Research Repository UCD

Title	VM reassignment in hybrid clouds for large decentralised companies: A multi-objective challenge
Authors(s)	Saber, Takfarinas, Thorburn, James, Murphy, Liam, B.E., Ventresque, Anthony
Publication date	2017
Publication information	Saber, Takfarinas, James Thorburn, Liam Murphy B.E., and Anthony Ventresque. "VM Reassignment in Hybrid Clouds for Large Decentralised Companies: A Multi-Objective Challenge" 79, no. Part 2 (2017).
Publisher	Elsevier
Item record/more information	http://hdl.handle.net/10197/9041
Publisher's statement	This is the author's version of a work that was accepted for publication in Future Generation Computer Systems. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Future Generation Computer Systems, 2017-06. DOI: 10.1016/j.future.2017.06.015
Publisher's version (DOI)	10.1016/j.future.2017.06.015

Downloaded 2023-04-07T10:52:29Z

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

VM Reassignment in Hybrid Clouds for Large Decentralised Companies: a Multi-objective Challenge

Takfarinas Saber^{a,*}, James Thorburn^b, Liam Murphy^a, Anthony Ventresque^{a,*}

^aLero@UCD, School of Computer Science, University College Dublin, Dublin 4, Ireland

^bIBM Software Group, Toronto, Canada

Abstract

Optimising the data centres of large IT organisations is complex as (i) they are composed of various hosting departments with their own preferences and (ii) reassignment solutions can be evaluated from various independent dimensions. But in reality, the problem is even more challenging as companies can now choose from a pool of cloud services to host some of their workloads. This hybrid search space seems intractable, as each workload placement decision (seen as running in a virtual machine on a server) is required to answer many questions: can we host it internally? In which hosting department? Are the capital allocators of this hosting department ok with this placement? How much does it save us and is it safe? Is there a better option in the Cloud? Etc. In this paper, we define the multi-objective VM reassignment problem for hybrid and decentralised data centres. We also propose H2-D2, a solution that uses a multi-layer architecture and a metaheuristic algorithm to suggest reassignment solutions that are evaluated by the various hosting departments (according to their preferences). We compare H2-D2 against state-of-the-art multi-objective algorithms and find that H2-D2 outperforms them both in terms of quantity (approx 30% more than the second-best algorithm on average) and quality of solutions (19% better than the second-best on average).

Keywords: Large Decentralized Data centers, Hybrid Clouds, VM Reassignment, Multi-objective Optimization, Hybrid Algorithms, Interval Objective

1. Introduction

There has been a proliferation of cloud services in the past years, from virtual machines (VMs) of different flavours to out-of-the-box platforms (e.g., ready to use Machine Learning tools). The many benefits of these cloud solutions [21], including but not limited to their cost, have accelerated the adoption of the Cloud for all sorts of companies [38]. However, modern large and often global organisations seem more reluctant to outsource to the Cloud than small and medium companies [38], with only 17% of them reported having 1000+ VMs in the Cloud. Some of the many reasons for this slow process are the complexity of their software systems [25], the types of their workloads [45] and the privacy/security of their data and products [37] – as well as the distribution and segmentation of the data centres (DCs) of these large and global companies [42]: they possess many hosting departments with ‘competing’ or even ‘conflicting’ demands and requirements.

However, the hybrid cloud solution [34], i.e., mixing private infrastructure and public cloud services, is now seen as a potential solution for these large companies [38] as it gives them the benefits of both worlds. On one hand, the Cloud provides quick infrastructure provisioning and deployment [31],

while on the other hand, companies can still maintain their own infrastructure when exact characteristics of servers [5], performances [32] and reliability [45] are important.

VM reassignment in data centres is a known challenge [14], especially as real problems are often *multi-objective* [7, 42], i.e., solutions are evaluated by decision makers based on different dimensions or objectives. However, most solutions to this problem in the literature use a weak definition of multi-objective [35], i.e., a linear aggregation of objectives [1, 29, 23]. Related work also often miss the complexity of large companies, where the infrastructure is not monolithic but distributed over decentralised hosting departments with their own preferences [43, 44]. The optimisation of the companies’ IT infrastructure is then more challenging as decisions have to take into account the preferences of the capital allocators of each hosting departments and the interests of the company as a whole. Hybrid cloud solutions, while explored extensively in the literature, have not been addressed in the context described above – multi-objective and decentralised. In particular, the various pricing policies [39, 24] and their variability make the question of finding good hybrid reassignment solutions even more challenging.

This paper is the first to address the problem of *multi-objective VM reassignment for large and hybrid decentralised data centres*. In this paper: (i) we consider that data centres are decentralised, i.e., capital allocators of hosting departments express their own preferences, (ii) we also consider the different dimensions of the infrastructure optimisation (cost of running

*Corresponding authors

Email addresses: takfarinas.saber@ucdconnect.ie (Takfarinas Saber), jthorbur@ca.ibm.com (James Thorburn), liam.murphy@ucd.ie (Liam Murphy), anthony.ventresque@ucd.ie (Anthony Ventresque)

the IT infrastructure, reliability, migration cost) as independent and we propose solutions optimising all these objectives together and (iii) we consider hosting some of the VMs in the Cloud as an option.

Example: As a motivating example, let's suppose a large distributed company with different hosting departments (or VCs, for Virtual data Centres), each of them managed by capital allocators (CAs) who have their own preferences regarding the placement of VMs on their servers. Some CAs may want to reduce the energy footprint of their VC, while others see the licensing cost as more important; or some CAs view the reliability of their infrastructure as crucial (e.g., if they run critical systems), while other CAs consider response time as essential and are mindful of VMs' co-location. When reassigning its VMs, the company can also choose one of the multiple cloud solutions out there and pay the cost (which may only be known partially in advance, i.e., an interval of possible prices) of the deployment of VMs. Either way, possible reassignments have to respect constraints of the IT infrastructure and satisfy the various preferences of CAs.

Figure 1 shows possible VM reassignments (from a to g) in two dimensions: (i) reliability and (ii) cost of running some of the VMs in the Cloud (the lower the better for both). Note that the cloud cost is not always known in advance so the values in this dimension are often intervals (except for solution e which does not outsource any VM). The broken segments with black dots represent the non-dominated reassignments, i.e., those that are better than the others in at least one particular objective (the good ones). For example, solution f is worse than b on both objectives and is not considered as one of the good reassignments (hence the white dots on f). Notice that while solution d has a worse reliability than c and potentially (given the overlapping intervals of c and d) a worse cloud cost, it is possible that d eventually gets a better cloud cost and we keep d in the list of non-dominated solutions at this stage. At the end, we end up with five good reassignments: a, b, c, d and e. These solutions can then be evaluated locally by decision makers, such as: "a and b have a good reliability but also have high cloud costs, whereas c's reliability is worse, but still not as worse as d's and brings a good reduction in the public cloud cost despite its large interval. Solution e does not cost anything in terms of cloud cost but has the worst reliability amongst them – so we favour c and keep d as a backup plan for when VM prices rise too much".

Figure 1: A decision space in two dimensions (reliability and cost of VMs hosted in the Cloud). The price variability in the Cloud means the solutions are actually in the form of intervals. The (potentially) non-dominated solutions are the broken lines with black dots.

H2-D2: In this paper, we propose *H2-D2*, a multi-objective VM reassignment system for large and hybrid decentralised DCs (*H2-D2* stands for Hybrid algorithm for Hybrid Decentralised Data centres). *H2-D2*'s search space is composed of the different hosting departments of the data centres of a large company and the different public cloud solutions. *H2-D2* suggests reassignments of VMs to either the hosting departments or the public cloud locations depending on various objectives: cost of running servers in the hosting departments, migration cost of the VMs, reliability of the servers and cost of hosting VMs in the Cloud. The non-dominated possible reassignments are then suggested to the company's decision makers who can navigate them and choose one solution over another based on their preference and current focus.

H2-D2 is an adaptation of our previous work [42] to the more challenging (and realistic) cloud environment. We propose a new model/problem definition, with new constraints and a new objective (cloud cost) – this particular objective is an interval objective, the cost of migrating workload to the Cloud being difficult to estimate precisely. This makes the problem significantly different and more challenging than the one we addressed previously.

We compare the performance of *H2-D2* against systems with various reassignment algorithms on a realistic data set. We show that *H2-D2* outperforms all of them both in terms of quantity (number of non-dominated solutions): *H2-D2* gets 29.99% more solutions than the second-best on average, and quality (hypervolume): *H2-D2* is ~20% better than the second-best on average.

After describing the related work (Section 2), we formally define the *multi-objective VM reassignment problem for hybrid and decentralised data centres* (Section 3). This formal problem definition includes a large number of constraints and four objectives, i.e., directions in the search space considered independent. Then we describe our solution, which is based on an architecture for large decentralised data centres [42] but adds the critical and challenging public cloud element (Section 4). This new element makes the problem more complex and *H2-D2* has to take into account: (i) individual CAs' preference, (ii) price fluctuation of cloud services and (iii) optimisation of the global infrastructure (at the general managers' level). We then compare *H2-D2* against various reassignment algorithms on a realistic data set – inspired from a challenge proposed by Google and modified to make it more realistic for the complex context we address here (Section 5). We show that *H2-D2* outperforms (Section 6) all of them both in terms of number of non-dominated solutions (*H2-D2* gets ~30% more solutions than the second-best on average) and quality of these solutions (*H2-D2* achieves ~20% better hypervolume in comparison to the second-best on average).

2. Related Work

This section is a short literature survey of three areas: (i) machine/VM reassignment, (ii) VM reassignment in Decentralised and hybrid DCs and (iii) multi-objective VM reassignment.

2.1. Machine/VM Reassignment

The VM reassignment problem (a.k.a., machine reassignment problem) is an extension of the d -Dimensional Bin Packing [33]. While the d -Dimensional Bin Packing aims at reducing the number of bins, the VM reassignment problem considers moving items between an already given set of bins (a.k.a., machines or servers). This problem tries to optimise different goals while integrating many topic-related constraints. In the context of managing resources in cloud computing environments, several optimisation problems have been defined and studied. Doddavula et al. [16] reassigned tasks to different machines by comparing First-Fit algorithms. Beloglazov et al. [7] tackled the energy consumption of clouds' data centres with resource allocation heuristics. Stillwell et al. [47] introduced heterogeneous machines in the problem definition.

Due to the increasing popularity and scale of the Cloud, Google (one of the big players with Google Cloud Engine) proposed a challenge at an Operations Research event (ROADEF / EURO 2012 [19]) and provided real large scale instances. The challenge attracted 48 participants, with algorithms of different types [1]: the majority were based on a local search (e.g., Large Neighbourhood Search (LNS [51])), but others used Greedy Randomized Adaptive Search (GRASP [18]), Hill Climbing (HC [50]), or a combination of a solver with some other optimisation techniques (e.g., Mixed-Integer Linear Programming (MILP) solver [23]).

2.2. VM Reassignment in Decentralised and Hybrid Contexts

VM reassignment has been studied for different types of data centres in the literature [30]. Some of them address the existing topologies within private DCs going from centralised ones where data centres are taken as a global entity, to hierarchical ones with two or multiple composition layers. Others study the possibility of improving the assignment through collaboration between the different DCs either with (cloud federations [28]) or without (multi-clouds [17]) prior agreements, and also as a combination of both approaches (inter-clouds [46]).

Large Enterprise data centres, however, are oftentimes decentralised and do not fit perfectly into the aforesaid types. A formulation of the VM reassignment problem for such data centres has been proposed [42] and considers that each site is independent when making its own machine placement. The study provides a full multi-objective model for the VM reassignment in such infrastructures and evaluates several algorithms. However, this work does not consider the availability of public clouds and the opportunities they offer in a hybrid cloud fashion.

Examining the available literature in hybrid clouds, we see that the majority of these studies focus on architecture / management aspects of such environments and their challenges [25]. Studies concerned with resource allocation are often either (i) focused on the selection of the public cloud by formalising it as a brokering problem (e.g., with multiple cloud providers [49] or with a distributed cloud [20]) or (ii) focused on the private DC with the possibility of bursting to public clouds [15] to cope with a momentary excess of demand in resources.

Works that deal with VM allocation in hybrid data centres are often mono-objective (e.g., driven by overall cost reduction [3], SLA satisfaction [15] or minimisation of VM leasing cost [10]). In addition, they either (i) address a small scale scheduling problem with a simplistic private DC architecture: Chunlin and LaYuan [13] define a set of routines for an optimal resolution of the problem, while Bittencourt et al. [10] surveys common scheduling algorithms and studies their performance on the problem, or (ii) put an emphasis on the private DC sub-problem [12]. However, there are a few that take the complex architecture [52] and workload [45] of large Enterprise DCs into account, but not as decentralised DCs though.

2.3. Multi-objective VM Reassignment

The multi-objective version of the VM Reassignment problem has been described recently as an important research challenge for data centres [7]. Although many works dealing with this problem can be found in the literature [35], most of them either consider systems of small and unrealistic scales [26], or use a 'weak' multi-objective formulation (e.g., combining the objectives using a weighted sum [1, 29, 23] or limiting their study to two objectives [57]).

It is only recently that the VM reassignment problem has been tackled in a true multi-objective way. The authors proposed a three-step method to address the problem, using a modified version of the ROADEF challenge [43]. In [44] and [41] the authors proposed a linear formulation of the problem and studied the relevance of using MILP and Constraint-Based LNS (CBLNS) solvers. They showed that MILP solvers (e.g., IBM ILOG CPLEX) can only solve small instances, whereas CBLNS achieves worse results than MILP solvers on small instances, but scales well to larger ones.

The few works in the literature which study the multi-objective VM allocation in hybrid data centres are formalising the problem as a scheduling problem with VMs/jobs running for a limited duration. Raju et al. [36] propose an algorithm that is inspired by the behaviour of bats (i.e., echo-localisation and hibernation) to schedule VMs in an energy efficient way, however, despite having an algorithm based on two objectives, the evaluation of its performance is not. Hu et al. [22] use a Genetic Algorithm for the job scheduling problem that tries to limit its number of evaluations in order to fasten its optimisation for both tasks completion and computation cost. Zuo et al. [62] extend the initial definition of the scheduling problem and add deadlines to jobs completion. They also propose an ant colony algorithm to address the problem.

3. Formal Problem Definition

This section provides the first formalisation of the multi-objective VM reassignment problem in large and hybrid decentralised data centres. Such data centres are composed of: virtual machines (VMs), physical machines (PMs, i.e., hosts or servers), hosting departments (VCs, i.e., virtual data centres), public cloud locations (Ps) with the type of VMs they offer (Fs). Each data centre is also defined by a large number of constraints (see below Section 3.1) and a set of objective functions

(i.e., dimensions that define the search space). The optimisation problem itself is defined by two types of mappings of VMs on PMs: (i) more ‘general’ reassignment of VMs to either one of the VCs that compose the data centre or the public cloud, and (ii) ‘exact’ placement of VMs on the PMs belonging to the VCs where the VMs have been assigned. The formal definitions below are inspired by recent papers proposed by Google [19] and the authors of the current paper [42]. The main differences are the introduction of the multi-objective aspect of the problem and the hybridisation of the decentralised data centres.

A decentralised data centre is composed of a set C of VCs: $C = \{c_1, \dots, c_n\}$. Each c_i is given a set $\mathcal{M}_i \subseteq \mathcal{M}$ of PMs, $\mathcal{M}_i = \{m_1, \dots, m_n\}$. Each m_j has several resources $r \in \mathcal{R}$ (e.g., RAM, CPU, disk), in a limited capacity $Q_{m_j,r}$. Resources are either transient ($r \in \mathcal{TR} \subseteq \mathcal{R}$, e.g., RAM and disk) if they are consumed by both origin and destination PMs during a VM migration, or non-transient otherwise (e.g., CPU): $r \in \mathcal{R} \setminus \mathcal{TR}$.

A hybrid data centre can outsource VMs to the different public cloud locations $P_l \in \mathcal{P}$. We consider that there is no limit on the resources offered by public cloud locations, but each P_l varies on the type of flavours $F^l \in \mathcal{F}$ of VMs they offer. Every flavour $f \in F^l$ provides a quantity $Q_{f,r}$ of resource $r \in \mathcal{R}$ and has a price varying within the interval $p_f = [\underline{p}_f, \overline{p}_f]$. To some extent, a flavour is a type of VM.

A hybrid decentralised data centre hosts a set $\mathcal{V}$ of VMs, $\mathcal{V} = \{v_1, \dots, v_l\}$. Each VM v_k has to be reassigned to either a VC $Re(v_k) = c_i \mid c_i \in C$ in a machine $M(v_k) = m_j \mid m_j \in \mathcal{M}_i$, or to a public cloud location $Re(v_k) = P_l$ with a flavour $M(v_k) = f_j^l$ (we also use the notation $M_0(v_k) = m_j$ for the original PM of v_k in case of a migration). The quantity of resource r that every v_k needs is fixed to $d_{k,r}$. VMs are sometimes grouped into services $\mathcal{S} = \{s_1, \dots, s_p\}$, with $s_p = \{v_p^1, \dots, v_p^q\}$. A service can be seen as a duplication of the same process to ensure the resiliency of the application to defects.

3.1. Constraints of the Problem

The problem of VM reassignment considers a large number of hard constraints at three different levels:

3.1.1. VC Level

Every VM placement on a PM within the same VC has to satisfy a set of constraints:

Capacity constraints: PMs’ resource capacity should not be exceeded by the demand of VMs that are either hosted on them (for non-transient resources) or VMs migrating from/to them (for transient resources). $\forall m_j \in \mathcal{M}$:

$$\begin{cases} \sum_{v_i \in \mathcal{V} \mid M_0(v_i)=m_j \vee M(v_i)=m_j} d_{v_i,r} \leq Q_{m_j,r} & \text{if } r \in \mathcal{TR} \\ \sum_{v_i \in \mathcal{V} \mid M(v_i)=m_j} d_{v_i,r} \leq Q_{m_j,r} & \text{otherwise} \end{cases} \quad (1)$$

Anti-cohabitation constraints: Services that run on multiple VMs may prefer placing each of them on distinct PMs.

$$\forall v_i, v_j \in \mathcal{V}, i \neq j, \forall s_p \in \mathcal{S}, (v_i, v_j) \in s_p^2 \Rightarrow M(v_i) \neq M(v_j) \quad (2)$$

The variation and difference in network connections between PMs of a VC can impact VM placement decisions made by CAs. We use the notion of neighbourhood to capture this and we denote the neighbourhood of every PM m_i by $N(m_i)$.

Dependency constraints: When a service s_t depends on s_u ($s_t \hookrightarrow s_u$), every VM v_i in s_t has to at least be in the neighbourhood of one VM v_j from s_u :

$$s_t \hookrightarrow s_u \Rightarrow \forall v_i \in s_t, \exists v_j \in s_u \mid N(M(v_i)) = N(M(v_j)) \quad (3)$$

3.1.2. Public Cloud Level

Deploying a VM to the Cloud is usually done through tools or APIs depending on the cloud providers, with service level agreements decided ahead of the deployment. Thus there is little control over the characteristics of the hosts on which the VMs will be deployed (e.g., exact characteristics of the PMs, location). Although cloud providers are known for the quality and reliability of their infrastructures, it is difficult with the Cloud to address conflict and dependency constraints, making the Cloud irrelevant for certain VMs.

Fitting VM flavour: Public clouds list a certain number of VM flavours to choose from, with a pre-set quantity of resources that the user/client should not exceed. This number can be large as there might be VMs set for specific targets (CPU intensive, RAM intensive, GPU, etc.), but VM flavours often do not exactly match the needs of our VMs. Therefore, a VM flavour with higher specifications has to be booked, leading to potential resource wastage (hence the $\leq$ sign in equation 4).

$$\forall v \in \mathcal{V}, M(v) = f_i^l \Rightarrow \forall r \in \mathcal{R}, d_{v,r} \leq Q_{f_i^l,r} \quad (4)$$

Conflicting VMs: In private DCs, VMs that belong to the same service are considered conflicting, and cannot share the same host. On the other hand, public clouds behave as black boxes and do not provide any knowledge regarding the placement of VMs within their location. Therefore, it is not possible to send two VMs belonging to the same service to the same location of the public cloud without taking a risk of having them sharing the same host. To avoid this, we have to reassign VMs of the same service to different locations. $\forall v_i, v_j \in \mathcal{V}, \forall s_t \in \mathcal{S}, \forall P_l, P_k \in \mathcal{P}, (v_i, v_j) \in s_t^2, i \neq j$:

$$Re(v_i) = P_l, Re(v_j) = P_k \Rightarrow l \neq k \quad (5)$$

Dependent VMs: A service s_t depending on a service s_u , obliges VMs of s_t to be in the neighbourhood of at least one VM from the service s_u . Since there is little or no notion of neighbourhood (equipment sharing a high-speed connection) when it comes to public clouds, it is not possible to outsource any VM from the service s_t .

$$s_t \hookrightarrow s_u \Rightarrow \forall v_i \in s_t, Re(v_i) \notin \mathcal{P} \quad (6)$$

3.1.3. Manager Level

Most of the constraints concern the two levels above (i.e., VC and public cloud), with the exception of the spread constraints. Spread constraints express the fact that for security

reasons, some services might require their VMs to be assigned to a number of VCs and/or public cloud locations higher than a value σ_s that is considered safe (e.g., for redundancy reasons).

$$\forall s \in \mathcal{S}, \sum_{l \in \mathcal{CUP}} \min(1, |\{v \mid v \in s \wedge Re(v) = l\}|) \geq \sigma_s \quad (7)$$

3.2. Objectives to Optimise

We introduce now the cost functions which define the various objectives. We consider in our work four objectives: two that depend on the characteristics of the VCs (i.e., reliability and electricity costs), one that is given by the price of hosting VMs on the Cloud (i.e., cloud cost), and one that is obtained from both VCs and public clouds' characteristics (i.e., migration cost).

3.2.1. Reliability Cost

Resource contentions (e.g., cache) can be generated when hosting multiple VMs on the same physical machine. VM re-assignment algorithms are often assumed to buffer the reserved resources as safety capacity [56], hence allowing them to not consider the potential issues of VMs co-location. Recently, it has been proposed to adapt the size of the buffers based on the workload of VMs [27] and their resource utilisation patterns. Therefore, trying to assign VMs with similar resource needs at any given time to different hosts, reducing the risk of violating resource requirements. We follow in our work the classical direction and consider a fixed safety capacity as the way of measuring the *reliability cost*, i.e., how likely is it to generate resource contentions in a host? We consider a physical machine m_j as reliable if its load $U_{m_j,r}$ for every resource $r \in \mathcal{R}$ is less than a safety capacity $\rho(m_j, r)$. We define the *reliability cost* that is associated with a machine as follows:

$$\rho(m_j) = \sum_{r \in \mathcal{R}} \max(0, U_{m_j,r} - \rho(m_j, r)) \quad (8)$$

3.2.2. Electricity Cost

$\epsilon(m_j)$ of a machine $m_j \in \mathcal{M}$ is defined by its electricity consumption multiplied by the price per unit of electricity at its location. It is complex to model the electricity consumption of a machine. We use in our work a simplified model [58] whereby the electricity consumption can be represented as a linear function based on the total usage in CPU:

$$\epsilon(m_j) \begin{cases} \gamma_{m_j} \times (\alpha_{m_j} \times Q_{m_j,CPU} + \beta_{m_j}) & \text{if } m_j \text{ is running} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where γ_{m_j} is the price of electricity at m_j 's location, β_{m_j} is the electricity consumption when m_j is idle, and α_{m_j} is the tangent of the CPU consumption.

3.2.3. Migration Cost

Migrating a VM v goes through three different steps, with each step having its own cost [54]: preparing the VM for the migration: $\mu_1(v, M_0(v))$, transmitting the VM: $\mu_2(v, M_0(v), M(v))$

and installing the VM in the destination PM: $\mu_3(v, M(v))$. These three costs are difficult to quantify as they are based on several parameters (e.g., VM size and network topology).

$$\begin{aligned} \mu(v, M_0(v), M(v)) = & \mu_1(v, M_0(v)) \\ & + \mu_2(v, M_0(v), M(v)) \\ & + \mu_3(v, M(v)) \end{aligned} \quad (10)$$

Note that when a VM v is reassigned to a public cloud location $Re(v) = P_l$ using a VM flavour $M(v) = f_i^l$, it is only possible to give an estimation for μ_2 depending on the public cloud location and the bandwidth associated with the VM flavour. This is also the case for μ_3 . Given that specifications of the hosting machines are often unknown, we can only approximate this time based on the size of the VM and the processing power conferred by the VM flavour.

3.2.4. Cloud Cost

The (public) cloud cost sums up the price of every VM assigned to the public cloud. Since the fee of a VM flavour in a public cloud location is given as an interval, this objective is computed using interval arithmetic [9]:

$$CC = \sum_{v \in \mathcal{V} \wedge Re(v) \in \mathcal{P}} \left[\underline{p_{M(v)}}, \overline{p_{M(v)}} \right] \quad (11)$$

Note that electricity and cloud costs cover different things in the finances of companies and are often impossible to compare. While they are both paid in "dollars" we see them as two independent objectives - which is really what they are in the enterprise world.

Definition 1 (Placement). Given a VC c_i , its PMs $\mathcal{M}_i$ and a set of VMs $\mathcal{V}_i = \{v_1^i, \dots, v_2^i\}$, a placement of $\mathcal{V}_i$ on $\mathcal{M}_i$ is a mapping: $Placement_i : \mathcal{V}_i \mapsto \mathcal{M}_i$, such that $Placement_i(v) \rightarrow m$, which satisfies the hard constraints in Section 3.1.1.

The preference of CAs in any particular VC c_i corresponds to minimising a function $Pref(c_i)$ that is defined by three constants w_i^1 , w_i^2 and w_i^3 :

$$\begin{aligned} Pref(c_i) = & w_i^1 \times \sum_{m_j \in \mathcal{M}_i} \rho(m_j) \\ & + w_i^2 \times \sum_{v_k \in \mathcal{V}_i} \mu(v_k, M_0(v_k), Placement(v, \mathcal{M}_i)) \\ & + w_i^3 \times \sum_{m_j \in \mathcal{M}_i} \epsilon(m_j) \end{aligned} \quad (12)$$

Definition 2 (Reassignment). A VM reassignment in a hybrid decentralised DC ReAssign is a mapping: $ReAssign : \mathcal{V} \mapsto \mathcal{P} \cup \mathcal{C}$, such that $ReAssign(v_k) \rightarrow c_i$ or $ReAssign(v_k) \rightarrow P_l$, that ensures the satisfaction of the constraints defined in Section 3.1.

Given that we are in the context of a pure multi-objective optimisation, we aim at minimising all the objectives independently (i.e., minimising the multiple objective functions 8, 9, 10 and 11) instead of minimising them in a combined fashion (i.e., as a single function).

We obtain from the optimisation a set of non-dominated solutions (solutions that are better than the others on at least one particular objective), thus defining a *Pareto front*.

4. H2-D2

We present a system that addresses the multi-objective VM reassignment problem in the context of large and hybrid decentralised data centres. H2-D2 aims at satisfying the preferences of CAs of individual VCs while offering managers of the private DC the ability to extend their resources using public clouds, hence improving their VM placement solutions in the multi-objective search space. H2-D2 has two different modules responsible for the reassignment of VMs and their placement (see grey boxes in Figure 2).

Figure 2: Overview of H2-D2. The grey boxes represent the two main components: (i) the reassignment of VMs to VCs and cloud platforms and (ii) the exact placement of VMs in the servers of the VCs. Note that each solution is evaluated by cloud providers and VCs’ CAs as the companies’ managers may not know the exact price of hosting the VMs in the Cloud or the characteristics of each placement (depend on VCs’ CAs preferences).

Reassignment

Decision makers at the top level of the private DC run an extension of GeNePi [43] to handle four objectives of possibly interval values. GeNePi is a hybrid metaheuristic that executes successively three metaheuristics to (i) find a set of good initial solutions scattered over the search space using a greedy algorithm (GRASP), (ii) combine them to generate solutions of higher variety in terms of objective trade-offs (using a genetic algorithm: NSGA) and (iii) find more non-dominated solutions using local search techniques (PLS). Decision makers ensure that all the outsourced VMs satisfy the hard constraints. They also send all of the VMs reassigned within the private infrastructure to their corresponding VCs, where a placement is tried: if every VC succeeds in finding a valid placement, then the real cost of the solution is updated (the solution is possibly discarded if considered not good enough with regards to the four objectives). In the event that the reassignment is violating some constraints, GeNePi modifies it and tries resubmitting it.

GeNePi starts with GRASP’s constructive phase [18] as a first step: VMs are arranged decreasingly following their dependencies and requirements. The top ones are successively assigned to VCs that allow the satisfaction of the spread and dependency constraints and have a utility value beyond a threshold α . For every solution, a certain ratio of VMs that are not concerned by dependencies and that are eligible to exit the private DC are sent to non-conflicting public cloud locations. The

higher the ratio, the more VMs are outsourced. This ratio is updated in a dynamic (using dichotomy to explore a large number of outsourcing degrees) and reactive (when constraints are violated we consider that we are reaching the limits in outsourcing capacity, thus the ratio can only decrease) fashion.

Then, GeNePi applies NSGA [8]: a genetic algorithm which in addition to coping with interval objectives distinguishes itself by (i) selecting and making a tournament between four parents before mixing the winners (using different operators: crossover and mutation) and (ii) mixing the offspring population (i.e., population resulting from an evolution) with the original one, thus only keeping the elite solutions for a faster convergence towards the optimal Pareto front. In our algorithm, we use One-point and Two-point ‘cuts’ as crossovers where random cut positions are selected in the parents’ chromosomes and their respective parts exchanged. We also use a mutation which randomly reassigns a VM to either a public cloud location or a VC.

Last but not least, GeNePi applies PLS [2]: a local search which only looks for new solutions in the neighbourhood of non-dominated solutions that are the most isolated (as they are the most likely to refine the non-dominated set). GeNePi uses 1-exchange (changing the assignment of one VM at a time) and swap (exchanging assignments between two VMs) as operators for generating the neighbourhood of each solution.

We use the same parameters for GeNePi as in [43]. Moreover, a ratio of the global execution time (i.e., the time limit) is allocated to each step which allows a better control over the execution time and a reactivity towards the complexity of the instance. Therefore, a third of the execution time is given to GRASP. In the event that the size of NSGA’s initial population is not reached, this time can be extended up to half of the time limit. As NSGA achieves large improvements, we allow it the longest execution time with half of the global execution time. PLS gets only one sixth of the global time as it has a vocation to refine the Pareto front by finding more non-dominated solutions, and not to improve its quality.

Placement

Every VC is running two different algorithms within a limited time that is the same for all VCs. The first one finds a feasible placement for the VMs they were assigned (using a First-Fit Decreasing (FFD) algorithm and reparation), and the second one optimises it according to the VC’s cost function (using a Hill Climbing algorithm). If the placement is feasible, its costs are sent back to the general manager.

Algorithm 1 describes the first step that is responsible for finding an initial placement that respects constraints of the VC. This step starts by placing VMs that were initially in the VC in their original machines, which minimises the consumption of transient resources and reduces the migration cost. The remaining VMs are placed using an FFD algorithm after sorting them based on their dependency and their resource requirements. A reparation of the placement is attempted with a maximum of two moves in case some VMs cannot be placed without violating any of VC’s constraints. The placement is cancelled if it is not successful after performing the reparation procedure.

Algorithm 2 summarises the second step which uses a late acceptance Hill Climbing algorithm to optimise the initial VM placement. Algorithm 2 runs until reaching one of the two stopping criteria: (i) no more positive savings can be made (impossible to make any extra improvement on the VC's objective function using a local move), or (ii) exceeding the allowed time limit (defined by the general manager for all VCs). As it was the case with PLS, HC repeatedly explores placements around the current one using local moves (i.e., 1-exchange and swap) and moves to the placement that improves VC's objective function the most (to the best positive saving).

Algorithm 1: VC's initial VM placement

```

input :  $\mathcal{M}_i$ ;  $PMs$ ,  $V_i$ ;  $VMs$ 
output:  $V_{not\_placed}$ :  $VMs$ 
 $V_{not\_placed} \leftarrow \emptyset$ ;  $V_{placed} \leftarrow \emptyset$ ;
// VC's initial VMs  $V_{init}$  are not moved
 $V_{new} \leftarrow V_i \setminus V_{init}$ ;
// Placing the rest of  $VMs$ 
 $sort(V_{new}, \text{Dependency and Requirement})$ ;
for  $v \in V_{new}$  do
    if  $exists(FFD(v))$  then
         $M(v) \leftarrow FFD(v)$ ;  $V_{placed} \leftarrow V_{placed} \cup \{v\}$ ;
    else
         $V_{not\_placed} \leftarrow V_{not\_placed} \cup \{v\}$ ;
// Reparation
for  $v \in V_{not\_placed}$  do
    if  $repaired(v, \mathcal{M}_i, V_{placed})$  then
         $V_{not\_placed} \leftarrow V_{not\_placed} \setminus \{v\}$ ;
return  $V_{not\_placed}$ ;

```

Algorithm 2: Optimisation of VC's initial VM placement

```

input :  $p$ : Placement,  $W$ : Weights,  $M$ : Moves,  $t$ : Time
output:  $p$ : Placement
do
     $\mathcal{N} \leftarrow generateNeighbours(p, M)$ ;
     $(p', saving) \leftarrow getBestSaving(p, W, \mathcal{N})$ ;
    if  $saving > 0$  then
         $p \leftarrow p'$ 
while  $saving > 0 \ \& \ execTime < t$ ;
return  $p$ ;

```

5. Experimental Setup

In this section, we evaluate the performance of H2-D2 and state-of-the-art multi-objective VM reassignment systems for large and hybrid decentralised DCs. We use in this evaluation a large data set provided by Google and two quality metrics: the quantity of non-dominated solutions found (how many non-dominated solutions are found) and the quality of the found solutions (how much of the search space do they cover).

Our algorithms are developed in C++ and the experiments are run on a machine with 24 Intel® Xeon® 2.20GHz CPUs and 64GB of RAM, running Ubuntu 12.4 LTS (64bit).

5.1. Data Set

The Google ROADEF challenge data set [19] consists of a large number of realistic data centres with their VMs, PMs, resources, services and VCs. The goal of the challenge was to compare optimisation techniques on a mono-objective problem: optimising a single weighted sum of objectives in geographically distributed DCs (called locations in the challenge) without the notion of decentralisation. We have modified the data set provided in the challenge to make it more realistic to large decentralised IT companies where various hosting departments “collaborate” and “compete” with the central managers [42]. In this new data set, we introduce different objectives for the central managers and various preferences for the capital allocators of the different VCs. In the current paper, we also add information about public clouds (VM flavours, i.e., resource capacities and price intervals).

Table 1 summarises the characteristics of the hybrid decentralised DCs in our data set (every instance corresponds to one DC). The data set consists of three sets of instances: (i) instances a_1_* (considered easy) have between 2 and 4 resources, 4 to 100 PMs, 100 to 1,000 VMs, 79 to 981 services, 4 to 50 VCs and 15 to 80 VM flavours in 3 to 9 public cloud locations, (ii) instances a_2_* (considered medium) have 12 resources, 50 or 100 PMs, 1,000 VMs, 129 to 180 services, 25 VCs and 30 to 100 VM flavours in 3 to 10 public cloud locations and (iii) instances b_* (considered difficult) have 6 or 12 resources, 100 or 500 PMs, 5,000 to 20,000 VMs, 1,732 to 15,025 services, 10 to 50 VCs and 12 to 117 VM flavours in 4 to 9 public cloud locations.

Instance a_1_3 for example represents a DC that has 3 resources, 100 PMs, 1,000 VMs, 216 services, 25 VCs and 9 public cloud locations offering 63 VM flavours all together (the total number of VM flavours = number of VM flavours per location $\times$ number of public cloud locations). All the instances have a number of public cloud locations and a number of VM flavours generated randomly within the intervals [3, 10] and [3, 14] following offers for ‘general purpose’ VMs by major public clouds. Table 1 also shows the maximum allowed execution time for each optimisation algorithm. These values were discussed with industry experts from the domain and reflect their common knowledge about the decision making process in large companies. In particular, they represent consolidation of workloads done on a regular basis, e.g., monthly or quarterly and the time that can be given to the tools supporting capital allocators’ decision process.

5.2. Metrics

It is a known fact that it is difficult to compare the quality of two multi-objective algorithms [59]. There exists in the literature several quality indicators for assessing their performance such as coverage, diversity and number of non-dominated solutions [55]. The selection of metrics depends on the desired characteristics of the obtained Pareto fronts. In our work, we consider two perspectives to evaluate the results of the different algorithms:

Quantity of non-dominated solutions: this measure is intuitive yet it is important as we would like to provide the decision

Instance	# Resources	# PMs	# VMs	# Services	# VCs	# Cloud Locations	# VM Flavours	Exec Time (s)
a_1_1	2	4	100	79	4	8	64 (8×8)	30
a_1_2	4	100	1,000	980	4	3	15 (5×3)	1,200
a_1_3	3	100	1,000	216	25	9	63 (7×9)	1,200
a_1_4	3	50	1,000	142	50	8	80 (10×8)	1,200
a_1_5	4	12	1,000	981	4	3	36 (12×3)	1,200
a_2_1	3	100	1,000	1,000	1	8	56 (7×8)	3,600
a_2_2	12	100	1,000	170	25	3	30 (10×3)	3,600
a_2_3	12	100	1,000	129	25	10	60 (6×10)	3,600
a_2_4	12	50	1,000	180	25	8	56 (7×8)	3,600
a_2_5	12	50	1,000	153	25	10	100 (10×10)	3,600
b_1	12	100	5,000	2,512	10	9	117 (13×9)	7,200
b_2	12	100	5,000	2,462	10	3	24 (8×3)	7,200
b_3	6	100	20,000	15,025	10	8	56 (7×8)	7,200
b_4	6	500	20,000	1,732	50	4	12 (3×4)	7,200

Table 1: Characteristics of the different data centres (one instance equals one data centre) and the allowed execution time for each of them.

makers with multiple options that can be navigated before they make a decision. We use the *size of the set of non-dominated solutions*, i.e., the number of solutions on the Pareto front (those that are better than any other on a particular combination of objectives).

Figure 3: The solutions found by an algorithm (on the left) have the form of intervals as the price of VMS in the Cloud is not known in advance. The two figures on the right show the results under the optimistic and pessimistic scenarios with the solutions on the Pareto front (black dots) and the hypervolume (grey area).

Quality of the set of non-dominated solutions: the more of the search space is covered by the set of solutions the better it is for the decision makers who can then explore a variety of solutions and pick the one that corresponds best to their needs. We use the *hypervolume* (or S-metric) [61] as a measure of the quality of a set of solutions. The hypervolume is now popular [11] in the optimisation community to compare non-dominated sets obtained by different multi-objective algorithms. It measures the dominated hypervolume of space between the non-dominated solutions and a reference point selected far from it. This reference point must be the same for all of the algorithms and differs only between instances.

In our model, the exact price of hosting VMs in the Cloud

is not known in advance but is taken from an interval of values (which reflects the fact that price in the Cloud is not always determined in advance). The reassignment solutions found by the algorithms then depend on the exact price of hosting VMs in the Cloud, which is represented in Figure 3 (left figure) by the broken segments, i.e., the intervals of cloud costs. The segments with black dots represent the solutions on the Pareto front (the good ones) and the segments with white dots represent the other (not interesting) solutions. Now, as the cloud cost depends on the exact price of hosting the VMs in the Cloud, the evaluation of the solutions formed by the algorithms may vary. To tackle this problem, we have created these different scenarios:

- one, that we call optimistic, where the price of hosting VMs in the Cloud is always the lowest. This corresponds to the case when the capital allocators are ‘lucky’ and have to pay less than expected for the VMs.
- on the contrary, the pessimistic scenario gives the highest possible price in the interval to all VMs.
- average scenario gives the average price to all of the VMs hosted in the Cloud.

Figure 3 shows the impact of the optimistic and pessimistic scenarios on the results of an algorithm: (i) the hypervolume (grey area) is different (bigger in the optimistic scenario) and (ii) the number of solutions also varies in the example (d being on the Pareto front in the pessimistic case but not in the optimistic case).

In the following section, we explore the performance of the different algorithms under the two extreme scenarios (optimistic and pessimistic) and we profile the behaviour of the different algorithms under the average scenario.

5.3. Other Algorithms

There is no direct comparison of *H2-D2* against other state-of-the-art solutions – as, to the best of our knowledge *H2-D2* is the first solution to the multi-objective VM reassignment in hybrid Decentralised DCs. However, *H2-D2* performs two main tasks, reassignment and placement, which can be evaluated. As

we show later in this paper (Section 6.1) the placement element of *H2-D2* is fast and efficient, so we do not think there is any room for improvement and evaluation here. On the contrary, the reassignment element of *H2-D2* is crucial and takes up most of the processing time. Hence we compare only the reassignment phase of *H2-D2* against solutions found in the literature. More precisely we compare *H2-D2* against nine other “systems” where the reassignment module is not implementing the three-step technique, as in *H2-D2*, but another state-of-the-art algorithm:

- eBSA [52]: a cutting-edge algorithm for the placement of VMs in hybrid Enterprise data centres, with a geographically distributed infrastructure. eBSA is an extension of a former approach [48] which combines a Biased Importance Sampling (BSA) and a Cross Entropy method (CE [40]). eBSA is based on three major concepts: (i) generating solutions probabilistically, (ii) biasing the generation of solutions to favour the creation of feasible ones, and (iii) optimising the solutions using a CE importance sampling. Given that eBSA was designed with a mono-objective optimisation in mind, we had to adapt it in our implementation in order to deal with multiple objectives. We, therefore, use the Pareto ranking algorithm described and implemented in [6] for the ranking of individuals and the generation of importance probabilities during the importance sampling phase.
- different elements of our three-step optimisation technique (GeNePi) taken individually (to evaluate the relative importance of each of them). The first algorithm: GRASP is run with the same maximally spread weighted sum vectors as in GeNePi during the whole execution time. The second algorithm: NSGA is run with the same parameters as in GeNePi, with a third of the execution time being dedicated to randomly generating its initial population. The third algorithm: PLS is run for the entire execution time, but unlike in GeNePi, it is run on all the iterative Pareto front solutions. Given that GeNePi is a three-step method, we also decided to compare it against GrNSGA, i.e., a combination of its two first algorithms: GRASP (one-third of the time limit) followed by NSGA (two-thirds of the time limit).
- other common and classical multi-objective optimisation techniques. We start with a Strength Pareto Evolutionary Algorithm (SPEA2 [60]): a well-known genetic algorithm, that is extensively used to compare existing evolution-based methods. As is the case with NSGA, SPEA2 is used in two different ways: (i) SPEA with a randomly generated initial population and (ii) GrSPEA with an initial population generated using GRASP. In addition, we compare our three-step algorithm to the multi-objective Cross Entropy (CE [6]) method. We finish with HC which runs multiple iterations of a Hill Climbing algorithm with a weighted sum objective, taking a different vector of weights on a maximally spread basis at every iteration.

5.4. Statistical Analysis and Tests

In order to validate the significance of our comparison, we perform a statistical test using a non-parametric test: the two-tailed Mann-Whitney U test (MWU). On every instance, MWU takes in the different performance values obtained on a given metric from each run (in our case 10) for two distinct algorithms. MWU returns the p-value that one of the algorithms obtains different values than the other. We consider tests significant when they are below a significance level of 0.05. Moreover, given the small number of runs in our experiment (justified by the long time each of them take), and in order to lower the risk of having incorrect rejection of true null hypothesis, we use a conservative but safe adjustment (i.e., the standard Bonferroni adjustment [4]) which reduces the chances of their erroneous rejection. Furthermore, following the advice in the practical guide proposed by Acruri and Briand [4], we also use the non-parametric $\hat{A}_{12}$ [53] effect size measure which evaluates the ratio of runs from the first algorithm that outperform the second one. It is considered in the literature that when $\hat{A}_{12}$ is above 0.71, differences between the algorithms are large. Given that these significance tests can only be performed on two algorithms at a time, and to avoid the combinatorial explosion when reporting the results in our revised manuscript, we only report the results for GeNePi against the other best algorithm.

6. Evaluation

We compare in this section the Pareto fronts found by *H2-D2* and the other systems presented above, both quantitatively (number of solutions) and qualitatively (space covered by the solutions, i.e., hypervolume). We are also interested in evaluating the HC used at the VC level, as it is essential for *H2-D2* that VCs find a quick placement for VMs assigned to them. Next, we profile how the different algorithms optimise as we are interested in knowing their individual behaviour and their composition in a hybrid technique. Finally, we report identified threats to the validity of our results.

6.1. Evaluation of the VM Placement Optimisation in VCs

FFD and HC are used at the VC level (see Figure 2, Section 4) respectively to find an initial VM placement and to optimise it. To understand how does HC improve the VM placement and how long does it take to converge, we present in Figure 4 the iterative savings obtained on *H2-D2*’s first VM assignment, when applied on instances with a number of PMs larger than their number of VCs. First, we show in details the optimisation process (improvements made by HC) for the four VCs that belong to the instance *a.1.2*, which all contain 25 PMs and respectively 225, 228, 284 and 233 VMs. We see that HC makes significant savings and these savings are decreasing over time. This means that general managers can allow a shorter time for the optimisation of the placement without deteriorating HC’s performance too much.

Figure 4 shows that HC converges quickly to optimum placements (that are sometimes local) on instance *a.1.2*, with all VCs finishing their optimisation within 4s (before the time limit)

– which may not always be the case. We observe that savings vary a lot between the different VCs: 2,192,652 and 6,081,944 respectively for c_0 and c_2 at similar execution times (2.95s and 3.64s), and that more time is required for c_2 to converge. The VC c_2 has more VMs than the others, thus making the placement’s improvement more complex (time costly), and at the same time showing larger savings.

Figure 4 also shows a simplified saving version for the other instances. Due to the large number of VCs in these instances (10 to 50), we only show the slowest (i.e., the last VC to finish its improvement) and the fastest VCs. We also report the mean savings of all VCs. We see a quick convergence in about 1s for instances a.* with the exception of a.1.2 and a.2.1, and in the order of tens of seconds for instances b.*. We also see that the slowest and fastest VCs do not differ a lot and that they are in the same order of magnitude (the largest differences being observed on b.1 with 19.5s-60.42s and on b.4 with 5.5s-19.93s). The means of VCs’ savings (the dark lines inside the grey areas) show that the majority of VCs tend to converge in a time that is close to the slowest one. However, this is not a huge problem as VCs are likely to be processed in parallel and the convergence of the slowest amongst them is reached in a reasonable time.

6.2. Quantitative and Qualitative Evaluation

We compare the different reassignment algorithms described earlier both quantitatively and qualitatively. We show the number of non-dominated solutions found by every algorithm. Given that the exact price of every VM flavour (in every public cloud) is not known but is to be picked from an interval, it is not possible to compare solutions given by the different algorithms as the price fluctuation has an impact on what is the best placement. In our study, we then consider two scenarios: (i) one scenario that we call *optimistic*, where the price of hosting VMs is set at the minimum value in the range and (ii) one scenario, *pessimistic*, where the price is set at the maximum. This gives an insight on the qualitative results of the algorithms in the extreme cases: when the cost of hosting VMs in the Cloud is low or high. Table 2 shows the average results of 10 runs obtained by every algorithm in terms of number of non-dominated solutions and hypervolume under those two scenarios.

We notice that GeNePi outperforms all of the other algorithms in terms of hypervolume when considering the optimistic scenario, with an improvement of 19.72% on average over the second-best algorithm. GeNePi also outperforms the other algorithms on most of the instances with the exception of b.4 in the pessimistic scenario and achieves an improvement in hypervolume of 18.22% on average against the second-best algorithm. Although the number of non-dominated solutions is a less important metric than the hypervolume as an algorithm can find more solutions with a poor quality, GeNePi finds more non-dominated solutions than the other algorithms in 11 cases out of 14 and GeNePi consistently attains at least the second-best results. This corresponds to an improvement of 29.99% on average in comparison to the second-best results. The statistical analysis suggests that results obtained using GeNePi on both hypervolume and number of non-dominated solutions are significantly different from those obtained by the other best algo-

rithm (with a p-value in the order of at least e-5) even after applying the Bonferroni correction. We also see that GeNePi outperforms the other best algorithm in hypervolume with $\hat{A}_{12} = 1$ (i.e., GeNePi outperforms the other algorithm on all the runs) on all instances (except on Pessimistic b.4). Moreover, GeNePi also outperforms the other best algorithm in number of non-dominated solutions with $\hat{A}_{12} > 0.98$ on 11 cases out of 14.

We also see that both GrNSGA and GrSPEA get relatively good improvements in terms of hypervolume (close to results of GeNePi) on all of the instances, but GrNSGA and GrSPEA do not obtain as many non-dominated solutions as GeNePi (GeNePi gets respectively 61.43% and 100.02% more non-dominated solutions than GrNSGA and GrSPEA on average). GrNSGA is better in general than GrSPEA as it outperforms it in hypervolume on all of the instances in the pessimistic scenario (GrNSGA gets an improvement of 21.62% in hypervolume on average) and on 11 instances out of 14 in the optimistic scenario (GrNSGA gets an improvement of 4.11% in hypervolume on average). However, the difference in terms of the number of non-nominated solutions is more significant as GrNSGA outperforms GrSPEA on all instances and gets 23.76% more non-dominated solutions on average.

GRASP gets a fair number of non-dominated solutions, but with an average quality (GRASP reaches 54.88% and 65.25% of the hypervolume obtained by GeNePi on average, respectively, on the pessimistic and optimistic scenarios).

NSGA and SPEA do not perform well, both quantitatively and qualitatively. NSGA and SPEA only reach respectively 29.38% and 32.41% in the pessimistic scenario, and 33.72% and 34.21% in the optimistic scenario of the hypervolumes obtained by GeNePi on average. NSGA and SPEA do not perform well in terms of number of non-dominated solution either as GeNePi gets respectively ~26 and ~5 times more non-dominated solutions than NSGA and SPEA on average. This is mainly due to the bad quality of the random generation of the initial population that affects negatively genetic algorithms. This effect can be seen when comparing their results with those obtained using a better initial population generated by applying GRASP. GrNSGA achieves on the optimistic and pessimistic scenarios respectively ~5 and ~27 times the improvement of NSGA in terms of hypervolume on average. GrNSGA also gets ~14 times more non-dominated solutions in comparison to NSGA on average, whereas GrSPEA achieves on the optimistic and pessimistic scenarios respectively ~8 and ~10 times the improvement of SPEA in terms of hypervolume on average. GrSPEA also gets ~2 times more non-dominated solutions in comparison to SPEA. Unlike GrNSGA and GrSPEA, SPEA performs relatively better than NSGA in most of the cases when given randomly generated initial populations, as SPEA gets 6.97% better hypervolume in the optimistic scenario and 153.91% more non dominated solutions on average.

PLS on its part gets a large number of non-dominated solutions and even has the second-best number of non-dominated solutions in 5 cases. However, the quality of these solutions is varying from one instance to another. PLS achieves a bad hypervolume on small instances, however, this trend changes when it comes to larger ones, even outperforming GeNePi and

Figure 4: Placement savings made by the Hill Climbing: in details for a.1.2 (4 VCs), and slowest/fastest VCs (in terms of convergence), in addition to the mean of savings for the other instances. Instances a.1.1 and a.1.4 have been intentionally omitted as their number of PMs equals their number of VCs, thus making the initial placement (from the FFD) optimal without any improvement from the Hill Climbing algorithm.

the other algorithms on the pessimistic scenario of instance *b.4*.

eBSA does not perform as well as expected as it only reaches 33.92% and 33.97% of GeNePi's hypervolume on average respectively on the optimistic and pessimistic scenarios. eBSA gets an average number of non-dominated solutions (always more than 10), but their number is very small when compared to GeNePi (GeNePi gets ~ 11 times more non-dominated solution on average). Despite eBSA's average performance, it is still outperforming basic genetic algorithms as eBSA gets 28.12% and 15.64% better hypervolume in the optimistic scenario, while getting 259.22% and 37.43% more non-dominated solutions when compared respectively with NSGA and SPEA. We clearly notice the advantage of biasing the importance sampling when looking at the performance of CE. CE achieves a poor performance both in terms of hypervolume and number of non-dominated solutions.

Jointly with CE, HC performs the worst overall, both quantitatively and qualitatively, despite some results improvements when applied on some instances (e.g., *a.1.1* and *a.1.3*).

We also notice large differences between hypervolumes of the optimistic and the pessimistic scenarios in same instances relatively to the improvements that are made by the different algorithms. These difference are reaching up to 63.77% in case of NSGA on average, which shows the impact of cloud price

fluctuations and the importance of considering the cloud price as an interval rather than a fixed cost. NSGA is followed by CE and HC with differences of 50.29% and 50.08% respectively on average. GrNSGA, eBSA and GeNePi are the best in terms of difference of hypervolumes between the optimistic and pessimistic scenarios with 11.30%, 12.63% and 15.45% respectively thanks to the large number of non-dominated solutions they find on every instance, which tone down the effect of VM prices' volatility in the Cloud.

6.3. Profiling

In this section we are interested in profiling the improvement in terms of hypervolume brought by the different reassignment algorithms on the different instances. We are particularly interested in the shape of the improvement curves. We plot on Figure 5 the evolution of the hypervolume improvement over time made by each algorithm on our studied instances. Given the complexity of showing the evolution of hypervolume as an interval, we choose to only plot the hypervolume when the price of every VM flavour in the public cloud is at its average.

As what has been seen in Table 2, GeNePi outperforms the other algorithms at the end of the time limit on all of the instances (with the exception of *b.4*) even when considering the

	Metric	Initial	GeNePi	GRASP	NSGA	SPEA	GrNSGA	GrSPEA	PLS	HC	CE	eBSA	MWU	$\hat{A}_{12}$
a.1.1	#Solutions	1	951	659	462	405	525	497	739	573	127	128	9.13E-05	1
	Hypervolume (e20)	Optimistic	5.769	9.029	8.566	8.664	8.430	8.890	8.817	6.972	8.905	8.279	8.147	4.22E-06
		Pessimistic	5.769	8.868	8.396	8.560	8.319	8.732	8.603	6.944	8.852	8.230	8.112	6.28E-06
a.1.2	#Solutions	1	336	238	6	18	192	127	33	8	8	23	8.98E-05	1
	Hypervolume (e25)	Optimistic	4.093	7.646	6.278	4.308	4.325	7.142	6.915	4.153	4.271	4.247	4.429	6.28E-06
		Pessimistic	4.093	7.494	6.191	4.223	4.265	7.042	6.748	4.128	4.237	4.216	4.403	6.28E-06
a.1.3	#Solutions	1	89	39	31	37	68	57	30	30	19	40	7.97E-05	1
	Hypervolume (e23)	Optimistic	6.376	6.950	6.662	6.517	6.590	6.825	6.805	6.550	6.666	6.388	6.610	9.66E-06
		Pessimistic	6.376	6.867	6.622	6.452	6.540	6.798	6.758	6.537	6.599	6.378	6.560	6.28E-06
a.1.4	#Solutions	1	4,877	574	1,536	1,674	1,855	1,372	3,744	125	18	48	9.13E-05	1
	Hypervolume (e22)	Optimistic	3.778	4.830	4.730	4.675	4.501	4.733	4.742	4.481	4.587	4.125	4.452	4.22E-06
		Pessimistic	3.778	4.829	4.709	4.655	4.483	4.712	4.705	4.481	4.587	4.125	4.452	6.28E-06
a.1.5	#Solutions	1	206	226	3	10	125	87	9	77	7	42	8.88E-05	0
	Hypervolume (e23)	Optimistic	5.472	7.245	6.658	5.596	5.720	6.763	6.761	5.584	5.661	5.682	5.737	1.36E-05
		Pessimistic	5.472	7.144	6.552	5.551	5.676	6.669	6.600	5.528	5.638	5.657	5.716	6.28E-06
a.2.1	#Solutions	1	555	284	3	43	206	168	12	15	8	33	8.93E-05	1
	Hypervolume (e24)	Optimistic	4.193	8.424	6.438	4.309	4.316	7.377	7.274	4.244	4.195	4.204	4.334	6.28E-06
		Pessimistic	4.193	8.258	6.323	4.232	4.249	7.274	7.135	4.202	4.193	4.202	4.332	6.28E-06
a.2.2	#Solutions	1	395	225	10	25	483	331	29	193	11	84	9.03E-05	0
	Hypervolume (e23)	Optimistic	4.093	5.287	4.862	4.235	4.269	5.090	4.939	4.237	4.241	4.336	4.536	4.22E-06
		Pessimistic	4.093	5.224	4.823	4.210	4.239	5.070	4.903	4.212	4.233	4.327	4.529	4.22E-06
a.2.3	#Solutions	1	103	33	47	53	50	48	36	26	19	51	7.92E-05	1
	Hypervolume (e24)	Optimistic	3.344	3.463	3.398	3.367	3.368	3.413	3.413	3.385	3.359	3.358	3.376	6.28E-06
		Pessimistic	3.344	3.414	3.344	3.346	3.356	3.391	3.381	3.371	3.345	3.346	3.365	6.28E-06
a.2.4	#Solutions	1	183	93	110	124	142	116	149	21	18	49	8.63E-05	1
	Hypervolume (e23)	Optimistic	1.087	1.912	1.411	1.556	1.615	1.830	1.752	1.401	1.567	1.354	1.560	4.22E-06
		Pessimistic	1.087	1.893	1.386	1.525	1.590	1.801	1.716	1.396	1.551	1.336	1.546	6.28E-06
a.2.5	#Solutions	1	135	92	61	74	117	101	44	55	18	24	8.25E-05	1
	Hypervolume (e23)	Optimistic	4.422	4.836	4.652	4.727	4.761	4.821	4.813	4.647	4.687	4.613	4.734	6.28E-06
		Pessimistic	4.422	4.788	4.587	4.652	4.671	4.754	4.716	4.601	4.669	4.590	4.715	6.28E-06
b.1	#Solutions	1	36	29	9	11	31	29	11	5	4	19	9.83E-05	0.985
	Hypervolume (e25)	Optimistic	6.244	6.401	6.378	6.265	6.247	6.377	6.398	6.266	6.264	6.244	6.247	6.28E-06
		Pessimistic	6.244	6.312	6.252	6.247	6.245	6.301	6.269	6.246	6.247	6.244	6.245	6.28E-06
b.2	#Solutions	1	57	32	9	15	36	28	7	4	14	30	8.20E-05	1
	Hypervolume (e24)	Optimistic	3.877	4.245	4.019	3.917	3.906	4.163	4.145	3.905	3.879	3.881	3.898	1.36E-05
		Pessimistic	3.877	4.182	3.955	3.879	3.884	4.097	4.036	3.881	3.878	3.880	3.897	6.28E-06
b.3	#Solutions	1	22	9	6	9	17	16	18	7	7	18	6.43E-05	1
	Hypervolume (e25)	Optimistic	2.300	2.405	2.396	2.310	2.311	2.392	2.382	2.371	2.341	2.303	2.318	1.36E-05
		Pessimistic	2.300	2.400	2.372	2.300	2.307	2.387	2.376	2.352	2.327	2.300	2.307	4.22E-06
b.4	#Solutions	1	21	16	11	15	15	12	23	9	12	12	8.89E-05	0.05
	Hypervolume (e24)	Optimistic	1.977	2.096	2.079	2.051	2.056	2.076	2.071	2.093	2.045	2.010	2.034	6.28E-06
		Pessimistic	1.977	2.059	2.066	2.039	2.042	2.072	2.066	2.080	2.025	1.989	2.018	4.22E-06

Table 2: Results obtained by the different algorithms on the ROADEF instances in terms of number of non-dominated solutions (#Solutions) and hypervolume (10^4) when considering the optimistic and the pessimistic scenarios by setting VM prices respectively to their lowest and highest. We also report the MWU significance values and $\hat{A}_{12}$ measures between GeNePi and the other best algorithm on every metric and on all instances.

average VM pricing scenario. GeNePi is followed closely by GrNSGA and GrSPEA.

Also, algorithms based on GRASP (i.e., GRASP, GrNSGA, GrSPEA and GeNePi) give better results due to their greedy nature and the usage of maximally spread weight vectors in their optimisation. Thus, improving significantly the hypervolume during the first fifth of the execution time. However, they often plateau until starting the next optimisation phase (i.e. either NSGA or SPEA) or continue on the same trend until the end of the execution time in the case of GRASP.

The NSGA step in both GrNSGA and GeNePi brings a quick and important improvement in hypervolume until reaching an elite population where only a marginal improvement can be obtained using the crossover and mutation operators. GrSPEA also improves the hypervolume significantly, however, not to the same extent as is observed for the GrNSGA's. This is mainly due to the ranking procedure in SPEA which favours mostly the elite and the most spread individuals rather than the introduction of new ones and therefore leads to a prema-

ture convergence. Furthermore, we notice that GrNSGA takes longer than GrNSGA to make any improvement between the different generations, which is mostly due to the time complexity of SPEA's ranking procedure. This 'time-wastage' is especially noticeable at the start-up of the SPEA part as the initial population received from GRASP can be of a larger size than the offspring populations, thus taking longer to rank its individuals.

GeNePi copes with the convergence of the population in GrNSGA and the stagnation of the hypervolume by introducing a PLS step that increases the number of non-dominated solutions and gains some extra hypervolume improvement.

NSGA struggles to improve the hypervolume during the first third of its execution time. This corresponds to the random part that does not find solutions or finds some that are not interesting enough. After the random phase, we often notice an improvement in the hypervolume thanks to the genetic algorithm step. However, NSGA never reaches results obtained with GrNSGA due to the handicap of the random generation of

Figure 5: Evolution of the hypervolume obtained using the different algorithms on the different instances over time when setting the VM prices to their average. Each dot corresponds to a solution (or a set of solutions) found by the algorithm.

the initial population.

As with NSGA, SPEA also struggles to improve the hypervolume during the first third of its execution time and does not reach GrSPEA's results. In addition, SPEA takes longer to improve its initial population. However, despite being given a poor initial population, SPEA's ranking procedure allows it to achieve a better improvement than NSGA.

CE's and eBSA's use of a random generation for their initial individuals does not bring much improvement either. After that phase, we see that CE is not improving significantly as it mostly finds non-feasible solutions. However, thanks to biasing the importance sampling, eBSA generates more feasible individuals. These individuals allow eBSA to achieve significant improvements that even outperform those of NSGA and SPEA

in most cases. However, without ever reaching GrNSGA's and GrSPEA's results though. In terms of time complexity, we see that both CE and eBSA take some time (longer than the ranking in SPEA) to start generating individuals, which indicates that the Cross Entropy method is very expensive in time. We also see that eBSA usually takes a bit longer than CE as biasing the importance sampling also adds a small but existing time complexity.

PLS goes through several improvements which are often-times marginal, particularly on small and medium instances. However, we see an increase in performance on very large scale instances (i.e., *b_3* and *b_4*). HC shows the opposite behaviour to that of PLS. HC does not make many improvements but makes more noteworthy ones when it makes any.

6.4. Threats to Validity

We have identified some threats to the validity of our evaluation. First, our work is based on instances of data centres from the Google/ROADEF data set. Therefore, results might not generalise to other data sets. In order to lower this risk, we used 14 instances of different scales (i.e., small, medium and large) and complexities (i.e., different number of resources and constraints). The second threat to validity is the influence of randomness on our algorithms. To reduce the influence of randomness on the results, we repeat our experiment 10 times thereby reducing its impact. We also use a statistical significance test (i.e., MWU) that is corrected (using the standard Bonferroni adjustment) to lower the risk of erroneous rejection of the null hypothesis. We also use the $\hat{A}_{12}$ measure to evaluate how many values from each run of GeNePi outperform the values obtained by the other best algorithm on each instance and show that GeNePi outperforms them on most of the runs. Another threat to validity we could identify is potential errors that might come from the used machine or differences in implementation of the algorithms. To lower this risk, we implemented all the algorithms using the same language and libraries. We also used the same machine to run each of the algorithms with no processes running in parallel (except those of the Operating System). The last threat is the implementation of the state-of-the-art algorithm eBSA, as we could not find a publicly available implementation of it. Hence we had to carefully implement and extend it ourselves to the multi-objective context after a thorough study of the paper that describes it.

7. Conclusion

This paper is a first attempt at addressing the complex options that capital allocators and managers of the infrastructure of large IT companies now face: (i) they can either deploy some of their workload in the Cloud or keep them locally, (ii) the cost of hosting VMs in the Cloud may vary over time which makes a comparison of solutions difficult, (iii) internally these large, often geographically distributed companies are composed of multiple hosting departments with various local preferences and (iv) any reassignment of the VMs can be seen from different perspectives (objectives) that cannot be compared.

We have proposed H2-D2, a hybrid algorithm which reassigns VMs to the Cloud or internal hosting department using a succession of three optimisation algorithms (greedy, genetic and local search) and the placement of VMs in the hosting departments using a greedy routine followed by a Hill Climbing algorithm. We have extended a previous data set proposed by Google for an Operations Research challenge [19, 42] and added information about public cloud services. Our comparison against nine multi-objective placement algorithms shows that H2-D2 outperforms them both quantitatively (H2-D2 gets approx 30% more non-dominated solutions on average than the second-best algorithm) and qualitatively (H2-D2 gets close to 20% improvement in hypervolume on average in two extreme scenarios we defined).

Our focus is now on modelling direct interactions between hosting departments, because, as is common in large decentralised data centres, capital allocators tend to collaborate with their counterparts in other hosting departments.

Acknowledgement

This work was partially supported by Science Foundation Ireland grants 10/CE/I1855 and 13/RC/2094 to Lero - the Irish Software Research Centre (www.lero.ie).

References

- [1] Afsar, H. M., Artigues, C., Bourreau, E., Kedad-Sidhoum, S., 2016. Machine reassignment problem: the roadeff/euro challenge 2012. *Annals of Operations Research*, 1–17.
- [2] Alsheddy, A., Tsang, E. E., 2010. Guided pareto local search based frameworks for biobjective optimization. In: CEC. pp. 1–8.
- [3] Altmann, J., Kashef, M. M., 2014. Cost model based service placement in federated hybrid clouds. *FGCS*, 79–90.
- [4] Arcuri, A., Briand, L., 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: ICSE. pp. 1–10.
- [5] Baset, S. A., 2016. Cloud service level agreement. *Encyclopedia of Cloud Computing*, 433–445.
- [6] Bekker, J., Aldrich, C., 2011. The cross-entropy method in multi-objective optimisation: An assessment. *EJOR*, 112–121.
- [7] Beloglazov, A., Abawajy, J., Buyya, R., 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *FGCS*, 755–768.
- [8] Bhunia, A., Biswas, A., Sen, N., 2014. An application of extended elitist non-dominated sorting genetic algorithm in multi-objective linear programming problem of tea industry with interval objectives. *USCM*, 245–256.
- [9] Bhunia, A. K., Samanta, S. S., 2014. A study of interval metric and its application in multi-objective optimization with interval objectives. *Computers & Industrial Engineering*, 169–178.
- [10] Bittencourt, L. F., Madeira, E. R., Da Fonseca, N. L., 2012. Scheduling in hybrid clouds. *IEEE Communications Magazine*.
- [11] Bringmann, K., Friedrich, T., 2013. Approximation quality of the hypervolume indicator. *Artificial Intelligence* 195, 265–290.
- [12] Casalicchio, E., Menascé, D. A., Aldhalaan, A., 2013. Autonomic resource provisioning in cloud systems with availability goals. In: CAC. pp. 1:1–1:10.
- [13] Chunlin, L., LaYuan, L., 2015. Optimal scheduling across public and private clouds in complex hybrid cloud environment. *IFS*, 1–12.
- [14] Corradi, A., Fanelli, M., Foschini, L., 2014. Vm consolidation: A real case based on openstack cloud. *FGCS*, 118–127.
- [15] Cunha, R. L., Rodrigues, E. R., Tizzei, L. P., Netto, M. A., 2017. Job placement advisor based on turnaround predictions for hpc hybrid clouds. *FGCS*, 35–46.
- [16] Doddavula, S. K., Kaushik, M., Jain, A., 2011. Implementation of a fast vector packing algorithm and its application for server consolidation. In: *CloudCom*. pp. 332–339.
- [17] Fadda, E., Plebani, P., Vitali, M., 2016. Optimizing monitorability of multi-cloud applications. In: CAiSE. pp. 411–426.
- [18] Gabay, M., Zaourar, S., 2012. A GRASP approach for the machine reassignment problem. In: EURO.
- [19] Google/ROADEF, 2012. Google/roadeff challenge'12. <http://challenge.roadeff.org/2012/en/>.
- [20] Hao, F., Kodialam, M., Lakshman, T., Mukherjee, S., 2016. Online allocation of virtual machines in a distributed cloud. *Transactions on Networking*.
- [21] Harms, R., Yamartino, M., 2010. The economics of the cloud. Microsoft whitepaper, Microsoft Corporation.
- [22] Hu, C., Yao, H., Liu, C., Zeng, D., 2016. Executing time and cost-aware task scheduling in hybrid cloud using a modified de algorithm. In: ISICA. pp. 74–83.

- [23] Jaśkowski, W., Szubert, M., Gawron, P., 2015. A hybrid mip-based large neighborhood search heuristic for solving the machine reassignment problem. *Annals of Operations Research*, 1–30.
- [24] Kash, I. A., Key, P. B., 2016. Pricing the cloud. *IEEE Internet Computing*, 36–43.
- [25] Khan, S. U., Ullah, N., 2016. Challenges in the adoption of hybrid cloud: An exploratory study using systematic literature review. *The Journal of Engineering*.
- [26] Li, R., Zheng, Q., Li, X., Wu, J., 2016. A novel multi-objective optimization scheme for rebalancing virtual machine placement. In: *CLOUD*.
- [27] Li, X., Ventresque, A., Omana Iglesias, J., Murphy, J., 2015. Scalable correlation-aware virtual machine consolidation using two-phase clustering. In: *HPCS*. pp. 237–245.
- [28] Liaqat, M., Chang, V., Gani, A., Ab Hamid, S. H., Toseef, M., Shoaib, U., Ali, R. L., 2017. Federated cloud resource management: Review and discussion. *JNCA*, 87–105.
- [29] Lopes, R., Morais, V. W., Noronha, T. F., Souza, V. A., 2015. Heuristics and matheuristics for a real-life machine reassignment problem. *ITOR*, 77–95.
- [30] Mann, Z. Á., 2015. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *CSUR*, 11:1–11:34.
- [31] Manvi, S. S., Shyam, G. K., 2014. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *JNCA*, 424–440.
- [32] Moothedan, M., Paul, V., 2016. Sla for iaas clouds. *IJES*.
- [33] Panigrahy, R., Talwar, K., Uyeda, L., Wieder, U., 2011. Heuristics for vector bin packing. *research.microsoft.com*.
- [34] Petcu, D., 2014. Consuming resources and services from multiple clouds. *Journal of Grid Computing*, 321–345.
- [35] Pires, F. L., Barán, B., 2015. A virtual machine placement taxonomy. In: *CCGrid*. pp. 159–168.
- [36] Raju, R., Amudhavel, J., Kannan, N., Monisha, M., 2014. A bio inspired energy-aware multi objective chiropteran algorithm (eamoca) for hybrid cloud computing environment. In: *ICGCCEE*. pp. 1–5.
- [37] Rezaeian, A., Abrishami, H., Abrishami, S., Naghibzadeh, M., 2016. A budget constrained scheduling algorithm for hybrid cloud computing systems under data privacy. In: *IC2E*. pp. 230–231.
- [38] RightScale, 2016. cloud computing trends 2016 state cloud survey. www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey.
- [39] Rittinghouse, J. W., Ransome, J. F., 2016. *Cloud computing: implementation, management, and security*. CRC press.
- [40] Rubinstein, R. Y., Kroese, D. P., 2013. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*.
- [41] Saber, T., Marques-Silva, J., Thorburn, J., Ventresque, A., 2017. Exact and hybrid solutions for the multi-objective vm reassignment problem. *IJAIT*.
- [42] Saber, T., Ventresque, A., Brandic, I., Thorburn, J., Murphy, L., 2015. Towards a multi-objective vm reassignment for large decentralised data centres. In: *UCC*. pp. 65–74.
- [43] Saber, T., Ventresque, A., Gandibleux, X., Murphy, L., 2014. Genepi: A multi-objective machine reassignment algorithm for data centres. In: *HM*. pp. 115–129.
- [44] Saber, T., Ventresque, A., Marques-Silva, J., Thorburn, J., Murphy, L., 2015. Milp for the multi-objective vm reassignment problem. In: *ICTAI*. pp. 41–48.
- [45] Salapura, V., Mahindru, R., 2016. Enabling enterprise-class workloads in the cloud. In: *IC2E*. pp. 228–229.
- [46] Sotiriadis, S., Bessis, N., 2016. An inter-cloud bridge system for heterogeneous cloud platforms. *FGCS*, 180–194.
- [47] Stillwell, M., Vivien, F., Casanova, H., 2012. Virtual machine resource allocation for service hosting on heterogeneous distributed platforms. In: *IPDPS*. pp. 786–797.
- [48] Tantawi, A. N., 2012. A scalable algorithm for placement of virtual clusters in large data centers. In: *MASCOTS*. pp. 3–10.
- [49] Tordsson, J., Montero, R. S., Moreno-Vozmediano, R., Llorente, I. M., 2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *FGCS*, 358–367.
- [50] Turkey, A., Sabar, N. R., Sattar, A., Song, A., 2016. Parallel late acceptance hill-climbing algorithm for the google machine reassignment problem. In: *AJCAI*. pp. 163–174.
- [51] Turkey, A., Sabar, N. R., Song, A., 2017. Neighbourhood analysis: A case study on google machine reassignment problem. In: *Artificial Life and Computational Intelligence*. pp. 228–237.
- [52] Unuvar, M., Steinder, M., Tantawi, A. N., 2014. Hybrid cloud placement algorithm. In: *MASCOTS*. pp. 197–206.
- [53] Vargha, A., Delaney, H. D., 2000. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 101–132.
- [54] Voorsluys, W., Broberg, J., Venugopal, S., Buyya, R., 2009. Cost of virtual machine live migration in clouds: A performance evaluation. In: *CloudCom*. pp. 254–265.
- [55] Wang, S., Ali, S., Yue, T., Li, Y., Liaaen, M., 2016. A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In: *ICSE*. pp. 631–642.
- [56] Wood, T., Shenoy, P. J., Venkataramani, A., Yousif, M. S., et al., 2007. Black-box and gray-box strategies for virtual machine migration. In: *NSDI*. p. 229–242.
- [57] Xu, J., Fortes, J., 2011. A multi-objective approach to virtual machine management in datacenters. In: *ICAC*. pp. 225–234.
- [58] Xu, J., Fortes, J. A., 2010. Multi-objective virtual machine placement in virtualized data center environments. In: *GreenCom*. pp. 179–188.
- [59] Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C. M., da Fonseca, V. G., 2002. Why quality assessment of multiobjective optimizers is difficult. In: *GECCO*. pp. 666–673.
- [60] Zitzler, E., Laumanns, M., Thiele, L., et al., 2001. Spea2: Improving the strength pareto evolutionary algorithm. In: *Eurogen*. pp. 95–100.
- [61] Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms — a comparative case study. In: *PPSN*. pp. 292–301.
- [62] Zuo, L., Shu, L., Dong, S., Chen, Y., Yan, L., 2016. A multi-objective hybrid cloud resource scheduling method based on deadline and cost constraints. *IEEE Access*.

Takfarinas Saber is a Ph.D. student in the Performance Engineering Laboratory, University College Dublin, Dublin, Ireland, under the supervision of Prof. Liam Murphy and Dr. Anthony Ventresque. His main research topic is about multi-objective VM reassignment for large data centres.

James Thorburn brings over 15 years of experience in designing and delivering enterprise-level solutions to optimization problems across IBM's global software development organization. He has designed and delivered data-driven global solutions supporting Fixed Assets Management, IT Security Automation, Data Centre Management, Software License Management, Capital Plan Management and IT Resource Optimization.

Liam Murphy received the B.E. degree in electrical engineering from University College Dublin, Dublin, Ireland, in 1985, and the M.Sc. and Ph.D. degrees in electrical engineering and computer sciences from University of California, Berkeley, CA, USA, in 1988 and 1992, respectively. He is a Professor of computer science with University College Dublin, where he is also the Director of the Performance Engineering Laboratory.

Anthony Ventresque received his Ph.D. degree in Computer Science from the University of Nantes, France, in 2008. He is currently Lecturer in the School of Computer Science at University College Dublin, Ireland, and a Funded Investigator with Lero, the Irish Software Engineering Research Centre. Previously he held positions as Research Fellow at NTU, Singapore (2010–2011), UCD, Ireland (2012–2014), and IBM Research Dublin, Ireland (2014–2015).