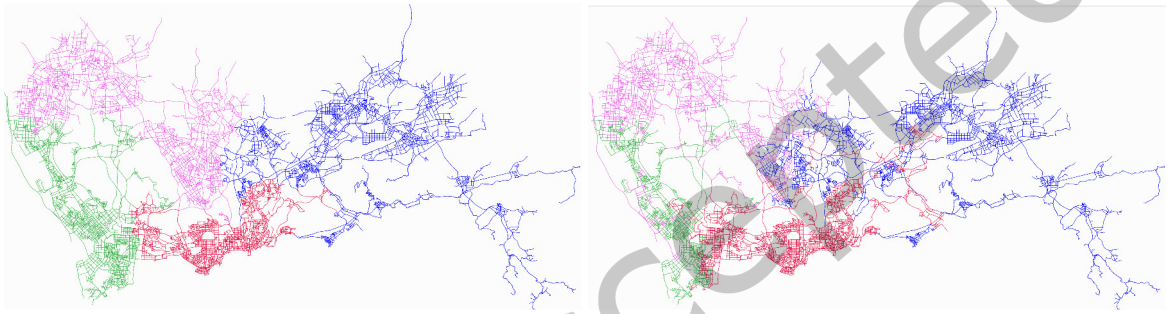
First, the gap between the minimum and maximum TPC is remarkable and more pronounced than in the in-vitro experiments from the previous section. The only aspect where the random seeds have any impact is the order in which vertices are processed in Algorithm 2. The features coming from the simulation which has higher and lower traffic regions in the road network increased the skewness of the features, making the processing ordering of the vertices to have higher impact in the final result of the partitioning process. This observation further motivates studying alternative ordering schemes exploiting characteristics of the underlying graph. Also, a Monte Carlo approach might be employed by invoking ENHANCE in parallel with different random seeds, and then using the best result obtained by any instance.

In most of the cases, ENHANCE was able to improve the average predicted cost over the partitioning obtained by METIS. The benefit of ENHANCE is particularly remarkable in the heterogeneous environment (see Table 1), which is not entirely unexpected since METIS does not consider any resource heterogeneity. Changing the perspective, the results may also be interpreted as the performance potential lost when ignoring resource heterogeneity.



(a) Shenzhen $\delta = 1$ (b) Email-Enron $\delta = 1$

Fig. 4. Breakdown of the computation and communication cost for partitionings produced by METIS and ENHANCE



(a) METIS partition (b) ENHANCE partition with heterogeneous nodes $\delta = 4$

Fig. 5. Initial METIS partitioning and ENHANCE partitioning for the Shenzhen road network. The corresponding values for the C_i are 1, $1.\bar{3}$, $1.\bar{6}$ and 2 for the colors green, pink, blue and red.

Also, by taking a look on the minimum obtained TPC, we can see that in five runs of the algorithm, it is was able to find at least a better partitioning than the original for all cases expect for two: for the 10 partition in the HM Heterogeneous case, and for the NHC Fast Homogeneous with 4 partitions. The fact that ENHANCE may yield partitionings performing *worse* than METIS calls for further analysis. As a safeguard, ENHANCE can be adjusted to always fall back to the initial partitioning in case it has better cost than the "enhanced" one, which would guarantee to never perform worse than the initial partitioning.

7.3 CityMoS Performance Evaluation

In the previous section we have shown performance benefits of ENHANCE over METIS without considering the accuracy of the cost model since only the predicted cost of the partitionings was evaluated. In the following we present evaluation results for end-to-end performance in terms of wall-clock time corresponding to the total cost (TC). Note that, in this case, the cost model used has to be reasonably accurate to enable ENHANCE to identify and apply beneficial modifications to the partitionings (Q2). For our evaluation, we used the partitionings produced by METIS and ENHANCE considered in the previous section, and we use the one with minimal TPC for CityMoS (as possible in practice by picking the best of multiple partitionings produced in parallel).

For the homogeneous deployments, we show end-to-end performance results in Figures 6a, 6b and 6c, corresponding to the Tables 2a, 2b and 2c. The results show that the partitionings produced by ENHANCE indeed

Table 1. TPC for the METIS and *enhanced* partitionings, along with speed up over METIS and the Average Moved Vertices (AMV) while enhancing METIS partitionings.

N	METIS TPC	Min TPC	Avg TPC	Max TPC	Avg Speedup	AMV
2	1,998.82	1,921.40	1,926.09	1,941.32	1.04	2,166.20
4	1,052.63	1,015.14	1,030.29	1,058.27	1.02	5,190.20
6	866.64	804.95	835.87	857.81	1.04	7,107.00
8	835.79	767.90	809.05	840.39	1.03	10,560.20
10	885.14	854.86	894.19	925.90	0.99	12,755.80

(a) HM Homogeneous

N	METIS TPC	Min TPC	Avg TPC	Max TPC	Avg Speedup	AMV
2	1,998.82	1,440.15	1,467.82	1,511.19	1.36	24,261.80
4	1,022.52	859.34	939.51	1,022.42	1.09	22,526.20
6	817.01	729.40	765.84	791.48	1.07	18,436.80
8	821.06	692.66	742.33	831.47	1.11	21,606.40
10	879.99	910.28	961.37	1,065.77	0.92	34,711.80

(b) HM Heterogeneous

N	METIS TPC	Min TPC	Avg TPC	Max TPC	Avg Speedup	AMV
2	1010.91	971.62	988.30	1002.47	1.02	1542.00
4	627.62	616.14	641.30	660.44	0.98	4139.80
6	601.65	558.84	571.56	590.77	1.05	6275.40
8	654.24	594.53	603.21	610.61	1.08	9507.00
10	748.93	729.61	766.16	815.45	0.98	12431.40

(c) NHC Slow Homogeneous

N	METIS TPC	Min TPC	Avg TPC	Max TPC	Avg Speedup	AMV
2	984.38	932.30	948.99	965.02	1.04	3515.60
4	627.62	585.88	623.14	648.50	1.01	4854.60
6	601.65	515.46	542.51	582.99	1.11	8822.40
8	654.24	551.17	588.67	613.14	1.11	13164.00
10	748.93	621.05	699.32	783.70	1.07	15553.80

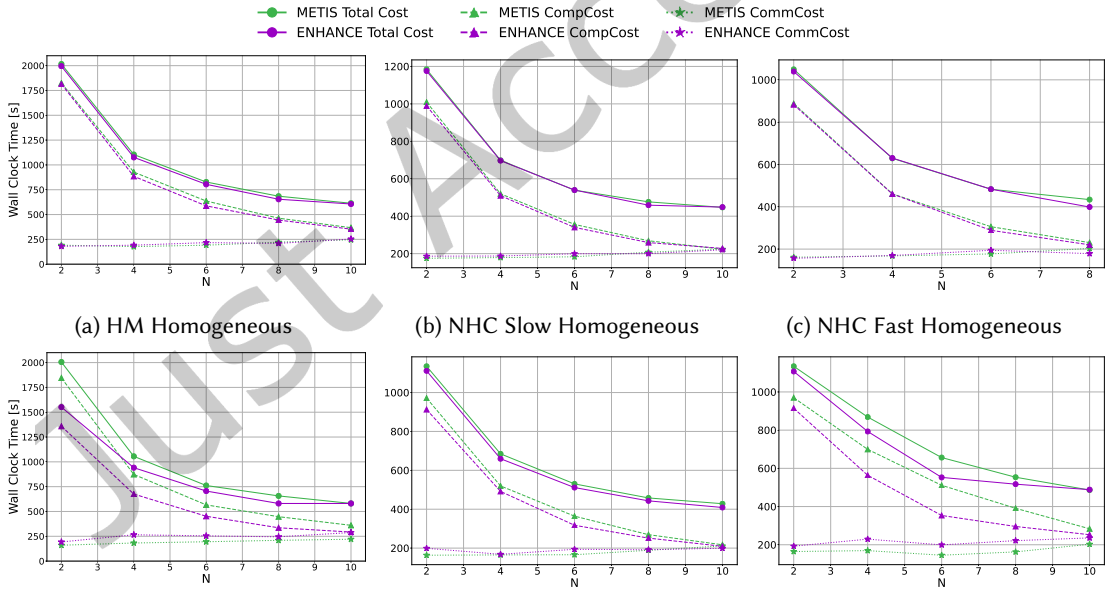
(d) NHC Natural Heterogeneous

N	METIS TPC	Min TPC	Avg TPC	Max TPC	Avg Speedup	AMV
2	886.02	848.60	863.97	876.94	1.03	1486.80
4	562.27	572.06	587.40	608.91	0.96	3797.00
6	556.03	506.90	525.83	557.61	1.06	6082.60
8	620.24	540.75	558.65	583.46	1.11	9617.80

(e) NHC Fast Homogeneous

N	METIS TPC	Min TPC	Avg TPC	Max TPC	Avg Speedup	AMV
2	984.38	932.30	948.99	965.02	1.04	3515.60
4	813.10	711.92	880.20	1047.69	0.92	35010.80
6	754.22	594.49	694.43	788.18	1.09	21337.00
8	774.07	701.34	746.05	805.95	1.04	34948.20
10	825.94	796.01	909.15	1029.76	0.91	28192.20

(f) NHC Degree Four Heterogeneous



(a) HM Homogeneous (b) NHC Slow Homogeneous (c) NHC Fast Homogeneous

(d) HM Heterogeneous (e) NHC Natural Heterogeneous (f) NHC Degree Four Heterogeneous

Fig. 6. Cost breakdown with METIS and *enhanced* partitionings in homogeneous (up) and heterogeneous (down) deployments.

yield better performance than the initial ones produced by METIS by a small margin. In particular for the HM homogeneous case the improvements were the most significant. To understand the reason of these improvements, consider the experiments in Table 3a summarizing the measured computation cost per partition together with the values of the feature vectors for the HM homogeneous deployment. We can see that in the METIS homogeneous case partition 1 is the slowest one which has both the maximum amount of agents and the maximum amount of active lanes. ENHANCE was able to identify this imbalance using the cost model and to improve upon it. We can see that the resulting partitions reduced the number of agents in partition 1 and increased it, for example, in partition 3. Partition 3 ended up with the highest number of average agents, but it does not have the highest number of active lanes. The slowest partition, curiously enough, is still partition 1 because even though it has the smallest number of agents it has the highest number of active lanes.

The decision to improve this balance was only possible because of the higher granularity of the cost model used. As we can recall from Section 6, for the HM we used polynomial model of order 2 cost model considering both features *AvgAgentNumber* and *AvgActiveLanes* which resulted in a $R^2 = 0.99$, and for the NHC fast and slow nodes we used linear regression with same features resulting on $R^2 = 0.99$ for both cases. By only focusing on balancing the amount of agents as in previous works, it is basically assumed a linear *CompCost* depending on the *AvgAgentNumber* feature. On the HM homogeneous node, this cost model results in a $R^2 = 0.94$ making in less precise to explain the variability of the data. In contrast, for the NHC fast and slow nodes, doing a linear fitting for only the *AvgAgentNumber* results in R^2 of 0.97 and 0.98 respectively. As consequence, most of the variability in the data is determined by the feature *AvgAgentNumber*, which is the main assumption of the state of the art, and therefore the starting point is closer to a better partitioning. This is one of the factors why the improvements on the HM homogeneous were more impactful compared to the NHC Slow and Fast homogeneous deployments.

For the heterogeneous case, the results depicted in Figures 6d, 6e and 6f and Tables 2d, 2e and 2f show a more pronounced improvement of ENHANCE. Taking a closer look at the HM heterogeneous 6d and Table 2d, in the case of $N = 2$, the speedup was 29%, followed by around 10% for higher numbers of partitions, except for the $N = 10$ where both partitionings had almost the same performance. Remarkably, the partitioning generated with ENHANCE at $N = 8$ is basically performing equally well as the METIS partitioning for the $N = 10$ case, meaning that the same performance can be achieved with 20% fewer resources and better partitioning. Taking also a closer look to the computational cost per partition in Table 3, we can see the reason of the performance improvement. Here it is very clear that machines 1 and 3 are the fast machines, so even though in the METIS partition the number of agents is very similar having difference of 4.07% between the highest and lowest average amount of agents, the imbalance in term of computation cost is of 47.04%. For the refined partitioning we can see that the resulted computation cost has imbalance of only 6.16% and it reduced the maximum computational cost by 22.62%. Therefore, accounting for the heterogeneity allowed ENHANCE to improve the initial partitioning (Q3).

8 CONCLUSION AND FUTURE WORK

In this paper, we addressed the challenge of deriving partitionings suitable for distributed CityMoS by incorporating a cost model. ENHANCE's approach is incremental by nature, and therefore, it can be used both in conjunction with other partitioning algorithms, such as METIS, as we did in this work, and as well for dynamic re-partitioning during run-time.

One key ingredient in ENHANCE's approach is to break the coupling between graph weights and the (implicit) cost models used for the partitioning process. This enables using cost models that more accurately reflect simulator performance characteristics than (edge or vertex) weights. Explicit cost models also account for resource heterogeneity, a common occurrence in cloud deployments. We presented experimental evaluation

Table 2. TC for the METIS and *enhanced* partitionings, along with the performance speedup over METIS.

N	METIS TC	ENHANCE TC	Speedup
2	2016.31	1994.66	1.01
4	1104.54	1076.17	1.03
6	827.83	804.50	1.03
8	684.76	654.20	1.05
10	613.80	606.34	1.01

(a) HM Homogeneous

N	METIS TC	ENHANCE TC	Speedup
2	1184.42	1175.61	1.01
4	698.60	696.70	1.00
6	539.22	539.53	1.00
8	476.56	458.80	1.04
10	447.17	448.52	1.00

(b) NHC Slow Homogeneous

N	METIS TC	ENHANCE TC	Speedup
2	1049.58	1039.24	1.01
4	629.01	630.36	1.00
6	482.97	483.19	1.00
8	433.92	398.93	1.09

(c) NHC Fast Homogeneous

N	METIS TC	ENHANCE TC	Speedup
2	2005.54	1551.47	1.29
4	1055.54	940.95	1.12
6	761.41	705.85	1.08
8	656.75	580.56	1.13
10	581.00	580.60	1.00

(d) HM Heterogeneous

N	METIS TC	ENHANCE TC	Speedup
2	1135.05	1111.64	1.02
4	684.73	659.60	1.04
6	530.26	511.39	1.04
8	457.99	443.01	1.03
10	428.32	408.89	1.05

(e) NHC Natural Heterogeneous

N	METIS TC	ENHANCE TC	Speedup
2	1134.31	1107.21	1.02
4	868.58	793.06	1.10
6	656.46	552.59	1.19
8	553.48	517.13	1.07
10	486.41	488.39	1.00

(f) NHC Degree Four Heterogeneous

Table 3. Features (average number of agents and active lanes) and computational cost per partition (n) for $N = 4$ nodes on the HM environment.

Experiment	n	*CompCost*	*AvgAgentNumber*	*AvgActiveLanes*
METIS	1	**927.41**	**76,081.01**	**20,011.63**
	2	872.70	75,815.19	17,059.32
	3	821.45	72,979.82	13,858.69
	4	858.59	75,749.65	14,889.10
ENHANCE	1	**884.96**	72,648.66	**19,296.33**
	2	858.09	74,623.15	16,962.01
	3	869.76	**76,890.69**	14,574.25
	4	864.22	76,454.71	14,988.52

(a) Homogeneous deployment

Experiment	n	*CompCost*	*AvgAgentNumber*	*AvgActiveLanes*
METIS	1	525.85	**76,075.53**	**20,011.38**
	2	**873.18**	75,822.46	17,058.36
	3	469.80	72,977.33	13,858.58
	4	870.16	75,751.48	14,889.44
ENHANCE	1	653.99	90,658.27	**22,782.30**
	2	643.91	56,784.20	13,324.99
	3	634.03	**94,934.14**	17,907.80
	4	**675.65**	58,236.23	11,807.16

(b) Heterogeneous deployment

results for CityMoS demonstrating the benefits of a prototype of ENHANCE. Next, we highlight potential directions to further improving ENHANCE as part of future work.

Even though the simplified cost models we used for CityMoS were sufficient to produce partitionings yielding substantially higher performance compared with METIS, there is potential for improving the accuracy of these cost models and possibly obtain further performance benefits. For instance, the cost models can be made more accurate by capturing the performance of different communication channels (e.g., shared memory and network), and accounting for peer-to-peer (rather than centralized) communication as well as the impact of latency hiding.

In Section 5.3 we pointed out that in our current prototype of ENHANCE, vertices are processed in random order when modifying partitions. The impact of ordering schemes, however, is currently not clear and may be analyzed in future work. Alternative ordering schemes, possibly based on vertex characteristics, may contribute to a faster convergence or partitionings yielding higher performance.

The randomization in the ordering of vertices may also be taken as an advantage to explore larger parts in the solution space: instead of applying ENHANCE only once, it can be applied multiple times (e.g., once for each available CPU core) in parallel, each time with different random seeds. This approach may reduce the risk of *all* invocations converging to local optima far away from global optima.

Finally, we aim to explore the use of ENHANCE for dynamic re-partitioning. We designed ENHANCE to work incrementally already, but we did not test it in this use case yet. In this context, one promising property of ENHANCE is that the cost models, which are already an integral part of ENHANCE, can be leveraged to estimate the potential benefit of a re-partitioning, and this estimate can be used in the logic to decide whether to re-partition.

REFERENCES

[1] Rishan Chen, Mao Yang, Xuetian Weng, Byron Choi, Bingsheng He, and Xiaoming Li. 2012. Improving Large Graph Processing on Partitioned Graphs in the Cloud. In *Proceedings of the Third ACM Symposium on Cloud Computing* (San Jose, California) *(SoCC '12)*. Association for Computing Machinery, New York, NY, USA, Article 3, 13 pages. https://doi.org/10.1145/2391229.2391232

[2] Dazhao Cheng, Jia Rao, Yanfei Guo, and Xiaobo Zhou. 2014. Improving MapReduce performance in heterogeneous environments with adaptive task tuning. In *Proceedings of the 15th International Middleware Conference* (Bordeaux, France) *(Middleware '14)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/2663165.2666089

[3] Laurent Y.M. Gicquel, N. Gourdain, J.-F. Boussuge, H. Deniau, G. Staffelbach, P. Wolf, and Thierry Poinsot. 2011. High performance parallel computing of flows in complex geometries. *Comptes Rendus. Mécanique* 339, 2-3 (2011), 104–124. https://doi.org/10.1016/j.crme.2010.11.006

[4] Bruce Hendrickson and Tamara G Kolda. 2000. Graph partitioning models for parallel computing. *Parallel Comput.* 26, 12 (2000), 1519–1534. https://doi.org/10.1016/S0167-8191(00)00048-X Graph Partitioning and Parallel Computing.

[5] Damian Igbe. 2010. *Dynamic load balancing of parallel road traffic simulation.* Ph. D. Dissertation. University of Westminster. https://doi.org/10.34737/90644

[6] Anirban Dasgupta Jure Leskovec, Kevin J. Lang and Michael W. Mahoney. 2009. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics* 6, 1 (2009), 29–123. https://doi.org/10.1080/15427951.2009.10129177

[7] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1998), 359–392. https://doi.org/10.1137/S1064827595287997

[8] G. Karypis and V. Kumar. 1998. Multilevel Algorithms for Multi-Constraint Graph Partitioning. In *SC '98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*. 28–28. https://doi.org/10.1109/SC.1998.10018

[9] B. W. Kernighan and S. Lin. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal* 49, 2 (1970), 291–307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x

[10] Tim Kiefer, Dirk Habich, and Wolfgang Lehner. 2016. Penalized Graph Partitioning for Static and Dynamic Load Balancing. In *Euro-Par 2016: Parallel Processing*, Pierre-François Dutot and Denis Trystram (Eds.). Springer International Publishing, Cham, 146–158. https://doi.org/10.1007/978-3-319-43659-3_11

[11] Zhuoxiao Meng, Xiaorui Du, Paolo Sottovia, Daniele Foroni, Cristian Axenie, Alexander Wieder, David Eckhoff, Stefano Bortoli, Alois Knoll, and Christoph Sommer. 2022. Topology-Preserving Simplification of OpenStreetMap Network Data for Large-scale Simulation in SUMO. In *SUMO User Conference 2022 (SUMO 2022)*. https://doi.org/10.52825/scp.v3i.111

[12] Henning Meyerhenke, Burkhard Monien, and Thomas Sauerwald. 2008. A new diffusion-based multilevel algorithm for computing graph partitions of very high quality. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. 1–13. https://doi.org/10.1109/IPDPS.2008.4536237

[13] Carlos Fernandez Musoles, Daniel Coca, and Paul 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* (Kyoto, Japan) *(ICPP '19)*. Association for Computing Machinery, New York, NY, USA, Article 19, 10 pages. https://doi.org/10.1145/3337821.3337876

[14] Kinjal Patel, Eric Hunsberger, Sean Batir, and Chris Eliasmith. 2021. A Spiking Neural Network for Image Segmentation. https://doi.org/10.48550/arXiv.2106.08921

[15] Tomas Potuzak. 2012. Methods for division of road traffic networks focused on load-balancing. *Advances in Computing* 2, 4 (2012), 42–53. https://doi.org/10.5923/j.ac.20120204.01

[16] Tomas Potuzak. 2016. Utilization of graph coarsening for improving of results of a genetic algorithm for road traffic network division. In *2016 9th International Conference on Human System Interactions (HSI)*. 28–34. https://doi.org/10.1109/HSI.2016.7529604

[17] Tomas Potuzak. 2021. Improved Road Traffic Network Division based on Genetic Algorithm and Graph Coarsening. In *2021 14th International Conference on Human System Interaction (HSI)*. 1–8. https://doi.org/10.1109/HSI52170.2021.9538696

[18] Tomas Potuzak. 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)*. 77–86. https://doi.org/10.1109/DS-RT55542.2022.9932112

[19] K. Schloegel, 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 (2001), 451–466. https://doi.org/10.1109/71.926167

[20] Isabelle Stanton and Gabriel Kliot. 2012. Streaming graph partitioning for large distributed graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Beijing, China) *(KDD '12)*. Association for Computing Machinery, New York, NY, USA, 1222–1230. https://doi.org/10.1145/2339530.2339722

[21] S. Subramanian, D.M. Rao, and P.A. Wilsey. 2000. Study of a multilevel approach to partitioning for parallel logic simulation. In *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*. 833–838. https://doi.org/10.1109/IPDPS.2000.846071

[22] Alan Valejo, Vinícius Ferreira, Renato Fabbri, Maria Cristina Ferreira de Oliveira, and Alneu de Andrade Lopes. 2020. A Critical Survey of the Multilevel Method in Complex Networks. *ACM Comput. Surv.* 53, 2, Article 39 (apr 2020), 35 pages. https://doi.org/10.1145/3379347

[23] Dali Wei, Feng Chen, and Xinxin Sun. 2010. An improved road network partition algorithm for parallel microscopic traffic simulation. In *2010 International Conference on Mechanic Automation and Control Engineering*. 2777–2782. https://doi.org/10.1109/MACE.2010.5536795

[24] Donghua Xu 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.* 455–463. https://doi.org/10.1109/MASCOT.2004.1348301

[25] Ning Xu, Bin Cui, Lei Chen, Zi Huang, and Yingxia Shao. 2015. Heterogeneous Environment Aware Streaming Graph Partitioning. *IEEE Transactions on Knowledge and Data Engineering* 27, 6 (2015), 1560–1572. https://doi.org/10.1109/TKDE.2014.2377743

[26] Yadong Xu, Wentong Cai, Heiko Aydt, and Michael 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

[27] Yadong Xu, Wentong Cai, David Eckhoff, Suraj Nair, and Alois 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* (Singapore, Republic of Singapore) *(SIGSIM-PADS '17)*. Association for Computing Machinery, New York, NY, USA, 209–219. https://doi.org/10.1145/3064911.3064914

[28] Daniel Zehe, Alois Knoll, Wentong Cai, and Heiko Aydt. 2015. SEMSim Cloud Service: Large-scale urban systems simulation in the cloud. *Simulation Modelling Practice and Theory* 58 (2015), 157–171. https://doi.org/10.1016/j.simpat.2015.05.005

[29] Daniel Zehe, Suraj Nair, Alois Knoll, and David Eckhoff. 2017. Towards CityMoS: A Coupled City-Scale Mobility Simulation Framework. *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication* 2017 (2017), 03. https://mediatum.ub.tum.de/doc/1612009/document.pdf

[30] Angen Zheng, Alexandros Labrinidis, Panos K. Chrysanthis, and Jack Lange. 2016. Argo: Architecture-aware graph partitioning. In *2016 IEEE International Conference on Big Data (Big Data).* 284–293. https://doi.org/10.1109/BigData.2016.7840614

Received 29 February 2024; revised 29 February 2024; accepted 17 May 2024