A Tabu Search Algorithm for the Location of Data Centers and Software Components in Green Cloud Computing Networks

Federico Larumbe, *Member*, *IEEE*, and Brunilde Sansò, *Member*, *IEEE*

Abstract—The ubiquity of cloud applications requires the meticulous design of cloud networks with high quality of service, low costs, and low CO_2 emissions. This paper presents a planning problem and an extremely efficient tabu search heuristic for optimizing the locations of cloud data centers and software components while simultaneously finding the information routing and network link capacities. The objectives are to optimize the network performance, the CO_2 emissions, the capital expenditures (CAPEX), and the operational expenditures (OPEX). The problem is modeled using a mixed-integer programming model and solved with both an optimization solver and a tabu search heuristic. A case study of a web search engine is presented to explain and optimize the different aspects, showing how planners can use the model to direct the optimization and find the best solutions. The efficiency of the tabu search algorithm is presented for networks with up to 500 access nodes and 1,000 potential data center locations distributed around the globe.

Index Terms—Application component placement, cloud computing, cloud federation, data center location, energy efficiency, environmental impact, green networking, network planning, tabu search, virtual network embedding

1 Introduction

With the evolution of the Internet, we are witnessing the birth of an increasing number of applications that rely on the network; what was previously executed on the user's computers as stand-alone programs has been redesigned to be executed on servers with permanent connections to the Internet, making the information available from any device that has network access. Instead of buying a copy of a program, users can now pay to obtain access to it through the network, which is one of the models of cloud computing [1], software as a service (SaaS). The continuous growth of Internet bandwidth has also given rise to new multimedia applications, such as social networks and video over the Internet; and to complete this new paradigm, mobile platforms provide the ubiquity of information that allows people to stay connected.

Service providers may own servers and data centers or, alternatively, may contract infrastructure providers that use economies of scale to offer access to servers as a service in the cloud computing model, i.e., infrastructure as a service (IaaS). As users become more dependent on cloud services and mobile platforms increasing the ubiquity of the cloud, the quality of service becomes increasingly important. A fundamental metric that defines the quality of service is the delay of the information as it travels between the user computers and the servers, and between the servers themselves.

Along with the quality of service and the costs, the energy consumption and the CO_2 emissions are fundamental considerations in regard to planning cloud computing networks. Data centers consumed 1.5 percent of the energy in the US in 2006, which is equivalent to the amount of energy consumed by 5.8 million households [2]. Each type of energy, such as wind, hydroelectric, nuclear, diesel, and coal, introduces a different amount of CO_2 into the environment, and each potential data center location will be provided by one or more of those energy sources. Thus, choosing data center locations supplied by green energy sources can greatly reduce environmental pollution.

In this study, we solve the problem of designing a cloud computing network by answering the following questions:

- What potential data centers should be used in the network?
- Which data center should host each of the software components of the cloud applications?
- How many servers will be hosted at each data center?
- How will the information be routed through the network?
- What are the link capacities required to carry that information?

All these questions are interrelated and have an impact on the quality of service, energy consumption, cost, and pollution. However, in practice, these questions are tackled separately and very often, in different time frames. In this paper, we propose the simultaneous decision of data center location, software component placement, and other planning elements to provide a comprehensive optimization framework for future reference. Once the

• *The authors are with École Polytechnique de Montréal and the Group for Research in Decision Analysis (GERAD), 3000, chemin de la Côte-Sainte-Catherine, Office 4414, Montreal, QC H3T 2A7, Canada. E-mail: {federico.larumbe, brunilde.sanso}@polytml.ca.*

Manuscript received 1 Mar. 2013; revised 8 May 2013; accepted 2 June 2013; published online 16 July 2013.
Recommended for acceptance by R. Bianchini.
For information on obtaining reprints of this article, please send e-mail to: tcc@computer.org, and reference IEEECS Log Number TCC-2013-03-0039.
Digital Object Identifier no. 10.1109/TCC.2013.2.

network is designed and in operation, complementary models can be used to manage the system in a dynamic way, and interesting research avenues are open in that direction [3], [4].

The criteria to choose the optimal solutions in our framework are embedded in a multiobjective function that allows planners to weight each attribute according to their priorities. The objective function is composed of the following metrics: traffic delay, energy consumption, CO_2 emissions, traffic cost, server cost, data center capital expenditures (CAPEX), and data center operational expenditures (OPEX).

The proposed problem is formalized as a mixed-integer linear programming (MILP) model and solved, first with AMPL-Cplex, and then with a very efficient tabu search heuristic. That heuristic is strongly needed because the numerous integer variables of the model produce very high execution times in a general optimization solver. In fact, AMPL-Cplex took up to 1 hour to solve small instances of 24 potential data center locations. The cases used for this study were networks with up to 500 access nodes and 1,000 potential data center locations, which corresponds to the size of the largest cloud networks currently available [5]. We show that the tabu search algorithm presented in this paper achieved very small optimality gaps, and the execution time ranged from some milliseconds to less than 10 minutes.

The framework presented in this paper helps understand each aspect of a cloud network in a formal way, and multiple actors may be interested in having optimal solutions to this problem. One actor may be an infrastructure provider that needs to deploy or extend a data center network used by service providers. Another actor may be a service provider that needs to choose the cloud data centers to deploy global distributed applications and to decide the number of servers to host in each one of them. Yet another actor may be organizations with private clouds who want to solve the whole network design in an integrated and optimal way.

The remainder of this paper is structured as follows: Section 2 presents a literature review of related problems and models. Section 3 describes the problem and proposes a MILP model. Section 4 specifies the tabu search heuristic that was developed to solve large problems. Section 5 presents a case study of a search engine application that requires data centers and software components to be placed around the world. Section 6 shows optimal solutions for the case study and analyzes how the multiple objectives interact. This section also shows the optimality gap of the tabu search heuristic and the execution time depending on the instance sizes. Finally, the conclusions are presented in Section 7.

2 Related Work

The problems of facility location, such as capacitated facility location [14], [15], [16], originate in operations research, and extensive literature exists on them [17]. As surveyed in [4], some articles on data center location were recently presented [6], [7], [8], [10]. Table 1 summarizes the problems of data center location and provisioning closest to our work. The authors in [6] considered access nodes that aggregate user traffic, and each one of them must be assigned to a primary data center and to a backup data center. The objective of the proposed linear programming model was to minimize the delay between users and data centers. In [7], a model that minimizes the total cost of traffic, servers, and data centers was presented. Potential locations in the US were studied, considering energy and land costs. That approach is the closest to the model presented in this paper because cost, energy, delay, and CO_2 were also considered. The main difference is that our model takes into account distributed applications with traffic between user devices and servers and between the servers themselves, which has an impact on the routing and capacity assignment of the backbone network. Furthermore, our approach includes a multicriteria objective function allowing planners to minimize delay, CO_2 emissions, as well as costs. Regarding the resolution method, [7] proposed a simulated annealing heuristic combined with a linear programming model. Dong et al. [8] focused on the minimization of optical and IP router power consumption as functions of the data center location through linear programming models. The proposed strategy was to place data centers close to energy sources to reduce power losses in the grid. Covas et al. [10] proposed a method to select potential data center locations by comparing multiple criteria, such as energy cost, security, availability of telecommunications networks, and labor. The delay between users and data centers was not taken into account in that model.

With respect to the application location, Stone [11] was the first to model the software component placement considering the information traffic between the components. The proposed resolution method based on a maximum flow algorithm was restricted to the case of two processors. Rao et al.

TABLE 1
Data Center Location and Provisioning Approaches

Article	Objective	Decisions	Resolution
[6]	- Net. delay	- DC location - AN to DC assig. - AN to backup DC	MILP model + solver
[7]	- Cost	- DC location - AN to DC assig.	Simulated anneal. + LP
[8]	- Energy	- DC location - AN to DC assig. - Routing	MILP model + solver
[9]	- Net. delay - Cost - Pollution	- DC location - AN to DC assig. - App. location - Link capacities	MILP model + solver
[10]	- Risk - Social - Cost - Pollution	- DC location	ELECTRE TRI
[11]	- N. delay	- App. placement	Max. flow
[12]	- Energy cost	- Workload assig.	Minimum cost flow
[3]	- Cost - Energy - Resp. time	- App. placement - Resource alloc. - Workload migration	CloudSim
[13]	- Cost - Load balance	- Virtual network embedding	LP + multi-com. flow

[12] served web requests from multiple data centers to minimize the total energy cost by changing the workload assignment, depending on the energy price, and guaranteeing the quality of service. The problem was linearized and then reduced to a minimum cost flow problem. Buyya et al. [3] studied dynamic provisioning of applications from multiple cloud vendors and did a performance study with the CloudSim toolkit. That paper shows that the cloud federation model yields important benefits in quality of service and costs.

Because the application layer and the network layer are two separate graphs and because the problem includes mapping the application layer on top of the network layer, a related problem is virtual network embedding, which maps a virtual network with virtual routers and virtual links on top of a physical network. Each virtual node must be located on a physical node, and each virtual link must be routed through a physical path. Chowdhury et al. [13] proposed an integer programming model and solved it using linear relaxation and rounding techniques. With respect to defining the routing and the link capacities in multilevel networks, the authors of [18] presented a survey on the models and resolution approaches for the network synthesis problem.

Regarding our previous work, the authors of [19] presented a model that simultaneously solved the location of data centers and software components with the routing. This model was an initial model that minimized the network delay but did not consider the energy and the pollution aspects. Larumbe and Sansò [9] solved a model that considered energy and pollution aspects through an AMPL-Cplex program by adding the search for optimal link capacities. The new aspect caused us to discard the assumption of Poisson traffic, and the delay of each link was taken from real measurements.

The present study includes the following original contributions:

1. A highly efficient tabu search heuristic to minimize delay, cost, energy, and CO_2 emissions in the jointly determined location of data centers and software components, routing, and link dimensions.
2. Calculation of the energy consumed by switches and routers, in addition to the servers.
3. Tradeoff analysis between the multiple objectives.

3 Problem Description

The problem is to define the location of data centers and software components, the number of servers in each data center, the information routing, and the capacity of each link in the network. The design objective is to minimize the network delay, the cost, the energy consumption, and the CO_2 emissions. This problem is formalized as an MILP model that minimizes a multicriteria objective function.

The problem setting is an organization that offers cloud services to its users through a data network. The organization must optimize the network design to provide the best possible quality of service while reducing costs, energy consumption, and CO_2 emissions. The applications that will be executed in the network are represented as a graph

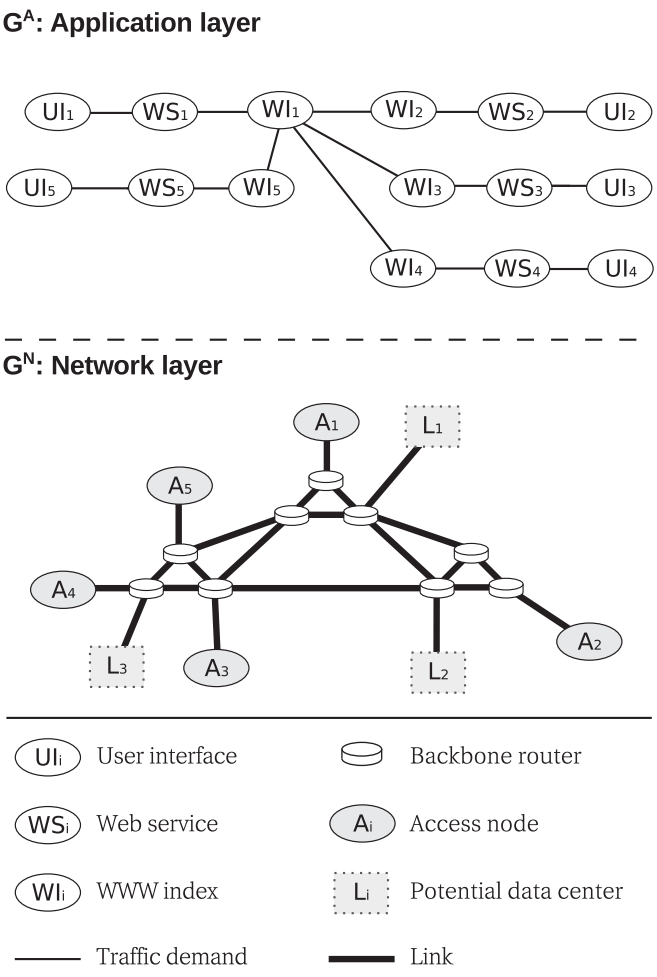

Fig. 1. Network and application layers.

$G^A(V^A, A^A)$, where each node is a software component and each arc is a traffic demand from one software component to another. Each software component may be executed on a user computer or on a server. A single software component can even span multiple servers to achieve its purpose. The applications will be deployed on a computer network $G^N(V^N, A^N)$ composed of access nodes $\mathcal{A}$, backbone routers $\mathcal{B}$, existing data centers $\mathcal{D}$, and possible locations to open new data centers $\mathcal{L}$. The set $\overline{\mathcal{D}}$ denotes all the potential data centers, i.e., the union of $\mathcal{L}$ and $\mathcal{D}$. Each access node represents an aggregation of users, which may be an Internet Service Provider (ISP), a mobile company, a city, a region, or a country, depending on the case. The users will access the cloud services hosted in the data centers. The nodes in G^N are connected through the backbone links, typically by optical fiber. The link capacities, measured in number of channels, are decision variables. Because not every node is suitable to host every software component, P is the set of possible assignments of software components in V^A to network nodes in V^N.

Fig. 1 shows an example of a web search engine that must be deployed on a data center network. The application is composed of user interfaces, web servers, and World Wide Web indexes. Each access node aggregates the users of a whole city. There is one user interface for each access node, representing the client component that is executed on

the user computers of that access node. There is also a web service that answers requests to that access node. The same web service may require multiple servers to handle all the requests in an efficient way. The web service makes index queries to the web index, where the associations between keywords and pages are stored. Each web index may also require multiple servers with multiple hard disks to store the entire index. In the example, WI_1 is the master index and all other indexes are replicas of it. In this case, the problem is to choose a subset of the potential data centers and locate each web service and each web index in one of the data centers. The following sections provide the parameters and decision variables of the problem.

3.1 Network Traffic

The software components send and receive information between them, and this information is divided into packets that will be sent through network paths. Each step in a path adds processing, waiting, transmission, and propagation delay. While the traffic in a link is under a maximal allowed utilization, the link delay is bound by a constant. For example, a specific link could present packet delays of less than 1 millisecond while its traffic is less than 70 percent of its capacity. In the same way, the delay incurred by the packets in a path is bound by the sum of the link delay bounds. The strategy in this problem is to locate software components that exchange traffic in close nodes in terms of delay such that the delay is minimized and the quality of service is increased. That implies, for instance, that web service components will be placed near the users to reduce the round trip time. The following is the network traffic notation:

- b_d^1 Throughput of demand $d \in A^A$ in peak time (in bps).
- b^2 Capacity per link channel (in bps), for example, 10 Gbps.
- u_e Maximal utilization ratio per channel in link $e \in A^N$ (e.g., 0.70).
- t_e Delay bound for link $e \in A^N$ (in seconds per packet) while its traffic is less than the allowed utilization.

3.2 Servers

Cloud computing applications have software components that are executed either on the user devices or on servers. In fact, a single software component may be executed on multiple servers. For instance, in a web search engine, multiple web servers answer HTTP requests. Each server has capacity allocated for each one of its resources, including network bandwidth, CPU, RAM, and hard disks. In this model, we consider that the number of replicas of each software component, i.e., the number of required servers, is a parameter that is estimated in advance and strictly depends on the nature of the application. The server-related notation is as follows:

- K Set of resources that each server has, which include 1 = CPU, 2 = network card, 3 = RAM, and 4 = storage.
- r_{ip}^1 Average consumption of resource $p \in K$ required by a copy of component $i \in V^A$, in the correspondent units, for example, GHz, Gbps, or GB.
- r_{ip}^2 Peak consumption of resource $p \in K$ required by a copy of component $i \in V^A$.
- r_i^3 Number of replicas of software component $i \in V^A$.
- r_p^4 Capacity of resource $p \in K$ on a server.
- r_k^5 Capacity of data center $k \in \overline{\mathcal{D}}$ in number of servers.
- r^6 Number of servers per local area network (LAN) switch.

The servers and LAN switches in a data center are organized in racks, and each data center has space for a maximal number of racks. Each switch can connect a maximal number of servers, which will determine the number of required switches. In summary, each software component requires a specific number of servers. The model solution defines what software components each data center hosts, and hence the number of servers required in each data center. From that the number of LAN switches needed for each data center in a particular solution can be calculated. The sum of servers and switches must be less than the data center capacity.

3.3 Energy

An important objective of cloud computing is to minimize the power consumption of the entire network. The power consumption is primarily from the servers, the network switches, the routers, the data center cooling, and the power provisioning equipment. The server power consumption depends on its utilization, having a minimal idle power and reaching a maximal power consumption when the resources are fully utilized [20]. The following is the energy-related notation:

- w^1 The power consumption of a server as a function of the utilization of each resource (in W).
- w^2 Network switch power consumption (in W).
- w^3 Router port power consumption (in W).
- w_k^4 Maximum power capacity of data center $k \in \overline{\mathcal{D}}$ (in W) for IT equipment.
- w_k^5 Power usage effectiveness (PUE) of data center $k \in \overline{\mathcal{D}}$ (ratio of total power/IT equipment power).
- w_k^6 CO_2 emissions in data center $k \in \overline{\mathcal{D}}$ (in g/kWh).
- w_i^7 Average power consumption of a server with a replica of software component $i \in V^A$.
- w_i^8 Peak power consumption of a server with a replica of software component $i \in V^A$.

CO_2 emissions depend on how the energy is produced because each type of energy generates a certain amount of CO_2 per kWh. For example, hydroelectric energy emits 10 g of CO_2/kWh, and diesel emits 778 g/kWh [21]. Because the power consumption, w_1, depends on each piece of hardware of the server used, the values w_i^7 and w_i^8 should be determined experimentally.

3.4 Cost

The following are the parameters that define the cost of each cloud element, the cost of the energy, and the penalties. Each element has a different amortization period; data centers are amortized between 12 and 15 years, and servers are amortized between 3 and 4 years [22]. Each cost below is the result of the total element cost divided by its amortization year:

- c_d^1 Penalty for delay in demand $d \in A^A$ (in \$/(ms/packet)/year).
- c^2 Cost of a LAN switch (in \$/year).
- c^3 Cost of a router port (in \$/year).
- c_e^4 Cost of a channel in link $e \in A^N$ (in \$/year).
- c^5 Cost of a server (in \$/year).
- c_k^6 CAPEX for using data center $k \in \overline{\mathcal{D}}$ (in \$/year).
- c_k^7 OPEX for using data center $k \in \overline{\mathcal{D}}$ (in \$/year).
- c_k^8 Electricity cost in data center $k \in \overline{\mathcal{D}}$ (in \$/MWh).
- c_k^9 Penalty for emitting CO_2 in data center $k \in \overline{\mathcal{D}}$ (in \$/ton).

A penalty is applied to the delay for exchanging information between the software components. Some traffic demands may suffer from the delay more than others. For example, a traffic demand could be used for copying files from a data center to calculate statistics once every day. That traffic demand might not be affected by an additional delay; therefore, its delay penalty is low. Traffic demands from the users doing search requests require a very short delay, and in that case, the penalty is high; this penalty represents a loss in revenue because users may change to the competitors.

The pollution penalty term comes from two sources. One is the country regulations that apply a carbon tax to the energy consumption. In that case, the electricity cost c_k^8 is set as the base cost without that tax in data center k, and c_k^9 includes the carbon tax. There is a second source of CO_2 penalty that may be even higher: the losses in corporate image. Nowadays, customers reward companies that make efforts to reduce pollution and its quantification should be included in the CO_2 penalty.

3.5 Decision Variables

- x_{ik} 1 if software component $i \in V^A$ is located in a network node $k \in V^N$; 0 otherwise.
- y_k 1 if node $k \in V^N$ hosts one or more software components; 0 otherwise.
- z_k Number of servers hosted in data center $k \in \overline{\mathcal{D}}$.
- s_k Number of LAN switches in data center $k \in \overline{\mathcal{D}}$.
- f_{de} Amount of demand $d \in A^A$ traffic carried by link $e \in A^N$ (in bps).
- b_e Number of channels, for example, wavelengths or fibers, in link $e \in A^N$.
- w_k Average power consumption of IT equipment in node $k \in V^N$.

The problem is modeled as a multicommodity flow problem, in which each flow variable contains the amount of traffic per demand, per link. These variables are subsequently related to the location variables of the software components because moving a software component moves its traffic demands.

3.6 Objective Function

Each criterion to optimize is included in a term of the objective function. The following are the formulas to calculate each cost. Note that some of these formulas contain constants to change units, for example, seconds to milliseconds, kWh to MWh, or Gbps to bps.

The flow variables that define how demands are routed are used to calculate each demand delay that is the end-to-end delay incurred by packets between software components. For instance, the delay of the demand between a user interface and a web service will be included in this computation. Then, each demand delay is multiplied by its delay penalty, and the total delay penalty is

$$c^D = \sum_{d \in A^A} c_d^1 \, 1{,}000 \sum_{e \in A^N} t_e \frac{f_{de}}{b_d^1}.$$

The network traffic cost is composed of the costs of the LAN switches, the router ports, and the links. Each pair of link channels, uplink and downlink, of a link is connected to two router ports, one in each end of the link. Therefore, the number of router ports required for a link is equal to the number of channels:

$$c^T = \sum_{k \in V^N} c^2 s_k + \sum_{e \in A^N} \left(c^3 + c_e^4\right) b_e.$$

The following is the cost of all servers in the network:

$$c^S = \sum_{k \in V^N} c^5 z_k.$$

The CAPEX of each data center includes the land cost, the cost of building the facility, the cooling artifacts, and the power provisioning instruments. The total CAPEX of a solution is calculated as follows:

$$c^C = \sum_{k \in V^N} c_k^6 y_k.$$

The data center OPEX includes the human labor, i.e., technicians and security guards, as well as the administrative costs. We exclude the electricity because we consider it in an explicit term:

$$c^O = \sum_{k \in V^N} c_k^7 y_k.$$

The energy cost for a whole year is calculated as a function of the data center energy consumption, its PUE, and the price of energy as follows:

$$c^E = \sum_{k \in V^N} c_k^8 (w_k \cdot 365 \cdot 24\text{h} \cdot 10^{-6}) w_k^5.$$

In a similar way, the environmental cost is proportional to the energy consumed in each data center, the ratio of CO_2 emissions, and the CO_2 emission penalty:

$$c^{\mathrm{CO}_2} = \sum_{k \in V^N} c_k^9 \left(w_k^6 \cdot 10^{-6}\right) \left(w_k \cdot 365 \cdot 24 \text{ h} \cdot 10^{-3}\right) w_k^5.$$

Finally, each cost in the objective function is weighted by a parameter, allowing planners to modify the priorities. The objective function is defined as follows:

$$z = \alpha c^D + \beta c^T + \gamma c^S + \delta c^C + \epsilon c^O + \eta c^E + \psi c^{\mathrm{CO}_2}. \quad (1)$$

3.7 Constraints

- *Location of software components and data centers:*

$$\sum_{(i,k) \in P} x_{ik} = 1 \qquad \forall \, i \in V^A, \quad (2)$$

$$\sum_{\substack{k \in V^N / \\ (i,k) \notin P}} x_{ik} = 0 \qquad \forall\, i \in V^A, \tag{3}$$

$$x_{ik} \leq y_k \qquad \forall\, (i,k) \in P. \tag{4}$$

Equations (2) and (3) state that each software component must be placed in one node, either access node, or data center. They also state that each assignment of a software component to a node must belong to the set of possible assignments P. Constraint (4) specifies that nodes containing software components are open.

• *Flow conservation:*

$$\sum_{e \in \Gamma_k^-} f_{de} + x_{ik} b_d^1 = \sum_{e \in \Gamma_k^+} f_{de} + x_{jk} b_d^1 \qquad \forall\, d = (i,j) \in A^A, \quad k \in V^N. \tag{5}$$

This constraint relates the software component location to the routing. It associates the application and the network layers. For each demand from software component $i \in V^A$ to software component $j \in V^A$ located in nodes $k_i, k_j \in V^N$, respectively, the traffic b_d^1 enters the network in k_i and exits in k_j. In each intermediate node k from k_i to k_j, flow is conserved.

• *Link capacity:*

$$\sum_{d \in A^A} f_{de} \leq u_e\, b^2\, b_e \qquad \forall\, e \in A^N, \tag{6}$$

$$b_{ij} = b_{ji} \qquad \forall\, (i,j) \in A^N. \tag{7}$$

Constraint (6) defines the link capacity variable b_e (in number of channels), requiring that the traffic be less than the maximal utilization threshold, $u_e\, b^2\, b_e$. Because full-duplex links are assumed, (7) states that the capacity in both directions of the link is the same.

• *Number of servers:*

$$z_k = \sum_{i \in V^A} r_i^3 x_{ik} \qquad \forall\, k \in \overline{\mathcal{D}}. \tag{8}$$

The number of servers required in each data center is the sum of the servers required by the software components that are located there.

• *Number of switches:*

$$r^6 s_k \geq z_k \qquad \forall\, k \in \overline{\mathcal{D}}. \tag{9}$$

Because each LAN switch can connect a maximum number of servers r^6, constraint (9) defines the minimal number of switches, s_k, required to connect the z_k servers in data center k.

• *Data center capacity:*

$$z_k + s_k \leq r_k^5 \qquad \forall\, k \in \overline{\mathcal{D}}. \tag{10}$$

The number of servers and LAN switches must be less than the maximal capacity of the data center.

• *IT equipment average power consumption:*

$$w_k = w^2 s_k + \sum_{e \in \Gamma_k^-} w^3 b_e + \sum_{i \in V^A} w_i^7 r_i^3 x_{ik} \quad \forall\, k \in V^N. \tag{11}$$

Equation (11) sums the average amount of power consumed by the server switches, the router ports, and the servers in a node. This value is used to calculate the cost of electricity used in the solution.

• *Data center maximal peak power:*

$$w^2 s_k + \sum_{e \in \Gamma_k^-} w^3 b_e + \sum_{i \in V^A} w_i^8 r_i^3 x_{ik} \leq w_k^4 \quad \forall k \in \overline{\mathcal{D}}. \tag{12}$$

The peak power consumption of data center k is calculated using the same methodology as for the previous constraint. This value has a fixed limit in each data center because of the electricity available and the power provisioning equipment in the data center.

• *Domain of variables:*

$$x_{ik} \in \{0,1\} \qquad \forall\, i \in V^A, k \in V^N, \tag{13}$$

$$y_k \in \{0,1\} \qquad \forall\, k \in V^N, \tag{14}$$

$$z_k \in \mathbb{Z}_{\geq 0} \qquad \forall\, k \in \overline{\mathcal{D}}, \tag{15}$$

$$s_k \in \mathbb{Z}_{\geq 0} \qquad \forall\, k \in \overline{\mathcal{D}}, \tag{16}$$

$$b_e \in \mathbb{Z}_{\geq 0} \qquad \forall\, e \in A^N, \tag{17}$$

$$f_{de} \in \mathbb{R}_{\geq 0} \qquad \forall\, d \in A^A, e \in A^N, \tag{18}$$

$$w_k \in \mathbb{R}_{\geq 0} \qquad \forall\, k \in V^N. \tag{19}$$

Each variable domain is defined. All the variables are nonnegative. The location variables may take the values 0 or 1. The number of servers, LAN switches, and network channels are integers. The flow and the data center power consumption variables are real numbers.

4 Resolution Approach

This section describes the tabu search heuristic [23], [24] proposed to solve the cloud network planning problem. The algorithm was programmed in C++ and compiled using gcc 4.5.1 with the optimization options activated. The quality of solutions obtained and the execution time are compared with the MILP model in AMPL with Cplex 12.4. Algorithm 1 presents the main structure of the proposed tabu search heuristic.

Algorithm 1. Tabu search for the cloud network planning problem.

function TABUSEARCH
  $M \leftarrow$ INITIALGREEDYSOLUTION
  Best $\leftarrow M$
  $L \leftarrow \{\}$
  $i \leftarrow 0$
  repeat
    Choose $(c,d) \in P - L$ that minimizes z when moving software component c to node d in M.
    Move c to d in M
    Add (c,d) to tabu list L for $\frac{\sqrt{|\mathcal{N}(\mathcal{M})|}}{2}$ iterations.
    $i \leftarrow i + 1$
    if $z(M) < z(Best)$ **then**
      Best $\leftarrow M$

```
            i ← 0
        end if
    until i = MAX_IT
    return Best
end function

function INITIALGREEDYSOLUTION
    M ← ∅
    for each component c in V^A do
        d ← x/(c, x) ∈ P and x is the node that least
increases z when placing c in x in M.
        M ← M ∪ {(c, d)}
    end for
    return M
end function
```

4.1 Solution Space

A solution of the proposed tabu search heuristic is a mapping of the software components to the network nodes, i.e., M: $V^A \rightarrow V^N$. In that solution M, each software component is placed at one of the network nodes. P specifies the set of possible assignments; hence, for each software component i, the pair $(i, M(i))$ must belong to P. Thus, the solution space of the proposed heuristic is the set of mappings, as follows:

$$\mathcal{S} = \left\{M : V^A \rightarrow V^N / (i, M(i)) \in P, \quad \forall i \in V^A\right\}.$$

Each solution M determines the location variables x_{ij} and satisfies constraints (2) and (3) in the model. From the software component placement defined by variables x_{ij}, all the other variables of data center location (y_j), data center capacities (s_k and z_k), routing (f_{de}), link capacities (b_e), and power consumption (w_k) can be calculated in a deterministic way as follows: The variables y_j of the data centers hosting the software components have a value of 1, as stated by constraint (4). Because the software component location is defined in M, the origins and destinations of the traffic demands are fixed. We use the shortest path between each pair of nodes of V^N for routing the demands and defining variables f_{de} that satisfy the flow conservation constraint (5). The distance considered in the shortest path algorithm is the delay bound t_e of each link. The routing defines the total amount of traffic carried by each link; therefore, we can calculate the minimal number of channels required for that link using the channel capacity, c_e, and the maximal allowed utilization ratio per channel, u_e, satisfying constraints (6) and (7). The numbers of servers, z_k, and switches, s_k, are also determined by the software components hosted in data center k through constraints (8) and (9). Variables w_k that correspond to the amount of power consumed by the IT equipment in data center k are calculated with the formula defined in (11). Finally, the terms of the objective function that correspond to a delay of traffic demands, c^D, network traffic c^T, servers c^S, CAPEX c^C, OPEX c^O, electricity c^E, and CO_2 emissions c^{CO_2}, are determined, and the value of the objective function is defined in (1). Thus, we see that the location of the software components determines all the variables in the model, and subsequently all the terms of the objective function.

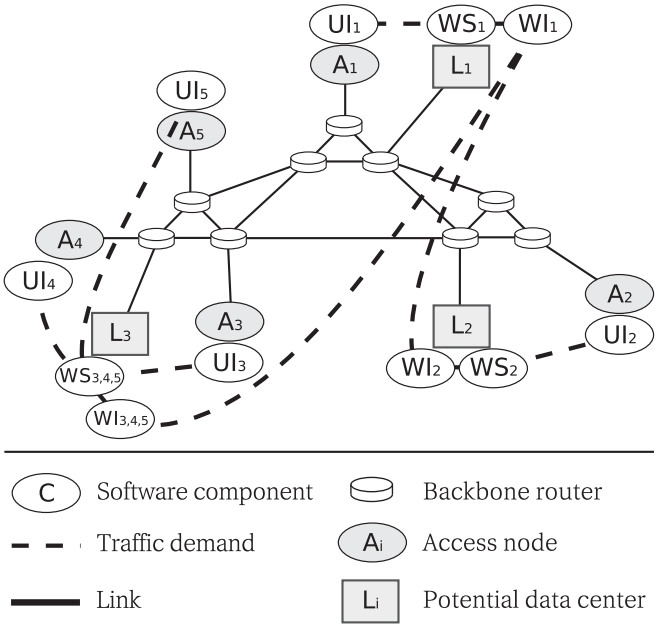

Fig. 2. Greedy solution for the web search engine example.

The solutions in $\mathcal{S}$ satisfy all the constraints except the data center capacity constraints (10) and (12). Instead of enforcing those constraints, one penalty term for each constraint is added to the objective function. The number of servers that exceeds the capacity in an overloaded data center is multiplied by a large constant, which is higher than all the other terms of the objective function. The same penalty is applied for power consumption that is higher than the data center limit. Thus, the tabu search will avoid those solutions because the objective function is much higher than in solutions that do not have overloaded data centers. The algorithm will, thus, accommodate corner cases, which have capacities that make feasible solutions difficult to find. In that case, the tabu search will search to reduce the overloaded data centers until a feasible solution is found.

4.2 Initial Solution

The greedy heuristic shown in Algorithm 1 locates one software component of V^A at a time. Each step chooses the node in V^N, where the objective function has the lowest value when the software component is placed in that node. Because the objective function includes delay, cost, energy, and CO_2, the node chosen will be the best in those aspects, according to the priorities defined. If the first priority is the delay, a web service will be hosted in the closest data center to its users. If it is the CO_2, then the data center with the cleanest energy will be chosen. As in every greedy heuristic, the placement is done in an arbitrary order, and that may be nonoptimal. The greedy algorithm provides a good starting solution that the tabu search improves to a near optimal solution.

Fig. 2 shows a greedy solution for the example of a web search engine. In the first five steps, each of the user interfaces UI_i is located in its correspondent node A_i, defined in the set P. In those steps, there is no choice because each UI_i has a unique possible assignment in P. Until that moment, the objective function remains zero because those components do not require servers and do

not have direct traffic demands among them. Then, the location of the web service WS_1 that serves the requests from UI_1 is the node that increments the objective function the least. If all the data centers have the same costs and capacities and all the links have the same delay bound, the best possible data center for WS_1 is L, as shown in Fig. 2. That is because UI_1 is located in A_1 and there is a traffic demand between UI_1 and WS_1 that contributes a delay that is part of the objective function. The shortest path between A_1 and L_1 has three links, and the shortest paths between A_1 and the other data center locations L_2 and L_3 have more links, and hence, more delay, given that all the links have the same delay. In the same way, WS_2, WS_3, WS_4, and WS_5 are located in the data center locations that are closest to the respective user interface. Given that the web indexes WI_i exchange a high amount of traffic with the corresponding web services WS_i, the locations that have a smaller increment in the objective function are those where the web index is in the same data center as the web service.

4.3 Neighborhood Relation

Each iteration of the tabu search moves from the current solution to a neighbor solution. One solution M is neighbor of a solution M' if and only if they differ in the location of only one software component i. The neighborhood of solution M is the following set:

$$\mathcal{N}(M) = \{M' \in \mathcal{S} / \exists!\ i \in V^A, M'(i) \neq M(i)\}.$$

Thus, one software component is moved from one node to another in each iteration of the tabu search. The chosen movement is that with the lowest objective function. The calculation of the objective function for each neighbor can be time consuming for large instances because the number of possible movements is high. However, the key of the tabu search heuristic is that each iteration must be performed as rapidly as possible. Calculating the objective function in an incremental way achieves that efficiency. Instead of performing the entire calculation for each neighbor, the difference between the current solution and the neighbor solution is calculated. That is, the amount that the objective function would change if a component is moved is what is calculated. In other words, the difference of each objective function term is calculated. Coding this approach with data structures with constant access time is more complicated than the straightforward calculation of the objective function, but the execution time of each iteration is several orders of magnitude shorter when performed incrementally. This means that the tabu search heuristic moves very rapidly among the solutions.

Limiting the number of neighbors to analyze can also reduce the execution time of each iteration. Instead of looking at the entire neighborhood, the tabu search analyzes a random subset of them [25]. The number of neighbors to analyze is $4\sqrt{|N(M)|}$, a number that grows as the instance size increases but is less than a linear function. The multiplier 4 was adjusted through experiments, making a tradeoff between the execution time and the solution quality obtained in each iteration.

4.4 Tabu List

The risk of this type of heuristic is to reach a local optimum and keep returning to it iteration after iteration. A mechanism to avoid repeated solutions prevents that situation. When a software component i is moved to a node j, then the pair (i, j) is added to a tabu list, L, so that movement will not be performed again for a number of iterations. This mechanism effectively avoids local optimums and the searching process continues. The number of iterations to keep the movement in L is related to the neighborhood size and is equal to $\frac{\sqrt{|\mathcal{N}(\mathcal{M})|}}{2}$. This value was carefully adjusted to grow when the neighborhood is larger, but in a less degree than a linear function.

4.5 Stop Criterion

If the best solution found does not improve after *MAX_IT* iterations, then the algorithm stops. *MAX_IT* is defined as the sum of the number of nodes in the network and the number of software components, i.e., $|V^N| + |V^A|$.

5 Case Study

In this section, we study the deployment of a hypothetical web search engine on a network of data centers.

5.1 Network and Application Topology

The network nodes V^N include a set of access nodes $\mathcal{A}$, backbone routers $\mathcal{B}$, and locations for new data centers $\mathcal{L}$. The nodes are distributed among different locations around the globe. To do that, we considered the 500 largest cities in the world [26], the number of Internet users in each of them [27], and their geographic location [28]. For each city, we created an access node in $\mathcal{A} \in V^N$ that aggregates all the traffic of its users. The access nodes are connected through intermediate backbone routers. Finally, for each of those nodes—routers and access nodes—a potential data center location was created in $\mathcal{L}$. This method allows us to produce scenarios of different sizes, with up to 500 access nodes and 1,000 potential data center locations.

Regarding the application layer G^A, we used the web search engine topology shown in Fig. 1 for n access nodes. There is a user interface UI_i for each access node that aggregates every web client in that city. There is also a web service WS_i that answers the users' HTTP requests and makes queries to the web indexes WI_i. All the web indexes keep synchronized with a master web index. The decisions to make are which data centers will host each web service and web index, as well as all the network planning variables of the proposed model, to minimize the delay, the cost, energy consumption, and the CO_2 emissions.

5.2 Network Traffic

The total interdomain traffic of the Internet was 154 Tbps in March 2013, considering the estimation of 39 Tbps in July 2009 [29] and a 45.5 percent annual growth. The hypothetical web search engine analyzed manages 1 percent of the Internet traffic, that is 1.54 Tbps. That total amount of traffic was distributed among the access nodes in proportion to the number of Internet users in each city [27]. Those are the traffic demands between the user interfaces UI_i and the web services WS_i (b_d^1). The same value was used between each WS_i and its WI_i. The average packet size is 770 bytes (ζ), the capacity of each link channel is 10 Gbps (b^2), the maximal channel utilization is 70 percent (u_e), and the maximal link delay is 1 ms plus the propagation delay (t_e). The propagation delay of each link is calculated with the distance between the nodes divided by the speed of light over optical fiber cable (200,000 km/s).

Fig. 3. Network topology with 10 cities and 10 potential data centers.

5.3 Servers

The number of servers required is 200,000, also distributed in proportion to the number of users in each city. From the servers required by each city, 60 percent are for web services and 40 percent for the web index (r_i^3). Each data center can accommodate either 40, 80, 160, 1,000, 32,000, 64,000, or 128,000 IT elements. Each LAN switch can connect up to 40 servers (r_{ip}^6).

5.4 Power

The average power consumption of each server (w_i^7) was defined as 150 W, and the peak power consumption (w_i^8) as 300 W. Each LAN switch consumes 400 W (w^2), and each 10-Gbps router port consumes 20 W (w^3). The data center power capacity for IT equipment (w_k^4) is either 12 kW, 24 kW, 49 kW, 303 kW, 9.7 MW, 19.4 MW, or 38.7 MW, depending on the data center size. The PUE of the data centers is 1.08 (w_k^5), that is the power provisioning equipment, cooling artifacts, and any other power consumption are 8 percent of the power consumed by the IT equipment. The type of energy that supplies a data center generates a specific amount of CO_2 emissions (w_k^6): offshore wind produces 9 g/kWh, hydroelectric 10 g/kWh, geothermal 38 g/kWh, nuclear 66 g/kWh, natural gas 443 g/kWh, diesel 778 g/kWh, and coal 960 g/kWh [21]. The amount of CO_2 produced by each data center in this case was randomly chosen from one of those values.

5.5 Cost

The delay penalty was defined as \$10,000/(ms/packet)/year for the traffic demands between the UI_i, WS_i, and WI_i, and \$10/(ms/packet)/year for the delay between the WI_i themselves (c_d^1). That means that the delay between UI_i, WS_i, and WI_i is much more important than the delay between the WI_i themselves. The cost of each LAN switch is \$1,000/year (c^2). Each 10-Gbps router port costs \$100/year (c^3) and each directed 10-Gbps link channel, \$60,000/year (c_e^4) [30]. The cost of each server is \$4,000 [31] and it is amortized in four years, resulting in \$1,000/year (c^5). The data center CAPEX is \$12/W amortized in 12 years giving a total of \$1/W/year (c_k^6), and the OPEX is \$0.24/W/year (c_k^7) excluding electricity that was considered as an explicit term of the cost function [22]. Electricity prices were uniformly distributed between \$30/MWh and \$70/MWh (c_k^8) [32]. Finally, the penalty for emitting CO_2 was defined as \$1,000/ton (c_k^9).

TABLE 2
Solutions for 10 Cities with up to 10 Data Centers

Solution:	A	B	C
Delay multiplier	1	1	1
CO_2 multiplier	1	1000	1
Cost multiplier	1	1	1000
Delay	**4.4 ms**	12.2 ms	21.2 ms
Total power	34.6 MW	34.6 MW	34.6 MW
CO_2 emissions	74,695 ton	**3,561 ton**	41,181 ton
Data centers	10	4	3
Delay penalty	**856.2 M**	2,563.5 M	5,336.2 M
CO_2 penalty	74.7 M	**3.6 M**	41.2 M
Traffic cost	52.4 M	78.0 M	77.0 M
Server cost	200.0 M	200.0 M	200.0 M
CAPEX	329.1 M	135.5 M	77.4 M
OPEX	79.0 M	32.5 M	18.6 M
Energy cost	17.7 M	16.8 M	17.5 M
Total cost	678.1 M	462.8 M	**390.5 M**
Execution time	0.7 s	0.4 s	178 s

6 Results

6.1 Tradeoff between Objectives

As a first step, we executed the AMPL-Cplex model in a case of 10 cities to analyze how the different aspects of the model interact. This case will also demonstrate how planners can use the framework with multiple objectives. The network has 10 access nodes and 10 potential data center locations. In Fig. 3, the access nodes are shown as black circles and the potential data center locations are shown as white boxes. The application graph is a web search engine with a user interface, a web service, and a web index for each access node, and the topology of traffic demands as discussed in Section 5.

The case was solved in three different contexts, depending on the optimization priorities: A) delay oriented, B) pollution oriented, and C) cost oriented. Table 2 presents the results of the three solutions, and Figs. 4, 5, and 6 show the data centers selected in each context. All the tests were executed on an Intel i7 with four cores at 2.8 GHz:

- In Solution A, 10 data centers are open, i.e., all of them are used to host servers and software components. That optimization initially favors the delay because the delay penalty is the highest term in the objective function at \$ 856.2 M. In this case, all the access nodes have a close data center that serves their requests,

Fig. 4. Solution A. Delay minimization.

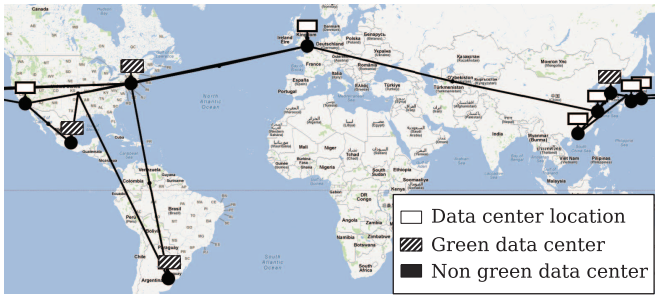

Fig. 5. Solution B. Pollution minimization.

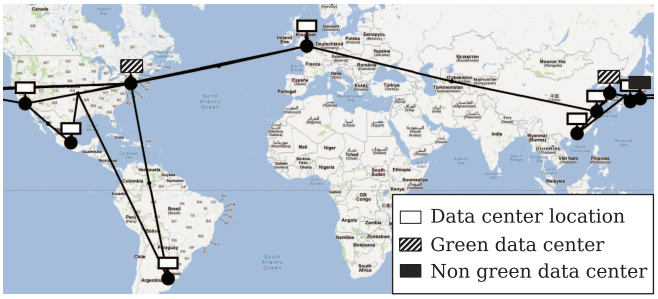

Fig. 6. Solution C. Cost minimization.

and the average delay is 4.4 ms. The total power consumed by all the data centers is 34.6 MW. The data centers are powered with all types of energy, in particular, four of them have high pollution emission ratios, between 443 and 960 grams of CO_2 per kWh. The total CO_2 emission is 74,695 ton per year, and the total cost is $ 678.1 M per year.

- Solution B is obtained by augmenting the pollution multiplier in the objective function (ψ) to 1,000. That makes the CO_2 penalty term greater than the delay penalty term. In this case, the optimal solution is to open four data centers. Three of the data centers are powered with hydroelectric energy producing 10 grams of CO_2 per kWh, and one of the centers is powered with geothermal energy introducing 38 grams of CO_2 per kWh. The total CO_2 emissions are 3,561 ton per year, 21 times smaller than Solution A. The CO_2 penalty decreases with the same ratio. The energy consumption remains the same because most of it depends on the number of servers used and the PUE of the data centers, and in these cases, both remain constant. The total cost is 68 percent of Solution A because fewer data centers are used; therefore, the CAPEX and the OPEX are reduced. The average delay is 12.2 ms, which is 2.8 times greater than Solution A.
- Solution C is obtained by setting the cost multipliers to 1,000, making the total cost the first priority to optimize. The number of open data centers is 3, and they have the minimal capacity to accommodate all the servers. The total cost is reduced to $ 390.5 M, the total CO_2 emissions are 41,181 tons per year, and the average delay is 21.2 ms.

From the three solutions, Solution B appears to be a good compromise between the network performance, pollution, and cost because the four chosen data centers imply a short delay with an acceptable cost and very low CO_2 emissions. Using this framework, planners can evaluate multiple alternative solutions and analyze the tradeoffs between them. Furthermore, once the optimal values for each aspect are known, additional constraints can be added to evaluate solutions around the optimal values. The execution time to find these solutions was between 0.7 and 178 s. We will see how the Cplex execution time increases very quickly as the network grows and how the tabu search performs very well for small and large cases.

6.2 Comparison between Tabu Search and Cplex

We executed the tabu search heuristic and the AMPL-Cplex model in 11 small cases to evaluate the execution time and the solution gaps in the tabu search with respect to the optimal values. The small cases had from 2 to 12 access nodes and from 4 to 24 potential data center locations. In these cases, the data center capacities were set at either 64,000 or 128,000 to be able to host 200,000 servers at a small number of data centers. Table 3 shows the results when the optimization priority is the delay, Table 4 shows the results with pollution minimization, and Table 5 presents the results when the priority was the network cost. All the costs are expressed in millions of dollars. Each case was executed

TABLE 3
Comparison between Tabu Search and Cplex

#AN	#DC	Tabu(s)	Cplex(s)	Delay(ms)	CO_2(ton)	Cost	z	Min.gap	Avg.gap	Max.gap	GR.gap
2	4	0.001	0.032	2.9	6,520	360.0	593.8	0.00%	0.00%	0.00%	0.00%
3	6	0.002	0.090	2.6	49,473	385.7	748.6	0.00%	0.00%	0.00%	24.44%
4	8	0.013	0.287	2.8	101,757	402.9	926.3	0.00%	0.00%	0.00%	22.44%
5	10	0.023	0.137	2.4	90,729	462.9	1,060.4	0.00%	0.00%	0.00%	0.38%
6	12	0.061	0.272	2.2	43,688	484.0	1,059.9	0.02%	0.02%	0.02%	0.02%
7	14	0.071	0.381	2.3	131,135	509.9	1,279.1	0.02%	0.02%	0.02%	1.64%
8	16	0.165	0.532	2.5	130,208	583.0	1,477.1	0.02%	0.08%	0.62%	0.62%
9	18	0.158	0.564	2.2	96,281	629.1	1,495.3	0.02%	0.02%	0.02%	0.02%
10	20	0.276	0.719	2.3	112,688	606.8	1,614.1	0.01%	0.01%	0.01%	0.01%
11	22	0.313	0.822	2.5	143,455	626.0	1,836.2	0.01%	0.01%	0.01%	0.01%
12	24	0.373	0.996	2.5	36,043	675.5	1,820.1	0.01%	0.01%	0.01%	0.01%

Delay priority.

TABLE 4
Comparison between Tabu Search and Cplex

#AN	#DC	Tabu(s)	Cplex(s)	Delay(ms)	CO_2(ton)	Cost	z	Min.gap	Avg.gap	Max.gap	GR.gap
2	4	0.002	0.031	2.9	6,520	360.0	7,138.7	0.00%	0.00%	0.00%	4.60%
3	6	0.005	0.130	13.5	5,645	408.8	8,085.0	0.00%	0.00%	0.00%	9.41%
4	8	0.022	0.361	8.6	18,178	393.8	19,983.1	0.00%	1.07%	10.22%	13.14%
5	10	0.033	0.336	6.6	6,108	435.4	7,998.7	0.00%	0.00%	0.00%	22.28%
6	12	0.047	0.354	10.3	4,650	393.4	8,042.6	0.00%	0.96%	1.59%	1.59%
7	14	0.137	1.084	22.9	6,932	416.6	13,509.8	0.00%	1.67%	4.40%	29.38%
8	16	0.247	0.846	7.4	3,427	429.7	6,437.7	0.00%	0.00%	0.00%	9.47%
9	18	0.230	0.580	16.7	5,151	501.5	10,339.3	0.00%	0.59%	4.86%	26.63%
10	20	0.458	4.146	11.9	4,040	432.5	8,714.3	0.00%	0.23%	0.95%	15.39%
11	22	0.303	3.297	7.9	4,149	435.8	8,480.7	0.03%	0.03%	0.03%	0.03%
12	24	0.559	0.763	4.6	3,817	512.2	6,456.5	0.00%	0.00%	0.00%	0.00%

CO_2 *priority.*

TABLE 5
Comparison between Tabu Search and Cplex

#AN	#DC	Tabu(s)	Cplex(s)	Delay(ms)	CO_2(ton)	Cost	z	Min.gap	Avg.gap	Max.gap	GR.gap
2	4	0.001	0.045	2.1	170,026	357.6	357,907.8	0.00%	0.00%	0.00%	16.09%
3	6	0.005	0.283	9.9	87,891	362.9	364,469.4	0.00%	0.45%	4.47%	20.02%
4	8	0.010	1.163	6.8	210,694	364.9	366,263.7	0.00%	0.28%	0.94%	24.85%
5	10	0.033	2.422	10.8	143,705	374.9	377,336.6	0.00%	1.90%	3.56%	22.88%
6	12	0.076	5.361	9.6	60,234	365.3	368,031.9	0.10%	0.29%	1.75%	24.28%
7	14	0.124	58.147	9.9	184,332	365.3	368,744.8	0.10%	0.75%	3.19%	19.42%
8	16	0.205	83.514	10.0	138,276	373.8	377,319.7	0.06%	1.99%	6.38%	36.51%
9	18	0.288	40.673	8.7	50,595	367.0	370,745.7	0.03%	2.45%	4.45%	37.35%
10	20	0.460	1057.181	11.8	75,585	369.8	374,711.4	0.03%	2.00%	5.71%	28.61%
11	22	0.867	1496.597	12.1	141,955	375.2	381,295.8	1.05%	3.27%	7.02%	29.95%
12	24	0.868	3482.393	13.5	59,401	380.5	387,864.8	1.95%	4.53%	5.74%	30.69%

Cost priority.

10 times using the tabu search heuristic to calculate the averages of each value, as well as the minimal, the average, and the maximal gaps. The minimal gaps are between 0 and 1.95 percent; most of them are below 0.1 percent, showing that the heuristic is very effective in finding near-optimal solutions. The last column of the tables exhibits the gap between the value found with the greedy heuristic and the optimal value. In two cases, greedy found the optimal values but most of the cases were above 10 percent. This reveals that the tabu search is what effectively improves the solution to near optimality.

The execution time of the tabu search ranged from less than 1 to 868 milliseconds, whereas Cplex used more than 58 minutes for the most difficult case. For that complex case, with 12 access nodes, 24 data centers, and cost minimization, the tabu search heuristic had an average gap of 1.95 percent and an average execution time of 868 milliseconds. The results also show that the cost-oriented optimization requires more execution time in all the cases. This extra execution time is needed because there are many near-optimal solutions that cannot be discarded until the optimum is found. In the delay-oriented case, the feasible solutions with small delay are those with servers near the users. Once one of those solutions is found, the components will tend to be fixed near the users, and other solutions are discarded, reducing the search space and the execution time. In the case of CO_2 optimization, feasible solutions with low emissions will discard data centers with high emissions. It is not possible to discard solutions in the cost-oriented case because many data centers have the same CAPEX and OPEX, resulting in many combinations with similar costs.

6.3 Large Cases

The tabu search heuristic is very efficient for small and medium problems. We will analyze larger cases, i.e., 50 to 500 access nodes and 100 to 1,000 potential data center locations, to gauge the performance of the tabu search heuristic for large problems. Considering a large number of cities makes it possible to guarantee a good quality of service for most of the Internet users around the world. Large networks, such as Akamai, have small data centers in more than a thousand locations around the world to guarantee a high quality of service [5], which is why it is important to analyze the large cases using an efficient heuristic. In these cases, the data center sizes were 40, 80, 160, 1,000, 32,000, 64,000, or 128,000 IT elements; solutions may contain multiple small data centers or a combination of small, medium, and large data centers. Tables 6, 7, and 8 show the network delay, costs, optimal values, and execution times for these cases. Each reported value is the average of 10 executions.

The results show that the execution time was less than 1 minute for the cases with up to 200 access nodes and 400 potential data centers, and it was less than 10 minutes in every instance. The delay was held to approximately

TABLE 6
Tabu Search Results for Large Cases

#AN	#DC	Exec.(s)	#DC open	Iterations	Delay(ms)	CO_2(ton)	Total cost	z	GR.gap
50	100	0.4	34	405	3.9	91,330	1,188.3	8,762.1	0.00%
100	200	2.5	73	908	3.6	106,711	1,833.2	14,594.4	7.06%
150	300	5.7	117	1,212	3.9	108,790	2,611.4	21,098.7	0.85%
200	400	12.1	137	1,669	4.4	99,655	2,976.8	27,455.9	3.81%
250	500	20.8	199	2,149	4.7	79,202	3,735.3	32,103.8	3.35%
300	600	32.3	218	2,631	5.0	97,388	3,597.0	38,335.7	5.03%
350	700	45.6	261	3,061	5.3	103,037	4,201.3	43,899.6	4.67%
400	800	66.1	306	3,681	5.7	101,242	4,859.2	49,963.2	10.34%
450	900	85.4	356	4,047	6.2	92,121	5,078.5	54,795.5	10.05%
500	1000	117.5	396	4,718	6.6	116,096	5,349.3	60,327.2	12.41%

Delay priority.

TABLE 7
Tabu Search Results for Large Cases

#AN	#DC	Exec.(s)	#DC open	Iterations	Delay(ms)	CO_2(ton)	Total cost	z	GR.gap
50	100	0.7	20	623	5.9	5,714	874.8	18,156.8	8.44%
100	200	2.2	52	806	4.8	5,344	1,259.0	23,013.4	0.74%
150	300	6.2	69	1302	5.0	6,982	1,848.2	33,031.9	6.37%
200	400	12.7	93	1744	5.1	7,500	2,330.5	40,687.0	4.05%
250	500	18.9	138	2027	5.3	5,746	2,765.8	42,423.4	0.67%
300	600	31.1	164	2658	5.5	7,171	3,183.9	50,408.5	3.40%
350	700	46.5	189	3094	6.0	7,620	3,705.2	58,609.4	3.41%
400	800	67.0	243	3712	6.4	8,609	4,267.8	65,433.0	7.98%
450	900	86.2	281	4182	6.8	8,325	4,737.6	69,195.0	12.77%
500	1000	112.4	314	4582	7.6	9,944	5,032.5	77,003.9	7.59%

Pollution priority.

TABLE 8
Tabu Search Results for Large Cases

#AN	#DC	Exec.(s)	#DC open	Iterations	Delay(ms)	CO_2(ton)	Total cost	z	GR.gap
50	100	1.4	7	1,295	17.6	101,729	426.2	463,528.1	33.84%
100	200	9.6	29	3,342	24.8	101,188	466.2	544,064.0	46.72%
150	300	19.7	44	3,905	22.3	82,974	470.9	590,651.9	46.45%
200	400	52.1	52	6,762	23.4	96,500	494.2	662,006.5	61.72%
250	500	126.8	85	12,418	18.5	149,779	505.1	637,902.5	56.34%
300	600	190.3	122	15,330	21.1	123,173	528.7	695,767.9	58.15%
350	700	258.3	176	17,098	22.6	116,601	554.6	795,113.5	50.70%
400	800	459.0	208	24,552	21.6	59,005	584.5	781,621.2	29.18%
450	900	467.2	296	21,193	23.6	76,514	617.3	863,840.6	13.77%
500	1000	553.8	359	21,329	26.1	134,860	670.2	951,827.6	16.10%

Cost priority.

5 ms in the delay-oriented optimization by adding more data centers. The smallest case opened 34 data centers, and the largest case opened 396, i.e., 12 times more, but the cost was only multiplied by 4.5 because smaller data centers can be used. However, the CO_2 emissions were high compared to the pollution minimization cases. The average CO_2 in the first table is 99,557 tons, and it is 7,295 tons in the second table. The average delay in the CO_2 optimization only increased to 5.8 ms, which is less than 1 ms more than the delay-oriented optimization. The average cost decreased from $ 3,543.0 M to $ 3,000.5 M, which are both very large compared to the average optimal cost of $ 531.8 M in the third case. In that cost-oriented strategy, the average cost was 18 percent of the cost of the CO_2 strategy, though the average delay increased to 22 ms, more than four times that in the delay-oriented strategy.

In these cases, that are too large for Cplex to solve to optimality, we show the performance of the tabu search compared to the initial greedy solution. Here, again the results exhibit a similar behavior to the near optimal results achieved in the small instances because the tabu effectively improves the initial solution: the gap between the greedy solution and the best tabu solution is up to 12.77 percent in Tables 6 and 7, and up to 61.72 percent in Table 8. To illustrate the gap, the cost-oriented case AN = 200, DC = 400 had a total cost of $ 494.2 M in tabu search and $ 630.2 M in greedy. The difference can also be seen in Fig. 7, where the

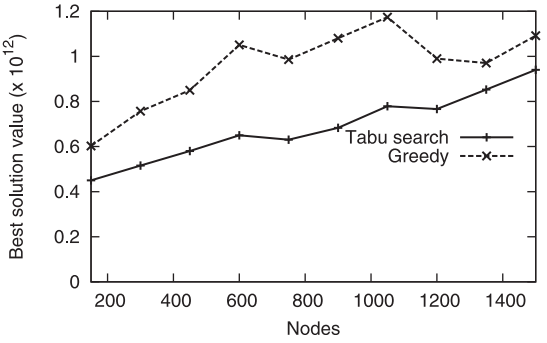

Fig. 7. Tabu search versus greedy heuristic for large cases and cost priority.

values obtained by the tabu and greedy algorithms in the cost-oriented cases are shown.

To our knowledge, these results exhibit the most flexible and efficient algorithm for data center location published in the literature. The flexibility is given by the possibility of defining an application graph that matches the forecasted traffic of the applications, and not only the traffic between users and servers. The model is also flexible in the sense of allowing planners to define the priorities in the multicriteria objective function. The efficiency between different approaches is difficult to compare because each one solves a variant of the problem and for different cases. Nevertheless, we can point that a simulated annealing heuristic combined with linear programming was tested in [7] with up to 500 data center locations and execution times of 30 minutes in a similar environment. Our case with 1,000 data centers was solved by tabu search in 10 minutes. The solution quality cannot be compared because optimal solutions cannot be guaranteed in neither of the two approaches. We understand that our approach achieves such results because each tabu search iteration is extremely quick compared to expensive linear programming resolutions. That was accomplished by using specialized data structures to calculate the objective function in an incremental way.

7 Conclusion

Cloud computing is expanding to increasingly more users, applications, and devices; therefore, the network needs to grow in an efficient and sustainable way. In this paper, we presented a cloud network planning problem through a mixed-integer linear programming model. The location of data centers, servers, and software components impacts the network efficiency, the pollution, and the total cost. The definition of the network link capacities and the information routing is also very important. The proposed model allows planners to evaluate different solutions and to make variations in the optimization priorities.

The AMPL-Cplex implementation is useful for analyzing small cases, but its algorithmic complexity makes the execution time too high for real instances. The deployment of global networks needs the evaluation of the traffic from many cities around the world, and hundreds of potential data center locations. We addressed that issue by developing an efficient tabu search heuristic. The tabu search algorithm evaluates potential neighbor solutions by calculating the difference from the current solution using the right data structures. The results showed that the tabu search heuristic could find near optimal solutions in a very short execution time. Instances of up to 500 access nodes and 1,000 potential data center locations were solved in less than 10 minutes, showing that every real instance can be modeled and solved in this fashion.

The mathematical model and the algorithm can be extended in multiple directions. One direction is the server virtualization, a key aspect of cloud computing. With that feature, each server can host multiple independent virtual machines. Then, fewer servers are used, reducing costs and energy consumption. Another possible extension is to consider dynamic demands. In that case, the variations of the resource requirements in each hour of the day and on each day of the week are considered; therefore, the solutions make better use of the data centers, servers, and link capacities.

References

[1] M. Armbrust, A. Fox, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View on Cloud Computing," Technical Report UCB/EECS-2009-28, Univ. of California, Berkeley, 2009.

[2] R. Brown, E. Masanet, B. Nordman, B. Tschudi, A. Shehabi, J. Stanley, J. Koomey, D. Sartor, P. Chan, J. Loper, S. Capana, B. Hedman, R. Duff, E. Haines, D. Sass, and A. Fanara, "Report to Congress on Server and Data Center Energy Efficiency," *Public Law,* vol. 109, p. 431, 2007.

[3] R. Buyya, R. Ranjan, and R. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," *Proc. 10th Int'l Conf. Algorithms and Architectures for Parallel Processing (ICA3PP),* pp. 328-336, May 2010.

[4] F. Larumbe and B. Sansò, "Location and Dynamic Provisioning Problems in Cloud Computing Networks," *Communication Infrastructures for Cloud Computing: Design and Applications,* M. Hussein and B. Kantarci, eds. IGI Global, to be Published.

[5] E. Nygren, R. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-Performance Internet Applications," *ACM SIGOPS Operating Systems Rev.,* vol. 44, no. 3, pp. 2-19, 2010.

[6] S. Chang, S. Patel, and J. Withers, "An Optimization Model to Determine Data Center Locations for the Army Enterprise," *Proc. IEEE World's Premier Military Comm. Conf. (MILCOM),* pp. 1-8, Oct. 2007.

[7] I Goiri, K. Le, J. Guitart, J. Torres, and R. Bianchini, "Intelligent Placement of Datacenters for Internet Services," *Proc. 31st Int'l Conf. Distributed Computing Systems (ICDCS),* June 2011.

[8] X. Dong, T. El-Gorashi, and J. Elmirghani, "Green IP Over WDM Networks with Data Centers," *J. Lightwave Technology,* vol. 29, no. 12, pp. 1861-1880, 2011.

[9] F. Larumbe and B. Sansò, "Cloptimus: A Multi-Objective Cloud Data Center and Software Component Location Framework," *Proc. IEEE First Int'l Conf. Cloud Networking (CLOUDNET),* pp. 23-28, Nov. 2012.

[10] M. Covas, C. Silva, and L. Dias, "Multicriteria Decision Analysis for Sustainable Data Centers Location," *Int'l Trans. Operational Research,* vol. 20, no. 3, pp. 269-299, May 2013.

[11] H. Stone, "Multiprocessor Scheduling with the Aid of Network Flow Algorithms," *IEEE Trans. Software Eng.,* vol. SE-3, no. 1, pp. 85-93, Jan. 1977.

[12] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment," *Proc. IEEE INFOCOM,* pp. 1-9, 2010.

[13] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," *Proc. IEEE INFOCOM,* pp. 783-791, 2009.

[14] A. Geoffrion and R. Bride, "Lagrangean Relaxation Applied to Capacitated Facility Location Problems," *AIIE Trans.,* vol. 10, no. 1, pp. 40-47, 1978.

[15] A. Klose and S. Gortz, "A Branch-and-Price Algorithm for the Capacitated Facility Location Problem," *European J. Operational Research,* vol. 179, no. 3, pp. 1109-1125, 2007.
[16] H. Pirkul, "Efficient Algorithms for the Capacitated Concentrator Location Problem," *Computers & Operations Research,* vol. 14, no. 3, pp. 197-208, 1987.
[17] Z. Drezner and H. Hamacher, *Facility Location: Applications and Theory.* Springer, 2004.
[18] M. Plante and B. Sansò, "A Typology for Multi-Technology, Multi-Service Broadband Network Synthesis," *Telecomm. Systems,* vol. 19, no. 1, pp. 39-73, 2002.
[19] F. Larumbe and B. Sansò, "Optimal Location of Data Centers and Software Components in Cloud Computing Network Design," *Proc. IEEE/ACM 12th Int'l Symp. Cluster, Cloud and Grid Computing, Workshop Cloud Computing Optimization (CCOPT),* pp. 841-844, May 2012.
[20] A. Beloglazov et al., "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers,* vol. 82, no. 2, pp. 47-111, 2011.
[21] B. Sovacool, "Valuing the Greenhouse Gas Emissions from Nuclear Power: A Critical Survey," *Energy Policy,* vol. 36, no. 8, pp. 2950-2963, 2008.
[22] L. Barroso and U. Hölzle, "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines," *Synthesis Lectures Computer Architecture,* vol. 4, no. 1, pp. 1-108, 2009.
[23] F. Glover, "Tabu Search—Part I," *ORSA J. Computing,* vol. 1, no. 3, pp. 190-206, 1989.
[24] F. Glover, "Tabu Search—Part II," *ORSA J. Computing,* vol. 2, no. 1, pp. 4-32, 1990.
[25] A. Hertz and D. Werra, "Using Tabu Search Techniques for Graph Coloring," *Computing,* vol. 39, no. 4, pp. 345-351, 1987.
[26] T. Brinkhoff, "City Population," http://www.citypopulation.de/, Jan. 2013.
[27] Internet World Stats, "Usage and Population Statistics," http://www.internetworldstats.com, Jan. 2013.
[28] Google, "Google Map," http://map.google.com, Jan. 2013.
[29] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet Inter-Domain Traffic," *ACM SIGCOMM Computer Comm. Rev.,* vol. 40, no. 4, pp. 75-86, 2010.
[30] Telegeography, "Executive Summary," http://www.telegeography.com, Jan. 2013.
[31] Dell, "Online Store," http://www.dell.com, Jan. 2013.
[32] U.S. Energy Information Administration, "Independent Statistics & Analysis," http://www.eia.gov/todayinenergy/detail.cfm?id=4530, 2011.

Federico Larumbe received the computer science degree from the University of Buenos Aires, Argentina. He is currently working toward the PhD degree at Ecole Polytechnique de Montreal. His research focuses on the planning and management of cloud computing networks by developing optimization models and algorithms. In the context of the PhD degree, he did an internship at Facebook, Menlo Park, California to analyze and model the energy consumption of the servers, network, and data centers in 2012. He received the best Computer Science Thesis award in 2009 from the University of Buenos Aires, Argentina. He also received awards from the ACM International Programming Contest in 2003. He is a member of the IEEE.

Brunilde Sansò is a full professor of electrical engineering at Ecole Polytechnique de Montreal and the director of the LORLAB, a research laboratory dedicated to the performance, reliability, design, and optimization of wireless and wireline networks. She has received several awards and honors, the coeditor of the books *Telecommunications Network Planning (Norwell, MA: Kluwer, 1998)* and *Performance and Planning Methods for the Next Generation Internet (Springer, 2005)*, an associate editor of *Telecommunication Systems*, and a TPC member of several major networking and optimization conferences. She is a member of the IEEE.