# A Multi-Commodity Network Flow Model for Cloud Service Environments

Ioannis M. Stephanakis[1], Syed Noorulhassan Shirazi[2], Antonios Gouglidis[2], and David Hutchison[2]

[1] Hellenic Telecommunication Organization S.A. (OTE), 99 Kifissias Avenue, GR-151 24, Athens, Greece, stephan@ote.gr

[2] InfoLab21, School of Computing and Communications, Lancaster University, LA1 4WA, UK {n.shirazi, a.gouglidis, d.hutchison}@lancaster.ac.uk

**Abstract.** Next-generation systems, such as the big data cloud, have to cope with several challenges, e.g., move of excessive amount of data at a dictated speed, and thus, require the investigation of concepts additional to security in order to ensure their orderly function. Resilience is such a concept, which when ensured by systems or networks they are able to provide and maintain an acceptable level of service in the face of various faults and challenges. In this paper, we investigate the multi-commodity flows problem, as a task within our $D^2R^2+DR$ resilience strategy, and in the context of big data cloud systems. Specifically, proximal gradient optimization is proposed for determining optimal computation flows since such algorithms are highly attractive for solving big data problems. Many such problems can be formulated as the global consensus optimization ones, and can be solved in a distributed manner by the alternating direction method of multipliers (ADMM) algorithm. Numerical evaluation of the proposed model is carried out in the context of specific deployments of a situation-aware information infrastructure.

**Keywords:** Resilience . big data cloud . multi-commodity flow networks . distributed algorithms . consensus optimization . alternating direction method of multipliers (ADMM)

## 1 Introduction

Cloud computing delivers computing services from large, highly virtualized network environments to many independent users, using shared applications and pooled resources. One may distinguish amongst Software-as-a-Service (SaaS) where software is offered on-demand through the internet by the provider and it is parametrized remotely (e.g., on-line word processors, spreadsheets, Google Docs and others); Platform-as-a-Service (PaaS) where customers are allowed to create new applications that are remotely managed and parametrized, and offer tools for development and computer interface restructuring (e.g., Force, Google App Engine and Microsoft Azure), and Infrastructure-as-a-Service (IaaS) where virtual machines, computers and operating systems may be controlled and parametrized remotely (e.g., Amazon EC2 and S3, Terremark Enterprise Cloud, Windows Live Skydrive, Rackspace Cloud, GoGrid, Joyent, AppNexus, etc.). The aforementioned service models of the cloud can be offered in three difference deployment models, i.e., public, private and hybrid. In public cloud systems, everyone may register and use the services. Private ones are accessible through a private network. Lastly, hybrid clouds refer to a combination of the previous two and usually used in the case where sensitive data is required to be kept in the private network and non-core applications are deployed in the public. The key functionality of the private deployment model is the ability to use and release resources from public clouds as and when required. This is used to handle sudden demand surges ('flash crowds') and is known as 'cloud-bursting'.

Cloud computing is an on-demand service whose size depends upon users needs and should feature scale flexibility. It is built upon such network elements as switches supporting novel communication protocols, specific servers based on Virtual Machine (VM) technology and dynamic resource management as well as Network-Attached-Storage (NAS). Specific software platforms may be used for service orchestration in cloud environments (e.g., OpenStack). Several next-generation implementations require widespread connectivity, security and a successful combination with machine-to-machine M2M applications[3] and cloud computing. Integration platforms are important facilitating the convergence of IoT[4], cloud computing, analytic, and big data. They support links among cloud applications and they tie together the distributed devices at one end of a network pipe with enterprise applications and analytic at the other end. Integration platforms shorten the development cycle for connecting devices to the cloud or enterprise systems. Other applications of cloud computing may be seen in the area of critical infrastructures. An example of that is the use of cloud computing services to perform analysis of the data conveyed between the various components of a Supervisory Control and Data Acquisition (SCADA) network [14].

Cloud systems and services can be applicable in a wide range of applications, as described above. Therefore, this further motivates us towards the investigation of concepts additional to security in order to ensure their orderly function. Resilience is such a concept that can ensure that a network or system can provide and maintain an acceptable level of service in the face of various faults and challenges to normal operations [20]. In order to accomplish the previous require-

[3] A typical M2M architecture includes an application domain, a network domain, an M2M device domain and one or more direct connections or gateways from the M2M area network to the network domain. M2M device area networks can use a variety of communication technologies (RFID, ZigBee, M-BUS, IEEE 802.15, 6LoWPAN), thus a gateway layer becomes important. The solutions for communication between the gateway and M2M applications include LTE, WiMAX, xDSL, and WLAN. In the application domain, clients will often include dashboards for data virtualization, status monitoring, reconfiguration and other functions.

[4] Among the consortia working on standards for IoT are AllSeen Alliance, HyperCat Consortium, and Industrial Internet Consortium. There are also initiatives such as the Eclipse M2M Industry Working Group and ITU-T Focus Group M2M initiative

ment we have proposed a resilience strategy entitled $D^2R^2 + DR$, i.e., Defend, Detect, Remediate, Recover, and Diagnose and Refine [20]. The first four consist processes of an internal loop process and the latter two of an off-line outer loop. In more detail, it is: Defend against challenges and threats to normal operation; Detect when an adverse event or condition has occurred; Remediate the effects of the adverse event or condition; Recover to original and normal operations; Diagnose the fault that was the root cause; and Refine behaviour for the future based on past $D^2R^2 + DR$ cycles. Based on our resiliency strategy, we further developed an architectural framework for resilience, which is able to operate in the context of cloud systems and used for diagnosing anomalies [18].

In this paper, we further investigate the multi-commodity network flow in the context of our resilience strategy and towards ensuring network resilience in cloud services. Multi-commodity network flow models provide the tools for optimal network design and dimensioning in telecommunications given a list of traffic nodes (sources or sinks) [17]. The basic mathematical models used to formulate and solve optimal network design problems make use of graph-theoretic and/or linear programming-based models (see e.g., [4],[13],[2]). The set of all possible topologies for the network to be constructed will typically be described by means of a given (undirected) graph $G = [V, E]$ where:

- the node set $V$ represents the various traffic sources/sinks to be interconnected;
- the edge set $E$ corresponds to the various pairs of nodes, which may be physically connected by installing transmission links.

A single-commodity flow between a source and a sink is a **M** vector, $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_M)$ such that $|\varphi_u|$ represents the amount of transmission resource used on edge $e = (i, j)$. The aforementioned commodity model is adopted to accommodate cloud computing and in-network processing [11],[9]. A walk-based as well as an edge based formulation is adopted. Applying such models in the context of distributed and parallel computing in the cloud is challenging. The sheer volume of data in next generation implementations requires advanced analytic capable of exploiting the big data and the computing power of the cloud. Scaling up to 50 and 200-billion connected devices requires innovative security solutions. Hybrid architectures focus on security at endpoints and when data is in transit: device security, cloud security, and network security. Virtualization must be done with resilient virtual machines (VM), resilient single-tenant and multi-tenant servers, and resilient software defined networks (SDN).

The structure of the remainder of this paper is: Section 2 elaborate on distributed proximal algorithms and on ADMM in cloud networks. A commodity network model for maximizing information processing in the cloud is presented in Section 3. An evaluation of the model is provide via simulations in Section 4. Conclusions are presented in Section 5.

## 2 Parallel Algorithms for Optimizing Big Data

### 2.1 Big Data Analytics and Distributed Proximal Algorithms

The information explosion propelled by the advent of online social media, Internet and global-scale communications has rendered big data analytics as well as data-driven statistical learning increasingly important [19]. Dealing with large-scale data sets poses formidable challenges. The sheer volume and dimensionality of data make it impossible to run analytics and traditional inference methods using standalone processors [3],[16]. Decentralized learning with parallelized multi cores is preferred [5],[12], while the data themselves are stored in the cloud or distributed file systems as in MapReduce/Hadoop [10]. Distributed signal processing can be used within the context of sensor networks as well (see for example [1]). Optimizing large scale data may be expressed as:

$$F^{*^{def}} = \min_{x}\{F(x) := f(x) + g(x) : x \in R^p\} \tag{1}$$

where $f$ and $g$ are convex functions. Efficient numerical methods to obtain $x$ in the context of large scale problems arising in big data applications are, namely, first order methods, randomization as well as parallel and distributed computing [8].

- First-order methods: First-order methods obtain low- or medium- accuracy numerical solutions by using only first-order oracle information from the objective, such as gradient estimates. They handle important non smooth variants of Eq. 1by making use of the proximal mapping principle. They feature nearly dimension-independent convergence rates, they are theoretically robust to the approximations of their oracles, and they typically rely on computational primitives that are ideal for distributed and parallel computation.
- Randomization: Randomization approaches stand out among many other approximation techniques since they enhance the scalability of first order methods. We can control their expected behaviour. Key ideas include random partial updates of optimization variables, replacing the deterministic gradient as well as proximal calculations with cheap statistical estimators, and speeding up basic linear algebra routines via randomization.
- Parallel and distributed computation: First-order methods naturally provide a flexible framework to distributive optimization tasks and perform computations in parallel. Surprisingly, one can further augment these methods with approximations to enormously scalable asynchronous algorithms with decentralized communications.

The three aforementioned classes of algorithms complement each other and offer surprising scalability benefits for big data optimization. For closed proper convex functions, one may define the proximal operator of the scaled function $\lambda f$, where $\lambda > 0$, as:

$$prox_{\lambda f}(\mathbf{u}) \overset{def}{=} \text{argmin}\left(f(\mathbf{x}) - \frac{1}{2\lambda}\| \mathbf{x} - \mathbf{u} \|_2^2\right) \tag{2}$$

This is also called the proximal operator of $f$ with respect to $\lambda$. The parameter $\lambda$ controls the extent to which the proximal operator maps towards the minimum of $f$, with larger values of $\lambda$ associated with mapped points near the minimum, and smaller values giving smaller movement towards the minimum. A proximal algorithm is an algorithm for solving a convex optimization problem that uses proximal operators of the objective terms. Proximal algorithms have been used for multi-commodity network flow optimization [15].

**Algorithm 1** Synchronous ADMM (sync-ADMM): Processing by the master.

```
1: initialize: $k = 0$.
2: repeat
3:     repeat
4:         wait;
5:     until receive updates from all $N$ workers;
6:     update $z^{k+1}$
7:     broadcast $z^{k+1}$ to all the workers;
8:     $k \leftarrow k + 1$;
9: until termination;
10: output $z^k$.
```

(a)

**Algorithm 2** Synchronous ADMM (sync-ADMM): Processing by worker $i$.

```
1: initialize: $k = 0, \lambda_i^0 = 0$.
2: repeat
3:     update $x_i^{k+1}$
4:     send $\lambda_i^k$ and $x_i^{k+1}$ to the master;
5:     repeat
6:         wait;
7:     until receive the updated $z^{k+1}$ from the master;
8:     update $\lambda_i^{k+1}$
9: until termination.
```

(b)

**Algorithm 3** Asynchronous ADMM (async-ADMM): Processing by the master.

```
1: initialize: $k = 0, \hat{x}_i = 0, \hat{\lambda}_i = 0, i = 1, 2, \ldots, N$.
2: repeat
3:     repeat
4:         wait;
5:     until receive a minimum of $S$ updates from the
       workers and $\max(\tau_1, \tau_2, \ldots, \tau_N) \leq \tau$;
6:     for worker $i \in \Phi^k$ do
7:         $\tau_i \leftarrow 1$;
8:         $\hat{x}_i \leftarrow$ newly received $x_i$ from worker $i$;
9:         $\hat{\lambda}_i \leftarrow$ newly received $\lambda_i$ from worker $i$;
10:    end for
11:    for worker $i \notin \Phi^k$ do
12:        $\tau_i \leftarrow \tau_i + 1$;
13:    end for
14:    update $z^{k+1}$
15:    broadcast $z^{k+1}$ to all the workers in $\Phi^k$;
16:    $k \leftarrow k + 1$;
17: until termination;
18: output $z^k$.
```

(c)

**Algorithm 4** Asynchronous ADMM (async-ADMM): Processing by worker $i$.

```
1: initialize: $\lambda_i^0 = 0, k_i = 0$.
2: repeat
3:     update $x_i^{k_i+1}$
4:     send $\lambda_i^{k_i}$ and $x_i^{k_i+1}$ to the master;
5:     repeat
6:         wait;
7:     until receive $\tilde{z}_i$ from the master;
8:     update $\lambda_i^{k_i+1}$
9:     $k_i \leftarrow k_i + 1$;
10: until termination.
```

(d)

Fig. 1: Synchronous and asynchronous processing from master and workers for ADMM

## 2.2 Synchronous and Asynchronous Consensus ADMM in Cloud Information Networks

Many machine learning problems can be formulated as the global consensus optimization problem, which can then be solved in a distributed manner by the alternating direction method of multipliers (ADMM) algorithm. The global variance consensus optimization problem [6],[5] in the context of minimization $f(x)$ reads:

$$\min_{x_1,\ldots,x_N,z} f(x)+g(z)=\sum_{i=1}^{N} f_i(x_i) : x_i = z, i=1,2,\ldots,N \tag{3}$$

where $z$ is the so-called consensus variable, and $x_i$ is node $i$ local copy of the parameter to be learned. The aforementioned problem may be reformulated as augmented *Lagrangian* optimization:

$$L(\{x_i\},z)=g(z)+\sum_{i=1}^{N} f_i(x_i)+\langle\lambda_i, x_i-z\rangle+\frac{\beta}{2}\parallel x_i - z \parallel^2 \tag{4}$$

where $\lambda_i$ are the Lagrangian multipliers, $\beta > 0$ is the penalty parameter, and $\langle,\rangle$ denotes the inner product. At the k-th iteration, the values of $x_i$ and $z$ denoted ($x_i^k$ and $z^k$) are updated by minimizing $L$ with respect to $x_i$ and $z$. Unlike the methods of multipliers, these are minimized in an alternating manner, which allows the problem to be more easily decomposed:

$$x_i^{k+1} = \underset{x}{\operatorname{argmin}} f_i(x)+\langle\lambda_i^k, x\rangle+\frac{\beta}{2}\parallel x - z^k \parallel^2 = prox_{f_i/\beta}(z_i^k-\lambda_i^k) \tag{5}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} g(z)+\sum_{i=1}^{N} -\langle\lambda_i^k, z\rangle+\frac{\beta}{2}\parallel x_i^{k+1} - z^k \parallel^2 = prox_{g/\beta}(x^{k+1}+\lambda^k) \tag{6}$$

$$\lambda_i^{k+1} = \lambda_i^{k+1} + \beta\big(x_i^{k+1} - z^{k+1}\big) \tag{7}$$

The updates can be easily implemented in a distributed system with one master and $N$ workers [21]. Each worker $i$ is responsible for updating ($x_i$ , $\lambda_i$) using the above equations. The updated $x_i^{k+1}$ are then sent to the master, which is responsible for updating the consensus variable $z$ as well as distributing its updated value back to the workers. Updating may be performed in a synchronous or an asynchronous manner (see algorithms in Fig. 1).

## 3 A Commodity Network Model for Maximizing Information Processing in Cloud Implementations

One may adopt the proximal-point method in order to optimize network flows in a cloud environment in an iterative fashion:

$$x^{k+1} = \underset{X\in S}{\operatorname{argmin}} \left( f(x)+\frac{1}{2\lambda}(x-x^k)^T(x-x^k) \right) = prox_{\lambda f}(x^k) \tag{8}$$

where $S$ is a convex set, $f(x)$ is a convex function and $1/2\lambda$ is a constant. One may assume $N$ clusters featuring a distributed processing power given by $\pi = (\pi_1, \pi_2, \ldots, \pi_N)^T$. Vector $x$ consists of network flows $x = [x_{s1}, x_{s2}, x_{s3}, \ldots]$

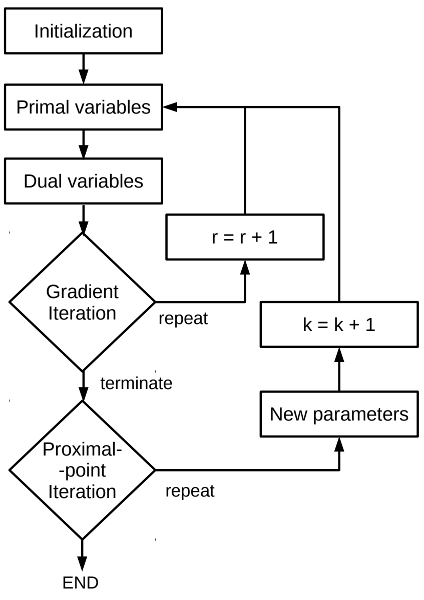


Fig. 2: Block diagram of the dual gradient algorithm.

where subscript $s$ runs over all network edges $s \in \{a, b, c \ldots\}$. The constraint optimization problem for one proximal-point iteration reads:

$$x = \min_{X \in S} \left( \left[ \frac{c_1}{\lambda_1}, \frac{c_2}{\lambda_2}, \ldots, \frac{c_n}{\lambda_N} \right] \mathbf{CF}_{clusters} x + \beta (x - z)^T (x - z) + {\theta_s}^T (\mathbf{CF}_s x - s_{in}) \right. \\ \left. + {\lambda_e}^T (x - \mathbf{BW}) + {\lambda_p}^T (\mathbf{CF}_c x - P) \right) \tag{9}$$

where $z$ is the approximation at the previous iteration step and $\mathbf{CF}_{clusters}$ is the incidence matrix for cloud clusters such that $\mathbf{CF}_{clusters} x$, gives the incident flows at all cloud processing nodes. Flow $(\mathbf{CF}_{clusters} x)_j$ is processed by processor $\pi_j$ within time $T_j \sim (1/\pi_j)(\mathbf{CF}_{clusters} x)_j$. Cluster processing utilization is analogous to processing time. It is assumed that processing cost per site is proportional to **CP** utilization, i.e. the term $[c_1/\lambda_1, c_2/\lambda_2, \ldots, c_n/\lambda_N]\mathbf{CF}_{clusters} x$ is equal to the total processing cost to be minimized according to Eq. 9. Dual variable $\theta_s$ accounts for flow incidence conditions at sensor nodes, i.e., $\mathbf{CF}_{sensors} x = s_{in}$ whereas dual variables $\lambda_e$ and $\lambda_p$ account for upper edge bandwidth limits $x_{edge_l} \leq \mathbf{BW}_{edge_l}, l = 1, \ldots, L$ and cluster processing capabilities. According to Slaters conditions (see for example [7]) strong duality holds for the optimization problem (i.e. the optimal values of the dual and the primal problem are equal. We carry out successive optimization over primal and dual and variables according to the block diagram in Fig. 2. Primal variables as well as dual variables are estimated iteratively several times during the execution cycle of an iteration step. Current estimation of $x^k$ is used as the proximal point $z$ for the next estimation $x^{k+1}$ (see Eq. 8). As an alternative approach, one may use distributed processing and synchronous ADMM and solve Eq. 9 using Eqs. 5 6 and 7 according to algorithms (a) and (b) in Fig. 1. Dual variables are estimated a number of times during each iteration step after the estimation of the primal variables $z$ and $x_i$. Global parameter $z$ is updated by the master so that flow incidence conditions at sensor nodes are satisfied, $\mathbf{CF}_{sensors}z = s_{in}$ and, finally, $z^k = \sum_{i=1}^{N} x_i^k$.

## 4 Numerical Simulations

Computing time $T_j$ is assumed to be normalized to CP utilization at cluster node $j$. Total computing cost is assumed to be analogous to total processing time, i.e.,$c_1T_1 + c_2T_2 + c_3T_3 + c_4T_4 \propto$ total processing cost. Two distinct cases of assigning processor costs are investigated. Case 1 assumes that $c_1 = c_2 = c_3 = c_4 = 100units$ whereas Case 2 assumes that $c_1 = 80units, c_2 = 60units, c_3 = 100units$ and $c_4 = 90units$. Convergence behaviour of the proposed proximal

Table 1: Terminal devices nominal source flows (**Mbps**)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $s_1$=1 | $s_2$=1 | $s_3$=2 | $s_4$=3 | $s_5$=0.5 | $s_6$=0.5 | $s_7$=4 | $s_8$=3 |
| $s_9$=2 | $s_1$0=2 | $s_1$1=2 | $s_1$2=1 | $s_1$3=1.5 | $s_1$4=5 | $s_1$5=2 | $s_1$6=1 |
| $s_1$7=2 | $s_1$8=1 | $s_1$9=1 | $s_2$0=2 | $s_2$1=4 | $s_2$2=2.5 | $s_2$3=3.5 | $s_2$4=2 |
| $s_2$5=2.5 | $s_2$6=1 | $s_2$7=1 | $s_2$8=2 | $s_2$9=3.5 | $s_3$0=5 | $s_3$1=1 | $s_3$2=1 |

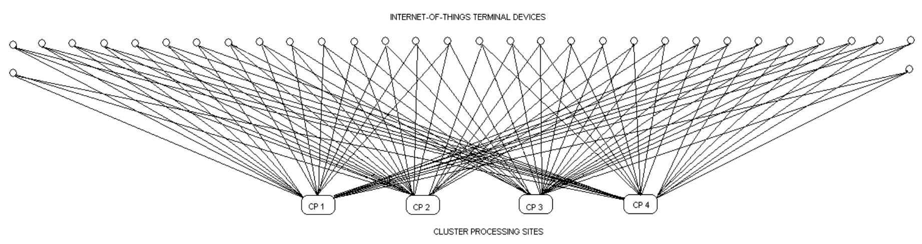
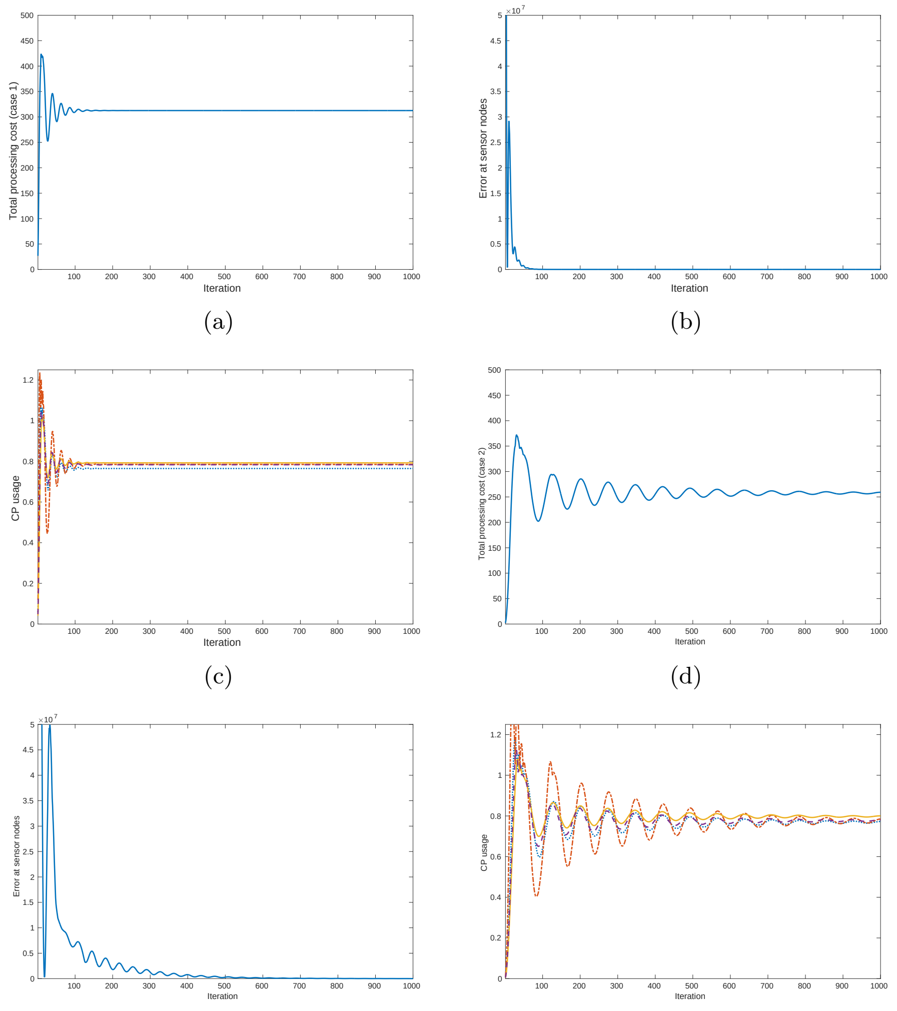


Fig. 3: Cloud interconnections

algorithm for $z$ equal to $x_k$ is depicted in Figs. 4. Fig. 4a depicts total cost, Fig. 4b depicts flow equilibrium conditions at sensor nodes during the execution of the algorithm for Case 1 and Fig. 4c depicts processor utilization per cluster for Case 1. Similar results are illustrated for Case 2 in Figs. 4d, 4e and 4f. Similar results of solving Eq. 9 for Case 1 and Case 2 using distributed processing and synchronized ADMM are depicted in Figs. 5. Figs. 5d and 5h present the difference of $z^k - \sum_{i=1}^{4} x_i^k$ over 1,000 iterations. Both approaches give similar total cost values for Case 1 and Case 2.

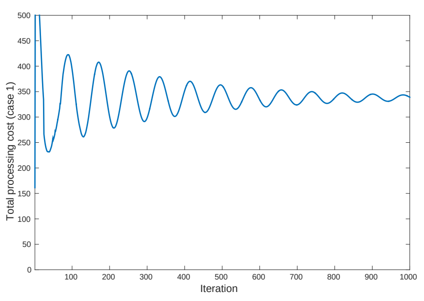


(a) (b)

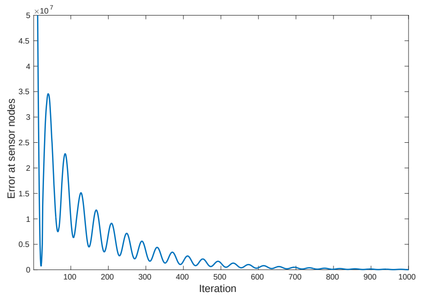
(c) (d)

(e) (f)

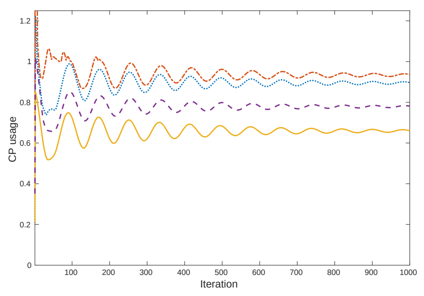
Fig. 4: Iterative solution of Eq. 9 in the proximity of the previous approximation, i.e.,$z = x_{k-1}$ (total processing cost for four processing clusters in (a) and (d), total error at sensor nodes in (b) and (e) and CP utilization for each of the four processing sites in (c) and (f))



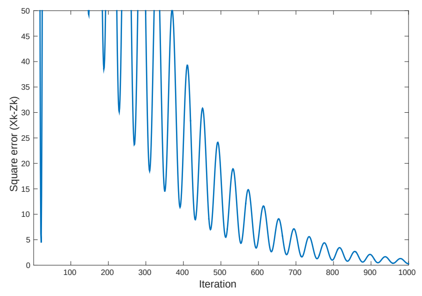
(a)



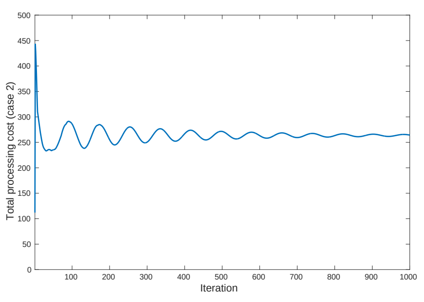
(b)



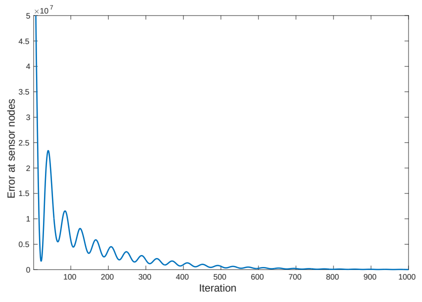
(c)



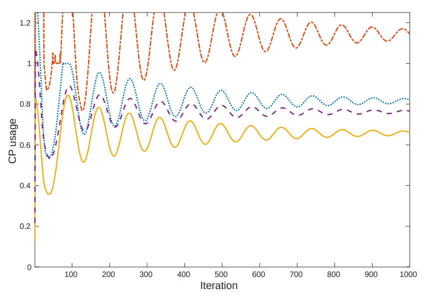
(d)



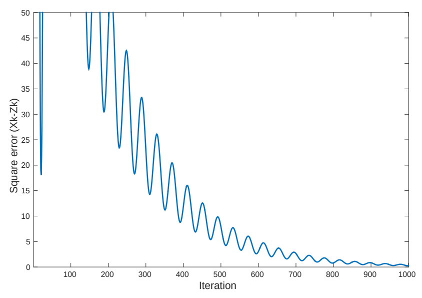
(e)



(f)



(g)



(h)

Fig. 5: Solution of Eq. 9 using distributed processing - sync ADMM (total processing cost for four processing clusters in (a) and (e), total error at sensor nodes in (b) and (f))

Numerical simulations are carried out for artificial data for four (4) processing sites (clusters) and a total of thirty-two (32) terminal devices (sensors) connected to cloud nodes. Each terminal device (sensor) is connected to a main processing cluster and two backup processing clusters. Link capacities vary from 0.5 Mbps to 5 Mbps. Each terminal device produces original source flows featuring values ranging from 0.5 Mbps to 5 Mbps (see Table 1). It is assumed that each flow may be diverted totally or partly from one processing site to the other. Each of the four (4) computing sites is capable of processing a total sum of flows, i.e. $\pi_1 = 20$ Mbps for **CP1**,$\pi_2 = 10$ Mbps for **CP2**,$\pi_3 = 30$ Mbps for **CP3** and $\pi_4 = 25$ Mbps for **CP4**). Cloud interconnections are illustrated in Fig 3.

## 5 Conclusions

An iterative proximal-point method is presented for optimizing commodity flows in the context of cloud computing. A novel approach that combines distributed sync ADMM and minimization over primal and dual variables split into two groups is proposed. Illustrative cases for four (4) processing sites and thirty-two (32) terminal devices are presented. The methods may be scaled to thousands of terminal devices and multiple cloud processing sites. Each terminal device is connected to a subset of processing clusters and may split data flows from one connected processor site to another according to available bandwidth. The proposed algorithm is directly generalized to accommodate variable network conditions as well. Optimization over processing and transmission costs is possible within the context of the proposed approach. Convergence time depends upon the updating method (synchronous or asynchronous ADMM).

**Acknowledgments.** This work is sponsored by UK-EPSRC funded TI3 project, grant agreement no. EP/L026015/1: A Situation-aware Information Infrastructure; and the European Union under Grant SEC-2013.2.5-4: Protection systems for utility networks – Capability Project, Project Number: 608090, Hybrid Risk Management for Utility Providers (HyRiM).

## References

1. Aduroja, A., Schizas, I.D., Maroulas, V.: Distributed principal components analysis in sensor networks. In: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. pp. 5850–5854. IEEE (2013)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: Theory, applications and algorithms. Prentice-Hall, Englewood Cliffs, New Jersey, USA Arrow, KJ (1963). Social Choice and Individual Values, Wiley, New York. Gibbard, A.(1973).Manipulation of Voting Schemes: A general result, Econometrica 41, 587–602 (1993)
3. Bengtsson, T., Bickel, P., Li, B., et al.: Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In: Probability and statistics: Essays in honor of David A. Freedman, pp. 316–334. Institute of Mathematical Statistics (2008)

4. Berge, C.: Graphes et hypergraphes.(dunod, paris.) english translation'(1973) graphs and hypergraphs (1970)
5. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and distributed computation: numerical methods, vol. 23. Prentice hall Englewood Cliffs, NJ (1989)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning 3(1), 1–122 (2011)
7. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
8. Cevher, V., Becker, S., Schmidt, M.: Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. Signal Processing Magazine, IEEE 31(5), 32–43 (2014)
9. Charikar, M., Naamad, Y., Rexford, J., Zou, K.: Multi-commodity flow with in-network processing. Manuscript, www. cs. princeton. edu/˜ jrex/papers/mopt14. pdf
10. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
11. Feizi, S., Zhang, A., Médard, M.: A network flow approach in cloud computing. In: Information Sciences and Systems (CISS), 2013 47th Annual Conference on. pp. 1–6. IEEE (2013)
12. Forero, P.A., Cano, A., Giannakis, G.B.: Consensus-based distributed support vector machines. The Journal of Machine Learning Research 11, 1663–1707 (2010)
13. Gondran, M., Minoux, M., Vajda, S.: Graphs and algorithms. John Wiley & Sons, Inc. (1984)
14. Gouglidis, A., Shirazi, N.u.h., Simpson, S., Smith, P., Hutchison, D.: A framework for resilience management in the cloud. In: International Conference on Telecommunications (ICT), 2016 23rd Annual Conference on. IEEE (2016)
15. Hanzalek, J.T.Z.: Distributed multi-commodity network flow algorithm for energy optimal routing in wireless sensor networks. Radioengineering 19(4) (2010)
16. Jordan, M.I., et al.: On statistics, computation and scalability. Bernoulli 19(4), 1378–1390 (2013)
17. Minoux, M.: Multicommodity network flow models and algorithms in telecommunications. In: Handbook of optimization in telecommunications, pp. 163–184. Springer (2006)
18. Shirazi, N.u.h., Simpson, S., Oechsner, S., Mauthe, A., Hutchison, D.: A framework for resilience management in the cloud. e & i Elektrotechnik und Informationstechnik 132(2), 122–132 (2015), http://dx.doi.org/10.1007/s00502-015-0290-9
19. Slavakis, K., Giannakis, G., Mateos, G.: Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge. Signal Processing Magazine, IEEE 31(5), 18–31 (2014)
20. Sterbenz, J.P., Hutchison, D., etinkaya, E.K., Jabbar, A., Rohrer, J.P., Schller, M., Smith, P.: Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. Computer Networks 54(8), 1245 – 1265 (2010), http://www.sciencedirect.com/science/article/pii/S1389128610000824, resilient and Survivable networks
21. Zhang, R., Kwok, J.: Asynchronous distributed admm for consensus optimization. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14). pp. 1701–1709 (2014)