

A Cloud Resources Portfolio Optimization Business Model - From Theory to Practice

Valentin Haag

Faculty of Computer Science
University of Vienna
Vienna, Austria
valentin.haag94@gmail.com

Maximilian Kiessler

Faculty of Computer Science
University of Vienna
Vienna, Austria
m.kiessler@hotmail.com

Benedikt Pittl

Faculty of Computer Science
University of Vienna
Vienna, Austria
benedikt.pittl@univie.ac.at

Erich Schikuta

Faculty of Computer Science
University of Vienna
Vienna, Austria
erich.schikuta@univie.ac.at

Abstract—Cloud resources have become increasingly important, with many businesses using cloud solutions to supplement or outright replace their existing IT infrastructure. However, as there is a plethora of providers with varying products, services, and markets, it has become increasingly more challenging to keep track of the best solutions for each application. Cloud service intermediaries aim to alleviate this problem by offering services that help users meet their requirements.

This paper aims to lay the groundwork for developing a cloud portfolio management platform and its business model, defined via a business model canvas. Furthermore, a prototype of a platform is developed offering a cloud portfolio optimization service, using two algorithms developed in previous research to create suitable and well-utilized allocations for a customer's applications.

Index Terms—Cloud Economics, Portfolio Optimization, Business Model

I. INTRODUCTION

Over the past years, the cloud resource market has been one of the fastest-growing IT segments. The biggest providers, Amazon Web Services (AWS), Microsoft Azure, and Google Cloud, have seen annual sales increases of over 20% for several years. Just AWS itself reported a revenue of 80 Billion US dollars for the year 2022, and the entire cloud market achieved revenue of 545.8 Billion dollars worldwide with 22% growth compared to 2021 [1]. This large and expanding market has resulted not only in a growing cloud service provider (from here on, often referred to as CSPs) market but also in several methods of delivery of cloud resources to the customer.

As mentioned by Pittl [2], one big challenge facing both industry and academia is finding a cost-effective solution when buying cloud capacities. Pittl's study concludes that almost all observed resource requests were oversized and offered significant cost reduction potential, which lies not only in reducing the amount of resources bought but also in their composition. Depending on the planning period of the operations to be performed, a different mix of procurement from different market spaces is optimal. The author suggests tackling this problem via a cloud resource trading intermediary.

Offering a service that aids businesses in managing their cloud portfolio and proposing efficient allocations to run

their applications, even across various providers, is of great interest in the market. While most larger CSPs offer some functionalities and services that claim to help prevent over-provisioning, like AWS Lambda and Fargate, it is ultimately not in the provider's best interest to reduce the costs for the customer. For the same reason, offering cross-platform support should not be expected of them either.

One way to better leverage the opportunities of the cloud market and make the market more accessible is the use of cloud intermediaries between the CSPs and the customer. Given that, this paper aims to tackle two main research questions:

- *Under what kind of business model could such an intermediary operate?* To answer this question, a detailed proposal and description of a viable business model for a cloud resource intermediary will be presented. Intermediaries in a similar fashion have been put forward, but as far as the authors' best knowledge, no conclusive business model exists on how these intermediaries could operate.
- *How could such a platform be implemented?* The other goal of this work is the design and implementation of a cloud resource intermediary.

Thus, the paper is structured as follows: The next section II gives the reader a literature survey on the target research area. Section III focuses on cloud portfolio optimization theory, and two respective algorithms, a genetic and a greedy one, are described and their performance evaluated. In section IV, we propose our proposal of a business model for a cloud portfolio management platform, consisting of the nine building blocks defined by the business model canvas framework of Osterwalder [3]. Based on the business model description, we present the implementation of our respective prototype in section V. The following section VI presents our application's user experience (UX) evaluation. Finally, section VII summarises the paper's findings, including what future work to expand upon the topic.

II. STATE OF THE ART

The cloud market currently consists of many cloud service providers, each featuring various ways and markets for cus-

tomers to access their products. There is no single space from which all suppliers can be accessed. The market is an oligopoly, dominated by the most prominent three players, Amazon Cloud Services (AWS), Microsoft Azure, and Google Cloud, according to [4]. In the course of this work, we focus on AWS, the largest cloud service provider today, which offers three different resource markets with varying pricing models, which are as follows:

- *On-demand marketplace*: For maximum flexibility and easy scalability with no advance notice, instances bought on this market are handled in the classic pay-per-use model, where customers can purchase computing power on-demand and by time period.
- *Saving plans marketplace*: This model offers significant price reductions (e.g., up to 72% cheaper) for the customer in exchange for a long-term commitment (e.g., 1- or 3-year plans) to a certain amount of usage.
- *Spot marketplace*: This market allows customers to benefit from the varying loads that AWS experiences on its services. Amazon EC2 offers instances of their currently unused capacity with even greater discounts of up to 90% reduced prices compared to the on-demand market.

The definition of a business model has not only been one of the earliest focuses of research, but also one of the most hotly debated. Almost every paper concerning this topic has defined its own take on this task, and to this day, there is no general agreement on a universally accepted definition. We follow the definition of [3], where a business model describes the rationale of how an organization creates, delivers, and captures value.

Similar to the definition of the term, there is also a wide array of frameworks describing the elements a business model comprises. The paper will focus on the Business Model Canvas from Osterwalder and Pigneur [3], one of the most established and widely used frameworks today. This framework proposes to describe any business model with nine building blocks, which can also be grouped into four areas of a business: customers, offer, infrastructure, and financial viability. They showcase how a company intends to make money and are as follows [5], [3]:

- The *Customer Segment* block describes the groups a company tries to offer its products to.
- The *Value Proposition* describes the reason why a customer chooses to work with one business over others
- *Channels* describe the way a company reaches its customer segment to make its proposition of value.
- The different kinds of relationships companies can have with their customers are described in the block *Customer Relationships*.
- The block *Revenue Streams* deals with the income a company receives from its customers.
- The *Key Resources* block describes the resources that allow the company to operate its business and earn

revenues.

- The block *Key Activities* defines the activities a company must perform to be successful. They are necessary to create value and earn revenue, and they can differ widely depending on the company.
- *Key partnerships* are the suppliers and partners a business model includes to make it work.
- The last block *Cost Structure* of the Canvas deals with all operating costs of the business model.

As many cloud providers are on the market nowadays, the question arises: What makes such a company and its business model successful? An analysis of the business model characteristics revealed three different types of business models, with differing value creation, proposition, and delivery that all providers could be divided into [6]:

- *Newcomer*: This type describes newcomers to the market that adapt already existing cloud strategies or form co-operations with already established cloud companies. Their value proposition focuses much on individual customization, often resulting in higher initial costs. The newcomer's main customer segment is niche markets, which they reach through traditional channels, such as print media and personal contact. A one-time charge generates revenue, which supplementary services or partner revenue models later generate. This cloud provider has to deal with a crowded market and stand out by having a well-developed market entry strategy and a defined target market. On the other hand, they have the advantage of working flexibly and developing a specialized cloud service instead of focusing on commodity services.
- *Experienced player*: Companies of this type mainly work with their existing know-how, offering a plethora of software and consultancy services. The services offered are standardized and provided via a public cloud with high-security standards, resulting in excellent scalability and possibilities for time and cost reductions. The experienced players target both the mass and individual markets and use support systems and online communities to reach their customer base. Subscription-based services usually account for the biggest source of revenue. While profiting from an economy of scale and a good understanding of their technology, they must be careful that good CRM, branding, and marketing strategies compensate for less direct customer contact and lower trust levels.
- *Specialised provider*: Providers of this type stand out by expanding the usual cloud services on a vertical level, adding services such as data processing, administration, marketplace, and migration services. Their target customer segment is branch-specific and may include the public sector. Firms following this business model often implement a usage-based revenue model. The specialized providers have the advantage of a high-quality and innovative model that, in combination with good security and customer orientation, results in high levels

of trust and loyalty by their customers. While not easily imitated, the model does not support high scalability and needs constant innovation to satisfy the customers' ever-developing needs.

Actual cloud broker implementations on the market are still sparse, while the high number of cloud service offerings makes it hard to find the right service as a customer. Brokerage can be several services the intermediary offers the cloud customer, such as decision support or enhancing the delivery of services. The benefits for the customer are lower costs and the ability to seamlessly switch between different cloud providers [7].

To gain a deeper understanding of the cloud intermediary market, Elhabbash et al. [7] presents a systematic survey of cloud brokerage literature, looking into the motivation for designing a cloud broker and its functionality.

While a plethora of research has been conducted in the field of cloud intermediaries, the number of implementations on the market is limited. This situation, in conjunction with the fact that the services provided can vary wildly, has resulted in a lack of research concerning the business models of cloud brokers so far. Nevertheless, the work of Filiopoulou et al. [8] gives an overview of benefits, common pricing models, and an evaluation of cloud brokers. It is concluded that brokers assist companies in developing themselves and creating a more competitive environment for providers while earning revenues themselves.

III. PORTFOLIO OPTIMIZATION MODEL

In this section, we will present a synopsis of the findings of our previous research work, which encompasses related approaches, a formulation of the problem at hand, a short description of two portfolio optimization algorithms developed, and finally, an evaluation of their performance [9].

The main goal of cloud portfolio management is usually achieving the lowest costs for running a specific set of applications over (a certain amount of) time. Some research like Jangjaimon and Tzeng [10] and Sharma et al. [11] tried to deal with this problem by creating a checkpointing mechanism or by focusing on preemptible servers in combination with concepts taken from financial modeling, to meet the requirements of applications when using spot instances. Meanwhile, Pittl et al. [2] took a more comprehensive approach to cloud portfolio management, which resulted in the findings that a more heterogeneous portfolio tends to be more cost-efficient. Another finding of that paper was to highlight the significance of right-sizing, where server instances should be chosen to most closely fit the capacity requirements of the applications the workload consists of. Otherwise, an expensive over-provisioning of resources could be the result [9]. Regarding right-sizing, Hwang and Pedram [12] developed a portfolio-based optimization approach with a probabilistic model, which, while created for data center operations, is also applicable to cloud portfolio management. Each assigned workload in this model has a

TABLE I
NOTATION FOR PROBLEM FORMULATION

Parameter	Description
I	A set of selected cloud instances (hosts)
A	A set of applications
T	A set of time slots for which to optimize
x_{ait}	Variable denoting assignments of applications to instances
S_a	The starting time of application a
F_a	The finishing time of application a
U_a	Indicates if application a is preemptible
μ_a	The expected resource demand of application a
σ_a	The std. deviation of the resource demand of application a
R_i	The resource capacity of instance i
C_i	The cost of instance i for one time slot
B_i	The first available time slot of an instance
E_i	The last available time slot of an instance
O_i	Indicates if instance i is suitable for non-preemptible apps
D_{it}	Aggregated resource demand of instance i at time t
Q_{min}	The desired quality of service

resource demand, not defined by a fixed value, but via a probabilistic model using a normal distribution. This idea of using probability-based problem formulation was also used in further research; for example, Martinovic et al. combined it with a stochastic bin-packing approach to finding efficient server allocations [13]. One additional approach proposed for optimizing virtual machine placement to reduce energy consumption in data centers is using a genetic algorithm, as was done by Wu et al. [14]. Another interesting concept to be regarded is the one by De Cauwer et al., which introduced the idea of a temporal component to server allocation schemes while using deterministic resource demands. This additional dimension better reflects real computational needs, where applications often do not run forever but with a specific start and end time [15], [9].

A. Problem formulation

This section will deal with formulating the problem, which has to be tackled by the cloud portfolio optimization approaches. The notations used for this task are defined in I [9].

First, we define a cloud portfolio as a set of *cloud instances* I , which are used to run a set of *applications* A on them. The main goal of our optimization problem is to find a cost-efficient allocation of these applications and instances [9].

All applications A have a specific resource demand, which, as proposed in the literature by Hwang and Pedram, have their resource needs not defined as a fixed value but with fluctuation taken into account [12]. Therefore, we denote the capacity requirements of our applications as an expected demand mean μ_a and with a corresponding standard deviation σ_a . As our workloads also have varying run times, each application sports a starting time S_a and a finishing time F_a , for which the statement $S_a < F_a$ must always be true. To model which applications are suitable for spot instances, meaning they can be interrupted at any time, we use the variable U_a which can either be 0 or 1 [9].

$$U_a = \begin{cases} 1 & \text{if } a \text{ is preemptible} \\ 0 & \text{else} \end{cases} \quad (1)$$

For modeling instances, each instance $I \in I$ has a pre-determined resource capacity R_i , representing, for example, the number of CPUs and RAM of an instance. Furthermore, each kind of instance has a price per time unit C_i , where the total cost of any instance is calculated by the price per time slot and the overall up-time. As with applications, each instance has a given starting time B_i and ending time E_i , with the inequation $B_i < E_i$ again having to be fulfilled at any time. Preemptible spot instances are denoted by the binary parameter O_i [9].

$$O_i = \begin{cases} 1 & \text{if } i \text{ is only suitable for preemptible applications} \\ 0 & \text{else} \end{cases} \quad (2)$$

To model the summed-up demand of all applications assigned to an instance $i \in I$, for a specific time slot $t \in T$, we use the parameter D_{it} . As with the demand of a single application, the aggregated demand is also not a fixed variable but a random variable, which means it can only evaluate the probability with which an instance stays within the designated capacity limits. Our proposed model also defines a desired minimum quality of service Q_{min} . An allocation is invalid if the probability of the aggregated demand D_{it} staying below the provided capacity of the instance R_i does not satisfy the minimum quality of service Q_{min} for time slot t . This approach also builds upon the model proposed by Hwang and Pedram (2012). Still, it has to consider the temporal component of our model, meaning that the capacity restrictions have to be fulfilled for every time slot an instance is used [9].

For modeling the formal assignment of applications to instances while considering the temporal restrictions of the problem, we used the approach postulated by Dell'Amico et al. [16]. The variable x denotes which hosting instance an application has been assigned to at a specific time. The resource demands for an application $a \in A$ are considered to be 0 for any time slot $t \in T$ if $t < S_a$ or $t > F_a$ [9].

$$x_{ait} = \begin{cases} 1 & \text{if app } a \text{ is assigned to instance } i \text{ at time slot } t \\ 0 & \text{else} \end{cases} \quad (3)$$

Having outlined the cloud portfolio optimization requirements and assumptions, we can now present our exact problem statement. The main optimization goal is to find the minimum of the following cost function [9]:

$$\min \sum_{i \in I} C_i * (E_i - B_i) \quad (4)$$

4 is the main function to optimize. It minimizes the costs of the entire portfolio, which consists of the sum of all prizes incurred by each assigned instance. Hereby, $(E_i - B_i)$ is the time an instance is running, multiplied by its costs per time slot C_i to arrive at the price the instance will account for. While trying to minimize 4, the following statements have to be fulfilled [9]:

$$s.t. \sum_{i \in I} x_{ait} = 1 \quad \forall a \in A, \forall t \in [S_a, F_a] \quad (5)$$

$$\sum_{i \in I} x_{ait} * U_a \geq O_i \quad \forall a \in A, \forall t \in [S_a, F_a] \quad (6)$$

$$P(D_{it} < R_i) \geq Q_{min} \quad \forall i \in I, \forall t \in [B_i, E_i] \quad (7)$$

$$x_{ait} \in \{0, 1\} \quad (8)$$

$$U_a \in \{0, 1\} \quad (9)$$

$$O_i \in \{0, 1\} \quad (10)$$

$$Q_{min} \in [0, 1] \quad (11)$$

The first constraint 5 asserts that each application has to be assigned to an instance, while at the same time, each, at any point of the run time of an application, can only be hosted by one instance. The next constraint 6 states that each host has to come from a suitable market space, which is necessary to guarantee that non-preemptible applications are not assigned to spot instances, which could be interrupted at any time. The equation formulated in 7 ensures that for every time slot an instance is running, the probability of the resource demand of the applications assigned being within the capacity of said instance is at least the quality of service Q_{min} . The final four constraints 8 to 11 state that the auxiliary variables have to be within a valid range from 0 to 1 [9].

B. Optimization approaches

Having modeled our cloud portfolio optimization problem in subsection III-A, this section will now present our approaches to solving it. As our problem is essentially a multi-dimensional packing problem, it is NP-hard, very complex to solve, and finding an optimal solution is usually not computationally feasible [17]. Therefore, we developed two optimization heuristics to find good approximations of an optimal solution [9].

1) *Greedy algorithm*: Our first algorithm is called *Efficient Resource Inference for Cloud Hosting* (ERICH). It integrates the approach of the widely known bin packing algorithm first fit decreasing (FFD) [18], combining it with the proposed portfolio management strategy by Hwang and Pedram [12]. It is executed in the following four stages [9]:

- *Stage 1*: As the first step, the algorithm sorts the preemptible and non-preemptible applications it receives as input by increasing starting dates, with applications that start earlier being allocated first. Applications with the same starting date are then sorted by non-increasing standard deviation of their resource demand, based on a

proposal by Hwang and Pedram, suggesting that reduced capacity needs can be achieved by grouping workloads with similar resource demand deviation[12]. To ensure that the algorithm prefers cost-efficient hosts, this step also includes sorting all received instance types by cost per time slot for the provided capacity in an increasing order [9].

- *Stage 2*: Next, the algorithm tries to allocate all non-preemptible applications to reserved instances by using the first fit decreasing approach. Iterating through all such applications, the algorithm first tries to find a suitable host that provides the needed capacity over the entire run-time in the existing portfolio. If so, the application is assigned to the said instance. Otherwise, a new instance covering the needs of the application is added to the portfolio [9].
- *Stage 3*: While reserved instances offer significant discounts in comparison to those procured on the on-demand market, the portfolio created in the previous step may be inefficient due to the requirements for minimum run-time in reserved instances. Therefore, step 3 tries to condense the portfolio by removing instances chosen in step 2 and replacing them with on-demand instances. To achieve this, the algorithm iterates through all instances, creating a new temporary portfolio with one reserved instance removed at each step. Next, applications from this instance are assigned to on-demand instances with the same first-fit-decreasing approach used in the previous step. Should this new allocation allow for a cheaper portfolio, it replaces the old one in the next iteration [9].
- *Stage 4*: The final step of the algorithm deals with finding fitting instances for all preemptible applications, which can be assigned to multiple hosts during their lifecycle and can, therefore, be allocated on an individual timeslot basis. To leverage this, the algorithm will first find any time steps for a suitable candidate host and assign preemptible apps to these instances for the respective periods. Should there be any more need for the workload to run the apps, the difference is made up by adding new spot instances [9].

Algorithm 1 describes the steps discussed in pseudocode. To check if an application fits into any given instance, the equation 7 is used.

2) *Genetic algorithm*: Using a genetic algorithm (GA) to solve a bin-packing problem is not an entirely new idea. Still, it has been proven to work well in dealing with combinatorial optimization problems [19], [20], [21], [22]. GAs are based on genetic operators that can be adapted to fit a particular problem, enabling them to perform an efficient and targeted search in the problem space. Our genetic algorithm has been named *Genetic Optimization of Resource Groupings* (GEORG) and will be described in this subsection. The algorithm is described in pseudocode in algorithm 2, followed by a description of the algorithm building blocks [9].

- *Encoding scheme*. Grouping of items and using bins are

Algorithm 1: Efficient Resource Inference for Cloud Hosting

Input: A set of non-preemptible apps $A1$; A set of preemptible apps $A2$; A set of reserved instance types RES ; A set of on-demand instance types ON ; A set of spot instance types $SPOT$

Result: Packing pattern $portfolio$

sort applications $A1$ and $A2$ by increasing start time and non-increasing σ_a

sort RES , ON and $SPOT$ by non-increasing C_i per R_i and time slot

$portfolio \leftarrow$ empty allocation variable

forall $a \in A1$ **do**

 assign a to $portfolio$ (FFD) while only considering RES instances

forall $i \in reserved\ instances\ from\ portfolio$ **do**

$tmp_portfolio \leftarrow$ copy of $portfolio$ without instance i

forall $a \in i$ **do**

 reinsert a into $tmp_portfolio$ (FFD) including ON instances

if $total\ cost\ of\ tmp_portfolio < total\ cost\ of\ portfolio$ **then**

$portfolio \leftarrow tmp_portfolio$

forall $a \in A2$ **do**

 assign a to $portfolio$ without allocating new instances

$gaps \leftarrow$ consecutive time slots where a is not yet assigned to $portfolio$

forall $gap \in gaps$ **do**

 assign a to $portfolio$ for time slots in gap by allocating $SPOT$ hosts

Algorithm 2: GENetic Optimization of Resource Groupings

Input: A set of non-preemptible apps $A1$; A set of preemptible apps $A2$; A set of reserved instance types RES ; A set of on-demand instance types ON ; A set of spot instance types $SPOT$

Result: List of packing patterns (portfolios) $population$

$population \leftarrow$ use semi-random heuristic to create initial population

while $termination\ criteria\ are\ not\ met$ **do**

$parents \leftarrow$ fitness-based selection of individuals from $population$

$offspring \leftarrow$ apply temporal biased crossover for each tuple in $parents$

$offspring \leftarrow$ repair broken chromosomes in $offspring$ after crossover

$offspring \leftarrow$ apply domination mutation operator to random $offspring$

$offspring \leftarrow$ repair broken chromosomes in $offspring$ after mutation

$population \leftarrow$ fitness-based merge of $offspring$ and current $population$

essential for building a GA, according to Falkenauer [20]. Their approach of encoding, where each chromosome consists of an array of bins holding a set of items, is not suitable to our problem with its temporal component. That is why we have chosen a temporal group encoding, where every chromosome locus represents a certain time step, while the allele is a set of instances running at a certain time. Finally, every host is assigned a set of applications, which are then allocated to the corresponding instance during this time slot. [9].

- *Population initialization.* To enable the operations of a GA, like crossover, mutation, and survivor selection, to occur, an initial set of individuals (a population) is needed. For our initialization process, a hybrid approach was chosen to achieve comparatively high fitness from the start while also offering good genetic diversity. Therefore, half of the assignments are done randomly, with the other half having applications that have been assigned to reduce the number of allocated instances and the overall costs [9].
- *Fitness evaluation.* This building block of our GA uses the equation 4, the main function to optimize, to evaluate the fitness of each individual [9].
- *Parent selection and crossover.* For each new generation of the GA, a group of individuals is chosen for parentage of the following generation. These individuals are chosen based on the fitness proportionate roulette wheel method. Out of these, there is a crossover applied by pairs of two parent solutions to create new solutions (offspring). To perform this crossover, a new biased temporal crossover operator was built on the concepts proposed by Quiroz-Castellanos et al. [21], which aims to encourage the passing on well-fitting genetic material to the following generation. Within each gene, which represents a time step, all active instances are sorted by decreasing the average capacity utilization rate and cost per time slot. Using a zip-merging approach, a new partial solution is created from both parent solutions. After removing hosts of already assigned applications, the partial solution is pruned. If any instances violate the constraints outlined in the problem formulation, they will be pruned in the subsequent time steps of the crossover process. Culling instances from solutions may break some chromosomes, as applications can end up with no or only partial assignments. This issue is addressed by employing a basic heuristic to reintroduce any applications that have not been completely assigned [9].
- *Mutation.* This operator is used on freshly produced offspring randomly to introduce new genetic characteristics to individuals and enhance the population's overall fitness. The concept of dominance, introduced by Martello and Toth [23], can lead to tighter packing patterns by replacing a subset of items with an item of larger or equal size. This approach has been shown to be incorporable into the mutation operator of a GA [20], [21], though for usage in this work, the definition of dominance has to

be adapted. Application a , hosted by instance $i \in I$ for the time slots $[S_t, F_t]$, dominates a partition of apps P from instance i if the period denoted by $[S_t, F_t]$ contains all assignments slots of the partition for the respective host. Additionally, the probability of the resource demand of the dominating application being higher than the summed-up capacity requirements of all elements of partition P has to exceed fifty percent. Applying the mutation operator on an individual results in the removal of several instances from the portfolio. Each application that is now unassigned is checked against candidate hosts from the portfolio. By creating partitions of applications of size two from the respective candidate instance, we can try to find partitions dominated by the unassigned applications. Should one be found, the application is swapped with the partition. As these operations may leave broken chromosomes with unassigned applications, just like the crossover operator, the insertion heuristic mentioned previously is used again in order to repair those corrupted individuals [9].

- *Insertion heuristic.* One of the main constraints of our problem formulation is equation 5, which requires each application to be assigned to a fitting instance at any time slot of its run time. As previously mentioned, the crossover and mutation operators may violate said constraint. The insertion heuristic chosen to alleviate this problem uses a naive first-fit approach, which chooses a candidate instance from the existing chromosome to fit orphaned applications. Should a non-fitting host exist in the portfolio, a new random instance is generated to accommodate the corresponding application. The element of randomness is used to reduce the risk of converging on a local optimum by creating additional genetic diversity [9].
- *Survivor selection.* The process of survivor selection decides which individuals at the end of an iteration will represent the next generation. This can be achieved in several ways, with one of the simplest being the selection of the fittest individuals. While more sophisticated methods exist, and simply choosing the best individuals could lead to a lack of diversity and converging on local optima, both our crossover and mutation operators introduce enough randomness, meaning that the fittest individuals are sufficient for our algorithm.
- *Termination.* For termination, our GA can use one of several common stopping criteria, like a maximum number of generations, the fitness scores of the individuals of a population converging to a certain degree and therefore becoming very similar, or the highest level of fitness of an individual not changing any more with more generations [24], [9].

C. Evaluation and Results

In this section, we will discuss how the previously described algorithms were evaluated and the results of this evaluation to present what kind of optimization results a customer could

expect from the cloud portfolio optimizer. Further examples created in the Cloud Portfolio Manager platform will be presented in section V. To evaluate our optimization heuristics, we used synthetic data. The implementation of the algorithms was done in Python 3.9, and the tests were conducted on a PC running a Windows operating system with an Intel Core i7-4770 processor (3.4 GHz base clock, 3.9 GHz turbo) and 16 GB of DDR3 memory at 1600 MHz [9].

1) *Data set description:* With our optimization problem, multiple dimensions influence the difficulty of any test set. As is the case with any bin packing problem, a primary contributor to this is the number of items to be assigned. Unlike many other bin packing problems, our problem must regard the temporal component as the primary driver of execution time, with the number of allocation periods increasing the difficulty. Therefore, the created test data sets reflect a variety in both these attributes [9].

TABLE II
SUMMARY OF APPLICATION DATA SETS

App. Set	Non-Pre.	Pre.	Avg. Res. Dem.	Std. Res. Dem.	Avg. Res. Dev.	Std. Res. Dev.	Avg. Alloc. Periods	Std. Alloc. Periods
apps_1	14.0	6.0	3.2	1.7	0.5	0.5	43.1	33.4
apps_2	59.0	41.0	3.0	2.6	0.5	0.7	63.9	43.9
apps_3	10.0	10.0	3.0	2.0	0.7	0.6	212.2	167.8
apps_4	42.0	58.0	3.1	2.6	0.5	0.6	237.2	171.5
apps_5	7.0	13.0	3.1	2.7	0.6	0.6	2758.5	1996.9
apps_6	41.0	59.0	2.8	2.0	0.5	0.6	2871.7	2055.6

Even though the test data used for the evaluation was synthetically created, it considers realistic scenarios, including anticipated price discounts for spot and reserved instances compared to on-demand instances. This was based on observations done on the major cloud service providers AWS, Google Cloud, and Microsoft Azure. Additionally, for creating our test instances, the price-to-capacity ratio has been modeled similarly to offerings observed on the previously mentioned CSPs. The table II above shows our chosen six sets of applications with their key resource demands and allocation characteristics, while the table III describes the three sets of instance types created for testing, each containing 500 instance types. These were combined in the following pairings to create six test cases: *case_1* (*apps_1*, *types_1*), *case_2* (*apps_2*, *types_1*), *case_3* (*apps_3*, *types_2*), *case_4* (*apps_4*, *types_2*), *case_5* (*apps_5*, *types_3*) and *case_6* (*apps_6*, *types_3*) [9].

TABLE III
SUMMARY OF INSTANCE TYPE DATA SETS

Instance type set	Avg. capacity	Std. capacity	Avg. Res. Prc.	Std. Res. Prc.	Avg. On. Prc.	Std. On. Prc.	Avg. Spot Prc.	Std. Spot Prc.
types_1	9.6	8.8	2.3	2.8	3.1	2.2	2.5	2.2
types_2	10.3	11.4	2.2	2.4	3.1	2.6	3.1	4.8
types_3	9.8	9.9	2.4	3.8	3.1	2.4	2.3	1.7

2) *Results:* The results of our tests focus on three key criteria for evaluation: speed of execution, packing density, and overall costs incurred by the resulting portfolio. To avoid side effects other tasks running on the test machine may have, each algorithm ran ten times to get data on execution speeds.

As you can see in Figure 1 and Figure 2, the optimization approach *ERICH* offered way faster execution speeds, being faster by the magnitude of close to 10. It also results in an almost static execution speed by being deterministic. In contrast, the optimization approach *GEORG* is not only slower, but also way more volatile in terms of execution speed. The slowest run can take 2-3 times longer than the fastest one due to genetic operators being highly influenced by randomness [9].

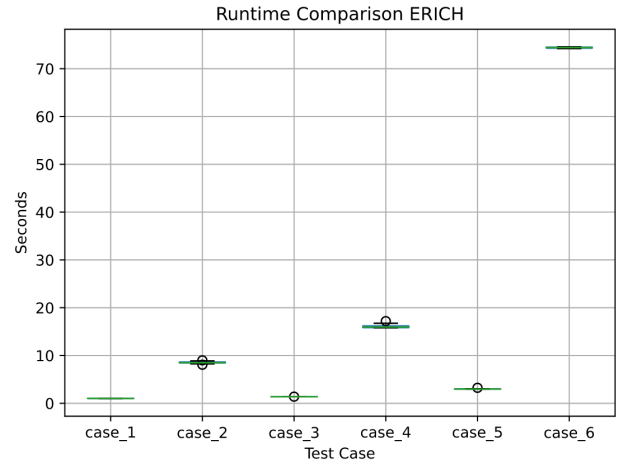


Fig. 1. Execution time ERICH

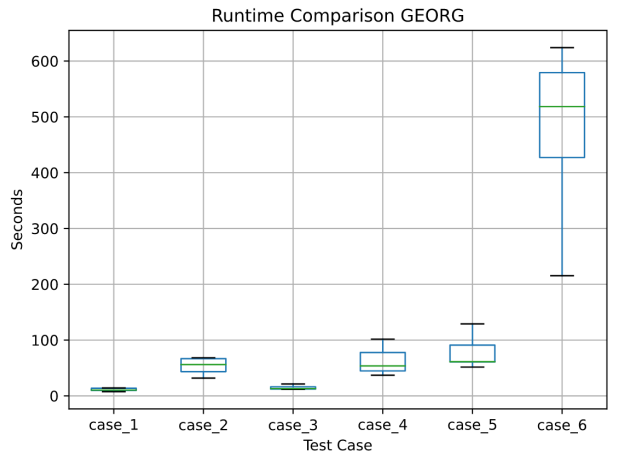


Fig. 2. Execution time GEORG

Our second criterion, the utilization rate, measures the relationship between the total expected resource demand of all assigned applications across relevant time slots and the absolute capacity provided for that period. Depicted in Figure 3, it is easy to see that once again *ERICH* delivers better results than *GEORG*. Depending on the test case, the gap can range

from minor, like in case 1, to very significant, as in cases 4 and 6 [9].

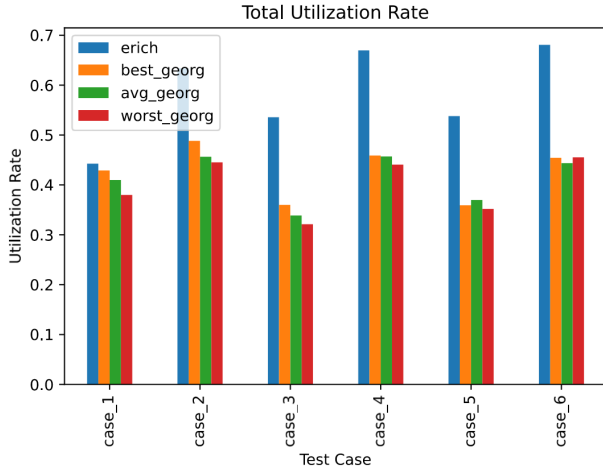


Fig. 3. Utilization comparison between ERICH and GEORG

Finally, in terms of overall costs of the generated portfolio, *ERICH* also outperformed the GA by a significant margin, which can be seen in Figure 4. While this may initially lead to the conclusion that the GA did not work correctly, this is untrue. In Figure 5, one can observe that the costs, meaning the fitness level of the multiple generations for the data set *case_6* improve continuously, with each new generation being fitter than the previous one. The initial population starts with a high degree of genetic diversity and improves with each generation. After ten generations, which is the duration the test ran, the average costs have been reduced by more than 50%. As the initial population of the GA can serve as an example of how an unplanned allocation of resources would look like, it also shows that both algorithms can deliver notable lower costs for a portfolio and, therefore, offer a usable basis to build our Cloud Portfolio Manager platform upon [9].

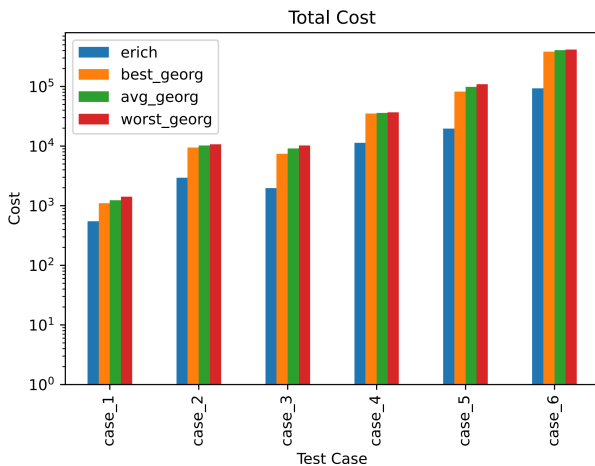


Fig. 4. Portfolio comparison between the two algorithms

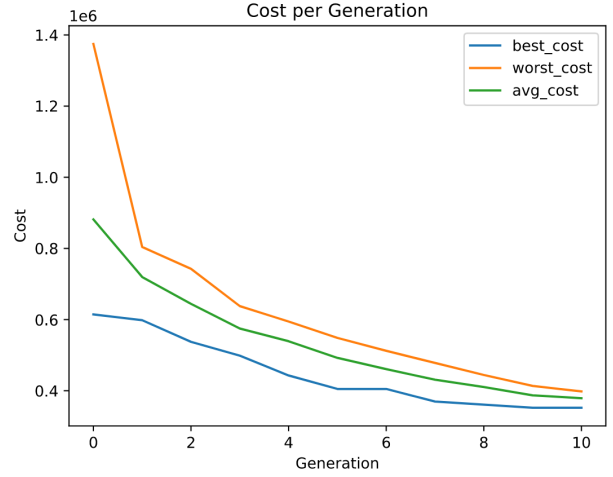


Fig. 5. Cost for each generation for set *case_6* using GEORG

IV. BUSINESS MODEL

In this section we will first present our business model of the Cloud Portfolio Manager using the Business Model Canvas framework by Osterwalder and Pigneur [3], where we describe our model with the help of the nine building blocks introduced in section II). Afterwards, we will discuss how our model can be classified within the context of various classification approaches. Finally, we will compare our model to contender platforms offering services with some similarity to our own.

A. The Cloud Portfolio Manager Business Model Canvas

1) *Customer Segments*: For our customer segment, the sole focus will be on the B2B area, as consumers, so far, have little to no reason to purchase cloud resources for personal use. Within the B2B sector, a big emphasis will be put on the IT sector, ranging from small businesses to large companies, seeing as large parts of this sector move more and more of their IT infrastructure into the cloud. One example would be Netflix, which, since 2015 has moved its entire IT infrastructure to AWS [25].

Surprisingly, when it comes to smaller businesses, as suggested in a study by Jonas et al. in 2013, start-up companies looking for cloud solutions prefer the reputation of a cloud provider over other aspects such as price, security, and reliability [26]. At least at the beginning, a small and not well-known company could mean targeting this customer segment could be more challenging than others. This situation also leads us to conclude that establishing a reputation should be a primary business goal. Besides this, we can increase our attractiveness to small businesses and start-ups by offering cloud consulting and optimization services. These services aim to help customers better understand cloud environments and set up and operate their own cloud portfolios. While it can be expected that larger companies already have access to this knowledge,

the same cannot be said for smaller businesses. Aided by our consultancy services, we can also market our optimization service to this customer segment.

One customer segment our business will specifically focus on within the IT sector is those companies, businesses, and possibly research facilities working with machine learning algorithms. These are uniquely well-suited to be run on cloud resources, as their implementations often offer out-of-the-box preemptibility like PyTorch and Google TensorFlow [27], [28]. Combining machine learning algorithms, resource-intensive applications, and our optimization approach, delivering the best results for preemptible applications, makes them a perfect match. Our value proposition, which will be detailed within its own building block, is well suited to reduce one of the great pains of implementing machine learning solutions, the lack of processing power, by offering cheaper access to computational resources.

While the previously described customer segment will be our primary focus, other sectors also offer a wide range of potential customers. Even in 2017, Mohit et al. suggested that over 90% of organizations were either already adopting cloud infrastructure or planning to do so within the next one to three years [29]. Therefore, our cloud resource optimization approach's potential market is large and diverse.

2) *Value Proposition*: Our primary value proposition, a cost reduction for their cloud portfolio, is unlike many other business models, targeting all customer segments. It can be applied to both existing portfolios and first-time cloud deployments, as long as the customer is roughly aware of their application's computational needs and run-time. A portfolio can be set up in two ways: First, directly via interacting with the platform via its website. Alternatively, outside initial registration, most interactions with the platform can also be done via API, allowing customers to integrate our service within their own systems and automate the process of creating portfolios and creating allocations.

The previously mentioned cost reduction is achieved by a combination of choosing the cheapest instances from the right marketplace and reducing their idle time through continuous monitoring of the needs of the applications. The resulting benefit for the customer entails the direct cost reduction itself and offers easier access to the complex world of cloud computing, which could be especially useful for small and medium-sized businesses. For these clients in particular, we can extend our value proposition by offering a consultancy service for customers who still need the know-how required to take advantage of cloud solutions. Said service would focus on the basics of cloudification, such as which applications are viable for being put into the cloud, the creation of portfolios, and first-time deployment.

3) *Channels*: Listed here, we will address the channels used throughout the five phases of customer interaction through which we aim to reach our customers:

- *Awareness*: For the first phase, raising the customer's awareness of our service, a mixture of online advertisement and direct contact with potential customers through an in-house sales force seems applicable. Furthermore, targeted online advertisement, for example, via Google ads¹, could be considered an option. While having the potential to result in a higher click-through rate, the advertiser has to be careful not to be too intrusive, as this can result in having the opposite effect. It also appears that targeted advertising does not work well with every demographic [30]. After the first customers have been acquired, it can also be reasonably expected that word-of-mouth between different businesses could raise further awareness levels for the company. Another option to reach potential customers is trade fairs focusing on the IT sector.
 - *Evaluation*: Our website is the primary channel used for evaluation and the center of the operation. It provides example calculations, which showcase the potential cost reductions offered by the service, gives an overview of the pricing structure, and offers guidance on how to set up an account. Later, customer success stories, a widely adopted practice among online businesses today, will be showcased on the website. These stories further highlight tangible implementations of our service and provide credibility to our claims of assisting customers in optimizing their cloud portfolio.
 - *Purchase*: Regarding purchasing our products, many online payment services such as PayPal, Amazon Payments, and Credit cards are available and can be provided with relative ease. Besides directly integrating single payment options, companies such as Stripe² offer a single API that enables the user to choose from a range of standard online payment options.
 - *Delivery*: The delivery of the optimization service to the customer can also be achieved through the platform's direct channel, either by direct customer interaction on the website or via API call. As for the consultancy side of the business, we expect delivery to be done via personal interaction, both physical and online, depending on the customer's preferences.
 - *After sales*: The Cloud Portfolio Manager will first focus on providing post-purchase support to customers through our website. Here, the user can overview his portfolios, optimizations, subscriptions, and general account information. A FAQ page can also help answer general questions. Direct support through a personal customer support force should also be implemented for more complex cases.
- 4) *Customer Relationships*: When it comes to the methods of interacting with the customer base, the Cloud Portfolio Manager will focus on two areas: For the majority of interactions, such as setting up an account, creating and managing

¹https://ads.google.com/intl/de_at/home/

²<https://stripe.com/en-gb-at/payments>

a portfolio, an automated service based on our website will be used. Complementing these services is a sales- and CRM- (Customer Relationship Management) force, which can be contacted personally via channels such as e-mail, phone, or video calls. This enables a more personal relationship with the customer and answers complicated and personal questions regarding single customers, which cannot be served easily via an automated service. This additional service is available to the customer during the whole interaction, from pre-sale evaluation until the purchase is completed. Another important aspect of this building block is the interaction between the consultancy force and the customers. As for offering a consultation service, direct human interaction is preferable, as each customer is assigned and mainly interacts with one consultant. This can be over various channels, but due to the close nature of the relationship, it will result in more face-to-face interactions than the other services of our business. A study by Roy et al. suggests that direct interaction with the customer is also preferable, as service experience is valued even higher than the actual service quality in B2B services [31].

5) *Revenue Streams*: Regarding revenue streams, a plethora of options are available at first glance. However, as we state in this section, most of them are not readily applicable to our platform for one reason or another, leaving us with one very widely used revenue stream as our primary source of revenue.

The first option we want to discuss is advertisement. While it is the main revenue stream of many large online platforms such as YouTube and Facebook, these are mass media B2C operations with millions or even billions of users, where each user only generates a relatively small amount of revenue through displayed advertisements. For our platform, which offers a specialized service to business customers, advertisements would mainly discourage users and possibly damage the brand reputation [32].

Next, we have considered the option of transaction fees. This could be implemented on a usage-based model, where the customer would pay a certain amount for each optimization based on the portfolio size. Another implementation could be a one-time transaction enabling unlimited access to the Cloud Portfolio Manager. Both options are not optimal for our product. The usage-based model does not lend itself well to a product that is meant for continuous optimization. This would either lead to a need to frequently pay for a new allocation or prevent customers from getting the full benefit of an approach that is meant to adapt to changing demands in their portfolio. The one-time charge option faces another drawback, making it unfeasible. Seeing as our portfolio manager is intended as a continuous service, this one-time charge would have to account for a long service time, which in turn would increase the price a level, which would turn it into a severe deterrent for new customers that are not entirely convinced of the benefits of the product yet.

Another alternative would be a system based on a brokerage

fee. In this case, a part of the cost reduction achieved by the Cloud Portfolio Manager would be taken as our revenue. The significant flaw with this idea, though, is that our system does not aim to directly access and manage the customer's cloud instances. This would result in customers needing to accurately and honestly report their current cloud expenses and their achieved cost reductions, which lends itself to be abused way too easily.

Finally, we propose that subscription fees are the revenue stream best suited to our business model. They tackle several disadvantages mentioned in the previously discussed systems, such as fitting well with a continuously running service, unlike pay-per-use transaction fees and a low entry barrier compared to a one-time charge. The subscription fee, due in a monthly interval, could either be based upon a system with different levels of subscriptions, offering support to differing sizes of cloud portfolios and varying levels of customer support, or directly scaling with the size of the optimized cloud portfolios.

As for revenue streams concerning the consultancy side of the business model, three monetization variants are possible. First, a classic hourly fee would most suit customers needing only a more minor assistance contingent. Another variant would be offering package deals with a fixed price, such as offering to help set up the first cloud portfolio for a customer. Finally, higher-level subscription models for the cloud portfolio optimizer platform could include a certain amount of consultancy services for free.

6) *Key Resources*: As for the differing categories of key resources, the following can be said: When it comes to physical key resources, there is little to be mentioned here. While server resources are necessary to host the platform, the hardware required is highly interchangeable and easy to come by. Furthermore, it could be more advantageous to completely forgo physical servers and host the platform itself on a cloud server.

Financial key resources may also not play a huge role in starting off. Of course, financial resources such as cash or credit will be needed to set up the business, but due to its nature, these will be of a small volume. One possible option to gain access to financial resources to start the business would be to apply for one of the many tech start-up sponsorships available in Austria. The most critical key resources are within the intellectual resource category, consisting of the portfolio management platform and, in particular, the optimization algorithms, which are at the heart of the operation and are needed to realize all of the other components of the business model.

Finally, when it comes to human key resources, the following groups can be expected to be part of those: Especially for development and improvements to the platform, further full-stack developers could be needed. Furthermore, a small team of cloud consultants would be responsible for providing customers with know-how on cloud solutions. Besides that, a

group of employees helping with sales and CRM-related topics should be employed as well.

7) *Key Activities*: The most important key activity of the business is the operation and maintenance of the Cloud Portfolio Manager platform. In this capacity, the platform offers the customer an automated service. After creating an account and logging in, customers can create, delete, and change their cloud portfolios. Besides a simple interface to directly manipulate a portfolio, the main feature for management is the possibility to upload load profiles based upon which an optimized portfolio of instances is calculated and displayed to the customer. The load profiles can be uploaded manually on the website and through a REST API.

In addition to the service provided by the platform, the other main activity is problem-solving for the customer by offering our consultancy service. This mainly entails sharing cloud-related know-how and aiding the customer with planning, creating, and monitoring their own cloud solutions and migrating their existing applications.

8) *Key Partnerships*: Within this building block, the most prevalent partnerships are the various CSPs for which the platform offers portfolio optimization. While actively managing the customer's portfolio is not part of the business plan so far, it is crucial to the platform's functionality to access the instances and their respective pricing offered by the various providers. Luckily, all major CSPs provide APIs that give access to live data on the current availability and pricing of their offered instances.

If the business grows beyond a small scale, it would be possible for certain activities, such as customer support or cloud consulting, to be outsourced to external partners, which would turn these into key partnerships as well.

9) *Cost Structure*: The cost structure of the business model is intended to lean towards being value-driven, focusing on creating value for the customer. As the business operates online with a web platform at its center, scaling should be achievable relatively easily. While an increase in the customer base will require additional staff for CRM and consulting, the platform performance for handling a certain amount of customers can be scaled almost infinitely and with a mediocre but easily projectable impact on costs.

In contrast to the platform's operating costs, which should be manageable, the same cannot be said about its initial creation, which can be expected to be one of the significant cost factors in starting the business. Once the platform is in operation, the major cost factors will be the personnel required for the CRM, consulting operations, and resources spent on updating and expanding the platform. Further costs that have to be taken into account come from actions taken towards the acquisition of customers, especially those mentioned in the awareness section of the "channels" building block.

The Business Model Canvas

Key Partners Cloud Service Providers (CSPs): Offer the products which the platform aims to optimize the use of	Key Activities Operation and maintenance of cloud manager platform Enable customers to optimize their cloud portfolios Offer know how on cloud operations, monitoring and migration	Value Proposition Reduction of a customers cloud portfolio costs Both for existing and new portfolios Finding cheapest instance from the right marketplace and reducing idle time	Customer Relationships Automated service via website with cloud portfolio manager Personal channels like phone, mails or meetings for consultation services or questions regarding the online service	Customer Segments Only B2B, not much market for personal cloud resources Focus on IT sector from small businesses to large companies Consultancy services more targeted at smaller businesses
	Key Resources Intellectual: Portfolio manager platform and optimization algorithms Human: Full stack developers, cloud consultants, sales and CRM team Financial: Payment Providers	Channels Online advertisement and personal sales force for awareness Evaluation, delivery and after sales via website and personal consultation Purchase via online payment services	Revenue Streams Monthly subscription fee for usage of website and cloud portfolio optimizer Varying levels of subscription based on support level and portfolio size Consultancy service partly included in higher subscription levels, otherwise hourly fees	Cost Structure Operational costs of platform low, but initial creation is a major cost factor Once operational, consulting personal major cost factor Costs for raising awareness has to be taken into account

Fig. 6. Business Model Canvas [3] of Cloud Portfolio Manager

B. Model Classification

Having described the Cloud Portfolio Managers business model, we will now discuss how our model can be classified within various classification systems developed in the literature. Besides the system offered by Timmers [33], we will also apply the classifications of Osterwalder and Pigneur [3].

1) *Classification according to Timmers*: Many authors worked on classifications for e-business models like the classification made by Timmers [33], which, despite the age of the paper, still applies well to today's e-business models. The classes are the E-shop, E-procurement, E-auction, E-mall, third-party marketplace, virtual communities, value-chain service provider, value-chain integrator, collaboration platform and information broker, trust, and other services. These different models are all aligned along two criteria: functional integration, from a single function to multiple, and their degree of innovation from lower to higher. Within the classification system of Timmers [33], both of our business models' primary services put the platform into the information broker model. Those focus on providing information by analyzing or finding data that can benefit the customer's operations, which is the case for both our optimization algorithms and our cloud consultancy operations. Aligning our model along the two axes Timmers' system uses, we will first find a high degree of innovation with our cloud portfolio optimization, being one of the first ever to offer this kind of service and cloud consultancy being a service that has only emerged in the past few years. Placing our business along the functional integration axis, we find that with two main functions, it falls towards the lower end of this spectrum. Both these placements fit well with the information broker classification, as seen in the Figure 7.

2) *Classification according to Osterwalder and Pigneur*: Now we will classify our model according to the patterns of Osterwalder and Pigneur [3]:

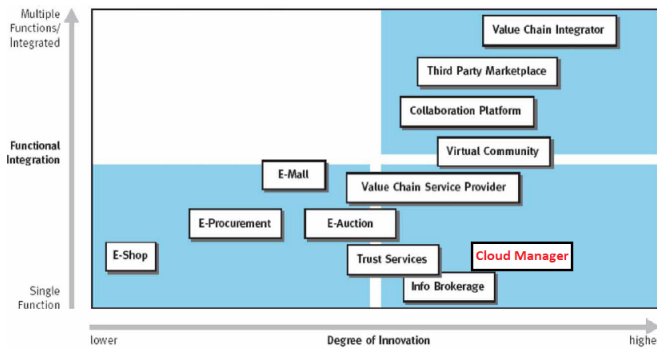


Fig. 7. Classification of Cloud Portfolio Manager within internet business models, figure from B. Wall et al., 2007, Production Planning and Control, page 248 [34]

- In the case of the "Unbundling" pattern, our business model falls within the product innovation category, with a relatively new service on the market and only a few small players present in it so far. The consultancy aspect of our business model could be seen more as a customer relationship management business, meaning that, to prevent this service from conflicting with our cloud optimization service, separating them into different entities like business units may be necessary.
- Next, considering the so-called "Long Tail", we would argue that our model does not adhere to this pattern. Our business focuses on optimizing portfolios consisting of widespread cloud instances that are sold frequently, not making them niche products. We also do not offer a wide range of niche products, only a few services.
- On the other hand, at first glance, it can be argued that our product is a "Multi-Sided Platform" of some kind, as it brings together two interdependent groups, and without the presence of CSPs, our platform could not exist. On the other hand, while customers optimizing their portfolio are profiting from our service, the same cannot be said for the CSPs themselves, as they rather stand to lose extra revenue generated by unused but paid-for instances running idle. Therefore, as the "Multi-Sided Platform" pattern should be of value for all involved groups, we argue that our model does not conform to this pattern.
- The same cannot be said for the "Free" pattern, as our business model will include a small part of our services free of charge. While there will be various levels of subscription that will cost a monthly fee, there will also be a free trial functionality, offering limited access to the service to lure in potential customers. This approach has been labeled as "Freemium".
- The final pattern to take into account is the "Open" business model. While the "inside-out" approach is not planned to be part of our business model, the "outside in" idea could potentially be explored in the future by integrating external cloud frameworks into the cloud optimization platform, which can make the product more

appealing to potential customers.

C. Business Model Contenders

There are only a few business models similar to our approach.

1) *spot.io*: First, we will look at the platform *spot.io*³, which offers a range of tools for customers to analyze, manage and optimize their cloud portfolios. Two of their products provide functionality similar to our cloud portfolio optimization approach. Elasticgroup uses AI predictions to automate infrastructure scaling with the help of *spot* instances fully. On the other hand, *Eco* tries to optimize the customer's cloud portfolio by finding and off-loading unused Reserved Instances and Saving Plans. Furthermore, in a similar fashion to our platform, *spot.io* also offers consulting options to their customers, though limited to their highest subscription plan. So overall, regarding the value proposition, this firm is similar to our business model.

2) *Densify*: Next up is the platform *Densify*⁴, which offers a cloud management and optimization service with a similar value proposition to our platform. There is a contrast to our platform regarding revenue stream and pricing model. *Densify* charges the customer for each managed instance per year, with the price depending on the number of instances. Besides the high prices and seemingly not offering solutions for a portfolio of under 1000 instances, *Densify* does not seem to be incentivized to optimize a customer's portfolio to need fewer instances, as they charge per instance.

3) *Terraform*: The final cloud optimization platform discussed is *Terraform*⁵. It allows users to express their computational infrastructure needs in their own semi-structured language, which then can be deployed to a range of resource providers like AWS or Google Cloud. The value proposition is to simplify and enable infrastructure management across multiple cloud providers. While there is also the capability of easy scalability of resource needs, there does not seem to be any optimization of the cloud portfolio. Therefore, this platform's value proposition does not directly compete with our Cloud Portfolio Manager but could work in a complementary way.

V. PORTFOLIO MANAGER PROTOTYPE

This section will present the prototype of our Cloud Portfolio Manager. It will present an overview of the various pages and showcase the various functionalities of the application.

A. Login, Registration and Landing Page

Starting off, the user is presented with the login page, as seen in Figure 8, where the user can enter their e-mail address and password to access the website. In case a new customer does not have an account yet, the *Register* button leads to the registration page as seen in Figure 9, allowing for a new

³<https://spot.io/>

⁴<https://www.densify.com/>

⁵<https://www.terraform.io/>

account to be created by entering a valid e-mail address, a username, and a password.

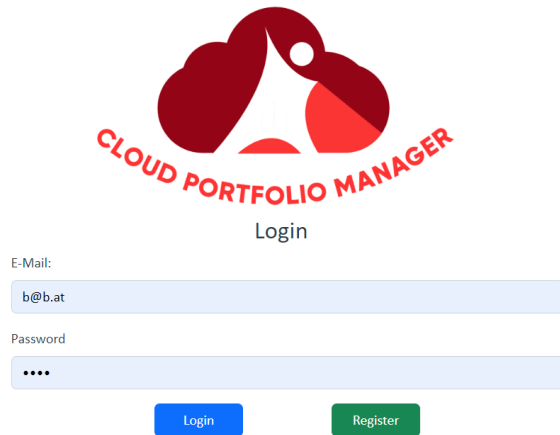


Fig. 8. Login Page

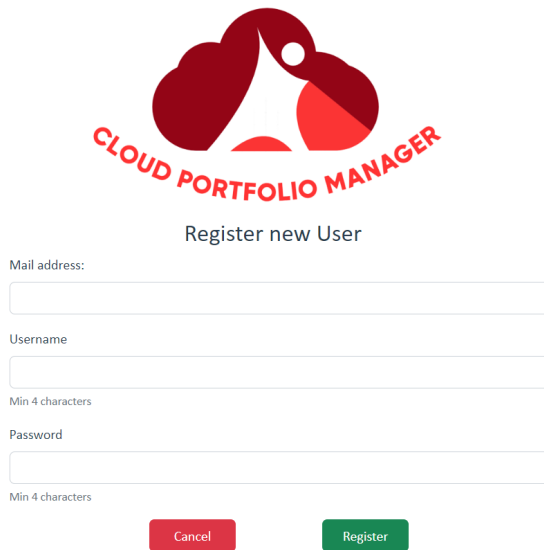


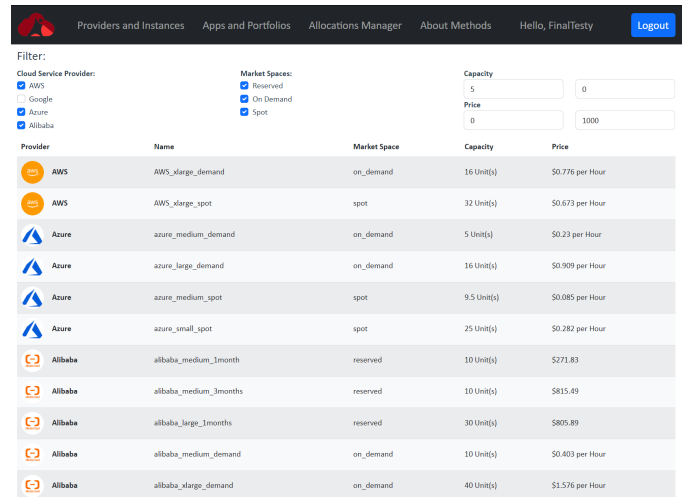
Fig. 9. Registration page

Logging into the website with the correct credentials brings the customer to the landing page displaying the platform’s logo and listing its creators. This page and all others besides the login and registration also feature a navbar for easy navigation between the various pages.

B. Instances Page

This view can be navigated to through the *Providers and instances* tab in the navbar and contains information about available instances from the various providers. For our prototype, we chose a range of instances from the four biggest CSPs: AWS, Google Cloud, Microsoft Azure, and Alibaba. They all give good examples of what is available on the market. This could be adapted for future development to

load instances provided by CSP APIs. The list on this page gives an overview of each instance’s main attributes: provider, name, market space, capacity, and price. Furthermore, a filter function allows users to simplify their search for instances. For example, as shown in Figure 10, the search does not include Google Cloud instances but instances from all market spaces, with a capacity of 5 or higher and a price of up to 1000\$.



Provider	Name	Market Space	Capacity	Price
AWS	AWS_xlarge_demand	on_demand	16 Unit(s)	\$0.776 per Hour
AWS	AWS_xlarge_spot	spot	32 Unit(s)	\$0.673 per Hour
Azure	azure_medium_demand	on_demand	5 Unit(s)	\$0.23 per Hour
Azure	azure_large_demand	on_demand	16 Unit(s)	\$0.909 per Hour
Azure	azure_medium_spot	spot	9.5 Unit(s)	\$0.085 per Hour
Azure	azure_small_spot	spot	25 Unit(s)	\$0.282 per Hour
Alibaba	alibaba_medium_1month	reserved	10 Unit(s)	\$271.83
Alibaba	alibaba_medium_3months	reserved	10 Unit(s)	\$815.49
Alibaba	alibaba_large_1months	reserved	30 Unit(s)	\$805.89
Alibaba	alibaba_medium_demand	on_demand	10 Unit(s)	\$0.403 per Hour
Alibaba	alibaba_xlarge_demand	on_demand	40 Unit(s)	\$1.576 per Hour

Fig. 10. Instances page

C. Apps and Portfolios Page

Next up, the *Apps and portfolios* page enables the user to manage two of the main components of the Cloud Portfolio Platform. As seen in Figure 11, the left side lists the user’s applications and details like mean resource demand, demand variance, preemptibility, and starting and finishing time. On the right side of the page is a list of the user’s portfolio, including details like which providers should be considered for any possible allocation, a minimum quality of service, the number of apps in the portfolio, and a list of which applications exactly the portfolio consists of. Finally, it also states the portfolio’s version, which is incremented every time a portfolio or one of its applications is changed. It is used to track which portfolio version a specific allocation has been calculated for. This enables the user to spot if any of their allocations are outdated or if any specifications or the makeup of the underlying portfolio have changed.

To create a new application or portfolio, two green buttons depict a plus sign on each side of the page. These open the respective application and portfolio forms, as seen in Figure 12 and Figure 13. To create an application, the user has to fill out the application form, including a unique name, mean resource demand, demand variance, a checkbox for preemptibility, and finally, the starting and finishing time chosen via a date-time picker. Should the user wish to create a portfolio, the portfolio form requires a unique name and a minimum quality of service, which gives a percentage of time the apps in the portfolio are required to run. The portfolio version, described

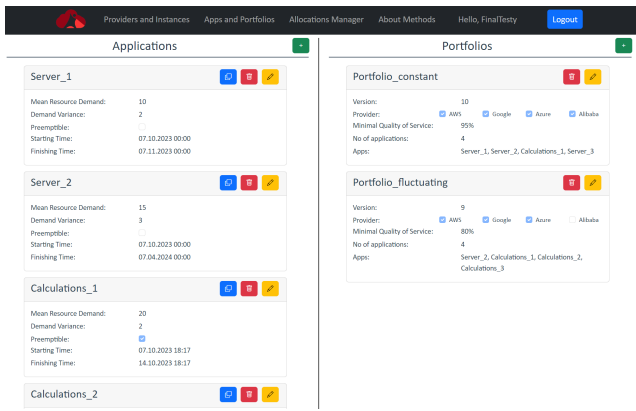


Fig. 11. Apps and portfolios page overview

previously, cannot be changed manually by the user. The portfolio form also requires the user to choose at least one CSP to be considered for allocations and which apps should make up the portfolio. To ensure suitable inputs for both forms, they also feature various checks, giving instant feedback to invalid inputs, such as an application's finishing time before its starting time.

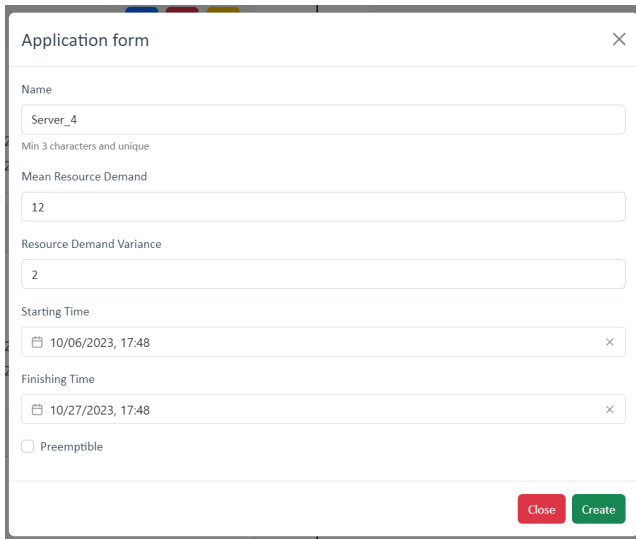


Fig. 12. Application Form

The user can update each application and portfolio by clicking the yellow button, which displays a pen icon for every application and portfolio. This will open up the respective form already filled out by the app's or portfolio's data. For ease of creating several applications with similar characteristics without having to fill out the entire form every time again, applications also feature a blue copy button, which will open the application form filled out with the characteristics of the chosen application and the suffix "_copy" added to its name. Furthermore, clicking the red button displaying a trashcan icon will delete the application or portfolio of choice. Deleting an application will also remove it from any portfolios it may be

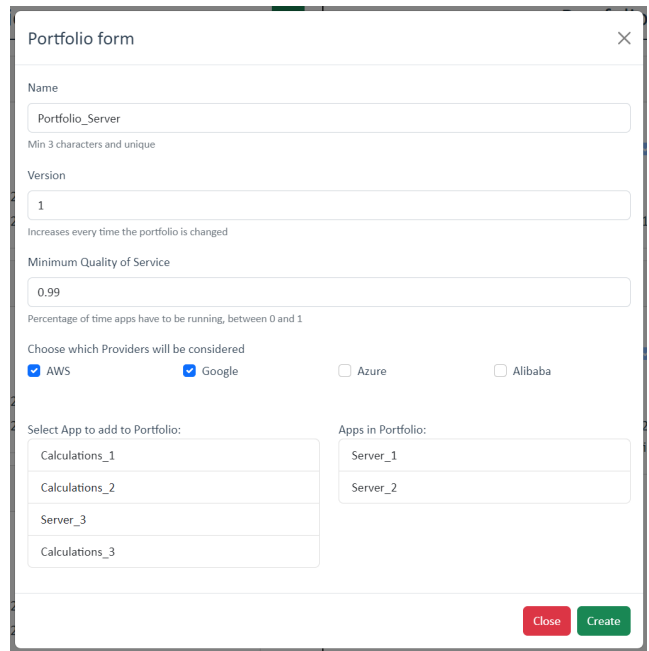


Fig. 13. Portfolio Form

part of and increment the portfolio's version. To give feedback on operations performed on this page, creating, updating, and deleting applications and portfolios will result in a short popup denoting a successful operation or, should any errors occur, give the user notice that there has been an error.

D. Allocations Page

This tab focuses on the primary feature of the Cloud Portfolio Manager: creating allocations for cloud portfolios. The user has a dropdown menu at the top of the page, which lists their created portfolios. Choosing a portfolio shows its details on the side, and all already existing allocations for this portfolio are below. Every allocation has an overview stating which algorithm was used, which portfolio version it was made for, its total costs, and the mean overall utilization achieved with this allocation. As allocations can take a while to be calculated, especially in the case of using the GEORG algorithm, there is also a field stating if the allocation is already completed. Below the general stats are two fields, which can be extended by clicking on them. The first contains a complete list of all instances used for this allocation and some statistics like capacity, price, and the beginning and end of the instance's run time. The second field contains more detailed statistics about the allocation, such as separate statistics for reserved, on-demand, and spot instances. Each allocation can also be deleted by clicking the red button with a trashcan icon in the header section of each entry. On the right side of the allocations list, this page also features some data visualization with graphs comparing the price, utilization overall, and utilization split by instance type for each allocation as bar charts.

To create a new allocation, the "New Allocation" button opens a form where the user can choose which algorithm should be

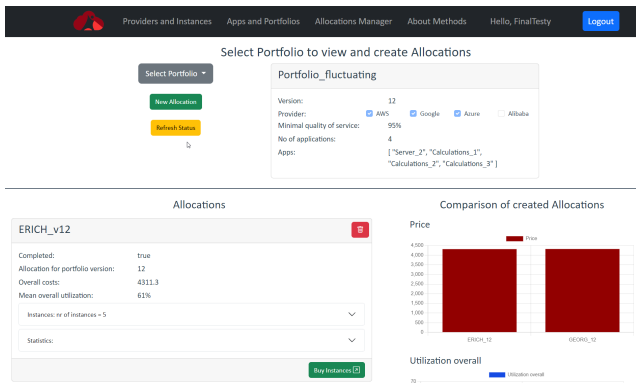


Fig. 14. Allocations page overview

used. In the case of using ERICH, that is all that is required to do, as there is no parameterization possible. Otherwise, when selecting GEORG, various options for configuring the genetic algorithm are available. Examples of this would be the size of the population, the number of generations, and the mutation rate. Should the user want to look into this topic sparingly, there is a default value for each. Otherwise, this enables experimentation with this algorithm, which can lead to varying results. The user should be aware that this will also impact performance; for example, a very high population size is more resource-intensive for the system the platform runs on. As the calculation of the allocations is run asynchronously, there is also a "Refresh" button on this page, which reloads the allocation data from the backend.

VI. USER EXPERIENCE EVALUATION

After the presentation of our prototype, this section will provide a brief evaluation of the user experience (UX). It is crucial for any platform targeting success in the public market to possess a convincing user interface. To this end, we will discuss the usability heuristics developed by Nielsen et al. and, based upon these same heuristics, present a brief survey on the quality of our user interface [35], [36].

A. Usability Heuristics

Heuristics are an approach to finding not perfect but adequate solutions to a problem, such as the optimization approaches we developed in section III. This methodology can also be applied to evaluate the quality of user interfaces. One of the most commonly used and widely known usability heuristics is those developed by Nielsen in the 1990s. He proposed ten general principles for designing user interfaces, which are the following [35], [36], [37]:

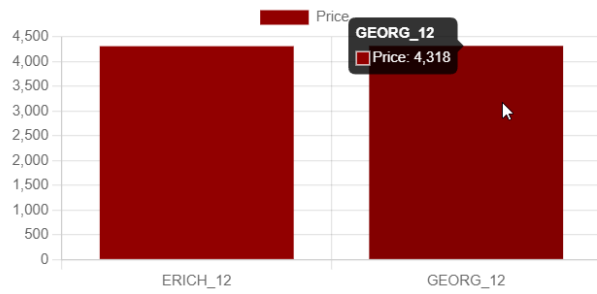
- *Visibility of system status:* This heuristic describes the ability of design to always communicate to the user about what is going on within the system in an easily understandable and immediate manner. A user must know what the previous interactions resulted in and understand which steps can succeed.

Fig. 15. Form for creating a new GEORG allocation

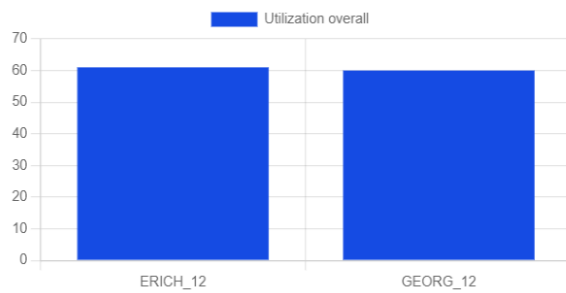
- *Match between the system and the real world:* This refers to a design that communicates by using words and concepts that a user recognizes instead of internal terms. A design that follows this heuristic enables intuitive use of the interface without needing to learn new words or concepts.
- *User control and freedom:* As users like to try various actions and mistakenly perform others, there should always be an obvious way to go back a step and cancel any action. It prevents users from getting stuck and gives them the confidence to try any action within a system freely.
- *Consistency and standards:* An interface should be consistent to offer a good user experience. It refers not only to consistency within the product you offer but also to similar products a customer could be accustomed to and have developed expectations from.
- *Error prevention:* This refers to designing an interface in a way that helps to prevent the occurrence of errors.

Comparison of created Allocations

Price



Utilization overall



Utilization detail

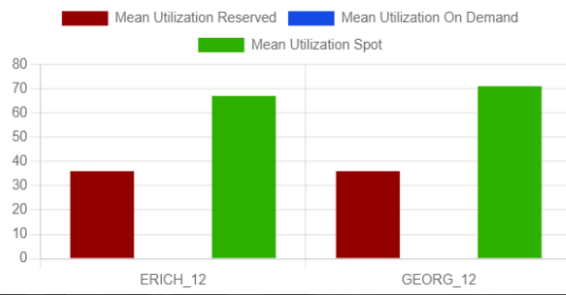


Fig. 16. Graphs for comparison between a GEORG and ERICH allocation for an example portfolio

Errors can be categorized into slips and mistakes. Slips are unconscious errors caused by inattentiveness and can be combated by setting helpful constraints and defaults. On the other hand, mistakes happen consciously and result from the design not properly communicating the model to the user. They can be alleviated by enabling the undoing of errors and reasonable warnings.

- *Recognition rather than recall*: As users have limited short-term memory, a good design does not rely on the recall of elements, actions, and options but encourages recognition. Additional information is needed and should be easily accessible if required.
- *Flexibility and efficiency of use*: This heuristic provides experienced users with possibilities to speed up interactions that inexperienced users may not need, e.g., keyboard shortcuts. These allow for flexible processes

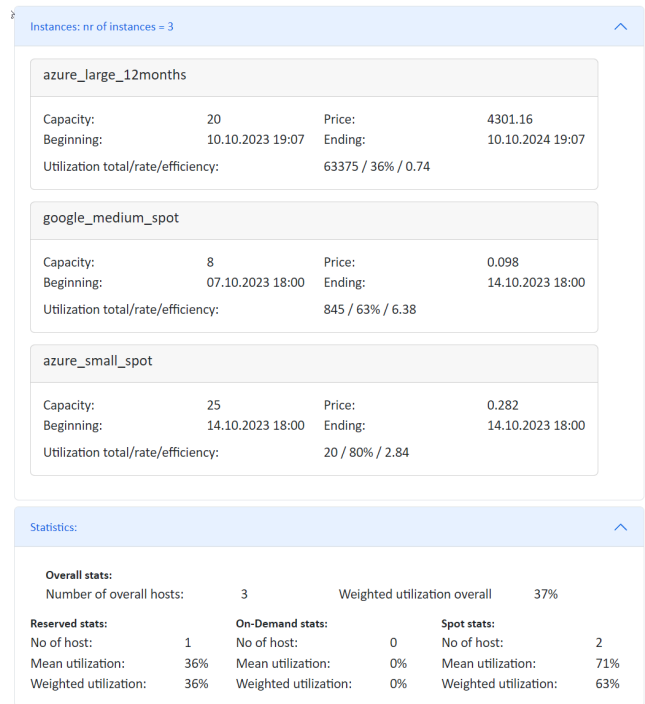


Fig. 17. Details for an allocation with statistics and instances

that can be executed in various ways.

- *Aesthetic and minimalist design*: The design of interface elements should prioritize essential information required for their functionality. Additional irrelevant or rarely needed information competes with relevant elements for visibility.
- *Help users recognize, diagnose, and recover from errors*: If errors occur, the system should inform the user using plain language, providing an accurate description and, when possible, a suggested solution. Technical terms, such as error codes and unusual visual design for error messages, should be avoided.
- *Help and documentation*: While in the best case, a system does not need any further explanation, it may be necessary to offer documentation to complete some tasks. Said documentation should be concise, easily searchable, and consist of concrete steps to be carried out.

B. Methodology

We will evaluate the Cloud Portfolio Optimizer frontend design based on these heuristics. This will consist of five testers going through a set of tasks on our platform and filling out a questionnaire afterward. While the number of testers may seem relatively low, Nielsen et al. mention in their work that in contrast to one tester often missing a lot of problems, three to five aggregated evaluations offer good results [35].

For the evaluation process, each tester is expected to complete the following list of tasks:

- Create a new account and log into it.

- Navigate to the 'about methods' page and skim over the description.
- Go to the instances page, look at the instances displayed, and use the available filters.
- Navigate to the applications and portfolios page, create four new applications, update one of them, copy another, and finally delete one.
- Create two portfolios, update one, and delete one.
- Finally, proceed to the allocations page and create one allocation for each optimization approach.
- Log out from the platform, which returns the user to the sign-in page.

Having completed the tasks above, the testers will be asked to complete a questionnaire based on Nielsen's heuristics. It was created using Google Forms⁶), and asked the user to rate the platform in regards to how well it adheres to each of the design heuristics on a scale from 1 to 5, with 5 representing the best possible adherence. Furthermore, the testers were questioned on whether they found any issues or had suggestions regarding each heuristic. To gauge the testers' expertise in regards to IT in general and the topic of cloud markets, the survey also contains a self-assessment of these topics.

C. Results

The questionnaire results gave helpful feedback on the design of the user interface and pointed out a couple of errors that were overlooked in development. Five testers completed the survey as described in subsection VI-B. Regarding IT-related experience, two testers reported no background in IT, one mentioned having educational knowledge, and two stated having work-related IT experience. The testers' knowledge about the cloud market was somewhat limited, with three testers stating their understanding to be cursory and two having no knowledge about the topic.

Overall, the platform was perceived as adhering to most heuristics pretty well, with some achieving better results than others. The best-rated heuristics were those of "aesthetic and minimalist design" as well as "user control and freedom," achieving an average of 4.6 and 4.4 points, respectively. The heuristics of "consistency and standards" and "recognition and standards" also scored well, averaging 4.0 points. All other heuristics got an average of 3.4 or 3.8, with the notable exception of "error prevention" only scoring 3.0 on average. The complete list of average scoring can be seen in Table IV.

These results point towards the platform having an aesthetic and mostly easy-to-use design, with room for improvement in error prevention and handling, as well as documentation and help. All testers, except for one, who provided no useful feedback by rating several heuristics poorly without offering any comments, also reported issues they encountered and provided suggestions for improvement.

⁶<https://docs.google.com/forms/u/0>

Visibility of system status	3.8
Match between the system and the real world	3.4
User control and freedom	4.4
Consistency and standards	4.0
Error prevention	3.0
Recognition rather than recall	4.0
Flexibility and efficiency of use	3.8
Aesthetic and minimalist design	4.6
Help users recognize, diagnose, and recover from errors	3.4
Help and documentation	3.4

TABLE IV
TABLE OF AVERAGE ASSESSMENT FOR EACH HEURISTIC

To sum up, the testers rated the platform positively regarding most heuristics and provided helpful feedback for improvement. The suggested feedback concerning minor changes has already been applied, while others inform future ways of improving the platform, such as adding a user guide.

VII. CONCLUSION AND FUTURE WORK

This paper focuses on cloud portfolio management platforms and related business models. Cloud portfolio management is primarily concerned with finding cost-efficient allocations for cloud resources. Having discussed the necessary concepts around the work built, we proposed our business model for a Cloud Portfolio Manager. With the business model in place, we implemented a prototype of the Cloud portfolio manager [38], incorporating two optimization algorithms that had been developed [39].

As the cloud computing market has been on a meteoric rise over the past years and is still expanding, the topic of cloud portfolio management will likely keep or expand its relevance in the coming years. Future work would see the prototype developed into a fully functional public platform operating with our business model or a modification of it. Furthermore, the platform's offered services could be extended into various monitoring functionalities and direct control of portfolios via the platform. Further work will improve the existing optimization algorithms and create new ones. In addition to tighter packing, these could, for example, elaborate on the resource constraints considered, such as network capabilities and storage. Finally, further research will build upon the business model presented in this thesis. Thus, we work on integrating the presented approach into our work on automatic and dynamic resource (re-)negotiation and contracting of resources between providers and customers [40].

REFERENCES

- [1] S. O. Services, "Annual revenue of amazon web services (aws) from 2013 to 2022," <https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/>, accessed: 2023-10-09.
- [2] B. Pittl, W. Mach, and E. Schikuta, "Cost-evaluation of cloud portfolios: An empirical case study," in *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER)*. SciTePress, 2019, pp. 132–143.
- [3] A. Osterwalder and Y. Pigneur, *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons, 2010, vol. 1.

- [4] M. Law, "Technology biggest cloud providers 2023," <https://technologymagazine.com/top10/top-10-biggest-cloud-providers-in-the-world-in-2023>, 2023, Accessed: 16.06.2023.
- [5] E. Fiel, "Conceptualising business models: Definitions, frameworks and classifications," *Journal of Business Models*, vol. 1, no. 1, pp. 85–105, 2013. [Online]. Available: <https://eprints.qut.edu.au/75316/>
- [6] S. Labes, N. Hanner, and R. Zarnekow, "Successful business model types of cloud providers," *Business & Information Systems Engineering*, vol. 59, no. 4, pp. 223–233, 2017.
- [7] A. Elhabbash, F. Samreen, J. Hadley, and Y. Elkhatib, "Cloud brokerage: A systematic survey," *ACM Comput. Surv.*, vol. 51, no. 6, Jan. 2019. [Online]. Available: <https://doi-org.uaccess.univie.ac.at/10.1145/3274657>
- [8] E. Filiopoulou, P. Mitropoulou, C. Michalakelis, and M. Nikolaidou, "The rise of cloud brokerage: Business model, profit making and cost savings," in *Economics of Grids, Clouds, Systems, and Services*, J. Á. Bñares, K. Tserpes, and J. Altmann, Eds. Cham: Springer International Publishing, 2017, pp. 19–32.
- [9] M. Kiessler, V. Haag, B. Pittl, and E. Schikuta, "Optimization heuristics for cost-efficient long-term cloud portfolio allocations," in *International Conference on Information Integration and Web*. Springer, 2022, pp. 309–323.
- [10] I. Jangjaimon and N.-F. Tzeng, "Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 396–409, 2013.
- [11] P. Sharma, D. Irwin, and P. Shenoy, "Portfolio-driven resource management for transient cloud servers," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 1–23, 2017.
- [12] I. Hwang and M. Pedram, "Portfolio theory-based resource assignment in a cloud computing system," in *2012 IEEE Fifth International Conference on Cloud Computing*. IEEE, 2012, pp. 582–589.
- [13] J. Martinovic, M. Hähnel, W. Dargie, and G. Scheithauer, "A stochastic bin packing approach for server consolidation with conflicts," in *Operations Research Proceedings 2019*. Springer, 2020, pp. 159–165.
- [14] G. Wu, M. Tang, Y.-C. Tian, and W. Li, "Energy-efficient virtual machine placement in data centers by genetic algorithm," in *International conference on neural information processing*. Springer, 2012, pp. 315–323.
- [15] M. De Cauwer, D. Mehta, and B. O'Sullivan, "The temporal bin packing problem: an application to workload management in data centres," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016, pp. 157–164.
- [16] M. Dell'Amico, F. Furini, and M. Iori, "A branch-and-price algorithm for the temporal bin packing problem," *Computers & Operations Research*, vol. 114, p. 104825, 2020.
- [17] C. Chekuri and S. Khanna, "On multidimensional packing problems," *SIAM journal on computing*, vol. 33, no. 4, pp. 837–851, 2004.
- [18] E. C. man Jr, M. Garey, and D. Johnson, "Approximation algorithms for bin packing: A survey," *Approximation algorithms for NP-hard problems*, pp. 46–93, 1996.
- [19] C. Reeves, "Hybrid genetic algorithms for bin-packing and related problems," *Annals of Operations Research*, vol. 63, no. 3, pp. 371–396, 1996.
- [20] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, no. 1, pp. 5–30, 1996.
- [21] M. Quiroz-Castellanos, L. Cruz-Reyes, J. Torres-Jimenez, C. Gómez, H. J. F. Huacuja, and A. C. Alvim, "A grouping genetic algorithm with controlled gene transmission for the bin packing problem," *Computers & Operations Research*, vol. 55, pp. 52–64, 2015.
- [22] K. Kang, I. Moon, and H. Wang, "A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem," *Applied Mathematics and Computation*, vol. 219, no. 3, pp. 1287–1299, 2012.
- [23] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete applied mathematics*, vol. 28, no. 1, pp. 59–70, 1990.
- [24] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, "On stopping criteria for genetic algorithms," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 405–413.
- [25] A. W. S. (AWS), "Solutions," 2017. [Online]. Available: <https://aws.amazon.com/solutions/case-studies/netflix-kinesis-data-streams/>
- [26] J. Repschlaeger, K. Ere, and R. Zarnekow, "Cloud computing adoption: an empirical study of customer preferences among start-up companies," *Electronic markets*, vol. 23, no. 2, pp. 115–148, 2013.
- [27] P. Foundation, "Pytorch Checkpoints," https://pytorch.org/tutorials/recipes/recipes/saving_and_loading_a_general_checkpoint.html, 2023, Accessed: 12.05.2023.
- [28] TensorFlow, "Tensorflow Checkpoints," <https://www.tensorflow.org/guide/checkpoint>, 2023, Accessed: 12.05.2023.
- [29] M. Agarwal and G. M. S. Srivastava, "Cloud computing: A paradigm shift in the way of computing," *International Journal of Modern Education & Computer Science*, vol. 9, no. 12, 2017.
- [30] S. C. Boerman, S. Kruijemeier, and F. J. Z. Borgesius, "Online behavioral advertising: A literature review and research agenda," *Journal of Advertising*, vol. 46, no. 3, pp. 363–376, 2017. [Online]. Available: <https://doi.org/10.1080/00913367.2017.1339368>
- [31] S. Roy, S. S., and S. Bhatia, "Service quality versus service experience: An empirical examination of the consequential effects in b2b services," *Industrial Marketing Management*, vol. 82, pp. 52–69, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0019850117301463>
- [32] M. Eurich, A. Giessmann, T. Mettler, and K. Stanoevska-Slabeva, "Revenue streams of cloud-based platforms: Current state and future directions." in *AMCIS*, 2011.
- [33] P. Timmers, "Business models for electronic markets," *Electronic markets*, vol. 8, no. 2, pp. 3–8, 1998.
- [34] B. Wall, H. Jagdev, and J. Browne, "A review of ebusiness and digital business—applications, models and trends," *Production Planning and Control*, vol. 18, no. 3, pp. 239–260, 2007.
- [35] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1990, pp. 249–256.
- [36] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 1994, pp. 152–158.
- [37] "Docker," <https://www.nngroup.com/articles/ten-usability-heuristics/>, 2024, Accessed: 01.02.2024.
- [38] V. Haag, "A cloud portfolio manager- a novel business model," Master's thesis, University of Vienna, 2024.
- [39] M. Kiessler, "A cloud portfolio manager - optimization strategies for efficient resource allocations," Master's thesis, University of Vienna, 2022.
- [40] W. Mach and E. Schikuta, "A generic negotiation and re-negotiation framework for consumer-provider contracting of web services," in *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*, 2012, pp. 348–351.