de Moraes *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2022) 11:5
https://doi.org/10.1186/s13677-021-00275-1

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Application of deterministic, stochastic, and hybrid methods for cloud provider selection

Lucas Borges de Moraes[2,3], Rafael Stubs Parpinelli[1,2,3] and Adriano Fiorese[1,2,3*]

Abstract

Cloud Computing popularization inspired the emergence of many new cloud service providers. The significant number of cloud providers available drives users to complex or even impractical choice of the most suitable one to satisfy his needs without automation. The Cloud Provider Selection (CPS) problem addresses that choice. Hence, this work presents a general approach for solving the CPS problem using as selection criteria performance indicators compliant with the Cloud Service Measurement Initiative Consortium - Service Measurement Index framework (CSMIC-SMI). To accomplish that, deterministic (CPS-Matching and CPS-DEA), stochastic (Evolutionary Algorithms: CPS-GA, CPS-BDE, and CPS-DDE), and hybrid (Matching-GA, Matching-BDE, and Matching-DDE) selection optimization methods are developed and employed. The evaluation uses a synthetic database created from several real cloud provider indicator values in experiments comprising scenarios with different user needs and several cloud providers indicating that the proposed approach is appropriate for solving the cloud provider selection problem, showing promising results for a large-scale application. Particularly, comparing which approach chooses the most appropriate cloud provider the better, the hybrid one presents better results, achieving the best average hit percentage, dealing with simple and multi-cloud user requests.

Keywords: Cloud provider selection problem, Cloud provider ranking, Evolutionary computing, Natural computing, Performance indicator, Hybrid methods, Stochastic algorithms, Deterministic algorithms

Introduction

Cloud Computing (CC) is a service model that allows a significant and on-demand hosting and distribution of optimized computing resources using computer networks [1] being a convenient and easily accessible service via Internet [2]. Popularization of CC usage inspired the emergence of a large number of new companies providing CC services [3] known as Cloud Providers (CPs) [1]. In spite of that, the increasing number of CPs does not guarantee service quality. In fact, careful and discerning cloud users can experience complexity in choosing a cloud computing company among all those available. Thus, selecting which CPs are the most suitable to satisfy each user's needs has become a complex issue called Cloud Provider Selection (CPS) problem.

From the last decade, the CPS became a significant research challenge and several works have already been developed trying to solve it by using deterministic approaches, for example, Multi-criteria Decision Methods (MCDM) [4–12] Other works cope with metaheuristics to address CPS problem [13, 14]. Deterministic, metaheuristic and hybrid methods, which combines deterministic ones, are well-used making them important to solve CPS problem according to state-of-the-art literature. Although these approaches deal with CPS, the complex user requests that require more than a single provider to be satisfied, is poorly scratched. At the best of our knowledge, none work used the deterministic and

*Correspondence: adriano.fiorese@udesc.br
[1]Department of Computer Science (DCC), Joinville, SC, Brazil
[2]Graduate Program in Applied Computing (PPGCA), Joinville, SC, Brazil
Full list of author information is available at the end of the article

metaheuristics approach involving different methods in a pipeline for fulfillment of complex multi-cloud user requests. Moreover, these works propose disconnected methods, particular approaches, and different problem definitions, lacking an integrative problem modelling and design. We advocate this happens because, beyond standardization absence, these works do not share a common ground such as an initial problem scenario composed of database and user requests format.

This work intends to find answers to the following research questions: Which can be the definitions (e.g., models, process, relationships, formats, etc.) and elements (e.g., software) that can be seem and used by other works as providing a common ground to solve the CPS problem? Which is the best approach/strategy/method to solve the CPS problem, according to those elements and definitions?

To address these issues, this work proposes a new and generic approach for solving the CPS problem[1], using CSMIC-SMI Performance Indicators (PIs) as metric to quantify CPs quality. In this sense, the proposed approach provides a pool of selection methods in three categories: deterministic or exact, stochastic or metaheuritics, and hybrid. Hence, a comparative study is performed.

This work presents several contributions. The main one is the CPS problem-solving model as a general software architecture that uses CP performance indicators as selection criteria. Also, the unordered subsidiary contributions cope with: 1) information model representing an individual, i.e., a CPS possible solution; and the suitable fitness function used by the developed metaheuristic algorithms; 2) Regarding the deterministic methods: 2.1) Mathematical model comprising a set of equations coping with transforming the CPS criteria, i.e., CP database's performance indicators and user requests, into inputs and outputs to feed the Data Envelopment Analysis (DEA) method, which is used in the developed CPS-DEA method; 2.2) A particular tolerance mechanism supporting small differences between user-requested PIs and cloud provider PIs; 3) A general and agnostic qualitative data[2] ontology (i.e., HT, LT, HLT PI characteristics), as seen in "Performance indicators" section, that enables the methods employed to deal with qualitative beyond quantitative[3] data; 4) An architectural design contributing to a potential standardization of CPS problem-solving.

The remainder of this paper is organized as follows: "General background" section presents the general background concerning the development of this work; "Related work" section presents important related work; "Cloud provider selection approach" section presents a detailed description of the proposed PI-based cloud provider selection approach; "Experiments and results" section presents the experiments performed, obtained results, and analysis. Finally, "Conclusions and future work" section elaborates on final considerations and future work.

General background

This section presents main concepts necessary for developing and understanding the proposed PI-based approach and selection methods for solving the CPS problem.

Performance indicators

An indicator is a tool that allows a synthesized collection (i.e., containing only the essential data) of information related to a particular aspect of reality [18]. It is possible to classify indicators into two categories [19]: **Quantitative:** They are those states, levels, or categories that can be expressed numerically and can be worked algebraically. The numeric values can be discrete or continuous. **Qualitative:** They are also called categorical. They can represent distinct states, levels, or categories defined by an exhaustive and mutually exclusive set of subclasses, which may or may not be ordered. The ordered subclasses have perceptible logical graduation among them, giving an idea of progression, whereas the unordered ones do not.

Quantitative indicators can also be classified according to the behavior of their utility function, that is, how good (useful, beneficial) the indicator becomes when its numerical value changes (increases or decreases). According to [19], there are three possible classifications: **a) Higher is Better (HB):** Users and/or system managers always prefer the highest possible values for this indicator. Examples are system throughput, resources (money, memory, and materials), and service's availability. **b) Lower is Better (LB):** Users and/or system managers always prefer the lowest possible values for this indicator. Examples are response time (delay) and costs. **c) Nominal is Best (NB):** Users and/or system managers prefer specific values (higher and lower values are undesired). A particular value is considered the best. The criteria system usage and security are examples of this utility function class.

The utility function of qualitative indicators is harder to map using numerical values. A trivial mapping is the binary one, i.e., the category or subclass "value" of the qualitative indicator has maximum utility whereas the others have minimal utility according to users choice. Quantitative indicators present a similar (but not equal) behavior with the NB utility function. Binary is a useful mapping process for unordered qualitative indicators.

[1] Unlike our previous works concerning the CPS problem [15–17], this paper brings up other selection methods, an entirely new set of experiments, new analysis, and exciting findings. Hence, the proposal of building a CPS approach offering several other cloud provider selection methods beyond previously developed ones makes this paper a significant new contribution to the community.

[2] criteria/attributes/performance indicators.

[3] To the best of our knowledge no CPS framework deals with qualitative data using similar approach.

However, numbers representing levels can be associated with each different category in ascending order for the ordered ones. Thus, smaller level numbers are assigned to categories that express lower values (e.g., "low", "little", "never") and larger numbers to categories that express higher values (e.g., "high", "many", "always"). Thus, this mapping can be useful when a user desires a specific category ("value") though higher and/or lower-level categories than the desired one can also help/satisfy the user. Therefore, to solve this problem, comprising this work, an ontology, similar to the one in [19], was created in order to model when an ordered qualitative indicator has tolerances for categories below and/or above the desired category level [15]. They are: **Higher is Tolerable (HT):** Categories above the desired one are tolerable; **Lower is Tolerable (LT):** Categories below the desired one are tolerable; **Higher and Lower are Tolerable (HLT):** Categories above and below the desired one are both tolerable.

In order to standardize the specification of PIs for the evaluation of CPs, the Cloud Service Measurement Index Consortium (CSMIC) was formed. The Service Measurement Index (SMI), proposed by CSMIC, represents a widely accepted set of PIs that can measure a CP's service performance and compare different services, regardless of whether the service is provided internally or by an outsider company [20].

SMI is a hierarchical structure whose upper level divides the measurement space into seven major categories and each major category is optimized by four or more attributes (subcategories) [20]. The seven major categories are: accountability, agility, assurance, financial, performance, security, privacy and usability. Each category contains a set of attributes, where each attribute has its own definition and collection form (how to obtain the value) [7, 21]. Some of the attributes also have sub-attributes that specializes them [22]. The major categories, also called Key Performance Indicators (KPI) [22], can be defined as follows:

- Accountability: This category measures if the CP is responsible and complies with good practices regarding actions and business practices. Functions critical to accountability include auditability, compliance, data ownership, provider ethicality, sustainability, etc.
- Agility: Agility depicts a change metric comprising how quickly new capabilities are integrated into IT as needed. When regarding a Cloud service's agility, attributes like service elasticity, portability, adaptability, and flexibility should be provided.
- Financial: Involves the cloud provider and client expenditure relationship aspects where cost is the most important attribute/indicator representing how much client pays to have the cloud service, for instance. It is a quantitative metric.
- Performance: This category covers the features and functions of the provided services. It can be assessed with respect to suitability, interoperability, accuracy and so forth. It has sub-attributes that are quantitative and qualitative.
- Assurance: This category includes key attributes that indicate how likely it is that the service will be available as specified. For selecting cloud providers some of these attributes are reliability, resiliency and service stability.
- Security and Privacy: This category includes attributes that indicate the effectiveness of a cloud service provider's controls on access to services, service data, and the physical facilities from which services are provided. It includes many qualitative attributes such as protecting confidentiality and privacy, data integrity and availability.
- Usability: It asses the ease with which a service can be used. Multiple factors are involved with usability such as Accessibility, Installability, Learnability, and Operatibility.

These KPI/categories, attributes and sub-attributes can be quantitative or qualitative and provide a wide range of cloud service providers' evaluation and comparison criteria. Moreover, they can be used according their utility function ontology. Therefore, they represent suitable performance indicators to be used per se or on decision-making support methods to rank and select cloud providers.

Data envelopment analysis

According to [23],

> Data Envelopment Analysis is a linear programming method for assessing the efficiency and productivity of units called Decision-Making Units (DMUs). Over the last two decades, DEA has gained considerable attention as a managerial tool for measuring organizations' performance. [23]

Data Envelopment Analysis is a well-known method for decision-making support introduced by Charnes, Cooper, and Rhodes in 1978 [24]. DEA can measure a set of similar DMUs [25] which are entities (e.g., manufacturing unit, bank, hospital, and cloud provider) that consumes a variety of identical inputs and produces a variety of identical outputs.

The performance of each DMU is assessed by DEA using the concept of efficiency or productivity, which is the ratio of total output regarding total input [25]. Noteworthy, outputs are criteria to be maximized, and inputs minimized for better efficiency. Thus, DEA aims to measure how efficiently a DMU uses the available resources to generate a set of outputs. Efficiency is a relative concept

that compares what has been produced with what could have been produced. Ultimately, it can be understood as a comparison between observed productivity among competing DMUs [4]. Thus, estimated efficiencies using DEA are relative to or compared to the best performing DMU, i.e., with the highest calculated efficiency value among the assessed DMUs [25]. Efficiency varies between 0 and 100% (or from 0 to 1).

Equation 1 shows DEA's optimization problem depicting the DMU's efficiency maximization as the sum of all its s weighted outputs, where u_k represents the weight k and O_{ki}, the output k of the DMU i. This is subject to the sum of all r weighted inputs equals 1, where v_l represents the weight l and I_{li} the input l of the DMU i. Moreover, the difference between weighted outputs and weighted inputs is another constraint and must be less than or equal to 0. It is important to note that the decision variables are the input and output weights $v = (v_1, v_2, ...v_r)$ and $u = (u_1, u_2, ..., u_s)$, respectively, and their values must be greater than or equal to 0 (non negative weights).

$$max\ Eff_i = \sum_{k=1}^{s} u_k \times O_{ki} \tag{1}$$

subject to:

$$\sum_{l=1}^{r} v_l \times I_{li} = 1$$

$$\sum_{k=1}^{s} u_k \times O_{kj} - \sum_{l=1}^{r} v_l \times I_{lj} \leq 0; \quad \forall j$$

$$u_k, v_l \geq 0; \quad k = 1, 2, ..., s;\ l = 1, 2, ..., r$$

It is noteworthy that the DMU that produces more outputs with fewer inputs than the others will present better efficiency values. The weights (u_k) and (v_l), assigned to outputs and inputs, respectively, are not allocated by users but automatically calculated by the method in a linear optimization procedure in order to allow the most beneficial results possible for each DMU [25], respecting the restrictions depicted in Eq. 1. The set of DMUs with an efficiency of 100% forms DEA's efficiency frontier. Thus, it is essential to note that the number of DMUs must be reasonable to use DEA. Otherwise, maybe the method cannot satisfactorily calculate the efficiency, and thus all DMUs fall in the efficiency frontier. To avoid that, one expects that the number of DMUs is greater than the product between the amount of input and output [25]. It is worth noting that the sample size should be at least two times larger than the sum of the inputs and outputs for an appropriate efficiency calculation [25].

Genetic algorithms

John Holland introduced Genetic Algorithms (GA) in the 70s [26] and popularized by David Goldberg in the 1980s [27]. They are based on the natural selection process proposed by Charles Darwin, where adaptability (fitness), heredity, and genetic mutation can influence the changes and selection of best-fit individuals. GA is probabilistic where several internal routines demand random number generation with threshold verification (e.g. crossover and mutation routines). Also, GA is population-based since it works on a set of candidate solutions, named individuals, of a population in each generation (iteration algorithm). Each individual represents a point in the search space, and it is composed of a genetic coding (genotype) implemented as a vector with different genes. Each gene is responsible for encoding one of the optimization problem variables. The GA looks for the best solution to a problem based on the fitness of its individuals. The higher the fitness, the greater the chances of that individual passing on its genetic load (portions of its chromosomes/solutions). The selection routine (e.g., roulette wheel and stochastic tournament) is responsible for guiding the evolution process over the generations, allowing the algorithm to converge to the best possible solutions. GA generally uses two genetic operators: crossover (for binary coding, e.g., one-point-crossover) and bit-flip mutation. One-cut-point crossover creates two new coding vectors (offsprings) by copying and exchanging two portions of the two individuals' coding vectors (parents) based on a single cut-off point. Bit flip is a mutation technique that changes one bit (an encoded problem variable, for instance, from 0 to 1) with a specified ratio in the individual encoding vector. An elitism routine can also help algorithm convergence to the optimal solution point and make an always upward fitness curve (best individual fitness in the current population). The elitism copies the best individual of the current population to the next algorithm generation [28].

Binary differential evolution

Binary Differential Evolution (BDE) is a probabilistic and population-based metaheuristic inspired by the canonical Differential Evolution (DE) algorithm [29]. BDE is adapted to handle binary search space problems using a simple modification of the DE/rand/1/bin variant for binary coding, which combines each individual of the current population with another randomly chosen one using the crossover operator [28]. Besides the binary representation, the main modification is the insertion of a bit-flip mutation operator inspired by the GA that improves its global search ability, which enables diversity [28]. Thus, the BDE consists of applying these simple operators (crossover and bit-flip mutation) in candidate solutions represented as binary strings. The adaptation starts with the population initialization. In this case, instead of continuous random values, the individuals are initiated with random binary ones. A random bit inversion replaces the original mutation, and the perturbation rate is a new

parameter added to establish how many individuals of the population will pass on to the next generation using the mutation and crossover processes. At least one individual will be mutated and the DE crossover is kept as the original but with binary values. This process is activated during the perturbation procedure and after mutation. In this case, the used selection routine is a greedy selection.

Discrete differential evolution

Discrete Differential Evolution (DDE) is another version of DE adapted to handle binary search space problems [30]. DDE encodes continuous variables, i.e., floating point numbers converted to binary before the fitness evaluation, while BDE encodes binary-only variables (0,1). In other words, DDE uses the genetic operators on floating-point (real) values codification. Thus, the i^{th} individual of the candidate solutions population is an n-dimensional vector, and each dimension j is composed of a random floating-point number between -1 and 1 [30]. The n parameter is equivalent to the number of DMUs in DEA modeling. The basic DE operator is the $DE/rand/1/bin$ mutation strategy over the generations. The selection routine used is the greedy selection [30]. Equation 2 presents this strategy,

$$x_{ij} = x_{aj} + F \times (x_{bj} - x_{cj}). \tag{2}$$

The DDE mutation operation creates a new individual x using three different random individuals (a, b, and c) and a scale parameter F that ranges from 0 to 1 [30]. To accomplish that, this new individual x is perturbed in every position j by receiving the random floating-point value of a, plus the weighted difference variation between b and c, at that same position j. Parameter F is the weighting factor used to control the amplification of the differential variation.

When DDE evaluates the fitness of an individual, it must to discretize the individual's genotype vector. That is, DDE transforms floating-point codification x_j to binary codification y_j. Equation 3 performs this discretization:

$$y_j = f(x_j) = \begin{cases} 1 \text{ if } x_j > 0 \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

Related work

This section presents various research works carried out by academicians and researchers using deterministic, stochastic (metaheuristic) and hybrid approaches for ranking and selection of CPs.

Deterministics

This subsection presents works addressing CPS problem with deterministic methods, involving multi-criteria decision methods, fuzzy logic and others.

Sundareswaran et al. [31] proposed an indexing technique to assist CP selection using CPs properties and users requirements. Service providers are analysed and indexed by an index key generated by concatenation of the encoding type of service and all the other encoded properties offered by the CP, using a "xor" operator. Each property (except service type) has a unique encoding. A "B+-tree" keeps CPs indexed. Thus, at receiving a cloud provider selection request, it is started a search in the index to identify an orderly list of candidate CPs according to how well their properties fit the users' needs. The technique is tested with six real CPs (Google Clouds, Joynet, Salesforce, Windows Azure, Amazon EC2, Rackspace) and nine properties (service type, operating system, Quality of Service (QoS) level, security level, measurement unit, instance sizes, pricing, and location-based prices, pricing unit).

Garg et al. [7, 32] proposed a framework called SMICloud for ranking CC services using SMI indicators. SMICloud performs CPs' QoS assessments to rank them using the Analytic Hierarchy Process (AHP) method [33]. The SMI indicators used can be of two types (essential, not essential) whose values can be boolean, numeric, unordered set, or range type. A small ranking study case evaluating SMICloud has been performed using a set of three real Infrastructure as a Service (IaaS) CP (Windows Azure, Rackspace and Amazon EC2) and six SMI PIs (agility, cost, assurance, security level - randomly generated, accountability level and performance). Although SMICloud seems appropriate, the assembly of many CPs and PIs hierarchy is tiresome and complicated.

Sun et al. [34] presented a cloud services selection framework called Cloud Service Selection with Criteria Interactions (CSSCI). The framework performs selection relying on criteria interactions modelled as user oriented priority orders and types of interaction among users. CSSCI applies fuzzy measures and the Choquet integral to assess the selection criteria interrelationship. Regardless of being an exciting work, SMI criteria are not used at the assessment of the selection criteria interrelationship. This makes it difficult to compare results with other works, including ours.

Chen et al. [35] presented another example of selection mechanism where the Logistic Selection Partner (LSP) problem faced by the omnichannel e-commerce marketplace realm is addressed. Notably, a solution for selecting the most suitable logistic partner for delivering goods is proposed. To accomplish that, fuzzy axiomatic design and extended regret theory are used to process functional requirements similar to human beings psychological behavior. The fuzzy axiomatic design computes the probability of success for each LSP to fulfill each criterion. The extended regret theory determines the criteria weights analysing the decision maker behavior. A

final phase computes the LSP final score combining previous results. Experiments with six LSP are conducted and results are compared with Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method.

Somu et al. [22] presented a Hypergraph based Computational Model (HGCM) for ranking Cloud Service Providers (CSPs). This model core uses the Helly's hypergraphs property to evaluate CSPs by means of the Minimum Distance-Helly Property (MDHP) technique. The MDHP takes into account the relationship between some SMI atributes, sub-atributes and KPIs and the cloud user requirements to select the most suitable CSP. Although the remarkable work done creating hyperedges and the recursive use of Helly property, this work does not complies with an attributes ontology that considers the more resources a cloud provider has the better is to fulfill the user requirement, which is taken into account in the proposed work.

Baranwal and Vidyarthi [8] proposed a CP ranking voting model that uses DEA to analyse user QoS expectation metrics as input data. Also, SMI metrics are used. Two kinds of metrics are used: application dependent and user-dependent. The metric values can be of different types such as range type, numeric unordered set, boolean, and data center value. In this voting system, each metric acts as a voter, and CPs are candidates. [36] presents a CP measure index framework that is a similar proposal to [8]'s voting model. In this case, the resulting cloud providers index provides the information about CPs that can be used to select the best one.

Abdel-Basset et al. [12] proposed the use of neutrosophic set, triangular neutrosophic numbers and their operations to deal with uncertainties, such as indeterminacy and falsity when performing selection criteria pairwise comparisons, which are not handled by fuzzy set theory and traditional MCDM. Thus, authors present the Neutrosophic Analytic Hierarchy Process (NAHP) method that uses a neutrosophic triangular scale instead of Saaty's scale in the AHP method to rank cloud providers using accessibility and usability, performance, security and, scalability as selection metrics. The proposed work deals with uncertainties at the criteria value collection level dismissing the need to pairwise criteria assessment.

Meesariganda and Ishizak [11] proposed the use of AHP with particular scale to rank cloud providers. In this case, a scale is chosen among nine different ones (e.g., Linear, Power, Geometric, Logarithmic, Root square, Asymptotical, Balanced, Inverse linear and Balanced power). The shortest euclidian distance between scales' values and the decision maker value defines the choice. Paper presents an experiment using six metrics (core competency, market opportunity, customer satisfaction, time to market, risk and financial benefits).

Hogben and Pannetrat [37] presented the challenges of defining and measuring the metrics availability for supporting real-world services. Their analysis is based on Service Level Agreements (SLA) that are not standardized generating ambiguity on the definition of availability as measured by service providers. This work highlights the importance of standardization to support comparisons among services.

Wagle et al. [38] proposed a services evaluation model coping with the commit of SLAs between CPs and users. The execution of this model arranges CPs in a heat map according to their performances comprising QoS. Therefore, this map portrays a reference and guide system for cloud computing users and brokers. This work uses as metrics, ones based on SMI: performance (throughput, response time, and latency), cost (storage and snapshot cost), reliability (load balancing, recoverability and mean time between failures - MTBF), security (authentication, encryption, and auditing) and availability (considering interruption frequency, downtime, and up-time). This selection approach does not rely on an automated method to decide which provider should be selected since it is visual. Although it is an excellent way to show data, it becomes complex to use if the number of CPs and QoS metrics scale.

Metaheuristics

This subsection presents works whose main approach uses metaheuristic methods, even though deterministic methods can be used to compare results.

Patra et al. [39] proposed a constrained multi-criteria federated cloud provider selection mathematical model. Authors present three metaheuristic algorithms, namely: 1) Bird Swarm Algorithm (BSA); 2) Teaching-Learning Based Optimization (TLBO), and 3) Jaya, a population-based gradient-free optimization method. The selection of the best Internet of Things (IoT) Cloud Software Platform is the application scenario for the algorithms. The algorithms use some optimized (maximized and minimized) criteria such as cost, suitability, assurance, reliability, availability, and usability. The paper presents algorithms flowchart. Algorithms evaluation results are similar but with BSA prevalence.

Mukherjee et al. [13] proposed a mathematical model for constrained multi-criteria federated cloud provider selection. That mathematical model is implemented as the Harris Hawks Optimizer (HHO) metaheuristic algorithm. Authors model several criteria, such as cost, performance, assurance, usability and accountability that are used as input for the HHO. A solution based on the AHP multi-criteria method is used to present feasible solutions that can be compared with the results offered by the HHO, TLBO and Jaya. A performance evaluation has been performed showing the HHO, TLBO and

Jaya results, respectively, are better than the AHP ones. Although the proposal presents a detailed modelling of the used criteria, particular features that represents user's needs to choose a cloud provider are not specified neither taken into account on those criteria. Our approach, beyond specifying the features (PIs), copes with ones that are in the SMI.

M. [14] proposed a trust estimation framework for infrastructure-based cloud service selection using the Non-dominated Sorting Genetic Algorithm (NSGA) II. NSGA-II provides a set of Pareto-optimal solutions assessing agility, finance, performance, security, and usability criteria. A proposal evaluation has been performed on generated service plans for 14 CP considering three models where one of the criteria is more important than the others (Performance-based, Finance-based and Security-based).

Mohamed and Abdelsalam [40] proposed a CPS solution where more than one single cloud provider can host the user services. In order to select multi CP, authors use Simulated Annealing (SA), Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) metaheuristic methods. Experiments are performed with a hypothetical dataset and they compare methods' performance against the best solution given by the AHP method. It is important to note that our proposed work goes further since it provides, all integrated, selecting single or multi CPs depending on the complexity of user needs. Moreover, it presents a broader set of deterministic (that they do not propose) and metaheuristic methods combining them into hybrid methods.

Hybrids

This subsection shows works whose main approach involves a combination of two or more methods regardless of deterministic or metaheuristic ones. As aforementioned, to the best of our knowledge, beyond ours none work has proposed a hybrid approach combining first a deterministic and then a metaheuristic methods.

Achar and Thilagam [9] presented a hybrid approach for ranking IaaS CPs using QoS measurements as selection criteria and using AHP and TOPSIS methods. The IaaS service selection comprises three steps, namely, 1) Identifying which criteria are appropriate to the request, i.e., identifying the essential PIs belonging to the SMI; 2) Assessing these criteria weights using the AHP method, and 3) Ranking each CP using TOPSIS [41]. The work evaluation is performed with six hypothetical providers and four PIs (security, availability, cost, and accountability). Although ranking using TOPSIS appears to be promising, more examples and analysis can lead to more reliable conclusions, especially with real data and user requests. Moreover, the approach lacks the use of qualitative PIs.

Jatoth et al. [42] proposed a SELCLOUD framework-hybrid multi-criteria decision-making approach i.e., a novel extended Grey TOPSIS using AHP method for cloud service selection. The Grey theory cope with the uncertainty issues; AHP to determine the weights of the criteria; and TOPSIS to obtain the CPs ranking and address the rank reversal problem. The performance of SELCLOUD has been validated using a set of seven IaaS CSPs (Amazon, Azure, CenturyLink, City Cloud, Google, HP, and Rackspace) on three categories i.e., large (two virtual cores), Extra-large (four virtual cores) & 2x-extra-large (eight virtual cores) and five PIs (price, processing performance, I/O operational consistency, disc storage performance, and memory performance) in terms of sensitivity analysis, adequacy under a change in alternatives, adequacy to support group decision-making, and handling uncertainty.

(Jaiswal and Mishra, [10]) proposed an approach that uses TOPSIS and Fuzzy TOPSIS [43] alongside AHP and Analytic Network Process (ANP) [44] to handle the CPS problem, based on quantified QoS attributes. AHP and ANP evaluates criteria weights and TOPSIS and Fuzzy TOPSIS rank CPs. The performance of the approach has been evaluated using four hypothetical CPs with data gathered from *cloudharmony.com* and eight arbitrary quantitative criteria (virtual core, memory, price/hour, CPU performance, disk input/output operations per second, disk consistency, disk performance, memory performance). Albeit a well-crafted approach, the showed instance evaluation results are subjective, and weights assigned by users make all the difference, making the proposed method's efficiency questionable for large quantities of CPs. Moreover, the proposed approach also does not seem to be able to handle subjective criteria.

Al-Faifi et al. [45] proposed a hybrid approach for selecting CPs in the context of smart data environments. To evaluate and rank cloud providers, authors consider the interdependencies and relationships between the performance measurements used as selection metrics. The approache's first step is to form groups of CPs with similar features using the k-means algorithm. After that, in order to provide a representative CP, each cluster applies the DEcision-MAking Trial and Evaluation Laboratory (DEMATEL) altogether with the ANP multi-criteria decision-making method. Further, the ANP method is applied to the result representatives set to rank and make a final decision. Although well designed, a user request is not considered in this work. On the other hand, our proposed approach, beyond considering user will, deterministic and metaheuristic approaches and their hybrid combinations, covers more CPS problem facets.

Table 1 presents a summarized overview of the main characteristics identified for comparison purposes between the related works discussed and also with the

Table 1 Overview and comparison between cloud providers rank and selection proposals

Work	Class	Method	Criteria	Filter	Ql	Multi-Cloud
[7, 32]	D	AHP	SMI	No	Yes	No
[31]	D	Own indexing method	$\subseteq$ SMI	Yes	Yes	No
[8]	D	Voting method using DEA	SMI	No	Yes	No
[38]	D	Own method	$\subseteq$ SMI	No	Yes	No
[36]	D	Voting method	SMI	No	Yes	No
[34]	D	Own Method	Open	No	Yes	No
[35]	D	Own Method	Open	No	Yes	No
[22]	D	Own Method	SMI	No	No	No
[37]	D	Own Method	SLA	No	No	No
[12]	D	AHP with particular scale	$\subseteq$ SMI	No	Yes	No
[11]	D	AHP with particular scale	Open	No	Yes	No
[39]	M	BSA, TLBO, Jaya	SMI	No	Yes	No
[13]	M	HHO	SMI	No	Yes	No
[40]	M	SA, GA, PSO	$\subseteq$ SMI	No	Yes	Yes
[14]	M	NSGA II	$\subset$ SMI	No	No	Yes
[42]	H	AHP + TOPSIS	$\subseteq$ SMI	No	No	No
[45]	H	Clustering + DEMATEL + ANP	$\subseteq$ SMI	No	Yes	No
[9]	H	AHP + TOPSIS	SMI	No	No	No
[10]	H	TOPSIS, Fuzzy TOPSIS, AHP, ANP	$\subseteq$ SMI	No	No	No
Proposed Architecture	C	Own method, Adapted DEA, Adapted metaheuristics (GA, DDE, BDE)	SMI	Yes	Yes	Yes

proposed approach. The identified characteristics are as follows:

- Method: The main math method/methodology/procedures used to select/rank CPs/services. In case of hybridism methods appear sumed. When more than one method is used a comma separates them.
- Class: Identifies work as deterministic (D), stochastic/metaheuristic (M), hybrid (H), and combination (C) representing works that combine at least two of these classes. The hybrid class represents a combination of methods of the same class.
- Criteria: The main CPs data used by the approaches/models/frameworks/methods to select/rank providers/services. When work does not mention SMI and criteria are generally provided by the user then result is "Open". The result "Open" means SMI is not mentioned and criteria are provided by the decision maker (user). "$\subseteq$ SMI" means that though work does not mention SMI some criteria are from SMI and some can alternatively be different. On the other hand, "SMI" depicts works that mention SMI and use only SMI criteria though not necessarily all of them.
- Filter: Reports if the work, as its first setp, discards CPs that does not minimally satisfies the user requirement. It will be "Yes" in affirmative case and "No", otherwise.
- Ql: Indicates if the work uses qualitative criteria for ranking CPs. It will be "Yes" in affirmative case and "No", otherwise.
- Multi-Cloud: Indicates if the work considers more than one single cloud provider as the CPS solution when only one does not fulfill all the criteria (the full service) at the same time and, different criteria (part of the service) can be satisfied separately from the others. It will be "Yes" in affirmative case and "No", otherwise.

As it is possible to note, several CPS solution proposals have already been developed. A shared characteristic among them is the use of different sets of information to perform their evaluations. This is a remarkable issue. Among the surveyed works, it is possible to find ones that use QoS metrics and information values found in SLAs. Therefore, although several works, including ours, do use indicators based on the SMI standard, there is still a lack of standardization regarding what should be the necessary information to use to evaluate cloud providers, including how to measure them.

Beyond those methods that involve a kind of direct comparison between metrics/indicators/measurements from different cloud providers, particular ones (some own methods) involve the use of fuzzy logic. However, most of them consider the use of MCDM per se or a combination of them. Few works propose the use of metaheuristic methods to solve the CPS even though they compare results with an initial solution given by a deterministic one.

The few hybrid methods found perform distinct steps using each one particular deterministic methods. Some of them use a step to calculate weights for the criteria using a deterministic method while the ranking step is performed by others.

Only one surveyed method, beyond this proposal, performs a filtering step, which discards CPs when they do not satisfy any of the user required criteria. Moreover, it seems that CPS solutions considering the possibility of user services or part of them running in more than one cloud provider are still in its infancy since beyond the proposed work only two other works deal with the multi-cloud environment.

All considered, none of the surveyed works are handling the CPS problem approaching deterministic, stochastic (metaheuristic) and their combination (i.e., hybrid deterministic and metaheuristic or vice-versa solutions, either exploring a modular architectural construction and using a well-established set of SMI key performance indicators as criteria, altogether. This work proposes to cover this lack, not only to present the mentioned approaches but also to demonstrate their behavior in comparison.

Cloud provider selection approach

This section presents the proposed Cloud Provider Selection PI-based approach to solve the CPS problem. The approach is composed of three indispensable components: 1) CP database; 2) user request and 3) selection methods, and an auxiliary one: user interface.

Figure 1 presents a general overview of the proposed CPS approach. This modelling aims to define a more generic and general scenario (database plus request) for any CP selection approaches using PIs, which provides a basic common ground where the proposed selection methods ("Selection methods modelling" section) are applied.

The CP database component stores CP PIs. In order to select the most suitable CP, user should choose a total of m PIs of interest according to its goals. The user interface

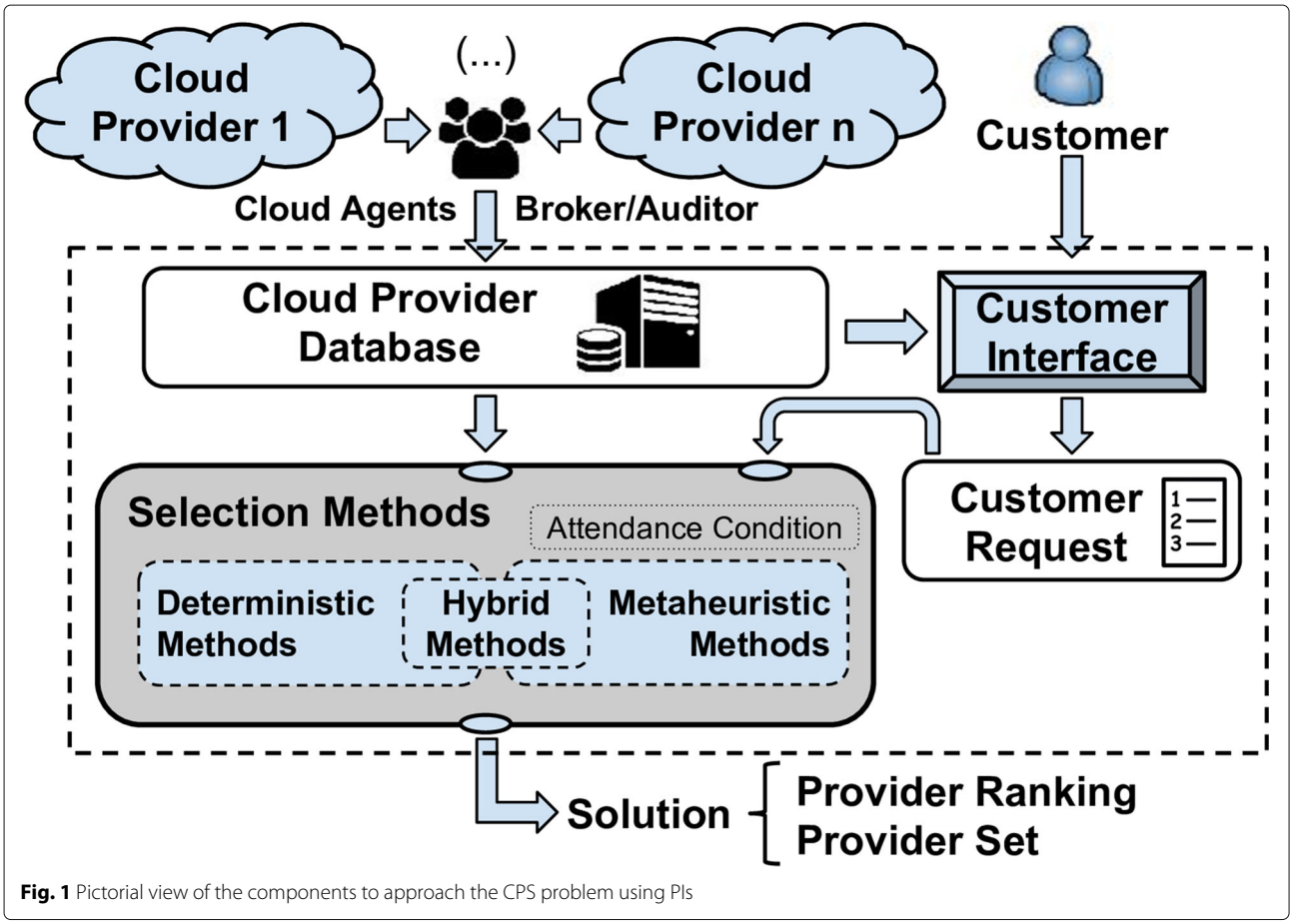


Fig. 1 Pictorial view of the components to approach the CPS problem using PIs

makes available this choice allowing communication with the database to retrieve the available PIs and collecting the selected one's desired value, importance weight, and other arguments.

In addition, CP selection methods are required in order to select appropriate CPs. The selection methods component provides a pool of them. A method execution can produce two kinds of responses: a CP ranking list, if it is a deterministic method; or a CP set (CPs with the highest fitness found in all method executions), in case it is a metaheuristic or hybrid method. Both kind of responses can be appropriate and returned to the user as a final solution for his request. In any case, the response with a CP set is expected when a multi-cloud solution is required to fulfill all the user PIs of interest.

Figure 2 presents the proposed approach flowchart from a user utilization point of view. This point of view assumes the CP database is already consolidated and accessible, and the user interface and selection methods are available.

Cloud provider database

The CP database stores all available candidate CPs with the name of all M registered PIs (plus price) with their own utility function type and values. Table 2 presents an example of a possible CP database. This database is agnostic and generic for all kind of PIs and CPs business models.

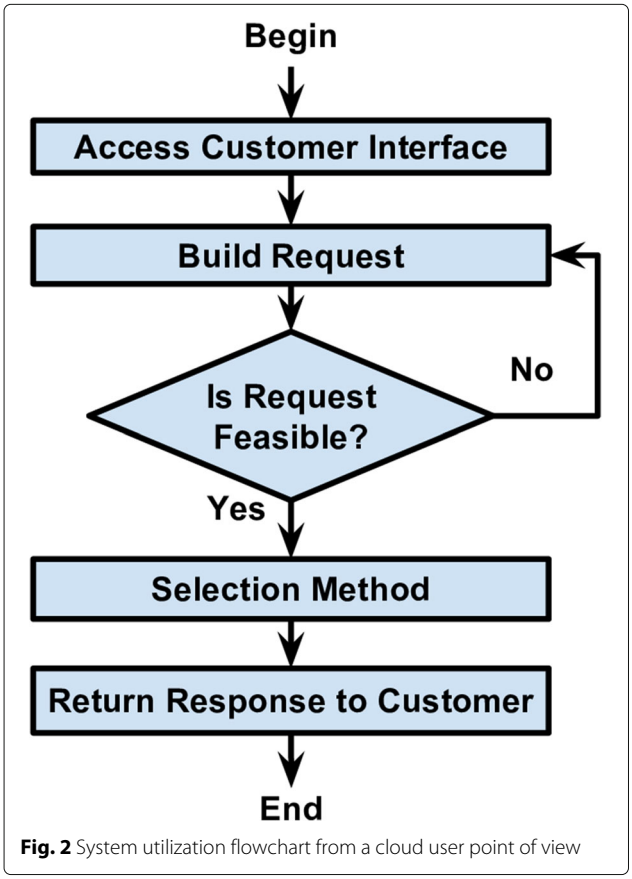


Fig. 2 System utilization flowchart from a cloud user point of view

Table 2 Generic CP database example

Name	Type	Group	P_1	P_2	...	P_n
PI_1	HB/LB/...	0/1,2,...	x_{11}	x_{12}	...	x_{1n}
PI_2	HB/LB/...	0/1,2,...	x_{21}	x_{22}	...	x_{2n}
PI_3	HB/LB/...	0/1,2,...	x_{31}	x_{32}	...	x_{3n}
...	...	...	...	...	...	...
PI_M	HB/LB/...	0/1,2,...	x_{M1}	x_{M2}	...	x_{Mn}
Price	LB	0	y_1	y_2	...	y_n

PIs can also be identified by a group id (a positive integer), where zero means that the PI is independent and not involved with any group. PIs on the same group, i.e., equal group id, are inseparable from each other. This means that a single CP must fulfill all PIs in the group.

If some values of Table 2 are not informed, default values are used. The default type for a PI is always NB (whether quantitative or qualitative), and the default group id is 0, meaning it is independent/separable from the other PIs. Default PI values for CPs vary according to each PIs meaning and type (quantitative or qualitative).

It is expected that the database feeding process is dynamic allowing the updating of PI values along the time. However, it is essential to note that all the database management process is out of scope of this work.

User request

The user request must inform all the m user's PIs of interest, which must be a subset of the database registered ones (i.e., $m \leq M$) with the respective desired value (X_j). User request can carry other features, such as the importance weight of each PI of interest (w_j), the tolerance value of the desired one (t_j), eventually the PI utility function type and whether this PI is essential to the user or not. In case PI utility type is informed, it will be used instead of the PI default utility type present in the CP database. Price is always LB and cannot be forced to another type. An essential PI indicates that it must be satisfied.

Table 3 shows a generic user request, with all fields that a selection method is able to process. Important to mention that the PI's name and desired value (integer or floating-point number) are mandatory whereas the other ones are optional.

Table 3 Generic user request

Name	Type	Value	Tol.	Weight	Essential
PI_1	.	X_1	t_1	w_1	true
PI_2	NB	X_2	t_2	w_2	false
PI_3	.	X_3	t_3	w_3	false
...	...	...	...	...	...
PI_m	.	X_m	t_m	w_m	false

Default PI tolerance value ("Tol.") is zero. If the PI is qualitative, the tolerance is implicit (HT, LT, and HLT – for ordered ones – or NB, otherwise). The default importance weight of each PI is 1. A PI is non-essential ("false") by default.

The pI attendance condition concept

The concept of PI attendance condition is central to this work, and the addressed selection methods use it several times. A PI value from a particular CP attend or satisfy a specific user desired value if this value is the desired one or it is better than the desired one or; at least, it is in the tolerance range from the desired one. To make such an evaluation, the utility type of each PI should be considered. The first step is to identify if the PI is quantitative or qualitative.

Thus, mathematically, a CP_i "Attend" certain quantitative PI_j, if its value (x_{ij}, numeric and present in the database), attends the user desired one (X_j, present in the request), with tolerance t_j (specified in the request, otherwise, tolerance is zero). Equation 4 presents this quantitative PI attendance condition function. Note that Eq. 4 takes into account the utility type of each PI and $i = 1, 2, ..., n$, where n is the amount of CPs in CP database, and $j = 1, 2, ..., m$, where m is the amount of PIs in the user request.

$$AttendQT(PI_j, x_{ij}, X_j, t_j) = \begin{cases} true \text{ if } x_{ij} \geq (X_j - t_j) \text{ and } PI_j \in HB \\ true \text{ if } x_{ij} \leq (X_j + t_j) \text{ and } PI_j \in LB \\ true \text{ if } (x_{ij} \geq (X_j - t_j) \text{ and } x_{ij} \leq (X_j + t_j)) \\ \text{ and } PI_j \in NB \\ false \text{ otherwise.} \end{cases} \tag{4}$$

Meanwhile, if PI_j is qualitative, it can be ordered or unordered [19]. If it is unordered, the attendance rule is simple: if x_{ij} (category, present in the CP database) is the value that the user desires (equals X_j, category present in the request), then PI_j is attended. Otherwise, it is not. However, if PI_j is ordered, then the attendance condition depends on its utility function is HT, LT, or HLT. Equation 5 presents the qualitative PI attendance condition function, with $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$.

$$AttendQL(PI_j, x_{ij}, X_j) = \begin{cases} true \text{ if } x_{ij} = X_j \text{ and } (PI_j \text{ is unordered or} \\ PI_j \in NB) \\ true \text{ if } x_{ij} \geq X_j \text{ and } (PI_j \text{ is ordered and} \\ PI_j \in HT) \\ true \text{ if } x_{ij} \leq X_j \text{ and } (PI_j \text{ is ordered and} \\ PI_j \in LT) \\ true \text{ if } PI_j \text{ is ordered and } PI_j \in HLT \\ false \text{ otherwise.} \end{cases} \tag{5}$$

Selection methods modelling

This section specifies the developed selection methods.

Deterministic methods calculate a score value for each CP that is used to rank and select the top-ranking one. The deterministic algorithms developed are the CPS-Matching and the CPS-DEA.

Metaheuristic methods provide the smallest non-empty set of CPs with the lowest price in order to maximize the request fulfillment. This work uses Evolutionary Algorithms (EAs), which are inspired on natural selection and survival of the fittest to accomplish that since they are one of the most used metaheuristic approaches to solve complex problems. Finding the smallest non-empty set of CPs coping with that restrictions makes the CPS problem much more complex because it creates a combination of possible different solution sets. Therefore, the computational complexity for n CPs is 2^n (search space grows exponentially), characterizing a complex optimization problem, justifying the use of EAs. In this work, the GA was chosen because of its ability to handle problems in binary representations, which suits the proposed CPS approach. The two DE variations are contributions of this work, where the versatility of the DE algorithm is employed to handle the CPS problem with discretization approaches. Thus, the EA-based methods developed are the CPS-GA, the CPS-BDE, and the CPS-DDE, which use a novel and PI-based particular fitness function also developed in this work and explained at "Metaheuristic methods: individual and fitness modelling" subsection.

The modelled hybrid methods are a merging of the previous ones and offer an answer according to the complexity of the user request. Thus, if PIs can be satisfied by a single CP, only the deterministic approach is executed. Otherwise, the metaheuristic one offers a solution. This work develops three hybrid methods: Matching-GA, which is the pair CPS-Matching and CPS-GA; Matching-BDE, the pair CPS-Matching and CPS-BDE; and Matching-DDE, resulting from the pair CPS-Matching and CPS-DDE.

It is essential to note that the proposed architecture somehow deals with the cloud elasticity concept. The architecture accomplishes that using a combination of indicator's classification ontology (HB, LB, NB, HT, LT, HLT) and the user providing tolerance values for the indicators requested. Thus, depending on indicators' nature (utility function - HB, LB, for example) and the tolerances set, the CP score is adjusted. This means that if a user chooses a CP and contracts those requested resources, it is probable that the CP can elastically provide more of those resources if he needs.

CPS-Matching

The CPS-Matching is a deterministic mathematical PI-matching algorithm that can score and rank an extensive list of CPs based on value, type (quantitative or qualita-

tive), nature or utility function (HB, LB, NB, HT, LT, HLT) and importance of each PI (essential or non-essential) requested by the user [15]. A PI classified by user as essential must be attended, otherwise CPS-Matching does not process the CP candidate associated. Non-essential PIs have different importance levels to the user ranging from "High" and "Low" and once attended are scored accordingly. The method's input corresponds to a list P of all n CPs from the CP database, and the user request.

CPS-Matching is divided into three main stages: 1) elimination of user's request incompatible CPs; 2) scoring quantitative and/or qualitative PIs by importance level; and 3) calculating CPs final score and fulfillment indicator, ranking them, and returning results to the user. At the first stage, the initial CPs list P is filtered, removing all CPs that do not satisfy all the users' essential PIs, generating a new list of P' with n' ($n \geq n'$) different CPs.

The second stage receives P' and scores each CP's PI importance level individually, according to the utility (real benefit) of each of its PIs, where the higher the utility, the higher the score. PIs with the same weight (user request provided) are at the same importance level and each level's score is the arithmetic average of the scores of each PI (quantitative or qualitative) in that level [15]. A PI_j scores 0 if its value does not attend the user desired one, otherwise it will be scored proportionally to how useful this value is, compared with the same PI_j value in all other CPs available, multiplied by a constant.

Equation 6 shows the evaluation function of a quantitative PI_j of a CP_i. There, x_{ij} is the PI value (numeric) in CP database, X_j is the desired value requested by the user, with a maximum tolerance t_j. It always returns a normalized floating-point number between 0 and 1 ($\forall\ x_{ij}, X_j, X_{max}, X_{min} \geq 0$ and $C_1, C_2, t_j > 0$). Function "AttendQT" is depicted in Eq. 4. The numerical constants (empirical parameters) C_1 and C_2 belong to the normalized open interval between $]0,1[$, and $C_1 + C_2 = 1$, mandatory. The number X_{max} is the highest value among all other n' CPs in the list P' for that PI_j ; as well as X_{min} is the lowest value and t_j is the maximum tolerated distance from the optimum point (X_j) for an NB PI (since that PI attends X_j, i.e., it belongs to the interval $[X_j - t_j; X_j + t_j]$).

$$PtQT(PI_j, x_{ij}, X_j, t_j) = \begin{cases} 0 & \text{if } AttendQT(PI_j, x_{ij}, X_j, t_j) = false, \\ C_1 + C_2 \times \frac{x_{ij} - X_j}{Max(x_{1j},\ldots,x_{nj}) - X_j} & \text{if } PI_j \in HB \\ C_1 + C_2 \times \frac{X_j - x_{ij}}{X_j - Min(x_{1j},\ldots,x_{nj})} & \text{if } PI_j \in LB \\ C_1 + C_2 \times \frac{t_j - |X_j - x_{ij}|}{t_j} & \text{if } PI_j \in NB. \end{cases} \tag{6}$$

Equation 7 shows the evaluation function of a qualitative PI_j of a CP_i. There, x_{ij} is the PI value (category or subclass) in CP database, X_j is the desired PI_j value requested by the user and K_1, K_2 are the total number of tolerable categories higher and lower, respectively, to X_j, and $K_3 = K_1 + K_2$. This function always returns a normalized floating-point number between 0.0 and 1.0 ($\forall K_1, K_2, K_3 \geq 0$ and $0 < C_3 < 1$). Equation 5 depicts function "AttendQL". Note that the score of tolerable categories (only for ordered PIs with tolerances, i.e., HT, LT, or HLT) will be directly influenced by the distance between the category X_j specified by the user, and x_{ij}, which is offered by the CP_i. It means the greater the distance, the lower the score for that PI. The distance between the categories x_{ij} and X_j is the difference between their levels, given by function $level(x)$ that returns a positive integer, from 1 to the total number of categories available in increasing order of graduation corresponding to the level of category x. Therefore, in case of perfect match ($x_{ij} = X_j$) the score is maximum, that is 1. In this case, the desired neighboring categories (above and below) will score C_3.

$$PtQL(PI_j, x_{ij}, X_j) = \begin{cases} 0 & \text{if } AttendQL(PI_j, x_{ij}, X_j) = false \\ 1 & \text{if } x_{ij} = X_j \\ C_3 \times \frac{K_1 - |level(x_{ij}) - level(X_j)| + 1}{K_1} & \text{if } PI_j \in HT \\ C_3 \times \frac{K_2 - |level(x_{ij}) - level(X_j)| + 1}{K_2} & \text{if } PI_j \in LT \\ C_3 \times \frac{K_3 - |level(x_{ij}) - level(X_j)| + 1}{K_3} & \text{if } PI_j \in HLT. \end{cases} \tag{7}$$

The third stage calculates the final score for each CP in the list P'. The CP's final score is the weighted average of the importance levels score. The weights of each level can be calculated analysing the relationship of importance between the level's PIs' weight values, using the AHP's Judgment Matrix [15], or just be specified by the user with floating-point numbers ranging from 0 to 1. When not specified, level's weights are equal to 1. Final score is calculated individually for each CP and varies in the range from 0 to 1, where the closer to 1, the more adequate the CP is to satisfy the user request. Finally, the CPS-Matching method returns a list of the highest-ranked CPs, containing their names and the PIs fulfillment indicator, regardless of PIs importance levels. The fulfillment indicator is the weighted average of all requested PIs attended by that CP. Likewise CP score, Eqs. 6 and 7 calculates the fulfillment indicator. Thus, in the absence of different PIs importance levels, fulfillment indicator equals final score.

As a CPS-Matching execution example, consider calculation performed for CP1 and CP2 $\in P'$ containing four PIs: RAM, HD, Availability and Price. Table 4 presents CPs PIs, and user request values. In this case, all PIs are quantitative. Tolerances are not informed neither if there are essential PIs. Moreover, as the user does not inform different PI weights, there is only one, whose weight is equal to 1. Therefore, considering C_1 and C_2 are both equal to 0.5, we have:

Table 4 CP PI database and user request example

PIs	Type	GID	CP1	CP2	Request
RAM	HB	0	2	8	4
HD	HB	0	20	25	20
Availability	HB	0	99.99	96.8	98
Price	LB	0	7	8.5	8

$$RAM_{CP1} = 0$$
$$HD_{CP1} = 0.5 + 0.5 \times \frac{20 - 20}{Max(20, 25) - 20} = 0.5$$
$$A_{CP1} = 0.5 + 0.5 \times \frac{99.9 - 98}{Max(99.9, 96.8) - 98} = 1$$
$$Price_{CP1} = 0.5 + 0.5 \times \frac{8 - 7}{8 - Min(7, 8.5)} = 1$$

$$RAM_{CP2} = 0.5 + 0.5 \times \frac{8 - 4}{Max(8, 4) - 4} = 1$$
$$HD_{CP2} = 0.5 + 0.5 \times \frac{25 - 20}{Max(20, 25) - 20} = 1$$
$$A_{CP2} = 0$$
$$Price_{CP2} = 0$$

$$Levels_{CP1} = \frac{0 + 0.5 + 1 + 1}{4} \approx 0.63$$
$$Levels_{CP2} = \frac{1 + 1 + 0 + 0}{4} = 0.5$$
$$Score_{CP1} = Fulfillment_{CP1} = 0.63$$
$$Score_{CP2} = Fulfillment_{CP2} = 0.50$$

CPS-DEA

CPS-DEA is a PI-based modelling of the classical DEA method ("Data envelopment analysis" subsection) applied to the CPS problem [16]. This work uses DEA instead of other multicriteria methods since DEA fundamentally asses efficiency that is considered a well-tailored metric do solve the CPS problem. For the CPS problem, each CP can be understood as a DMU, and each DEA input and output variables are related to the user requested PI values (user request) and CPs PI values (CP database). This method is divided into three main stages: 1) DEA input converter, 2) DEA method application and, 3) Final ranking routine.

DEA input and output values for each CP are generated by PIs aggregation functions. Thus, the CP database and requested PI values are converted to a format that DEA can use to calculate each CP's relative efficiency accurately [16]. Each CP has a predefined constant number of inputs and outputs regardless database and request sizes. Therefore, regarding the CPS problem as addressed in this work, each CP has two variables of input and two variables of output identified:

- **Inputs (criteria to be minimized):**
  1. **Resources ("Res."):** It is the weighted average of the normalized PI values present in the CP database. The used weights are the PI's weight present in the user request.
  2. **Costs ("Cost."):** It is the normalized CP's price regarding the most expensive price in the CP database.
- **Outputs (criteria to be maximized):**
  1. **Suitability ("Suit."):** It is the weighted average of the attending condition of each CP's PI, according to the user request (except price). It indicates how appropriate the CP is to the request.
  2. **Leftovers ("Left."):** It is the weighted average of all the resources that had left in the CP after attending the request (for quantitative PI HB and LB, only).

Equations 10 and 11 show how the inputs "Resources" ($Res(P_i)$) and "Costs" ($Cost(P_i)$) are converted for the provider P_i, respectively, and Eqs. 14 and 16 calculate the outputs "Suitability" ($Suit(P_i)$) and "Leftovers" ($Letf(P_i)$), respectively. Maximum and minimum functions are considered to normalization aspects when converting PI values to suitable input and output DEA values. All inputs and outputs are normalized between 0 and 1.

$$Dif = Max(x_{1j}, ..., x_{nj}, X_j) - Min(x_{1j}, ..., x_{nj}, X_j) \tag{8}$$

$$R_i = \sum_{j=1}^{m} w_j \times \begin{cases} 1 - \frac{x_{ij}}{Max(x_{1j},...,x_{nj},X_j)} & \text{if } PI_j \in HB \\ \frac{x_{ij}}{Max(x_{1j},...,x_{nj},X_j)} & \text{if } PI_j \in LB \\ \frac{|x_{ij} - X_j|}{Dif} & \text{if } PI_j \in NB \end{cases} \tag{9}$$

$$Res(P_i) = \left(\frac{R_i}{\sum_{j=1}^{m} w_j} \right)^{\alpha}. \tag{10}$$

$$Cost(P_i) = \left(\frac{y_i}{Max(y_1, y_2, ..., y_n)} \right)^{\beta}. \tag{11}$$

$$Attend(PI_j, x_{ij}, X_j, t_j) = \begin{cases} true & \text{if } (PI_j \in HB \text{ or } PI_j \in LB \text{ or } PI_j \in NB) \\ & \text{and } AttendQT(PI_j, x_{ij}, X_j, t_j) = true \\ true & \text{if } (PI_j \in HT \text{ or } PI_j \in LT \text{ or } PI_j \in HLT) \\ & \text{and } AttendQL(PI_j, x_{ij}, X_j) = true \\ false & \text{otherwise.} \end{cases} \tag{12}$$

$$V_i = \begin{cases} 1 \text{ if } Attend(PI_j, x_{ij}, X_j, t_j) = true \\ 0 \text{ otherwise} \end{cases} \quad (13)$$

$$Suit(P_i) = \left(\frac{\sum_{j=1}^{m} w_j V_i}{\sum_{j=1}^{m} w_j} \right)^{\gamma}. \quad (14)$$

$$L_i = \sum_{j=1}^{a} w_j \times \begin{cases} \frac{x_{ij} - X_j}{Max(x_{1j},...,x_{nj},X_j) - X_j} & \text{if } PI_j \in HB \\ & \text{and } Attend(PI_j, x_{ij}, X_j, 0) = true \\ 1 - \frac{x_{ij} - X_j}{Max(x_{1j},...,x_{nj},X_j) - X_j} & \text{if } PI_j \in LB \\ & \text{and } Attend(PI_j, x_{ij}, X_j, 0) = true \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

$$Left(P_i) = \left(\frac{L_i}{\sum_{j=1}^{a} w_j} \right)^{\delta}. \quad (16)$$

Where $i = 1, 2, ..., n$; $j = 1, 2, ..., m$; $\alpha, \beta, \gamma, \delta = 1, 2, 3, ...$; $m_{HB}, m_{LB}, m_{NB} = 0, 1, 2, ...$, subject to $m = m_{HB} + m_{LB} + m_{NB}$, where m_{HB}, m_{LB} and m_{NB} represent the number of PIs whose utility function is HB, LB and NB, respectively. Also, it is informed that $a = m_{HB} + m_{LB}$.

It is important to note that n is the total number of CP in database and m is the total number of PIs in the user request. Also, y_i is the service price of CP *i*. *Dif* is the difference between the maximum value of a particular NB PI_j value and the user requested PI_j value (X_j), and the minimum value of that PI. In addition, in case $X_j \geq Max(x_{1j}, ..., x_{nj})$ then the "Leftovers" value for that PI_j is always zero $\forall P_i, x_{ij}$.

The adjustable factors $\alpha, \beta, \gamma, \delta$ are responsible for transforming all the inputs/outputs "Resources", "Costs", "Suitability" and "Leftovers", respectively, into functions with a variation (increase/decrease), e.g., linear (1) is default, quadratic (2), cubic (3), etc. That is, the higher the value of such factors, the more significantly, numeric speaking, these inputs/outputs will be affected for each change made. The higher the factor, the greater the punctuation loss for every CP that does not reach 1 (efficient). The farther from 1 and closer to 0, the more score CPs will lose. For more critical input/output, it is appropriate to increase its associated factor. Bearing this in mind, the considered most critical variable is "Suitability". Therefore, its factor will be 2 ($\gamma = 2$), and for the others, it will be 1 (linear variation).

The second stage starts after these inputs and outputs calculation for each candidate CP feeding software that executes the DEA method, generating an efficiency frontier among CPs. Particularly to the CPS problem, the DEA input orientation (the one that tries to maximize outputs) was used since maximization of "Suitability" as output is an essential characteristic to be observed to select CPs. Comprising the DEA model, the Variable Returns to Scale (VRS) is used since it is more realistic for the problem's scope considering that the variations of input and output are not proportional, especially taking into account that "Suitability" and "Leftovers" (outputs) mostly are database and user request dependent. In contrast, "Resources" and "Costs" (input) are only dependent on the CP database.

The third stage uses each CP's input and output values for ranking all CPs when there is more than one in the DEA efficiency frontier. To accomplish that, CPs belonging to the efficiency frontier will be ranked first by their "Suitability" value, followed by the value of "Costs", then "Leftovers" and finally by "Resources" value in a non-compensatory way (*Suitability* > *Costs* > *Leftovers* > *Resources*, always). The CP with the highest value of "Suitability" will be at the top of the ranking. In the case of a tie, the CP with the lowest "Costs" will be the first in the ranking. If two or more CPs have the same "Suitability" and "Costs", then the highest value of "Leftovers" will be considered a tiebreaker, and, for the last case, the lowest value of "Resources" will be used. If one or more CPs tie in the four features, they will be sorted alphabetically.

Let us consider the CPs, PIs and user request in Table 4, to a CPS-DEA example. The first step is to convert PIs and request into input an output variables. Thus, we have:

DEA inputs calculation example:

$$RAM_{CP1} = 1 - \frac{2}{Max(2, 8, 4)} = 0.75$$

$$HD_{CP1} = 1 - \frac{20}{Max(20, 25, 20)} = 0.2$$

$$A_{CP1} = 1 - \frac{99.9}{Max(99.9, 96.8, 98)} = 0$$

$$Price_{CP1} = \frac{7}{Max(7, 8.5, 8)} \approx 0.83$$

$$\mathbf{Resources_{CP1}} = \left(\frac{0.75 + 0.2 + 0 + 0.83}{4} \right)^1 = 0.4$$

$$\mathbf{Costs_{CP1}} = \left(\frac{7}{Max(7, 8.5, 8)} \right)^1 \approx 0.83$$

$$RAM_{CP2} = 1 - \frac{8}{Max(2, 8, 4)} = 0$$

$$HD_{CP2} = 1 - \frac{25}{Max(20, 25, 20)} = 0$$

$$A_{CP2} = 1 - \frac{96.8}{Max(99.9, 96.8, 98)} = 0.031$$

$$Price_{CP2} = \frac{8.5}{Max(7, 8.5, 8)} = 1$$

$$\mathbf{Resources_{CP2}} = \left(\frac{0 + 0 + 0.031 + 1}{4} \right)^1 \approx 0.26$$

$$\mathbf{Costs_{CP2}} = \left(\frac{8}{Max(7, 8.5, 8)} \right)^1 \approx 0.94$$

DEA outputs calculation example:

$$RAM_{CP1} = 0$$
$$HD_{CP1} = 1$$
$$A_{CP1} = 1$$
$$\textbf{Suitability}_{\textbf{CP1}} = \left(\frac{0+1+1}{3}\right)^2 \approx 0.45$$
$$RAM_{CP1} = 0$$
$$HD_{CP1} = \frac{20-20}{Max(20,25,20)-20} = 0$$
$$A_{CP1} = \frac{99.9-98}{Max(99.9,96.8,98)-98} = 1$$
$$\textbf{Leftovers}_{\textbf{CP1}} = \left(\frac{0+0+1}{3}\right)^1 \approx 0.34$$

$$RAM_{CP2} = 1$$
$$HD_{CP2} = 1$$
$$A_{CP2} = 0$$
$$\textbf{Suitability}_{\textbf{CP2}} = \left(\frac{1+1+0}{3}\right)^2 \approx 0.45$$
$$RAM_{CP2} = \frac{8-4}{Max(2,8,4)-4} = 1$$
$$HD_{CP2} = \frac{25-20}{Max(20,25,20)-20} = 1$$
$$A_{CP2} = 0$$
$$\textbf{Leftovers}_{\textbf{CP2}} = \left(\frac{1+1+0}{3}\right)^1 \approx 0.67$$

Next steps comprise to execute DEA with that inputs and outputs for each CP, and finally to rank CPs using the efficiency frontier or the tiebreaks. Thus, the resulting ranking order is CP1 and CP2 since both are in the efficiency frontier, have the same Suitability but CP1 costs less than CP2.

Metaheuristic methods: individual and fitness modelling

This subsection elaborates on the proposed individual and the fitness function developed for the CPS-GA, CPS-BDE, and CPS-DDE metaheuristic methods. As already stated, the individual represents a candidate solution to the problem. The individual is coded as a vector using a binary encoding, where the number of coded variables is equal to the amount of CPs registered in the CP database. Each variable uses a position in the binary vector, and it is represented by a single bit, indicating whether that CP belongs to the solution set (bit 1) or not (bit 0) [17]. All CPs with bit 1 will be called "employed CPs" for that individual, i.e., CPs that will be effectively used in the solution set. This information is used in the fitness function to generate a fitness value for each individual.

Figure 3 illustrates this modelling with an example comprising ten hypothetical CPs and the resulting solution set that this individual encodes. Thus, in this case, the employed CPs are P1, P3, P4, and P8.

The individual's fitness is proportionally linked to the minimization of three factors: i) the total amount of employed CPs; ii) the total price of the individual; iii) the difference between the requested PI values and the ones from the individual's employed CPs. Thus, if this problem's objective function is to combine these factors for an individual, the fitness function comprises the minimization of that function altogether with the application of penalty when the individual is not a satisfactory answer. Equation 17 presents the fitness function (*Fit*) of the i^{th} individual of the population (*Ind*), where w_{pen} is a constant value for the penalty weight applied to individuals, whose default value is 1.

$$Fit(Ind) = 1 - Obj(Ind) - w_{pen} \times penalty(Ind). \quad (17)$$

Equation 18 presents the objective function (*Obj*), which is the weighted average of the employed CPs score (*ecp*) and the individual *Ind* total price (*price*), both normalized between 0 and 1. The weight values for the number of employed CPs (w_n) and price (w_p) ponder the importance of the least amount of CPs, and the lowest price desired for the final solution, respectively.

$$Obj(Ind) = \frac{w_n \times ecp(Ind) + w_p \times price(Ind)}{w_n + w_p}. \quad (18)$$

Since *Ind* is an encoding vector of size n, it is possible to that $n0$ represents the number of bits 0 in *Ind* and, $n1$ the number of bits 1. Equation 19 presents the *ecp* score given to *Ind* comprising its number of employed CPs.

$$ecp(Ind) = \begin{cases} 1 & \text{if } n1 = 0 \\ \frac{|n1 - minCP|}{n - minCP} & \text{otherwise,} \end{cases} \quad (19)$$

where *minCP* is a constant integer representing the minimal possible number of CPs estimated to maximize the attendance of the current request. Note that the extreme cases (encoding filled with all bits 0, or all bits 1) are penalized to the maximum (1) because they are undesired to the CPS problem solution.

So, let $P = \{P_1, P_2, ..., P_n\}$ be the set containing the n different candidate CPs available in the CP database. Now,

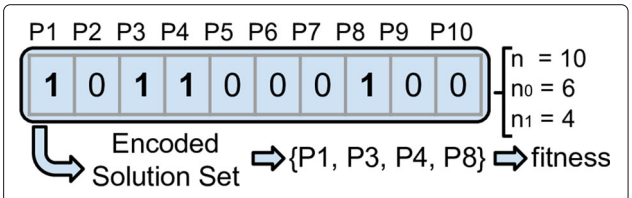


Fig. 3 Example of a possible solution encoded in a binary vector individual [17]

let $Y = \{y_1, y_2, y_3, ..., y_n\}$ be the respective prices associated with each of the n different CPs of P. The total price (*tprice*) of *Ind* is the sum of the prices of all employed CPs encoded by *Ind*, i.e., the sum of $y_i \in Y$ whose position i at the encoding vector *Ind* is bit 1. Thus, if a CP does not belong (bit 0, i.e., non-employed CP) to the encoded solution, its price will be zero. In order to be usable to the fitness function (Eq. 17), the *Ind* total price must be normalized according Eq. 20, where *maxPrice* is the sum of all prices of Y, i.e., $maxPrice = \sum_{j=1}^{n} y_j$, where $j = 1, 2, ..., n$.

$$price(Ind) = \frac{|tprice(Ind) - minPrice|}{maxPrice - minPrice}. \tag{20}$$

The floating-point number *minPrice* represents the minimal possible price value estimated to maximize the current request's attendance. It is the total sum of the *minCP* lowest prices in the database.

Penalties are applied to fitness computation according to Eq. 17. These penalties are a way to handle constrained optimization problems by significantly decreasing the individual's fitness value if it presents an inadequate solution to the problem [46]. The penalty is calculated proportionally as a function of how much the solution *Ind* infringes on the problem constraint. For the CPS problem, there are two constraints [17]: the individual must not be all encoded with bit 0, and the employed CPs must maximize attendance of all user requested PIs. The first constraint prevents functions that generate the initial random population and the genetic operators (modify individual's codification vector) from generating an encoding vector filled with bit 0 for any individual. The second constraint will proportionally penalize every individual that does not attend all PIs of the user request. So, Eq. 23 shows the penalty calculation, where the ordered list w contains the weights of each PI_j, function $pen(PI_j)$ returns a value between 0 and 1, representing the penalty value comprising the attendance or not of the PI_j or the PIs that belong to the PI_j group; m is the number of PIs from the user request, X_j is the PI_j user-requested value, x_{ij} is the PI_j value for the CP that best attends it, and t_j is the PI_j tolerance value provided by user. The function named *grouID* returns the group id number of a particular *PI*; s represents the number of PIs that belong to a particular PI_j group, according to a group id. A PI that does not belong to a group is group id 0. A user requested PI is fully attended if at least one CP encoded in *Ind* matches it. Another point to be emphasized is that if those PIs are arranged into groups which sizes are greater than 1, then that PI group is fully attended if at least one CP encoded in *Ind* attends all the PIs in that group. Otherwise, this individual will be proportionally penalized to how many PIs on that group are not attended by the encoded CP that better attend that group.

$$Att(PI_j, x_{ij}, X_j, t_j) = \begin{cases} 0 \text{ if } Attend(PI_j, x_{ij}, X_j, t_j) = true \\ 1 \text{ otherwise} \end{cases} \tag{21}$$

$$pen(PI_j) = \begin{cases} Att(PI_j, x_{ij}, X_j, t_j) & \text{if } groupID(PI_j) = 0 \\ \frac{\sum_{j=1}^{s} Att(PI_j, x_{ij}, X_j, t_j)}{s} & \text{otherwise} \end{cases} \tag{22}$$

$$penalty(Ind) = \frac{\sum_{j=1}^{m} w_j \times pen(PI_j)}{\sum_{j=1}^{m} w_j}. \tag{23}$$

The fitness calculation is applied to every individual in the population for each algorithm generation, up to a predefined number of generations. At the end of the generations, the best fitness individual is sought and its encoded CPs are returned as the response.

Hybrid methods

The modelled hybrid methods combine the CPS-Matching deterministic method and one metaheuristic method (e.g., CPS-GA), in a pipeline. For the hybrid methods, the user must choose the minimum desired percentage of fulfillment for his request (until 100%) regarding the CPS-Matching method. The CPS-Matching is first applied on the initial CP list resulting in a CPs best-rated list with their fulfillment indicator values. If the best CP fulfillment indicator value is equal or greater than the desired fulfillment value, the method returns that single CP to the user. However, if there is more than one best-rated CP with the same fulfillment value, the one with lowest price is returned. If the fulfillment value of the best-rated CPs is lesser than the desired, then the metaheuristic algorithm is executed providing as result the employed CPs in the highest fitness individual found at the end of all iterations.

Experiments and results

This section describes the experimentation protocol and algorithms' configuration, problem instances, and results, followed by their analysis. The algorithms were implemented in Java (JDK version 1.8) to the 64 bits Windows 10 operating system and are executed in a host with 8 gigabytes of RAM and an Intel Core i5, 3.0 GHz CPU.

Experimental setup

The minimal suitability value used for the hybrid methods Mtc-GA, Mtc-BDE and Mtc-DDE is 100%. Parameters used to CPS-Matching method (hybrid counterparts too) are: $C_1 = 0.9$ and $C_2 = 0.1$. Regarding CPS-DEA method, factors used are: $\alpha, \beta, \delta = 1$ and $\gamma = 2$, as well as DEA variable VRS model with output orientation.

Table 5 presents the main parameters and their values, which are defined empirically, applied to the EAs methods and their hybrid counterparts. The stop condition of these

Table 5 Parameters used by the EA methods

Parameter	GA	BDE	DDE
Population size	50	50	50
Number of generations	2000	2000	2000
Crossover Rate	95%	–	–
Mutation Rate	1%	5%	–
Probability of perturbation	–	50%	50%
Stochastic tournament size	5	–	–
Mutation weighting factor	–	–	0.5

EAs is 2000 generations (IT). As the size of the population (POP) is 50 individuals and the number of iterations (IT) is 2000, a total of 100000 fitness evaluations ($POP \times IT$) are performed for each EA to reach stop condition. The number of dimensions/variables (DIM) is always equal to the number of CPs (n) registered in the CP database for all the algorithms.

The CPS-GA selection routine is the stochastic tournament. It presents a parameter k that represents the size of the group that compete in each GA iteration, which is equal to 5 in the performed experiments. BDE and DDE use greedy selection that is applied on two individuals (parent and offspring) choosing the one with the highest fitness.

GA uses the one-cut-point crossover and bit-flip mutation. BDE has its own specific crossover ("Binary differential evolution" section) and bit-flip mutation too. DDE uses only its specific mutation routine, according to Eq. 2, with a mutation weighting factor of 0.5. The CPS-GA is the only EA that uses an elitism routine that is executed after the fitness evaluation of the new generated population. This routine replaces the individual with lower fitness on the current generation with the previous generation's best fitness one.

Moreover, comprising the EAs, 30 independent runs were performed for each problem instance. Regarding the deterministic methods, a single execution is performed. The weights for price (w_p), quantity of CPs (w_n) and penalty (w_{pen}) on the fitness function (Eq. 17) have the same importance and are constant and unitary ($w_p = w_n = w_{pen} = 1$). EAs and hybrid algorithms use this fitness function.

A statistical analysis of the results was performed. The null hypothesis of normality was rejected by the Shapiro-Wilk [47] test with a 5% significance level, i.e., $\alpha = 0.05$, presenting p-values smaller than 0.05. Hence, the non-parametric Dunn's test is employed also with a 5% significance level, i.e., $\alpha = 0.05$ [48]. Due to the multiple comparisons involved, the Bonferroni α correction is used.

Problem instances

Table 6 informs the basic CP candidates database used in the experiments. This database is fictitious and involves ten CPs with six PIs and their values that occur in real scenarios. The used PIs are the total amount of "RAM" available (Gb); maximum "HD Memory" for storage (Gb); the maximum amount of usable "CPU Power" (e.g., CPU frequency times the number of CPU cores); average CP resources "Availability" available via the Internet (percentage of accessible resources per year); estimated CP level of information "Security" (levels 1-5); and, finally "Price" of all that resources available for each CP (US$ per month of use). Each PI has one type of utility function behavior associated (HB, LB, or NB), one group id and one value for each CP.

On the other hand, Table 7 informs eight possible cloud user requests considering the CP database shown in Table 6. Note that not necessarily all database stored PIs are needed to the user. Request 4, for example, ignores PI "Security", that is, its value does not matter to the user and it can assume anyone.

Price is a special PI, which is not stated explicitly in the user request though it is always considered in the fitness function and the CPS-DEA method. However, in the CPS-Matching method, the price must be explicitly specified in each request. Otherwise, it will be ignored. Therefore, for the sake of comparison, to the experiments with the CPS-Matching the prices used are 8.25 (Req. 1 and 2); 10.0 (Req. 3); 1.25 (Req. 4); 7.5 (Req. 5 and 8); 2.5 (Req. 6) and 4.5 (Req. 7).

In this example, each PI has the same importance weight (1.0) as all others. Last line of Table 7 presents the known

Table 6 Simulated CPs database with their PIs

PIs	Type	ID	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
RAM	HB	1	1	2	4	8	16	32	64	32	16	8
HD Memory	HB	1	5	20	40	25	50	100	200	150	20	10
CPU Power	HB	1	2.0	2.5	3.5	4.0	5.5	6.0	10.0	8.0	7.0	5.0
Availability	HB	0	99.99	95.2	98.3	96.8	97.9	98.9	99.3	98.5	96.4	97.5
Security	NB	0	1	2	2	3	3	5	4	4	3	4
Price	LB	0	0.50	1.25	2.50	3.80	4.50	7.50	10.0	8.25	7.80	6.50

Table 7 Simulated user requests

PIs	Req. 1	Req. 2	Req. 3	Req. 4	Req. 5	Req. 6	Req. 7	Req. 8
RAM	8	16	64	2	16	4	–	–
HD Memory	20	50	200	20	100	40	50	–
CPU Power	4.0	8.0	10.0	2.0	5.0	–	–	–
Availability	98.0	98.0	99.9	99.0	–	98.0	97.0	99.99
Security	4	5	5	–	–	2	3	5
Optimal Answer	P8	P6 & P8	P1 & P6 & P7	P1 & P2	P6	P3	P5	P1 & P6

answers (optimal) for each request. The easiest requests are 1, 5, 6, and 7, because a single CP, of the 10, is already able to attend them completely. The most complex request to be answered in this scenario is the third request since the answer demands the highest values of resources and criteria levels being composed of three CPs.

Results and analysis

Tests with all methods discussed were performed for 10, 50, 100 and 200 CPs using data from Table 6. To accomplish that, those data were replicated 5, 10, and 20 times, respectively, to make new databases with 50, 100, and 200 CPs (horizontal scalability). Thus, data replicated to 50 CPs will generate the following data repetition on CPs: cloud provider P11 with the same original data of P1; P12 tied to P2, P13 tied to P3, and so on for each replication. This data replication was performed because it is a simple way to increase the database size without modifying the known optimum. The requests (Table 7) are the same for all these databases.

Table 8 presents results obtained when applying the CPS-Matching ("Mtc") method, CPS-DEA ("DEA"), CPS-GA ("GA"), CPS-BDE ("BDE") and CPS-DDE ("DDE") methods on the database on Table 6 replicated for 200 CPs, taking as input each one of the eight requisitions. The results for databases with fewer CPs (instances with 50 and 100 CPs) were omitted because they have identical results with 200 CPs. For all methods, the hits percentage (based on the known answer) has been calculated. For the evolutive methods, average and standard deviation of the fitness has been calculated as well. CPS-Matching and CPS-DEA methods did not use a fitness function. Thus, they have different answers for each request that can be satisfied by more than one CP. The top-scored CP for each request according to CPS-Matching and CPS-DEA methods are P8 (Req. 1), P6 (Req. 2), P7 (Req. 3), P7 (Req. 4), P6 (Req. 5), P3 (Req. 6), P5 (Req. 7), P1 (Req. 8).

According to Table 8, it is possible to notice that the performance of deterministic methods ("Mtc" and "DEA") reached the expected limits by attending half of the requests. They are optimal for simple requests but unsuitable attending the most complex ones. Each EA's performance was surprisingly the maximum because each of the 30 executions finds an optimum CP set, counting each as one hit. It is important to remember that there are several optimal CP sets because of the multimodality problem instances created by the several initial database replications. So, at using the aforementioned replicated CP database, the higher the number of CPs, the more optimal combinations exist (e.g., request 3). This fact increases the probability of the evolutive methods converging to any of these optima. Therefore, in order to explore a search scenario without these multiple optima, the price of the first 10 CPs that composes each solution of the current request was arbitrarily lowered, generating the results present in Table 9. These instances configure a more demanding search space to be solved by any of these evolutive methods. The results with 50 and 100 CPs were

Table 8 Hit percentage results for 200 CPs without arbitrary price decrease (multiple global optimal)

Request	Mtc	DEA	GA	BDE	DDE	GA (Av. ± Sd.)	BDE (Av. ± Sd.)	DDE (Av. ± Sd.)
Req. 1	100%	100%	100%	100%	100%	0.9963 ±0.0	0.9963 ±0.0	0.9963 ±0.0
Req. 2	0.0%	0.0%	100%	100%	100%	0.9930 ±0.0	0.9930 ±0.0	0.9930 ±0.0
Req. 3	0.0%	0.0%	100%	100%	100%	0.9921 ±0.0	0.9921 ±0.0	0.9921 ±0.0
Req. 4	0.0%	0.0%	100%	100%	100%	0.9969 ±0.0	0.9969 ±0.0	0.9969 ±0.0
Req. 5	100%	100%	100%	100%	100%	0.9967 ±0.0	0.9967 ±0.0	0.9967 ±0.0
Req. 6	100%	100%	100%	100%	100%	0.9990 ±0.0	0.9990 ±0.0	0.9990 ±0.0
Req. 7	100%	100%	100%	100%	100%	0.9981 ±0.0	0.9981 ±0.0	0.9981 ±0.0
Req. 8	0.0%	0.0%	100%	100%	100%	0.9967 ±0.0	0.9967 ±0.0	0.9967 ±0.0
Average	50%	50%	100%	100%	100%	–	–	–

Table 9 Hit percentage results for 200 CPs with arbitrary price decrease (single global optimal)

Request	A[a]	GA	BDE	DDE	GA (Av. ± Sd.)	BDE (Av. ± Sd.)	DDE (Av. ± Sd.)	Best Fit.
Req. 1	100%	56.67%	100%	83.33%	0.9966 ±0.0002	0.9968 ±0.0	0.9967 ±0.0002	0.9968
Req. 2	0.0%	40%	36.67%	76.67%	0.9935 ±0.0004	0.9936 ±0.0002	0.9938 ±0.0002	0.9939
Req. 3	0.0%	26.67%	6.67%	53.33%	0.9926 ±0.0004	0.9926 ±0.0003	0.9929 ±0.0003	0.9931
Req. 4	0.0%	83.33%	50%	86.67%	0.9973 ±0.0001	0.9972 ±0.0001	0.9973 ±0.0001	0.9974
Req. 5	100%	73.33%	100%	73.33%	0.9970 ±0.0002	0.9971 ±0.0	0.9970 ±0.0002	0.9971
Req. 6	100%	80%	100%	90%	0.9994 ±0.0002	0.9995 ±0.0	0.9994 ±0.0002	0.9995
Req. 7	100%	73.33%	100%	90%	0.9984 ±0.0002	0.9986 ±0.0	0.9985 ±0.0002	0.9986
Req. 8	0.0%	43.33%	53.33%	83.33%	0.9969 ±0.0003	0.9970 ±0.0002	0.9970 ±0.0002	0.9971
Average	50%	59.58%	68.33%	79.58%	–	–	–	–

[a]Column A shows the equal results from **Mtc** and **DEA** methods

omitted because they show almost 100% hits in all the requisitions with price changes. Table 9 last column presents the best possible fitness value, i.e., the fitness of the known optimum answer encoded.

Therefore, according to Table 9, EAs present consistent results and have a satisfactory average assertiveness (above 50%), specially CPS-DDE (almost 80%). The fitness standard deviation is in the order of 10^{-4}, showing EA's robustness for that instance. CPS-Matching and CPS-DEA present the same top ranking CPs, so they have the same hits. Moreover, Table 10 presents results of Dunn's statistical test with 5% significance level ($\alpha = 0.05$) and Bonferroni α correction for the hits values presented in Table 9 for each request and each EA. Table 10 shows p-values associated with EA hits in its respective row and column for each request. Values in bold are Dunn's test results with α correction of those EA considered statistically different.

Analyzing Table 10 and results obtained by the EA with the best average result (i.e., CPS-DDE), 3 out of 8 requests (i.e., requests 2, 3, and 8) makes CPS-DDE statistically better than CPS-GA and CPS-BDE. It is essential to notice that these requests are considered the most complex requests in this experimentation scenario. On the other hand, in 5 out of 8 requests (i.e., requests 1, 4, 5, 6, and 7), CPS-DDE statistically achieved the same results compared to CPS-GA and CPS-BDE.

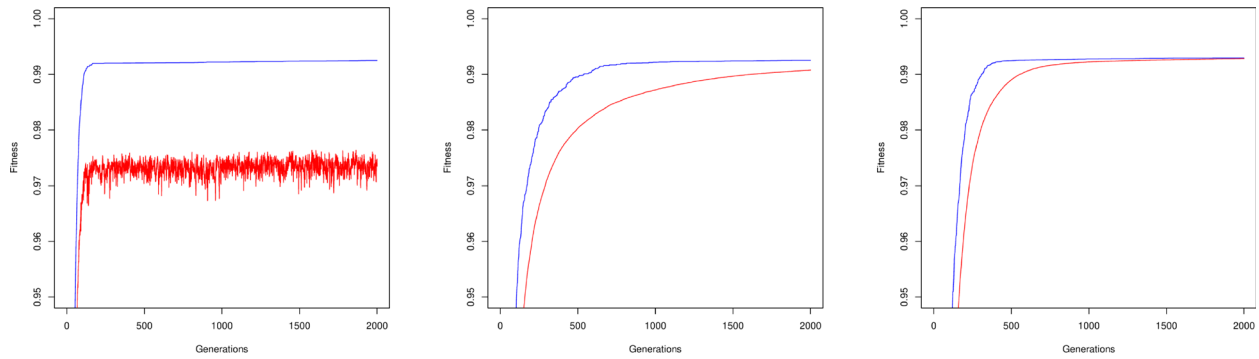
Figure 4 presents an average convergence plot regarding 30 executions for the CPS-GA, CPS-BDE, and CPS-DDE in the hardest problem instance (request 3 with 200 CPs and price decrease). The other requests have similar convergence behavior than request 3, for each EA. The y-axis shows fitness values and the x-axis shows generations. The y-axis scale starts at 0.95 in order to allow a better view and analysis. The upper curve represents the fitness of the best individual in the population, and the bottom curve represents the average fitness of the entire population.

It is possible to note that the amount of generations suffices good convergence at CPS-GA and CPS-DDE. The fitness improved is very noticeable for up to 500 generations. After that, small improvements still occur, and they are essential for convergence to the optimal solution,

Table 10 Dunn test with Bonferroni α correction for the hit values in Table 9

	GA	BDE
Req. 1		
BDE	**0.0**	–
DDE	**0.0154**	0.1628
Req. 2		
BDE	1.0	–
DDE	**0.0071**	**0.0031**
Req. 3		
BDE	0.1338	–
DDE	**0.0352**	**0.0001**
Req. 4		
BDE	**0.0055**	–
DDE	1.0	**0.0021**
Req. 5		
BDE	**0.0108**	–
DDE	1.0	**0.0108**
Req. 6		
BDE	**0.0154**	–
DDE	0.2988	0.2988
Req. 7		
BDE	**0.0026**	–
DDE	0.075	0.3595
Req. 8		
BDE	0.6477	–
DDE	**0.0025**	**0.0275**

Values are in bold when results are statistically different



(a) CPS-GA convergence curve (b) CPS-BDE convergence curve (c) CPS-DDE convergence curve

Fig. 4 Convergence plots for request 3 with 200 CPs, with price decrease

resulting in a hit. CPS-BDE has a smoother convergence curve and would probably benefit from more generations. Conversely, CPS-DDE has the best performance with these parameter values. The curve shape for average fitness in CPS-GA is different from the others because of the selection routine adopted (stochastic tournament).

Table 11 presents the hit percentages obtained with the hardest instances (requests with 200 CPs, with arbitrary price decrease) for the three hybrid methods: Mtc-GA, Mtc-BDE, and Mtc-DDE. The difference in the percentage of average hits of Mtc-GA and Mtc-DDE methods is 14.59 and 7.92 compared to GA and DDE methods, respectively. The Mtc-BDE hits percentage remained the same. In this case, starting the process with the CPS-Matching method did not make any difference.

Table 12 presents results of Dunn's statistical test (p-values) with 5% significance level and Bonferroni correction. These results correspond to the hit values presented in Table 11 for requests 2, 3, 4, and 8, and each hybrid method. Requests 1, 5, 6, and 7 were not tested because their hits percentage are the same for all methods (same performance), which is 100%. Values are in bold for those methods whose results (hits percentage) are considered statistically different.

Analyzing Table 12 and results obtained by the hybrid method with best average results (i.e., Mtc-DDE), it is possible to notice that in 3 out of 4 requests (i.e., requests 2, 3, and 8) the Mtc-DDE is statistically better than Mtc-GA and Mtc-BDE. Statistically, Mtc-DDE achieved the same results that Mtc-GA for request 4.

Finally, Table 13 shows the average execution time for all main selection algorithms, in milliseconds (ms), per execution. Thus, in ascending order, the execution time ranking is CPS-Matching, CPS-DEA, Matching-BDE, Matching-GA, CPS-BDE, CPS-GA, Matching-DDE, and CPS-DDE.

Therefore, according to Tables 8, 9, 11 and 13, it is possible to state that the EAs are able to find very satisfactory answers, even for the most complex cases (e.g., request 3 with 200 CPs) in an acceptable execution time. This behavior ensures the validity of the proposed fitness function for the problem, one of the main contributions of this work.

Deterministic methods like CPS-Matching and CPS-DEA presents minimal execution time. They are more stable (the same type of response, regardless of database size) than the others and can find optimal solutions for the more straightforward requests with 100% of request attendance. However, they are not adequate for more complex cases (requests 2, 3, 4, and 8), where more than one CP is required to attend each request fully.

The hybrid methods present the most optimized answers for more extensive databases. They merge both the natures of CPS-Matching (quick answers, stability, single run, efficient finding of the single best CP) and EAs (find the best CP set for complex multi-cloud requests),

Table 11 Hit percentages of the hybrid methods for 200 CPs with arbitrary price decrease

Request	Mtc-GA	Mtc-BDE	Mtc-DDE
Req. 1	100%	100%	100%
Req. 2	40%	36.67%	76.67%
Req. 3	26.67%	6.67%	53.33%
Req. 4	83.33%	50%	86.67%
Req. 5	100%	100%	100%
Req. 6	100%	100%	100%
Req. 7	100%	100%	100%
Req. 8	43.33%	53.33%	83.33%
Average	74.17%	68.33%	87.5%

Table 12 Dunn test with Bonferroni α correction for the hit values in Table 11

	Mtc-GA	Mtc-BDE		Mtc-GA	Mtc-BDE
Req. 2			**Req. 3**		
Mtc-BDE	1.0	–	Mtc-BDE	0.1338	–
Mtc-DDE	**0.0071**	**0.0031**	Mtc-DDE	**0.0352**	**0.0001**
Req. 4			**Req. 8**		
Mtc-BDE	**0.0055**	–	Mtc-BDE	0.6477	–
Mtc-DDE	1.0	**0.0021**	Mtc-DDE	**0.0025**	**0.0275**

Values are in bold when results are statistically different

making them an appropriate choice for real applications. Statistical analysis shows that Mtc-DDE hybrid method achieved the overall best performance.

Conclusions and future work

This work proposes a general approach for solving the Cloud Provider Selection problem. To accomplish that, several methods are developed using cloud provider PIs as selection criteria: deterministic using MCDM methods (e.g., CPS-Matching and CPS-DEA), metaheuristic (evolutionary algorithms, e.g., CPS-GA, CPS-BDE, and CPS-DDE) and hybrid ones (Matching-GA, Matching-BDE, and Matching-DDE). The experiments performed show that the hybrid methods present the most optimized answers, concerning hit percentage and execution time, for more demanding cases (200 CPs with price decrease), showing promise for a real large-scale application. Among them, the Matching-DDE hybrid method is the best one to solve the CPS problem. Moreover, this work provides a common ground by means of modelling the raw CP PIs into a usable search space for deterministic, metaheuristic, and hybrid methods to solve the CPS problem.

Table 13 Execution time for the developed experiments with 200 CPs, in milliseconds

Request	Mtc	DEA	GA	BDE	DDE	Mtc-GA	Mtc-BDE	Mtc-DDE
Req. 1	99	234	588	477	1611	99	99	99
Req. 2	89	284	592	487	1635	681	576	1724
Req. 3	70	254	593	488	1621	663	558	1691
Req. 4	85	218	580	465	1598	665	550	1683
Req. 5	74	192	573	458	1588	74	74	74
Req. 6	82	221	584	471	1602	82	82	82
Req. 7	76	184	580	464	1610	76	76	76
Req. 8	45	143	561	441	1575	606	486	1620
Average	78	216	581	469	1605	368	313	881
With arbitrary decrease in price (single global optimal)								
Req. 1	100	224	584	476	1603	100	100	100
Req. 2	88	272	587	487	1610	675	575	1698
Req. 3	75	262	585	489	1614	660	564	1689
Req. 4	94	209	576	465	1591	670	559	1685
Req. 5	83	198	571	457	1587	83	83	83
Req. 6	86	217	578	471	1596	86	86	86
Req. 7	78	172	575	465	1588	78	78	78
Req. 8	46	132	554	440	1567	600	486	1613
Average	81	211	576	469	1594	369	316	879

The uniqueness of the proposed approach can be depicted from its contributions: **i)** It can handle qualitative PIs (e.g., using the newly developed utility function - HT, LT and HLT - ontology); **ii)** A suitable PI-based fitness function allowing the use of metaheuristics; **iii)** A proper PI tolerance mechanism for deterministic methods and the cornerstone conversion equations of raw CP PIs into DEA inputs and outputs; The proposed approach design itself contributes to a potential standardization regarding the information and mechanisms to be used to solve the CPS problem.

This work can be improved. The access to a real and large scale database to perform more experiments is a limitation. Moreover, as future work, more selection methods can be added with standardized inputs and outputs. Also, a CP PI historical set of values can also be further considered to enhance the methods output. In addition, there are plans for working on a cloud broker where these issues are addressed including dealing with more dynamic user requests and database management. Moreover, the implementation of the selection method automatic choice for a particular CPS scenario is also planned.

Acknowledgments
The authors would like to express their gratitude to Santa Catarina State University, Joinville, Brazil for providing administrative and technical support.

Authors' contributions
Leading authors, Profs. Rafael Stubs Parpinelli and Adriano Fiorese: text structure, content advisement and review. Lucas Borges de Moraes: Text writing, programming, data collection and data analysis. All authors read and approved the final manuscript.

Funding
There was not funding to this research.

Availability of data and materials
All data and materials are available at: https://github.com/LBMbr/SelectionMethodsProject:/tree/master/.

Declarations

Ethics approval and consent to participate
The authors declare that they have no conflict of interest. All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version. This article does not contain any studies with human participants or animals performed by any of the authors.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Department of Computer Science (DCC), Joinville, SC, Brazil. [2] Graduate Program in Applied Computing (PPGCA), Joinville, SC, Brazil. [3] Santa Catarina State University (UDESC), Joinville, SC, Brazil.

Received: 17 December 2020 Accepted: 26 November 2021
Published online: 28 January 2022

References

1. Hogan MD, Liu F, Sokol AW, Jin T (2013) Nist Cloud Computing Standards Roadmap. NIST Special Publication 500 Series, USA
2. Senyo PK, Addae E, Boateng R (2018) Cloud computing research: A review of research themes, frameworks, methods and future research directions. Int J Inf Manag 38(1):128–139
3. Lee Y-C (2019) Adoption Intention of Cloud Computing at the Firm Level. J Comput Inf Syst 59(1):61–72
4. Ishizaka A, Nemery P (2013) Multi-Criteria Decision Analysis: Methods and Software. John Wiley & Sons, Ltd, United Kingdom
5. Whaiduzzaman M, Gani A, Anuar NB, Shiraz M, Haque MN, Haque IT (2014) Cloud service selection using multicriteria decision analysis. Sci World J 2014:1–10
6. Dhivya R, Devi R, Shanmugalakshmi R (2016) Parameters and methods used to evaluate cloud service providers: A survey. In: 2016 International Conference on Computer Communication and Informatics (ICCCI). IEEE, Coimbatore. pp 1–5
7. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Futur Gener Comput Syst 29:1012–1023
8. Baranwal G, Vidyarthi DP (2014) A framework for selection of best cloud service provider using ranked voting method. In: 2014 IEEE International Advance Computing Conference (IACC 2014). IEEE, Gurgaon. pp 831–837
9. Achar R, Thilagam PS (2014) A broker based approach for cloud provider selection. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2014). IEEE, Delhi. pp 1252–1257
10. Jaiswal A, Mishra R (2017) Cloud service selection using TOPSIS and fuzzy TOPSIS with AHP and ANP. In: Proceedings of the 2017 International Conference on Machine Learning and Soft Computing (ICMLSC 2017). ACM, Ho Chi Minh City. pp 136–142
11. Meesariganda BR, Ishizaka A (2017) Mapping verbal AHP scale to numerical scale for cloud computing strategy selection. Appl Soft Comput 53:111–118
12. Abdel-Basset M, Mohamed M, Chang V (2018) NMCDA: A framework for evaluating cloud computing services. Futur Gener Comput Syst 86:12–29
13. Mukherjee P, Patra SS, Pradhan C, Barik RK (2020) HHO Algorithm for Cloud Service Provider Selection. In: 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE). IEEE, Bhubaneswar. pp 324–327
14. M. S (2020) Cloud service provider selection using non-dominated sorting genetic algorithm. In: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI). IEEE, Tirunelveli. pp 800–807
15. Moraes L, Fiorese A, Matos F (2017) A multi-criteria scoring method based on performance indicators for cloud computing provider selection. In: Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017), vol. 2. INSTICC, Porto. pp 588–599
16. Moraes L, Cirne P, Matos F, Parpinelli RS, Fiorese A (2018) An Efficiency Frontier Based Model for Cloud Computing Provider Selection and Ranking. In: Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018). INSTICC, Madeira. pp 543–554
17. Moraes L, Fiorese A, Parpinelli RS (2017) An evolutive scoring method for cloud computing provider selection based on performance indicators. In: Proceedings of the 16th Mexican International Conference on Artificial Intelligence (MICAI 2017). Springer, Baja California. pp 1–12
18. Bishop DA (2018) Key Performance Indicators: Ideation to Creation. IEEE Eng Manag Rev 46(1):13–15
19. Jain R (1991) The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, Littleton
20. CSMIC (2014) Service measurement index framework. Technical report. Carnegie Mellon University, Silicon Valley, Moffett Field
21. Siegel J, Perdue J (2012) Cloud services measures for global use: The service measurement index (SMI). In: Annual SRII Global Conference 2012. SRII/IEEE, San Jose. pp 411–415
22. Somu N, Kirthivasan K, Shankar Sriram VS (2017) A computational model for ranking cloud service providers using hypergraph based techniques. Futur Gener Comput Syst 68:14–30. https://doi.org/10.1016/j.future.2016.08.014
23. Emrouznejad A (2005) Measurement efficiency and productivity in sas/or. Comput Oper Res 32(7):1665–1683
24. Charnes A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision making units. Eur J Oper Res 2(6):429–444
25. Khezrimotlagh D, Chen Y (2018) Decision Making and Performance Evaluation Using Data Envelopment Analysis, International Series in Operations Research & Management Science, 1st ed. Springer, Cham
26. Holland JH (1975) Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Bradford Books
27. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, University of Michigan
28. André L, Parpinelli R (2015) The multiple knapsack problem approached by a binary differential evolution algorithm with adaptive parameters. Polibits 51:47–54
29. Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution - An updated survey. Swarm Evol Comput 27:1–30
30. Krause J, Lopes HS (2013) A comparison of differential evolution algorithm with binary and continuous encoding for the MKP. In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence. IEEE, Recife. pp 381–387
31. Sundareswaran S, Squicciarin A, Lin D (2012) A brokerage-based approach for cloud service selection. In: 2012 IEEE Fifth International Conference on Cloud Computing. IEEE, Honolulu. pp 558–565
32. Garg SK, Versteeg S, Buyya R (2011) SMICloud: A framework for comparing and ranking cloud services. In: Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC 2011). IEEE, Melbourne. pp 210–218
33. Saaty TL (1990) How to make a decision: The analytic hierarchy process. Eur J Oper Res 48:9–26
34. Sun L, Dong H, Hussain OK, Hussain FK, Liu AX (2019) A framework of cloud service selection with criteria interactions. Futur Gener Comput Syst 94:749–764
35. Chen W, Goh M, Zou Y (2018) Logistics provider selection for omni-channel environment with fuzzy axiomatic design and extended regret theory. Appl Soft Comput 71:353–363
36. Shirur S, Swamy A (2015) A cloud service measure index framework to evaluate efficient candidate with ranked technology. Int J Sci Res 4(3):1957–1961
37. Hogben G, Pannetrat A (2013) Mutant apples: A critical examination of cloud sla availability definitions. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, vol. 1. IEEE Computer Society, Los Alamitos. pp 379–386
38. Wagle S, Guzek M, Bouvry P, Bisdorff R (2015) An evaluation model for selecting cloud services from commercially available cloud providers. In: 7th International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, Vancouver. pp 107–114
39. Patra SS, Jena S, Mund GB, Gourisaria MK, Gupta JK (2021) Meta-Heuristic Algorithms for Best IoT Cloud Service Platform Selection. In: Integration of Cloud Computing with Internet Of Things, chap 17. John Wiley & Sons, Ltd, New York City. pp 299–318
40. Mohamed AM, Abdelsalam HM (2020) A multicriteria optimization model for cloud service provider selection in multicloud environments. Softw Pract Exp 50(6):925–947
41. Hwang C-L, Yoon K (1981) Methods for multiple attribute decision making. In: Multiple Attribute Decision Making. Springer, USA. pp 58–191
42. Jatoth C, Gangadharan GR, Fiore U, Buyya R (2019) SELCLOUD: a hybrid multi-criteria decision-making model for selection of cloud services. Soft Comput 23(13):4701–4715
43. Kore NB, Ravi K, Patil SB (2017) A simplified description of FUZZY TOPSIS method for multi criteria decision making. Int Res J Eng Technol (IRJET) 4:2047–2050
44. Saaty TL (1996) Decision Making with Dependence and Feedback: The Analytic Network Process. RWS Publications, Pittsburgh
45. Al-Faifi A, Song B, Hassan MM, Alamri A, Gumaei A (2019) A hybrid multi criteria decision method for cloud service selection from smart data. Futur Gener Comput Syst 93:43–57
46. Mirjalili S (2019) Introduction to Evolutionary Single-Objective Optimisation. In: Evolutionary Algorithms and Neural Networks: Theory and Applications, Studies in Computational Intelligence. Springer, Cham. pp 3–14

47. Wilk MB, Shapiro S (1968) The joint assessment of normality of several independent samples. Technometrics 10(4):825–839
48. Dunn OJ (1964) Multiple comparisons using rank sums. Technometrics 6(3):241–252

Publisher's Note