3-2011

Multi-objective zone mapping in large-scale distributed virtual environments

Nguyen Binh Duong TA
Singapore Management University, donta@smu.edu.sg

Suiping ZHOU

Wentong CAI

Xueyan TANG

Rassul AVANI

Multi-Objective Zone Mapping in Large-Scale Distributed Virtual Environments

Duong Ta, Suiping Zhou, Rassul Ayani, Wentong Cai and Xueyan Tang

Abstract

In large-scale distributed virtual environments (DVEs), the NP-hard zone mapping problem concerns how to assign distinct zones of the virtual world to a number of distributed servers to improve overall interactivity. Previously, this problem has been formulated as a single objective optimization problem, in which the objective is to minimize the total number of clients that are without QoS. This approach may cause considerable network traffic and processing overhead, as a large number of zones may need to be migrated across servers. In this paper, we introduce a multi-objective approach to the zone mapping problem, in which both the total number of clients without QoS and the migration overhead are considered. To this end, we have proposed several new algorithms based on meta-heuristics such as local search and multi-objective evolutionary optimization techniques. Extensive simulation studies have been conducted with realistic network latency data obtained from real measurements using millions of pairs of geographically distributed IP addresses, and different workload distribution models. Simulation results demonstrate the effectiveness of the newly proposed algorithms.

I. Introduction

Distributed Virtual Environments (DVEs) are distributed systems that allow multiple geographically distributed clients (users) to explore and interact with each other in real time within a shared, computer-generated 3D virtual world, where each client is represented by an *avatar*. A client controls the behavior of his/her avatar by various *inputs*, and the *updates* of an avatar's state need to be sent to other clients in the same *zone* of the virtual world to support the interactions among clients. Examples of DVEs can

Duong Ta, Suiping Zhou, Wentong Cai and Xueyan Tang are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798. Emails: {binhduong, asspzhou, aswtcai, asxytang}@ntu.edu.sg

Rassul Ayani is with the School of Information and Communication Technology, Royal Institute of Technology, Sweden. Email: rassul@imit.kth.se

be seen in multiple areas, such as collaborative design, military simulations, e-learning, virtual shopping mall, and multiplayer online games [1].

Typically, in large-scale DVEs with thousands of clients interacting simultaneously, the resource requirements in terms of network bandwidth, CPU cycles, memory, etc. are huge, and will increase very fast as the number of concurrent clients increases. A distributed server infrastructure is usually required [2], [3], [4] for such resource-intensive applications. In this architecture, each client connects to one of multiple geographically distributed servers in the system, and clients interact with each other through these servers. For load distribution, the large virtual world is spatially partitioned into several distinct *zones*, with each zone managed by only one server. A client only interacts with other clients in the same zone, and may move from one zone to another. As a server only needs to handle a few zones instead of the entire virtual world, the distributed server architecture has good scalability.

The zone mapping problem in large-scale DVEs with a distributed server architecture arises due to the heterogeneous nature of the Internet and the fact that clients in a DVE are usually geographically distributed. In such situation, it is likely that a large number of clients in a zone may be far away (in terms of round-trip network latency) to the server hosting that zone, thus the interactivity of the DVE for those clients may be greatly degraded. Hence, there is a strong need for mechanisms to assign (map) the zones to servers in such a way that reduces the network latency between clients and servers. This is referred to as the *zone mapping* problem. Existing research on how to assign zones (with all of its clients) to servers in DVEs has usually been formulated as a *load balancing* problem, i.e., the objective is to balance the workload among servers rather than to directly reduce the latency between clients and their servers [5], [6].

Previous work [2], [3], [7] had formulated the zone mapping problem and showed that it is actually NP-hard. Several heuristic algorithms were then devised and evaluated. Although offering excellent performance in terms of interactivity, these algorithms may suffer from potentially significant overhead as the original problem has been formulated as a single-objective optimization problem. That is, the only objective is to minimize the overall mapping cost, which is measured by the total number of clients that have client-server round-trip network latencies larger than a predefined delay threshold, i.e., without QoS. We shall call it the single-objective zone mapping problem (SOZP).

One major problem of this single-objective approach is that it may cause great network and processing overhead. Usually the zone mapping procedure will start from an existing mapping. Such an initial mapping can be a random assignment of zones to servers, which may have poor performance; or a previously good mapping whose performance has gradually degraded over time because of network and

DVE dynamics, e.g., client joining, leaving or switching zones. In this case, zones will need to be migrated across distributed servers according to the result of the zone mapping algorithm. Each zone, in fact, is a small virtual world with a number of clients, hence zone migration may require significant data movement across network. Clients currently connect to the server hosting that zone also need to be switched to the new server. If zone mapping is carried out on the fly and frequently, this migration problem would become even more significant. Therefore, it is desirable to reduce the zone migration overhead, while maintaining a reasonable level of interactivity.

The primary contributions of this paper are as follows.

- We introduce a multi-objective approach to the zone mapping problem, in which we consider both the mapping cost as well as the migration overhead.
- We propose two new zone mapping algorithms, HMOEA and LPLS. These algorithms are based on a combination of multi-objective evolutionary algorithms (MOEAs), local search and linear programming relaxation technique.
- We use realistic network and workload models to evaluate the effectiveness of the new algorithms. Unlike previous work, e.g., [3] , in this paper we use realistic Internet delay spaces modeled after real measurements between millions of pairs of IP addresses distributed all over the Internet. We also employ different network topologies, both real and synthetic ones, and various client distributions, both in the network and the virtual world.
- We conduct extensive simulation studies to evaluate the effectiveness of the new algorithms. The results have been validated via Mann-Whitney U statistical tests [8], which show that HMOEA and LPLS significantly outperform a baseline algorithm with similar execution times.

The rest of the paper is organized as follows. Section II formulates the multi-objective zone mapping problem. The proposed algorithms are presented in Section III. Simulation methodology and results are described in Sections IV and V, respectively. Related work is discussed in Section VI, and Section VII concludes the paper.

II. Problem formulation

A. Multi-objective optimization

Informally stated, a multi-objective optimization problem (MOP) can be defined as a problem of finding one or more vectors of *decision variables* which satisfies given constraints and optimizes a vector of objective functions [9]. The vector X of decision variables x_i, $X = [x_1, x_2, ..., x_n]$, represents a solution to

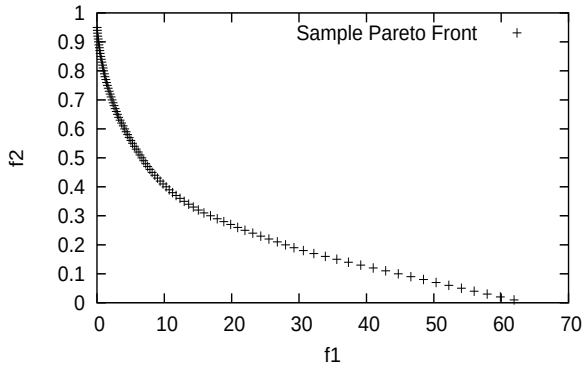


Fig. 1. Example of a Pareto front

a MOP. A vector of objective functions of the solution X is denoted as $f(X) = [f_1(X), f_2(X), ..., f_k(X)]$ where k is the number of objectives of the MOP.

Given two solutions $X = [x_1, x_2, ..., x_n]$ and $Y = [y_1, y_2, ..., y_n]$ to a MOP, assuming all the objectives of this problem are to be minimized, we say that solution X *dominates* solution Y if and only if $f(X) = [f_1(X), f_2(X), ..., f_k(X)]$ is *partially* less than $f(Y) = [f_1(Y), f_2(Y), ..., f_k(Y)]$, i.e., $\forall i \in \{1, ..., k\}$, $f_i(X) \leq f_i(Y) \wedge \exists i \in \{1, ..., k\} : f_i(X) < f_i(Y)$ [9].

If neither X dominates Y nor Y dominates X, then X and Y are called *non-dominated* solutions. In this case, we cannot tell which one is the better solution between X and Y.

In multi-objective optimization, the most commonly adopted notion of optimum is called *Pareto optimum*. A solution is Pareto optimal if there is no other solution that dominates it. In other words, Pareto optimal solutions are those solutions whose corresponding objective vector's elements cannot be all simultaneously improved [9].

The goal of a MOP is then to find a set of Pareto optimal solutions, rather than a single optimal solution as in single-objective optimization problems. These Pareto optimal solutions constitute the *Pareto front*, which is often illustrated as a diagram showing the corresponding values of the objective functions. A typical example of the Pareto front for an optimization problem with two objectives f_1 and f_2 is shown in Fig. 1.

B. The multi-objective zone mapping problem

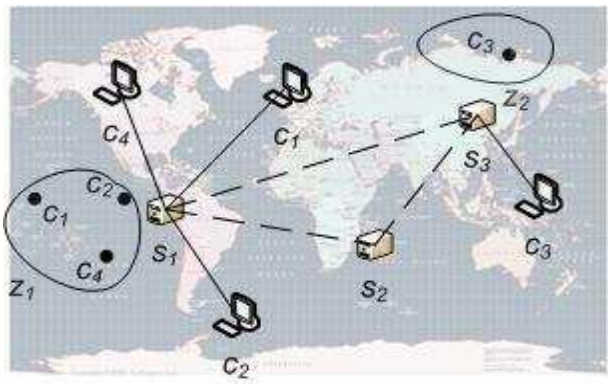


Fig. 2. Distributed server architecture with zone-based partitioning

In this paper, we focus on large-scale DVEs with a zone-based partitioning approach which maps the distinct virtual world zones over a set of geographically distributed servers (Fig. 2). Each zone is managed by only one server, and a client always connects directly to the server that manages its zone. Given an existing zone mapping, the multi-objective zone mapping problem (MOZP) concerns how to re-map each zone of the virtual world to an appropriate server to improve interactivity, and at the same time, maintain low overhead of zone migrations across servers due to the re-mapping process.

We use the following notations/definitions in the problem formulation.

- c_i - A client in the DVE.
- $C = \{c_1, ..., c_k\}$ - The set of all clients in the DVE.
- z_i - A zone in the DVE. This is also used to denote the set of all clients in a zone z_i.
- $Z = \{z_1, ..., z_n\}$ - The set of all zones in the DVE.
- s_i - A server in the DVE.
- $S = \{s_1, ..., s_m\}$ - The set of all servers in the DVE.
- R_{s_i} - The resource consumption on a server s_i. In this paper, R_{s_i} is measured by the total number of clients that are currently connecting to server s_i.
- R_{z_i} - The total amount of server resource used by all the clients that are interacting in the same zone z_i on the server that is hosting z_i. This is measured by the number of clients in zone z_i.
- C_{s_i} - The resource capacity of a server s_i.
- $d_{c_i s_j}$ - The round-trip network delay between a client c_i and a server s_j.
- D - The *delay bound* of a DVE. The delay bound indicates the maximum round-trip communication delay between a client and its server to maintain a desired level of interactivity for the DVE. For different types of DVEs, there are different delay bound requirements. For example, the delay bounds of First-

Person Shooter (FPS) games and car-racing games are about $250ms$ [10] and $100ms$ [11], respectively.

Similar to previous work in this area, we assume that the client-server communication delay in DVE is determined by the client-server network delay, since generally it would be easier to reduce the processing delays at the server side by adding more computing power, than to reduce message transmission delays in the network. In this paper, we assume that each server has a fixed capacity, say 400 clients, and if the load is within that limit, the server processing delay is negligible comparing to the network delay. In the following, the term "network delay" and "communication delay" are used interchangeably.

For interactive applications like DVEs, communication delay is the most important *Quality of Service* (QoS) parameter that the system provides to the clients [10]. In this paper, we say that a client is *with QoS* or *without QoS* if the communication delay between the client and its server is smaller or larger than the delay bound, respectively.

- X - A zone mapping solution, $X = [x_{ij}]$, where the decision variables $x_{ij} = 1$ if zone z_j is mapped to server s_i or $x_{ij} = 0$ otherwise.
- C_{ij} - The cost of mapping a zone z_j to a server s_i, which is calculated as follows.

$$C_{ij} = |\{c_k \in z_j : d_{c_k s_i} > D\}| \tag{1}$$

where $|\cdot|$ denotes the cardinality of a set.

C_{ij} measures the number of clients in a zone z_j that *do not* satisfy the delay bound D, i.e., without QoS. Therefore, by minimizing the total mapping cost when all zones are assigned, the total number of clients with QoS in the DVE would be maximized.

- $C(X)$ - The mapping cost of a solution $X = [x_{ij}]$, which is given by

$$C(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} x_{ij} \tag{2}$$

- $M(X)$ - $M(X)$ denotes the migration ratio, which is used to measure the overhead incurred by migrating zones across servers starting from an existing mapping $Y = [y_{ij}]$ to reach a new mapping $X = [x_{ij}]$. It is desirable to minimize this overhead when carrying out zone mapping. The migration ratio is calculated by normalizing the total number of clients in the zones that need to be migrated with the total number of clients in the system.

$$M(X) = \frac{\sum_{j=1}^{n} |z_j| M_j}{|C|} \tag{3}$$

where $M_j = 1$ if $\exists k \in \{1, ..., m\} : y_{kj} \neq x_{kj}$, and $M_j = 0$ otherwise.

- $[C(X), M(X)]$ - The vector of objective functions of each zone mapping solution X.

The multi-objective zone mapping problem can be formally stated as follows.

Definition 2.1: Multi-objective zone mapping problem (MOZP)

Let $I = \{1, ..., m\}$ and $J = \{1, ..., n\}$ be the set of indexes of servers and zones in the system, respectively ($n \gg m$). Find $X = [x_{ij}]$ that minimizes $[C(X), M(X)]$, with $x_{ij} = 1$ if zone z_j is mapped to server s_i or $x_{ij} = 0$ otherwise, subject to

$$\sum_{j=1}^{n} R_{z_j} x_{ij} \leq C_{s_i}, \forall i \in I, \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j \in J, \tag{5}$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J. \tag{6}$$

In the above formulation, Constraint (4) ensures that the capacity of each server is not exceeded. Constraints (5) and (6) are introduced to ensure that each zone is assigned to only one server.

Remark 2.1: The MOZP is NP-hard.

Remark 2.1 follows from the fact that the MOZP can be reduced to the original SOZP, which is NP-hard. In practice, this property implies that it is only feasible to find a good approximation to the true Pareto optimal set of solutions of the MOZP.

III. Multi-objective zone mapping algorithms

A. Linear programming (LP) relaxation with local search (LPLS)

The first algorithm, referred to as LPLS, that we propose for the MOZP uses LP relaxation technique to generate initial solutions, and then improves each of these solutions with a local search procedure, creating a pool of diverse solutions. An approximation of the Pareto optimal set for the MOZP is then created by removing all dominated solutions from the pool.

1) LP relaxation of the MOZP: LP relaxation is a widely used technique to design approximation algorithms for computationally hard problems. This technique is often used to transform an NP-hard problem, e.g., Integer Programming (IP), into a related LP problem, which can be solved in polynomial time. Usually, solutions obtained from solving the relaxed problem are not feasible for the original problem, but they can be used to gain useful insights into the solution that we are looking for [12].

An LP relaxation of the MOZP described in Section II can be obtained by replacing Constraint (6) by:

$$0 \leq x_{ij} \leq 1, \forall i \in I, \forall j \in J. \tag{7}$$

Note that with this new constraint, the variable x_{ij} now can take any real (fractional) value in $[0, 1]$, instead of the only two values 0 and 1 as in the original Constraint (6). Hence, a relaxed LP problem may provide infeasible solutions to the original problem.

Since the MOZP has two objectives, in order to solve the relaxed MOZP using available LP solvers like lp_solve (http://lpsolve.sourceforge.net), we need to convert one of the objectives into a constraint in order to have a single-objective LP problem. Here we choose to convert the migration overhead objective into a new constraint by limiting the percentage of clients that are migrated during the re-mapping process:

$$M(X) \leq \Delta \tag{8}$$

By setting a limit on migration ratio, say $\Delta = 0.5$, and solving the relaxed MOZP, we can obtain a mapping vector $X' = [x'_{ij}]$, and a corresponding vector of objective values $[C', 0.5]$, where C' is the relaxed mapping cost with a migration ratio of 0.5

Each value x'_{ij} is then rounded to the closest integer value (in this case the integer values are just 0 and 1). If $x'_{ij} = 1$, then this means that zone z_j is mapped to server s_i. The vector $X' = [x'_{ij}]$ with $x'_{ij} \in \{0, 1\}$ now stores the rounded variables as the initial solution for the next part of the LPLS algorithm. Note that this first part may result in an infeasible solution due to server capacity violation. The pseudocode of this first part is shown in Algorithm 1.

2) A local search procedure: The second part of LPLS is a local search procedure (Algorithm 2) aiming to produce a feasible and good solution X from each initial solution X' generated by Algorithm 1. This is done by attempting to fix the server capacity violations and further improving the quality of X' if possible. The search procedure forms a new solution from an infeasible solution by repeatedly reassigning a randomly selected zone z_j from an overloaded server to a new server s_k with sufficient capacity available which can provide the best (smallest) mapping cost C_{kj}. We also search for zones that are currently not assigned to any server[1], and assign these zones to the best available servers. Note that this local search procedure may not produce any feasible solution due to insufficient server capacity. In this case, a null solution will be returned.

[1]Due to the rounding process in Algorithm 1, some zones may not be assigned to any server.

Algorithm 1: Solving the relaxed MOZP to obtain an initial solution

Data: a migration ratio limit Δ

Result: a relaxed mapping solution X'

begin

relax the constraint (6) of the MOZP;

convert the overhead objective into a constraint based on the given migration ratio limit;

solve the relaxed LP problem to obtain mapping vector $X' = [x'_{ij}]$;

round all x'_{ij} to the nearest integer;

return X';

end

3) LPLS algorithm: Algorithm 3 shows the details of the LPLS algorithm, which relies on Algorithm 1 and Algorithm 2. To obtain a set of diverse zone mapping solutions, a set of migration ratio limits is predefined. For example, limits on migration ratios may be selected ranging from 0.1 to 1 by a step of 0.1. In this way, the final result set may have a wide range of solutions, from ones with low migration overhead but high mapping cost to those with high migration overhead but low mapping cost. For each migration ratio, an initial solution X' is obtained using Algorithm 1. With each X', at most a pre-specified number of t feasible solutions are then generated by repeatedly executing Algorithm 2. Finally, all dominated solutions are removed, and the algorithm returns the remaining set of non-dominated solutions, which is an approximation of the true Pareto optimal set.

B. Hybrid multi-objective evolutionary algorithm (HMOEA)

Evolutionary algorithms (EAs) refer to a class of meta-heuristics that are inspired by the natural evolution and selection process for solving computationally hard optimization problems. Recently, quite a number of MOEAs have been proposed [9] to tackle multi-objective optimization problems. The main motivation behind is due to the nature of all MOEAs employing a population-based approach, hence, multiple Pareto optimal/approximate solutions could be found in a single run of the algorithm. Recall that for multi-objective problems, a single solution that simultaneously optimizes all objectives is generally not possible, therefore the capability of MOEAs to generate multiple approximate alternative solutions in a single run is really appealing.

In this section, we propose a modified version of the NSGA-II algorithm [13] for our MOZP. NSGA-II is a well-known and highly successful MOEA in various application domains. The modified algorithm

Algorithm 2: Local search procedure to correct and improve an initial solution

Data: a relaxed mapping $X' = [x'_{ij}]$

Result: a new mapping solution $X = [x_{ij}]$, or NULL if the new solution is not feasible

begin
  initialize $x_{ij} = x'_{ij}$, $\forall i, j$;
  foreach $s_i \in S$ **do**
    while $R_{s_i} > C_{s_i}$ **do**
      randomly select a zone z_j currently mapped to s_i;
      if *no server with sufficient capacity to take* z_j **then**
        return NULL;
      end
      map z_j to a server s_k with smallest C_{kj} and $R_{s_k} + R_{z_j} \leq C_{s_k}$: set $x_{kj} = 1$;
    end
  end
  foreach $z_j \in Z$ **do**
    if z_j *is not mapped to any server* **then**
      if *no server with sufficient capacity to take* z_j **then**
        return NULL;
      end
      map z_j to a server s_l with smallest C_{lj} and $R_{s_l} + R_{z_j} \leq C_{s_l}$: set $x_{lj} = 1$;
    end
  end
  return X;
end

incorporates informed initialization and local search into the original NSGA-II, and is hence named Hybrid MOEA.

1) Non-dominated sorting genetic algorithm (NSGA-II): NSGA-II [13] uses a fast non-dominated sorting approach for ranking a population according to the level of non-domination and a density estimator to maintain population's diversity. Here, we briefly review these key components below. We refer the readers to the original paper [13] for more details.

Algorithm 3: LPLS algorithm

Data: a set of migration ratio limits $MR = \{\Delta | 0 \leq \Delta \leq 1\}$

Data: t - number of times to run Algorithm 2 in each loop

Result: a set of non-dominated solutions P

```
begin
    initialize the set of non-dominated solutions P = Φ;
    foreach Δ ∈ MR do
        run Algorithm 1 with migration ratio limit Δ to get initial mapping X';
        cnt = 0;
        while cnt < t do
            run Algorithm 2 with X' as input;
            if a feasible solution X is generated then
                add X to P;
            end
            cnt = cnt + 1;
        end
    end
    remove all dominated solutions in P;
    return P;
end
```

- **Non-dominated sorting:** NSGA-II uses a mechanism called non-dominated sorting to organize the entire population P into mutually exclusive classes referred to as fronts. In each front, member solutions are all non-dominated. Solutions belong to the first front are assigned the highest rank, while those of the last got the lowest rank. Generally, solutions in the higher ranked fronts produce more offsprings during the evolution process towards the true Pareto front.
- **Density estimation:** To ensure the diversity of the population, NSGA-II calculates an estimation of the density of solutions surrounding a particular solution in the population using a crowding distance measure. Essentially, each solution will be assigned a crowding distance value equal to the average distance of the two solutions on either side of this solution along each of the objectives.

2) HMOEA: The key components of the HMOEA are as follows.

- **Genetic representation and operations:** For the MOZP, we adopt a similar solution representation to the one used in [7]. Each gene in the chromosome represents a zone, where the position of the gene corresponds to the zone index, e.g., the second gene in a chromosome represents zone z_2. Each gene takes an integer value, which is the index of the server that this zone is mapped to. In this way, we can ensure that a zone is mapped to only one server, while one server can take multiple zones.

For the genetic operators, standard single-point crossover, standard mutation (randomly change the value of a gene) and binary tournament selector are used, as recommended by the original NSGA-II algorithm.

- **Informed initialization:** Generally, good methods of generating a population of initial solutions would help EAs quickly obtain a decent approximation of the Pareto optimal set. Hence, instead of using randomly generated solutions as usually seen in existing MOEAs, we propose to use LP relaxation to create a set of meaningful and relatively good initial solutions for the MOEA. The method used here is similar to the previously proposed LPLS algorithm, but for generating initial solutions we do not evaluate and eliminate dominated solutions.

- **Local search to improve offsprings:** Note that during the evolution process of a MOEA, offsprings generated by performing crossovers and mutations are usually infeasible. Hence, we propose to use local search (as described in Algorithm 2) on each offspring to meet the constraints of server capacity, as well as to further improve the quality of the solution.

Algorithm 4: HMOEA algorithm

Data: M - number of initial solutions

Data: N - a limit on the number of new solutions to be generated by the algorithm

Result: a set of non-dominated solutions

```
1 begin
2     generate M initial solutions;
3     while total number of new solutions generated is less than N do
4         apply NSGA-II's genetic operators to the initial population;
5         run Algorithm 2 for each new solution X;
6         apply NSGA-II's ranking and density estimation operators for the new population;
7     end
8     return the front with the highest rank;
9 end
```

Algorithm. 4 shows the HMOEA algorithm. First, we generate M initial informed solutions. Then, in each evolution round, the binary tournament selector is used to choose two solutions for mating, and the standard single-point crossover as well as the mutation operator are then applied. The offsprings will further be improved via Algorithm 2. The last operators to be executed at the end of a round are NSGA-II's ranking and density estimation operators. Finally, the front with the highest rank is returned after N new offspring solutions have been generated. For this paper, we implement the HMOEA algorithm into the open-source software package JMetal (http://jmetal.sourceforge.net/).

IV. Evaluation methodology

For evaluating the proposed multi-objective zone mapping algorithms, we employ realistic network latency models based on real Internet-wide measurements plus network topologies, and use different client distributions in both the virtual world and the network.

A. Network latency models

1) Latency models based on real measurements: We use the tool DS^2 (Delay Space Synthesizer) [14], which has been built to help designers/researchers of distributed systems to conduct more realistic simulations/emulations for evaluating different design alternatives. We generate two latency matrices using DS^2 (denoted as REAL1 and REAL2 in Table I).

Generating latency matrices for our experiments using DS^2 is a procedure with two steps. First, DS^2 creates two Internet latency models using two real-life measurements, each of them measuring approximately 16 million pairs of geographically distributed IP addresses (the data are downloadable from [15]). Each model is constructed with a default *scaling* factor of 10, i.e., given the size of the measured latency matrix is 4000×4000, DS^2 can scale this data and build a realistic Internet latency model for 40000 nodes with this scaling factor. This scaling factor helps researchers to model much larger Internet delay spaces with limited measurements.

In the second step, for each model, we construct a latency matrix by uniformly selecting 3000 nodes, and calculate the round-trip latency between each pair of nodes using data provided by the model. For more information on latency models in DS^2, please refer to the original paper [14].

2) Latency models based on network topologies: For diversity, we also employ latency models based on network topologies generated by the popular topology generator BRITE (http://www.cs.bu.edu/brite), and real topologies collected from the Internet. Table I lists these latency models, with TOPO1 and TOPO2 produced by BRITE. For the hierarchial topology (TOPO3), we collected a real-world AS (Autonomous

TABLE I

NETWORK LATENCY MODELS

Name	Model	Nodes	Links
REAL1	DS^2, Matrix1 from [15]	3000	-
REAL2	DS^2, Matrix2 from [15]	3000	-
TOPO1	Flat, Waxman (BRITE)	3000	6000
TOPO2	Flat, Barabasi-Albert (BRITE)	3000	5997
TOPO3	Hierarchical (real topology)	4466	8963

System[2])-level topologies (29 nodes) from http://www.ssfnet.org/Exchange/gallery. We then used the US AT&T continental IP backbone topology (154 nodes) [16] as the router-level topologies to construct a hierarchical topology.

To calculate network delays from the topologies, we take the link delays generated by BRITE, and employ both shortest-path routing and AS-level hierarchical routing [17] where applicable.

B. Workload models

TABLE II

DISTRIBUTION TYPES

Type	1	2	3	4
Cluster in PW	No	Yes	No	Yes
Cluster in WW	No	No	Yes	Yes

Similar to previous studies [2], for more robust and reliable simulation results, we have simulated a number of combinations of client distributions in both the virtual world (VW) and the physical world (PW) (the network). Table II shows these combinations. The number of clients may be larger in some specific zones of the virtual world than others, due to the clustering of clients in these more popular zones. For example, in online games, clients may be clustered in the zones with large amounts of game resources such as energy, gold, etc. In the physical world, due to the differences in time zones of geographically distributed clients, at a specific time, the number of online clients in the DVE may be quite different for different geographic regions [18].

[2]An Autonomous System refers to a group of IP networks that is usually under the administration of a single organization.

More specifically, in the clustered client distribution in the VW, a popular zone would have about 4 times more clients on average compared to a "normal" zone. Similarly, to simulate the clustering of clients in the PW, some nodes in the network are randomly selected to have a larger number (about 3 times more) of clients than the rest.

C. Other alternative algorithms

1) Baseline: The Baseline algorithm starts from an existing zone mapping. It then runs the local search procedure (Algorithm 2) for a pre-specified number of times to correct server capacity violation and re-assign zones to further improve the current mapping. In each iteration, it generates a new, feasible, and possibly better solution. In the end, dominated solutions will be removed, and the algorithm returns the set of remaining solutions. We should note that Baseline algorithm tends to produce solutions with low migration ratios but potentially high mapping cost, since the algorithm's main goal is to improve an existing mapping. Nevertheless, being simple and easy to implement, it serves as a useful basis for comparison with LPLS and HMOEA.

2) Super-optimal: The Super-optimal (SP) algorithm provides the lower bound (MOZP is a minimization problem) of the true Pareto front by repeatedly solving the relaxed MOZP using Algorithm 1 with a large number of different limits on migration ratios, say from 0 to 1, by a step of 0.01. Recall that MOZP is NP-hard, so the true Pareto front in this case is not attainable. Though in most cases solutions provided by the SP algorithm are not feasible, it can be used as a reference for comparing various performance measures of multi-objective algorithms.

D. Performance measures

How to accurately assess the performance of multi-objective optimization algorithms is a challenging research topic in its own right [19]. In this paper we use two of the most popular performance measures, namely hypervolume [20] and epsilon [19]. These measures assess different properties of a non-dominated solution set, and provide a single performance value for the set.

The epsilon measure for a pair of solution sets P^A and P^B, denoted as $\epsilon(P^A, P^B)$, calculates the minimum shift necessary for P^A to be dominated by P^B, i.e., after this shift operation, we can say that P^A is worse than P^B. In order to compare two solution sets produced by two alternative algorithms, we would need to calculate their epsilon values against a reference solution set. For example, using the solution set provided by the SP algorithm as a reference, we can say that LPLS outperforms HMOEA if $\epsilon(P^{SP}, P^{LPLS}) < \epsilon(P^{SP}, P^{HMOEA})$.

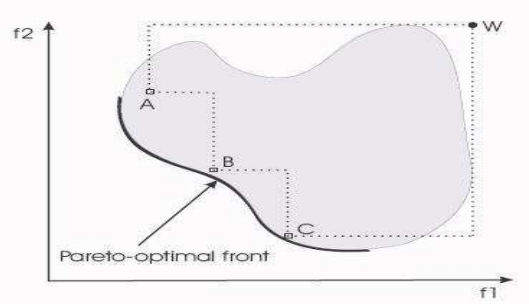


Fig. 3. An example of the hypervolume

The hypervolume measure calculates the volume covered by members of a solution set in the objective space. Algorithms with larger hypervolume values are more desirable. An example of the hypervolume for a minimization problem is shown in Fig. 3 (which has been taken from [21]). The hypervolume formed by solution set $\{A, B, C\}$ is the region enclosed within the discontinuous line, where W is a point with worst objective values in the objective space provided by a reference algorithm.

E. Default parameters

Unless otherwise stated, the following assumptions and default values are used in the simulations. The clients are uniformly distributed in the physical world as well as in the virtual world. There are 5000 clients, 100 zones, and 20 servers with a total capacity to accommodate 7000 clients. The minimum capacity of each server is 100 clients. Locations of servers are uniformly distributed in the physical world. The latency model in use is REAL1, and the DVE delay bound D is set to $100ms$.

To initialize LPLS and HMOEA, we solve the relaxed MOZP (using Algorithm 1) for a number of migration ratio limits ranging from 0.1 to 1 by a step of 0.1. Each algorithm (Baseline, LPLS, and HMOEA) generates 1000 new solutions before termination. All algorithms have similar execution times (roughly two minutes for each algorithm in all experiments). The number of initial solutions, default crossover and mutation rates of HMOEA are set to 100, 0.9 and 0.01, respectively, as recommended by the software package JMetal. All algorithms start from a random zone mapping scheme.

Mann-Whitney U statistical significance tests [8] are used to validate experiment results where applicable. It is a non-parametric statistical test[3] for comparing two unpaired groups of samples. The main result is the p-value ranging from 0 to 1. Informally speaking, if this value is small, we can conclude

[3]Such a test does not rely on assumptions that the data in question follow a given probability distribution.

that the two populations have different medians with high confidence. On the other hand, if p-value is large, we might not be able to conclude that the medians are different.

V. Results and analysis

In this section, we describe the experiment results which are obtained from 30 independent simulation runs for each algorithm.

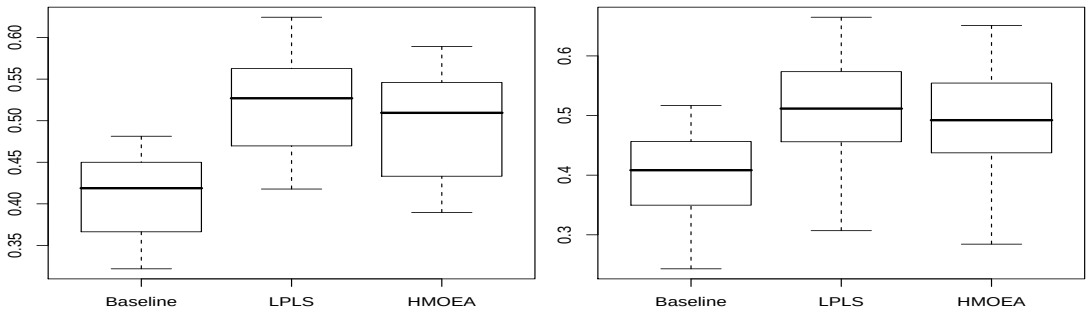
A. Impacts of client distributions

We have tested and compared Baseline, LPLS, and HMOEA with different client distributions to verify their performances in a wide range of practical situations. Fig. 4 shows box-and-whisker plots[4] of the representative results for the REAL1 latency model. The results show that, regardless of the client distributions in the physical or virtual world, LPLS and HMOEA obviously outperform Baseline in terms of both hypervolume and epsilon values. Indeed, the Mann-Whitney U statistical tests give extremely small p-values (the highest p-value is 0.0038 for Fig. 4(b)) when comparing LPLS to Baseline, implying high confidence in the performance difference.

We have also noticed that HMOEA offers almost no performance advantage over LPLS. This comes as a surprise, as HMOEA is based on the well-known NSGA-II which has been applied successfully in various application domains. It employs much more complicated operations compared to LPLS, such as fast non-dominated ranking and density estimation to evolve the solution set towards the true Pareto front. As we have observed in some cases, e.g., Fig. 4(a) and 4(b), Mann-Whitney U tests show that LPLS performs even better than HMOEA in terms of hypervolume with some substantial degree of confidence (the p-values obtained are 0.09 and 0.12, respectively).

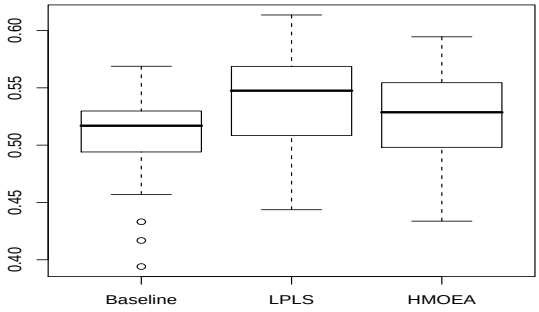
In addition, Fig 5 plots the Pareto fronts obtained by different algorithms. In Fig. 5, the x-axis shows the performance gaps in percentage between the mapping cost $C(X)$ provided by each solution X and the best possible lower bound $C(X^*)$, where X^* is the solution obtained by solving the relaxed MOZP (as described in Section III-A.1) without any constraint on the migration ratio. The performance gap is defined as $\frac{C(X)-C(X^*)}{C(X^*)} \times 100$. It is seen from Fig. 5 that HMOEA does not provide better solutions than LPLS.

[4]Box-and-whisker plots display groups of data using the smallest observation, lower quartile, median, upper quartile, and largest observation, together with outliers.

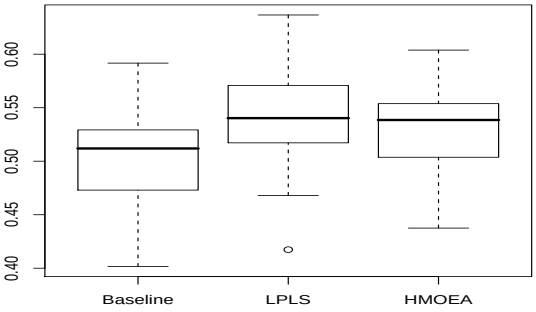


(a) Dist. type 1 - Hypervolume

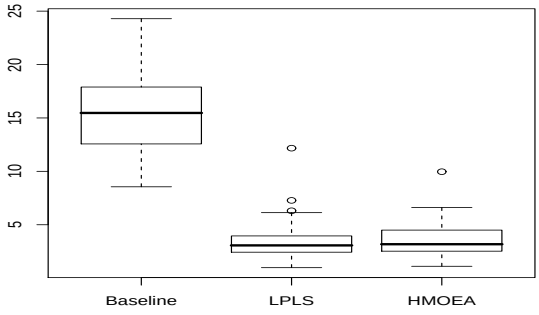
(b) Dist. type 2 - Hypervolume



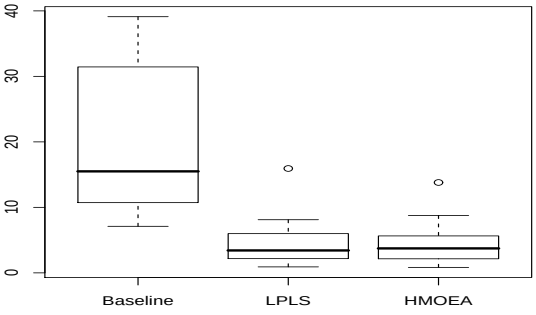
(c) Dist. type 3 - Hypervolume



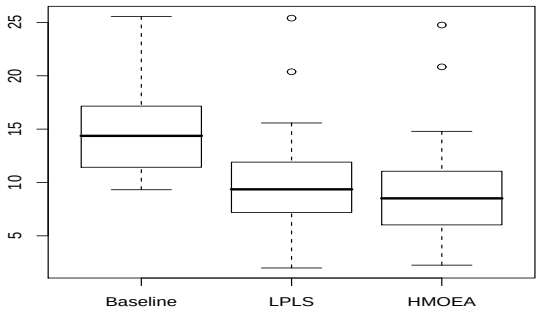
(d) Dist. type 4 - Hypervolume



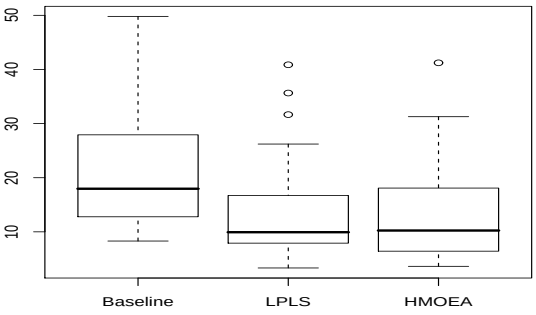
(e) Dist. type 1 - Epsilon



(f) Dist. type 2 - Epsilon

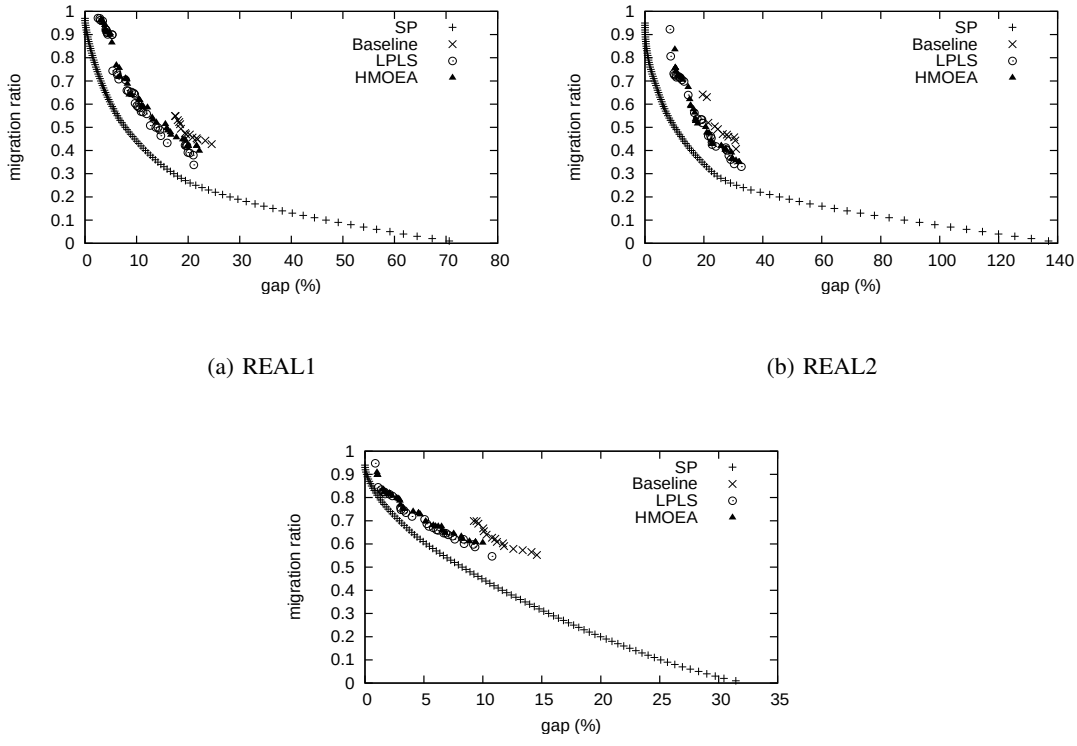


(g) Dist. type 3 - Epsilon



(h) Dist. type 4 - Epsilon

Fig. 4. Impacts of client distributions



(a) REAL1

(b) REAL2

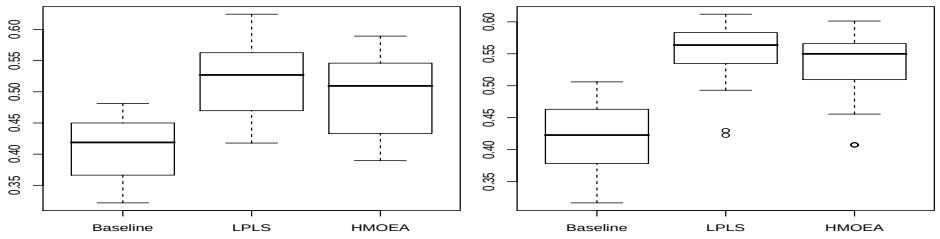
(c) TOPO3

Fig. 5. Pareto fronts of different algorithms with different latency models

B. Impacts of latency models

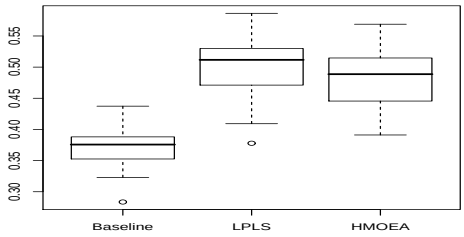
Due to the dynamic nature of the Internet, and the fact that there is no definitive network latency models currently, we feel that it is necessary to validate the performance of our algorithms with a diverse range of latency models. In this paper, we use five different models as previously discussed in Section IV for our experiments.

Fig. 6 shows the box-and-whisker plots of hypervolumes and epsilons for our algorithms when tested using different latency models. From the observations and the Mann-Whitney U statistical tests that we have carried out, we can conclude that regardless of latency models, LPLS and HMOEA always significantly outperform the Baseline algorithm, while performance differences between LPLS and HMOEA are not really significant.

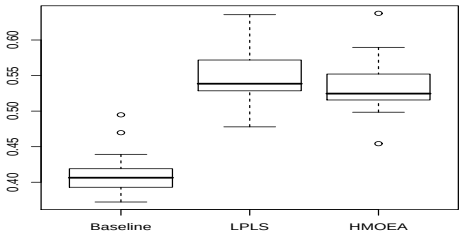


(a) REAL1 - Hypervolume

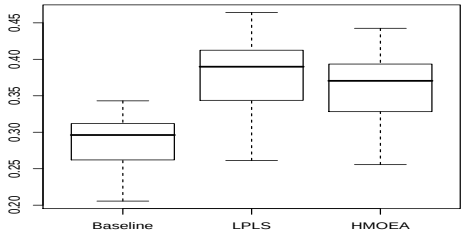
(b) REAL2 - Hypervolume



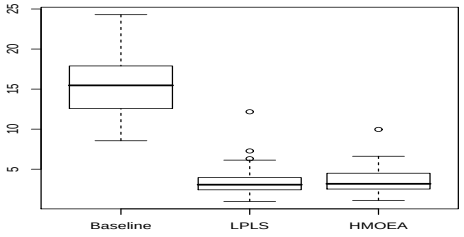
(c) TOPO1 - Hypervolume



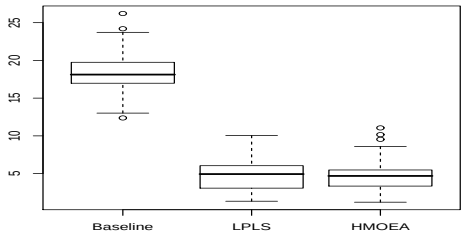
(d) TOPO2 - Hypervolume



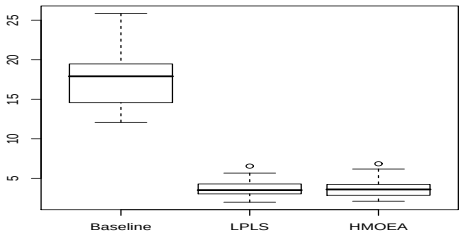
(e) TOPO3 - Hypervolume



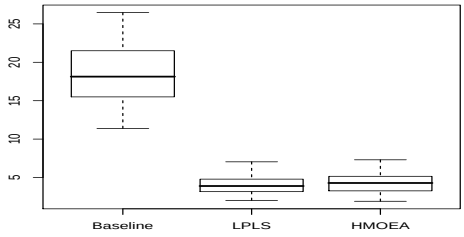
(f) REAL1 - Epsilon



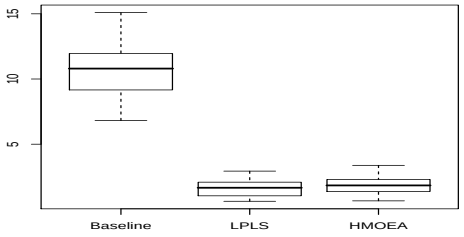
(g) REAL2 - Epsilon



(h) TOPO1 - Epsilon

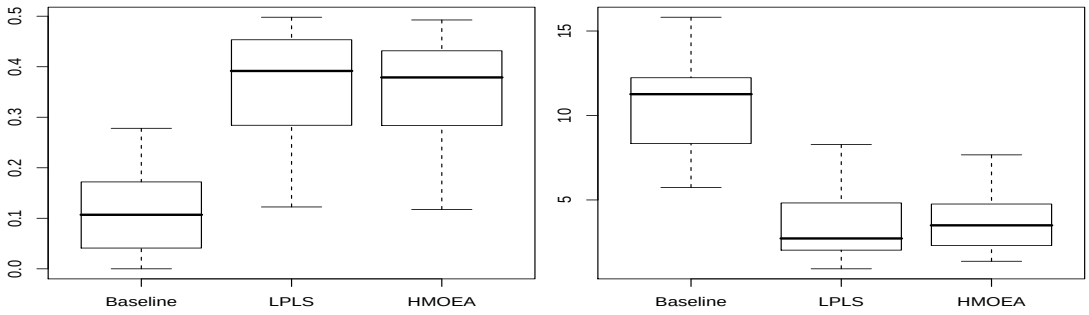


(i) TOPO2 - Epsilon



(j) TOPO3 - Epsilon

Fig. 6. Impacts of network latency models



(a) Hypervolume (b) Epsilon

Fig. 7. Impacts of existing zone mapping - Hypervolume and epsilon

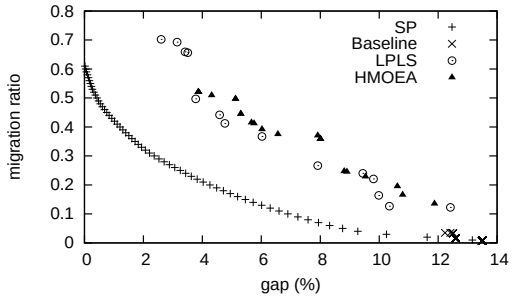


Fig. 8. Impacts of existing zone mapping - sample Pareto front

C. Impacts of client dynamics

Finally, we evaluate the new algorithms considering client dynamics. Typically, in a DVE, after sometime some clients may leave the virtual world or move from one zone to another zone, while new clients will be coming in. Due to such movements, it may be more practical to start the re-mapping operation early to prevent the current mapping from significant degradation in terms of interactivity. Hence, the multi-objective zone mapping algorithms should be periodically executed, say after a certain number of clients have changed server or have left/joined the virtual world.

In this experiment, we start re-mapping the zones after 1000 new clients have joined, 1000 existing clients have left and another 1000 existing clients have changed their zones. Note that the clients and their new zones (in case of joining/changing) are selected randomly for the experiment. From the results shown in Fig. 7 and 8, we can confirm that LPLS and HMOEA still outperform Baseline, while HMOEA

offers no significant advantage over LPLS.

VI. Related work

To the best of our knowledge, there is no existing work that directly addresses the latency-driven multi-objective zone mapping problem described in this paper. Previous work [2], [3], [7] had dealt with the single-objective zone mapping problem, that is, the only objective to be minimized is the mapping cost, which is measured by the total number of clients that have client-server round-trip network latencies larger than a predefined delay threshold. This single-objective zone mapping approach may cause significant network and processing overhead due to zone migrations across servers.

On the other hand, research on how to assign clients to servers in DVEs is usually formulated as a resource-driven problem [5], [6], [22], [23], [24]. In [25], the authors also focused on providing QoS for DVEs, but they only considered the processing delay caused by limited CPU resource. Such approaches may damage the interactivity of the DVE, since clients may be assigned to servers that are far away from them in terms of network delays. [26] considered a latency-driven assignment of clients to servers, but this work is limited to voice communication in massively multi-player games.

In [27], a distributed algorithm has been proposed for game clients to select the best server in a mirrored architecture, taking into account the network delay between clients and servers. The mirrored architecture replicates the DVE zones at multiple servers. This approach shares some similarities with the web server replica placement problem in Content Distribution Networks (CDNs) [28]. Similarly, [29] considers optimal client-to-mirror-assignment, for which the objective is to minimize average client-to-mirror delay considering player joins and leaves, and mirrors with limited capacity. However, unlike web replications, DVE replication faces more complicated consistency issues [30] which may damage the users' experience in interacting with the virtual world. In our approach, only one server has the control over the state of a zone, thus consistency is easier to maintain.

Following a latency-driven approach, [31] and [32] have proposed a decentralized middleware capable of utilizing information on clients' geographical locations to reduce the overall latency for the majority of users in a region of the virtual world. The latency reduction is accomplished by migrating a game region to a server closer to the clients. The focus of this work seems to be more on supporting services, e.g., name service, for game state migration. Such services can be of great benefit for implementing our proposed algorithms. On the other hand, the heuristics for determining good server locations in [31], [32] are rather simple, and without any provision for minimizing the game migration overhead.

VII. Conclusions

The MOZP aims to determine how to map the virtual world zones to different distributed servers without incurring much zone migration overhead. Most of the time, there is no single, definitive answer to this multi-objective optimization problem. Instead, the results are often expressed as a set of non-dominated solutions. In this paper, we have formulated this NP-hard problem, which calls for efficient zone mapping heuristics. We then proposed two new algorithms, LPLS and HMOEA, which could provide approximations to the set of Pareto optimal solutions for the MOZP.

Extensive simulation results on realistic models built from Internet-wide latency measurements between millions of host pairs have shown that our new algorithms significantly outperform a baseline algorithm in terms of two mostly used measures in multi-objective optimization, namely hypervolume and epsilon, while having similar execution time. This implies that LPLS and HOMEA are able to provide good quality and diverse solution sets, which can help DVE designers/developers/maintainers to select the most appropriate re-mapping solution depending on the situation. A final note is that we prefer LPLS over HMOEA, since the former has simpler implementation, and produces comparable or sometimes even better results than HMOEA.

Acknowledgement

This work is supported in part by the Singapore National Research Foundation under Grant NRF2007IDM-IDM002-052.

References

[1] S. Singhal and M. Zyda, *Networked virtual environments: design and implementation*. Reading, MA: Addison-Wesley, 1999.

[2] D. N. B. Ta and S. Zhou, "A Network-centric Approach to Enhancing the Interactivity for Large-Scale Distributed Virtual Environments," *Elsevier Computer Communications*, vol. 29(17), pp. 3553–3566, 2006.

[3] D. Ta and S. Zhou, "A Two-phase Approach to Interactivity Enhancement for Large-Scale Distributed Virtual Environments," *Elsevier Computer Networks*, vol. 51(14), pp. 4131–4152, 2007.

[4] V. Nae, A. Iosup, S. Podlipnig, R. Prodan, D.H.J.Epema, and T. Fahringer, "Efficient Management of Data Center Resources for Massively Multiplayer Online Games," in *Proc. of ACM/IEEE SuperComputing Conference on High Performance Networking and Computing*, 2008.

[5] D. N. B. Ta and S. Zhou, "A Dynamic Load Sharing Algorithm for Massively Multi-Player Online Games," in *Proc. of the 11th IEEE International Conference on Networks*, 2003.

[6] J. Lui and M. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," *IEEE Transaction on Parallel and Distributed Systems*, vol. 13(3), 2002.

[7] D. Ta, S. Zhou, W. Cai, X. Tang, and R. Ayani, "Efficient Zone Mapping in Distributed Virtual Environments," in *Proc. of the ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulations*, New York, 2009.

[8] G. W. Corder and D. I. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, 2009.

[9] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.

[10] T. Henderson and S. Bhatti, "Networked games: a QoS-sensitive application for QoS-insensitive users?" in *Proc. of the ACM SIGCOMM*, 2003.

[11] L. Pantel and L. Wolf, "On the Impact of Delay on Real-Time Multiplayer Games," in *Proc. of NOSSDAV*, 2002, pp. 23–29.

[12] S. Walukiewicz, *Integer programming*. Kluwer Academic Publishers, 1991.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[14] B. Zhang, A. N. T. S. Eugene Ng, R. Riedi, P. Druschel, and G. Wang, "Measurement-based analysis, modeling, and synthesis of the internet delay space," in *ACM SIGCOMM/USENIX Internet Measurement Conference (IMC)*, 2006.

[15] "Internet delay space synthesizer - data," *Available at http://www.cs.rice.edu/ bozhang/ds2/matrix/*, retrieved May 2009.

[16] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "Generating realistic isp-level network topologies," *IEEE Communication Letters*, 2003.

[17] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin, "The Impact of Routing Policy on Internet Paths," in *Proc. of IEEE INFOCOM*, 2001.

[18] W. chang Feng and W. chi Feng, "On the geographic distribution of online game servers and players," in *Proc. of NetGames*, 2003.

[19] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, "Performance Assessment of Multiobjective Optimizers: An Analysis and Review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[20] E. Zitzler and L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study," in *Conference on Parallel Problem Solving from Nature (PPSN)*, 1998, pp. 292–301.

[21] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham, "Abyss: Adapting scatter search to multiobjective optimization," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 4, pp. 439–457, 2008.

[22] R. Chertov and S. Fahmy, "Optimistic load balancing in a distributed virtual environment," in *Proc. of NOSSDAV*, 2006.

[23] D. T. Admed and S. Shirmohammadi, "A microcell oriented load balancing model for collaborative virtual environments," in *Proc. of the IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, 2008.

[24] M. Lim and D. Lee, "A task-based load distribution scheme for multi-server-based distributed virtual environment systems," *Presence: Teleoperators and Virtual Environments*, vol. 18, no. 1, pp. 16–38, 2009.

[25] P. Morillo, S. Rueda, J. Ordua, and J.Duato, "A Latency-Aware Partitioning Method for Distributed Virtual Environment Systems," *IEEE Transaction on Parallel and Distributed Systems*, vol. 18(9), 2007.

[26] C. D. Nguyen, F. Safaei, and P. Boustead, "Optimal assignment of distributed servers to virtual partitions for the provision of immersive voice communication in massively multiplayer games," *Elsevier Computer Communications*, vol. 29(9), 2006.

[27] K. W. Lee, B. J. Ko, and S. Calo, "Adaptive Server Selection for Large Scale Interactive Online Games," *Computer Networks*, vol. 49, pp. 84–102, 2005.

[28] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *Proc. of IEEE INFOCOM*, 2001.

[29] S. D. Webb and S. Soh, "Adaptive client to mirrored-server assignment for massively multiplayer online games," in *MMCN '08: Fifteenth Annual Multimedia Computing and Networking*, 2008.

[30] S. Zhou, W. Cai, B. S. Lee, and S. J. Turner, "Time-space consistency in large-scale distributed virtual environments," *ACM Transactions on Modeling and Computer Simulation*, vol. 14(1), pp. 31–47, 2004.

[31] P. B. Beskow, K.-H. Vik, P. Halvorsen, and C. Griwodz, "Latency reduction by dynamic core selection and partial migration of game state," in *Proc. of Workshop on Network and Systems Support for Games (NetGames)*, 2008, pp. 79–84.

[32] P. Beskow, K.-H. Vik, P. Halvorsen, and C. Griwodz, "The partial migration of game state and dynamic server selection to reduce latency," *Springer's Multimedia Tools and Applications Special Issue on Massively Multiuser Online Gaming Systems and Applications*, 2009.