Dynamic Cloudlet Assignment Problem: a Column Generation Approach

Alberto Ceselli[1], Marco Fiore[2], Marco Premoli[1], and Stefano Secci[3]

[1]Department of Computer Science, Università Degli Studi di Milano, Crema, Italy
[2]CNR-IEIIT, Torino, Italy
[3]UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France, e-mail: stefano.secci@upmc.fr

Major interest in network optimization is currently given to the integration of clusters of virtualization servers, also referred to as 'cloudlets', into mobile access networks for improved performance and reliability. Mobile access points (APs) are assigned (i.e., route their packets) to one or more cloudlets, with a cost in terms of latency for the users they provide connections to. Assignment of APs to cloudlet can be changed over time, with a cloudlet synchronization cost. We tackle the problem of the optimal assignment of APs to cloudlets over time, proposing dedicated mathematical models and column generation algorithms.

1 Model

Given a set of mobile access point sites (APs in the remainder), a set of virtualization server facility sites (cloudlets in the remainder) and the network connecting them, we aim at finding the best schedule for the assignment of APs to cloudlets over a planning time horizon so that the full AP demand is satisfied, no cloudlet capacity is exceeded and the management cost is minimum. Let A be a set of APs locations, K be a set of cloudlets and T be a set of time-slots in which the planning time horizon is split. For each $i \in A$ and $t \in T$, let d_i^t be the mobile traffic demand of AP i in time-slot t. For each $k \in K$, let C_k be the amount of demand that cloudlet k can handle in each time-slot and let $U \in [0, 1]$ represent the maximum allowed cloudlet utilization ratio. The assignment of an AP to a cloudlet implies a cost for the users connected to the AP in terms of communication latency, which is computed as the product of the demand traffic and the physical distance $m_{i,k}$ between AP $i \in A$ and cloudlet $k \in K$ in the network. Assignments can change over time, and each change implies a *switching* cost for the network, which is computed as the product of the demand traffic to be re-routed in the time-slot and the distance $l_{k',k''}$ between the pair of cloudlets $k', k'' \in K$ in the network. There is a trade-off between users' and network costs: to minimize the former an AP has to be assigned to its nearest cloudlet, while to minimize the latter an assignment to a distant cloudlet might be preferable, as long as it is not changing over time; let α and β be two non-negative parameters used to weight the relative importance of users and network costs.

We introduce two sets of variables: (i) variables $x_{i,k}^t$ model AP-cloudlet assignment, taking value 1 if AP $i \in A$ is assigned to cloudlet $k \in K$ in time-slot $t \in T$; and (ii) variables $y_{i,j,k}^t$

model the change of assignment in consecutive time-slots, taking value 1 if AP $i \in A$ is assigned to cloudlet $j \in K$ at time $(t-1) \in T$ and to cloudlet $k \in K$ at time $t \in T$.

We formulate our Dynamic Cloudlet Assignment Problem (DCAP) as follows:

$$\min \alpha \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik}^t + \beta \sum_{t\in T}\sum_{i\in A}\sum_{\substack{(j,k)\in \\ K\times K}} d_i^t l_{jk} y_{ijk}^t \tag{1}$$

$$\text{s.t.} \sum_{i\in A} d_i^t x_{ik}^t \leq UC_k \qquad \forall t \in T, \forall k \in K \tag{2}$$

$$\sum_{k\in K} x_{ik}^t = 1 \qquad \forall i \in A, \forall t \in T \tag{3}$$

$$x_{ik}^t = \sum_{j\in K} y_{ijk}^t \qquad \forall i \in A, \forall t \in T \setminus \{1\}, \forall k \in K \tag{4}$$

$$x_{ik}^t = \sum_{j\in K} y_{ikj}^{t+1} \qquad \forall i \in A, \forall t \in T \setminus \{|T|\}, \forall k \in K \tag{5}$$

$$\boldsymbol{y} \in \{0,1\}, \boldsymbol{x} \in \{0,1\} \tag{6}$$

Objective function (1) minimizes the trade-off between users' and network costs. Constraints (2) impose that for each cloudlet the total demand assigned in each time-slot does not exceed a fraction U of its capacity; constraints (3) impose that the demand of each AP is completely assigned; constraints (4) and (5) link $\boldsymbol{x}$ and $\boldsymbol{y}$ variables. In the literature, both single or multi source assignment models are popular: in the former an AP is assigned to a single cloudlet in each time slot, in the latter the demand of an AP can be split and served by several cloudlets in the same time slot. We consider both possibilities: the single source conditions are enforced by keeping constraints (6), while in the multi source variant constraints on $\boldsymbol{x}$ variables are relaxed to $\boldsymbol{x} \in [0,1]$. We also remark that in the single source model, due to constraints (4) and (5), $\boldsymbol{y}$ variables automatically take integer values when integrality conditions on $\boldsymbol{x}$ are enforced.

2 Algorithm

In order to tackle data instances with large set of time-slots and large scale networks, we devised a column generation approach with a rounding strategy to restore integrality. Our Dantzig-Wolfe decomposition on model (1) – (6) yields to the following master problem:

$$\min \sum_{i\in A}\sum_{p\in\Omega^i} \left(\alpha \sum_{t\in T}\sum_{\substack{(j,k)\in \\ K\times K}} d_i^t l_{jk} \tilde{y}_{ijk}^{t,p} + \beta \sum_{t\in T}\sum_{k\in K} d_i^t m_{ik} \tilde{x}_{i,k}^{t,p} \right) z^p \tag{7}$$

$$\text{s.t.} -\sum_{i\in A}\sum_{p\in\Omega^i} d_i^t \tilde{x}_{ik}^{t,p} z^p \geq -UC_k \qquad \forall t \in T, \forall k \in K \tag{8}$$

$$\sum_{p\in\Omega^i} z^p = 1 \qquad \forall i \in A \tag{9}$$

$$z^p \geq 0 \tag{10}$$

where for each AP $i \in A$, Ω^i is the set of feasible *assignment paths*, that model the sequence of cloudlets to which the AP is assigned in the complete sequence of time-slots. For each path

$p \in \Omega^i$, let variable z^p take value 1 if path p is chosen to assign the related AP to the sequence of cloudlets. A path $p \in \Omega^i$ is encoded via binary parameters $\tilde{x}_{ik}^{t,p}$ and $\tilde{y}_{ijk}^{t,p}$, which behave as the corresponding variables $\boldsymbol{x}$ and $\boldsymbol{y}$. Objective function (7) aims in minimizing the management cost related to the chosen paths; (8) impose the maximum cloudlet utilization in a time-slot; (9) impose that for each AP a combination of assignment paths are chosen and hence that its demand is fulfilled in every time-slot.

Since formulation (7) – (9) contains a combinatorial number of variables, we optimize it with column generation techniques. Let $\lambda_{t,k}$ be the non-negative dual variables of constraints (8) and η_i be the free dual variables of constraints (9). The pricing problem is a shortest path for each AP. The corresponding formulation, given a fixed AP $\hat{i} \in A$, is the following:

$$\min \; -\eta_{\hat{i}} + \alpha \sum_{t \in T} \sum_{\substack{(j,k) \in \\ K \times K}} d_{\hat{i}}^t l_{jk} y_{\hat{i}jk}^t + \sum_{t \in T} \sum_{k \in K} \left(\beta d_{\hat{i}}^t m_{ik} + d_{\hat{i}}^t \lambda_{t,k}\right) x_{\hat{i},k}^t \tag{11}$$

$$\text{s.t.} \sum_{k \in K} x_{\hat{i}k}^t = 1 \qquad \forall t \in T \tag{12}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}jk}^t \qquad \forall t \in T \setminus \{1\}, \forall k \in K \tag{13}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}kj}^{t+1} \qquad \forall t \in T \setminus \{T\}, \forall k \in K \tag{14}$$

$$\boldsymbol{x} \geq 0, \boldsymbol{y} \geq 0 \tag{15}$$

This shortest path problem involves a directed layered graph $G(N, A)$, with $|T|$ layers, one for each time-slot. Each layer has one node for each cloudlet and one arc for each pair of nodes in consecutive layers. Each node $(t, k) \in T \times K$ has an associated cost given by $d_{\hat{i}}^t(\beta m_{ik} + \lambda_{t,k})$, while each arc connecting nodes (t, j) and $(t+1, k)$ has an associated weight given by $\alpha d_{\hat{i}}^{t+1} l_{j,k}$. This shortest path problem can be exactly solved with computational complexity $\mathcal{O}(TK^2)$ by means of dynamic programming.

Restoring Integrality In order to restore integrality, a rounding algorithm is executed at every CG iteration. Given the fractional solution $\tilde{S}$ of the CG master problem and the fractional variables values $\tilde{\boldsymbol{z}}$, we can compute the values of the corresponding $\tilde{\boldsymbol{x}}$ variables. For each time-slot $t \in T$ and for each AP sorted by descending highest fractional value of $\tilde{x}_{ik}^t$, the assignment is made with the cloudlet with enough residual capacity corresponding with the highest $\tilde{x}$. After each assignment the residual capacity of the chosen cloudlet is updated. Unfortunately, the rounding problem is a generalized assignment, which is APX-Hard: no a-priori guarantee on reaching feasibility is given by our algorithm, even if experimentally it proved to be successful in almost all CG iterations.

Greedy Initial Solution A simple greedy heuristic is used to initialize the master problem. It works as follows: for each time slot, APs are sorted by descending demand in the time slot and each AP is associated to a cloudlet chosen according to these rules: (i) take the cloudlet to which the AP was associated in the previous time-slot if the demand of the AP does not exceed its residual capacity and if it is not the first time-slot; (ii) find the nearest cloudlet for which the AP demand does not exceed the residual capacity, otherwise. A solution is provided in $\mathcal{O}(TA \log(A)K)$. Still no guarantees neither on quality nor on feasibility of the solution are given.

3 Computational Results

We implemented our algorithms in C++, using CPLEX 12.6 to solve the master LP subproblems, running tests on an Intel i7 4GHz workstation equipped with 32 GB of RAM. We created two datasets. The first one aims at reproducing realistic scenarii. We used a synthetic set of 1400 APs, whose coordinates are randomly drawn from two normal distributions, in order to model a metropolitan circular area with an higher density of APs in the city center. Ten clusters of APs were created with a standard k-means algorithm: their centers represent cloudlet locations and distances m_{ik} and l_{jk} were computed as euclidean distances accordingly. Planning time horizon was set to a single day. We experimented on four time discretizations: two hours, one hour, thirty minutes and fifteen minutes, corresponding to 12, 24, 48 and 96 time-slots, respectively. We drew mobile traffic demands for the fifteen-minute time-slots so that: (i) within a time-slot the distribution of APs demand follows a truncated heavy-tail power law distribution, and hence the majority of APs have low demand but a significant number of APs have high demand; (ii) the sum of all demands in a time-slot follows the standard daily activity profile, which shows a steep rise of the demand during morning rush-hour, followed by a stable rise until the peak of the evening rush-hour, that is in turn followed by a fall; and (iii) for the single AP, the change of demand during the day follows the same trend of the sum of demands. Demand for larger time slots has been obtained by aggregation. The second dataset aims at stressing our algorithms from a computational point of view. Demands were drawn uniformly at random for the fifteen-minute time slots, with no relationship between demands in consecutive time-slots. We set the demand of an AP in each time slot as the average of the demands in the fifteen-minute time-slots that are covered by it. Moreover for each demand matrix we computed five different instances by perturbing all demands with noise drawn uniformly at random in the interval $[-5\%, +5\%]$. Cloudlet capacity was set equal to $\left(\max_{t\in T} \sum_{i\in A} d_i^t/|K|\right) \cdot 1.05$. Finally, parameters α and β were both set to 0.5, while parameter U was set to 1.

Table 1 reports for each time-slot granularity (columns), and for the realistic and random datasets (rows), the mean, minimum and maximum values computed over the five perturbed instances of: (i) the execution time of our algorithm in seconds and (ii) the percentage gap between the final fractional solution and the best integer one found with rounding. We can notice that the realistic demands show better results both in terms of final gap and execution times. In particular, while in the realistic dataset the optimality gap is always lower than 1% and the finer discretization requires less than one minute of execution time, in the random dataset the gap is always higher than 1.5% and the execution takes up to several minutes. We also notice that on the realistic dataset, the optimality gap increases together with the number of time-slots, while an opposite behavior is observed on the random dataset. We impute this behavior to the smooth trend of realistic demands during the day, while random dataset involves sudden changes in consecutive time-slots.

	no. time-slots	12		24		48		96	
data type	stats.	gap	time	gap	time	gap	time	gap	time
Realistic	mean	0.96%	1.8s	0.70%	4.6s	0.61%	12.2s	0.42%	54.4s
	min	0.69%	1s	0.60%	4s	0.44%	11s	0.32%	51s
	max	1.31%	2s	0.85%	5s	0.73%	14s	0.48%	56s
Random	mean	1.77%	5s	1.89%	18s	2.25%	110.2s	2.53%	1165.6s
	min	1.69%	5s	1.63%	17s	2.13%	106s	2.42%	1098s
	max	1.88%	5s	2.05%	19s	2.41%	112s	2.67%	1244s

Table 1: Computational Results Statistics