Reduction Applications*

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

CS363-Computability Theory

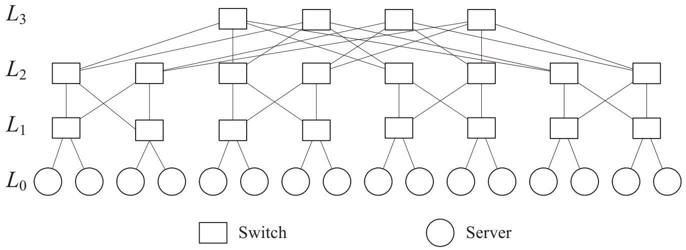
*Special Thanks is given to Dr. Zaixin Lu and Dr. Lidong Wu for sharing their talk materials.

Data Centers

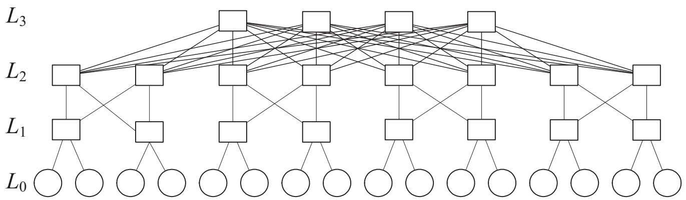
Data Center: a facility used to house computer systems and associated components.

From http://img.clubic.com/05468563-photo-google-datacenter.jpg

Switch-Centric Topology

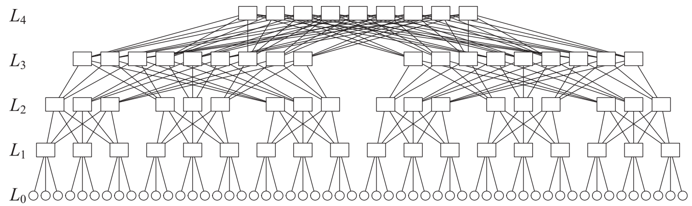


A Fat-Tree ($k = 4$)

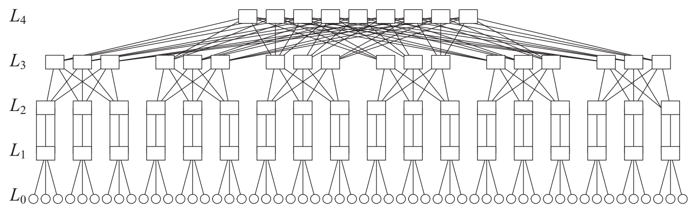


A VL2 (Virtual-Layer Two)

Switch-Centric Topology (2)



An Aspen Tree $C = \langle 1, 1, 3 \rangle$



An Aspen Tree $C = \langle 3, 1, 1 \rangle$

A Controller

Controller: monitor, manage network resources, update routing information, and prepare Virtual Machine migrations.

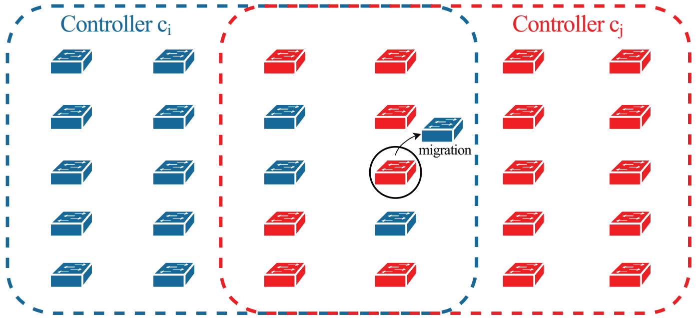
Traffic Load Monitoring: monitor the traffic of switches in a data center.

Workload: the workload of a controller is the sum of traffic loads from its monitored switches.

Our Objective

If a data center has m controllers to monitor n switches, then we hope that the workload of each controller is almost the same.

An Example:



Controller c_j dominates 17 switches and Controller c_i dominates 13 switches. The traffic between c_i and c_j is unbalanced, and c_j is migrating one of its switch to c_i.

Balancing Devolved Controllers (BDC) Problem

Given n switches $S = \{s_1, \cdots, s_n\}$, each has traffic load w_i, and m controllers $C = \{c_1, \cdots, c_m\}$.

Due to physical limitations, each s_i can only be monitored by its potential controller set $PC(s_i)$. Every c_i can only control switches in its potential switch set $PS(c_i)$. After the partition, the real controller and switch subset is denoted by $rc(s_i)$ and $RS(c_i)$ respectively.

The weight of a controller $w(c_i) = \sum_{s_i \in RS(c_i)} w(s_i)$.

Objective: get an *m-partition* for switches such that each controller will has similar amount of workload, say, to minimize the *Standard Deviation* $\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(w(c_i) - \overline{w(c)})^2}$, where $\overline{w(c)}$ is the average weight of controllers.

Non-Linear Programming

Define $x_{ij} = \begin{cases} 1 & \text{If } c_i \text{ monitors } s_j \\ 0 & \text{otherwise} \end{cases}$, Formulat BDC as:

$$\min \quad \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} w(s_j)\cdot x_{ij} - \overline{w(c)}\right)^2} \qquad (1)$$

$$s.t. \quad \overline{w(c)} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} w(s_j)\cdot x_{ij} \qquad (2)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \forall 1 \le j \le n \qquad (3)$$

$$x_{ij} = 0, \quad \text{if } s_j \notin PS(c_i) \text{ or } c_i \notin PC(s_j), \forall i, j \qquad (4)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \qquad (5)$$

Hardness Discussion

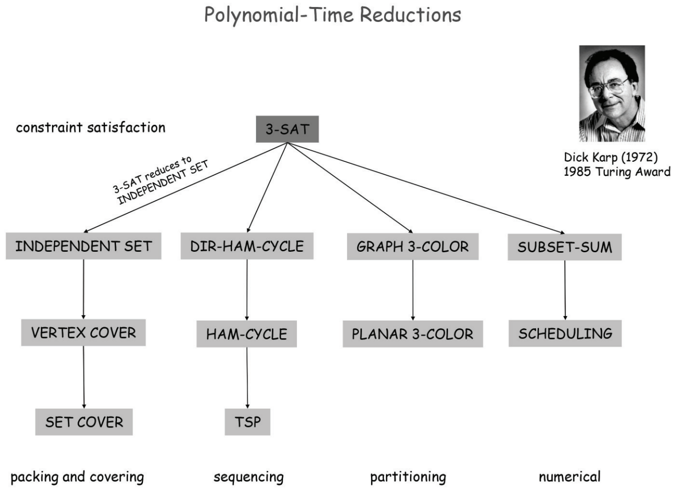
Decision Version of BDC: Given n switches $S = \{s_1, \cdots, s_n\}$, each has traffic load w_i, m controllers $C = \{c_1, \cdots, c_m\}$, a threshold w, does there exist an *m-partition* for switches such that the *Standard Deviation* σ among controllers $\leq w$.

Theorem: BDC $\in \mathbb{NP}$.

Proof: A certificate of BDC is an m-partition with $rc(s_i)$ and $RS(c_i)$ sets. The certifier is to check whether the standard deviation $\sigma \leq w$.

NP Reduction (1)

Theorem: BDC is NP-Complete.



NP Reduction (2)

Proof: PARTITION $\leq_p$ BDC.

An instance of PARTITION is: given a finite set A and a $size(a) \in \mathbb{Z}^+$ for each $a \in A$, is there a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} size(a) = \sum_{a \in A \setminus A'} size(a)$$

Now we construct an instance of LBDC. In this instance there are 2 controllers c_1, c_2 and $|A|$ switches. Each switch s_a represents an element $a \in A$, with weight $w(s_a) = size(a)$.

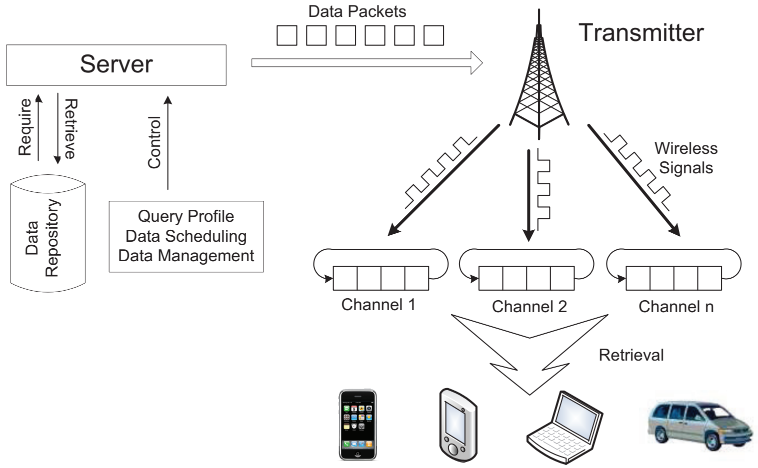
NP Reduction (3)

"$\Rightarrow$" Then, given a YES solution A' for PARTITION, we have a solution that c_1 controls $\{s_a \mid a \in A'\}$, c_2 controls $\{s_a \mid a \in A \setminus A'\}$, and $\sigma = 0$.

"$\Leftarrow$" given a solution for BDC with $\sigma = 0$, we can partite A into $A_1 = RS(c_1)$, $A_2 = RS(c_2)$, then it is a YES solution for PARTITION problem.

The reductions can be done within polynomial time, which completes the proof. $\square$

Wireless Data Broadcast System

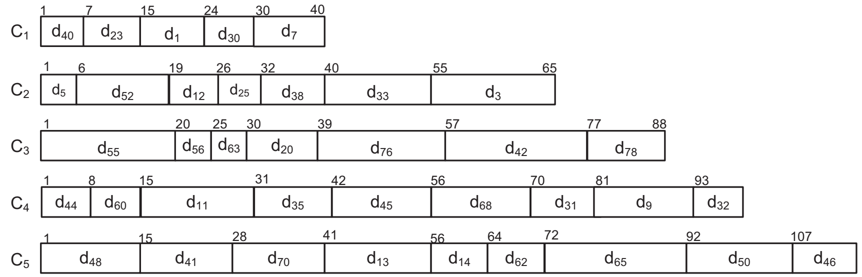


Broadcast Channel and Data Set

$D = \{d_1, d_2, \cdots, d_k\}$ data items, each with different size l_i.

$C = \{c_1, c_2, \cdots, c_n\}$ channels.

An example scenario:

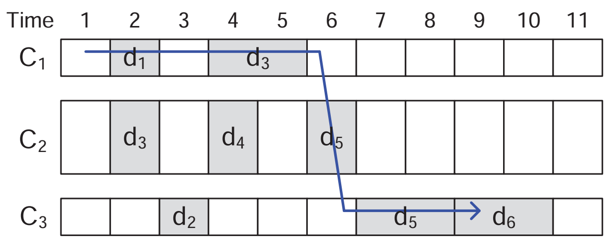


Client Request and Constraint

Request of client: $D_q \subseteq D$;

Switch constraint: switch require one time slot.

An example scenario:



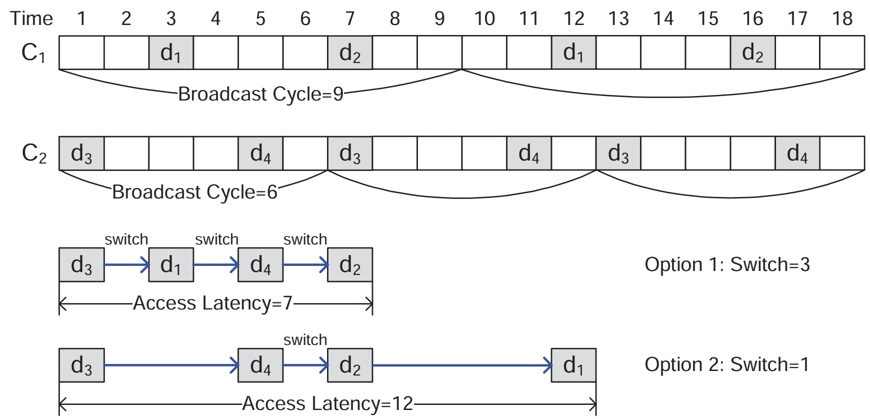
Query Data Set: $\{d_1, d_3, d_5\}$

Data retrieval process

Overall downloading time: 7

Saving Energy Consumption

Energy Consumption: downloading and switching, A Confliction!



Objective: A Constraint Minimization Problem

Definition: Minimum Constraint Data Retrieval Problem (V1)

Given $D = \{d_1, \cdots, d_k\}$ located on n channels $C = \{c_1, \cdots, c_n\}$. Each d_i has length l_i, and located at some position on channel c_j. If we fix a switch parameter h, then the *Minimum Constraint Data Retrieval Problem* (MCDR) is to find a minimum access latency data retrieval schedule to download $D_q \subseteq D$, with at most h switches.

Definition: Minimum Constraint Data Retrieval Problem (V2)

If we fix a latency parameter t, then the MCDR is to find a minimum switch-number data retrieval schedule to download $D_q \subseteq D$, with at most t access latency.

Definition: Minimum Cost Data Retrieval Problem (V3)

If we set parameters α and β, then the MCDR is to find a minimum cost $(\alpha \cdot hop + \beta \cdot time)$ data retrieval schedule to download $D_q \subseteq D$.

A Decision Version

Decision MCDR

Given a data set D, a channel set C, a time threshold t, a switching threshold h, find a valid data retrieval schedule to download all the data in D_q from C before time t with at most h switchings. (the cost is at most $\alpha h + \beta t$)

Theorem: MCDR $\in \mathbb{NP}$

Proof: A certificate of MCDR is a downloading schedule as a sequence of (c_i, d_j) pairs. The certifier is to check whether this schedule can be achieved within t time and h switches.

NP-Completeness

Theorem: MCDR is NP-Complete.

Proof: We prove by VERTEX-COVER $\leq_p$ MCDR.

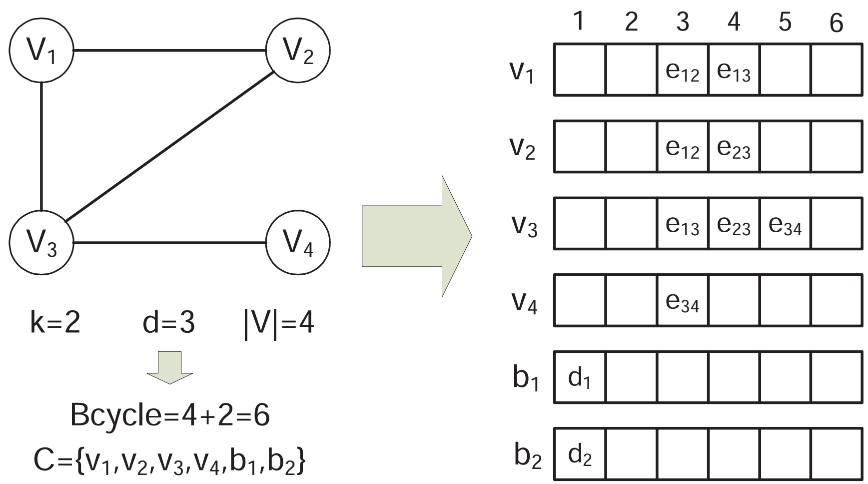
Decision Vector Cover: Given a graph $G = (V, E)$ and an integer k, does it have a vertex cover VC with size k.

Then we will construct an instance of MCDR from G and k.

Conversion Steps

- For each vertex $v_i \in V$, define a channel v_i. Define another k channels $b_1, \cdots, b_k$. Then the channel set is $C = \{v_1, \cdots, v_{|V|}, b_1, \cdots, b_k\}$. Totally $|V| + k$ channels. Let δ be the maximum vertex degree in G, then each channel has a broadcast cycle length of $\delta + 3$.
- For each edge $(v_i, v_j) \in E$, define a unit length data item e_{ij} in data set D_e, and append it on channel c_i and c_j (the order can be arbitrary, and starting from the third time unit).
- For each channel b_i, define a unit length data item d_i in data set D_d, and allocate it on the first time unit of channel b_i.
- The data set $D_q = D_e \cup D_b$.

An Example



Reduction Proof

Equivalence Relation: G has a vertex cover with size k iff there is a valid data retrieval schedule with $t = k(\delta + 3)$ and $h = 2k - 1$.

$\Longrightarrow$: If G has a vertex cover VC with size k, then we can select these k channels in $\{v_i \mid v_i \in VC\}$ to receive all the data in k cycles.

At i^{th} iteration, download b_i at $t = 1$, and hop to some $v_i \in VC$ channel, download needed data items, and then hop to b_{i+1}.

There are k b_i's, so in each iteration client will download one of them. VC is a vertex cover, so we can download every e_{ij}.

The length of each broadcast cycle is $\delta + 3$, totally $k(\delta + 3)$. In each iteration the client will switch twice (except the last cycle), so $h = 2k - 1$.

Reduction Proof (2)

$\Longleftarrow$: Assume MCDR has a valid schedule S with $t = k(\delta + 3)$ and $h = 2k - 1$.

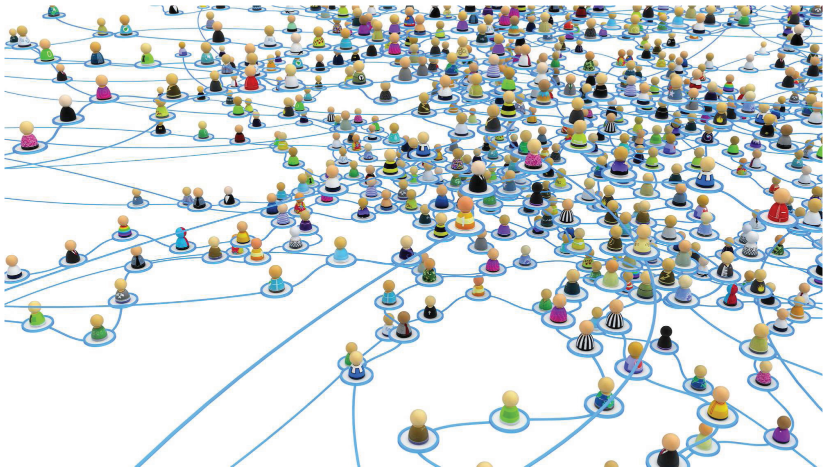
Consider D_b first. There are k b_i's located at the same position on k different channels $\Rightarrow$ have to switch $k - 1$ hops. Then we only have k hops for D_e $\Rightarrow$ can visit at most k channels in $\{v_i\}$.

At the beginning of each iteration, we stay at some b_i to download d_i, then switch to some v_i. At the end of this cycle, we have to switch to channel b_{i+1} for d_{i+1}. This means we cannot switch to two vertex channels within one broadcast cycle, otherwise we cannot download $D = D_e \cup D_b$ in k iterations.

Since S is valid, we visit k vertex channels and download all D_e data items, it means these k vertices form a vertex cover with size k. □

Social Network

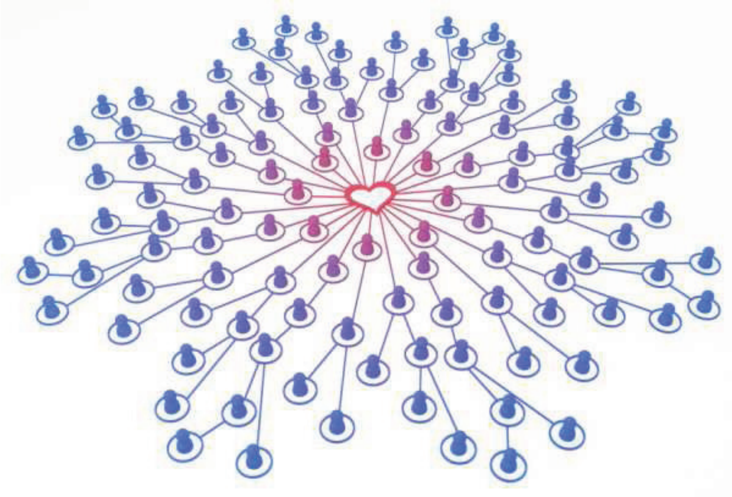
Social Network: a graph of relationships and interactions within a group of individuals.



From http://thenextweb.com/wp-content/blogs.dir/1/files/2013/11/social-network-links.jpg

Social Influence

Social Influence: ideas, information, opinions spread among the members in a social network.



Influence Maximization Problem

Influence Maximization Problem: Given a social network $G = (V, E)$ and k nodes are allowed to be activated initially, how do we select them in order to gain the maximum influence?

Decision Version: Given a social network $G = (V, E)$, a parameter k, and a threshold m, there exists a selection of k activated seeds to influence m members.

Influence Models

Linear Threshold model: A node i has a weight b_{ij} to influence node j and $\sum_{i \in N_j} b_{ij} \leq 1$ (if $(j, i) \notin E$, $b_{ij} = 0$). Node j is preassigned a threshold θ_j. At any single step, node j is successfully activated if the sum of weights from its active neighbors exceeds θ_j.

Independent Cascade model: If node i becomes active at step t, it has a probability p_{ij} to successfully activate each inactive neighbor j in step $t + 1$. Furthermore, whether or not i succeeds, it does not have any chances to activate j again.

Influence Maximization under Linear Threshold Model

Theorem: The Influence Maximization problem is NP-hard under Linear Threshold model.

Proof: VERTEX-COVER $\leq_p$ INFLUENCE-MAX

Given an instance of Vertex Cover with G and k, construct G' by directing all edges of G in both directions. For each node $v_i \in V$, $\theta_i = 1$. For each edge $(v_i, v_j) \in E$, $b_{ij} = 1/Indegree(v_j)$.

Equivalence Relation: G has a vertex cover with size k iff k seeds in G' influenced $|V|$ members.

$\Rightarrow$ If there is a vertex cover S of size k in G, then we can activate all nodes in G by selecting the nodes in S;
$\Leftarrow$ Conversely, this is the only way to activate all nodes in G.

Influence Maximization under Independent Cascade Model

Theorem: The Influence Maximization problem is NP-hard under the Independent Cascade model.

Proof: SET-COVER $\leq_p$ INFLUENCE-MAX

Given an instance of Set Cover with $U = \{u_1, \cdots, u_m\}$, $\mathbf{S} = \{S_1, \cdots, S_n\}$, and k, define a directed bipartite graph with $n + m$ nodes: a node i for each set S_i, a node j for each element u_j, and a directed edge (i, j) with activation probability $p_{ij} = 1$, whenever $u_j \in S_i$.

Equivalence Relation: U has a set cover with size k iff there is a set A of k nodes which can active n elements.

Proof

$\Rightarrow$: Note that for the instance we have defined, activation is a deterministic process, as all probabilities are 0 or 1. Initially activating the k nodes corresponding to sets in a Set Cover solution results in activating all n elements corresponding to the ground set U.

$\Leftarrow$: If any set A of k nodes can active n elements, then the Set Cover problem must be solvable.