Scheduling in switching networks

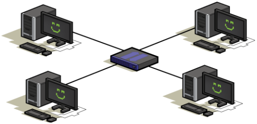
Leonidas Tsepenekas

National Technical University of Athens

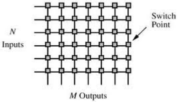
June 10, 2015

Overview

A simple example



- We have two unique roles. The receivers and the transmitters.
- All communication stream goes through the black box.
- The black box is actually a crossbar network.



- The overall traffic is already known and it consists of messages between transmitter i and receiver j with known duration.
- Construct transmission frames.
- Objective: Minimize the total transmission time, with regards to the switching cost.

Mathematical formulation (Graph Notation)

- The system can be seen as a bipartite graph $G(U, V, E, w)$.
- Every transmitter corresponds to a node of U.
- Every receiver corresponds to a node of V.
- A message between transmitter u and receiver v corresponds to an edge $(u, v) \in E$.
- The transmission time required by a communication task $e \in E$ corresponds to $w(e)$.
- We also consider d to be the cost of reconfiguring the crossbar switch.

Objective

Find a collection $\{M_1, M_2, \ldots, M_s\}$ of s matchings (schedule) such that
$\forall i, j, i \neq j : M_i \cap M_j = \varnothing$, $\bigcup_{i=1}^{s} M_i = E$ and $\sum_{i=1}^{s} w(M_i) + s \cdot d$ is minimized.
Where $w(M_i) = max\{w(e) \mid e \in M_i\}$.

Important Results

- The problem in known to be NP-complete.
- It is also known to be $\frac{4}{3} - \epsilon$ inapproximable $\forall \epsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$.
- If $d = 0$ then is proven to be solvable in polynomial time.
- If all the edges of the graph are of the same weight then again it is solvable in polynomial time, since it is equivalent to the bipartite edge coloring problem.
- Many clever approximation algorithms developed, as well as a very good performing heuristic.

- It is based on a heuristic.
- Their main concern was to minimize the number of switchings.

Lower bound on the number of switchings

$B = max\ (\Delta(G),\ \lceil |E|/K \rceil)$, where K is the number of the available transponders.

- The algorithm indeed achieves a schedule with B switchings.

The Gopal-Wong algorithm 2/3

Construct a B-regular graph by adding new vertices and edges to G.
Assign zero weight to the newly added edges.
Sort the edges in ascending order, $e_1, e_2, \ldots, e_{|E'|}$, according to weight.
$i \leftarrow 1,\ j \leftarrow 1$
$P \leftarrow \{e_1\},\ Q \leftarrow \{e_1\}$
while $j < B$ **do**
  while *Q is not a perfect matching* **do**
    $P \leftarrow P \cup e_i$
    if *there is an augmenting path for Q in P* **then**
      augment Q
    else
      $i \leftarrow i + 1$
    end
  end
  $P \leftarrow P \cap Q,\ Q \leftarrow \{\}$
  $j \leftarrow j + 1$
end

The Gopal-Wong algorithm 3/3

- The main intuition behind the algorithm is to group together messages of the same magnitude.
- The Gopal-Wong algorithm has unbounded approximation ratio.

Proof

Assume $|V| = |U| = 2n + 1$.
Consider also the following weight assignments: $\forall i \in [1, n+1]$,
$\forall j \in [1, 2(n-i)+2] : w(i, 2i-1) = M > 1,\ w(i+j, 2i-1+j) = 1$
Otherwise $w(i,j) = 0$. If run on this graph, the algorithm gives $n+1$ matchings each containing one edge with weight M. Thus the total duration time is $(n+1)M$. The optimal is achieved when we take all the edges of cost M in one matching and pack the rest into n matchings. So the optimal cost is $M + n$ and the approximation ratio is $\frac{(n+1)M}{M+n} \approx n+1$, as M goes to infinity.

The A-PBS algorithm 1/2

- Published by Afrati, Aslanidis, Bampis and Milis.
- It is based on the idea of preemption.
- Define $W(u) = \sum_{\{u,v\} \in E} w(u,v)$ and $W(G) = max\{W(u) \mid u \in G\}$.

The algorithm

(a) Round up the weight of every edge of the initial graph $G = (U \cup V, E, w)$ to a multiple of a given value α. Call the obtained graph $G' = (U \cup V, E, w')$.

(b) Split every edge e_{ij} of G' into $w'(e_{ij})/\alpha$ edges having each a weight equal to α. Call the induced polygraph G_α.

(c) Find exactly $\frac{W(G_\alpha)}{\alpha}$ matchings in G_α, covering all of its edges.

The A-PBS algorithm 2/2

- A straight forward lower bound of the problem is $W(G) + d \cdot \Delta(G)$.

A-PBS(d+1) has approximation ratio $2 - \frac{1}{d+1}$

- The cost of the algorithm is $\frac{W(G_\alpha)}{\alpha} \cdot \alpha + \frac{W(G_\alpha)}{\alpha} \cdot d$.
- Because of the rounding $W(G_\alpha) \leq W(G) + (\alpha - 1)\Delta(G)$.
- Thus, the total cost is bounded above by $W(G) + (\alpha - 1)\Delta(G) + \frac{W(G)+(\alpha-1)\Delta(G)}{\alpha} \cdot d = \frac{d+\alpha}{\alpha} \cdot (W(G) + (\alpha - 1)\Delta(G))$.
- Given that $\alpha = d + 1$ we get $cost \leq \frac{2d+1}{d+1}(W(G) + d \cdot \Delta(G)) = (2 - \frac{1}{d+1}) \cdot OPT$.

The Spilt-Graph algorithm 1/3

SGA

(a) Split the initial graph $G(U, V, E)$ in two graphs $G_M(U, V, E_M)$ and $G_m(U, V, E_m)$, where $E_m = \{e \mid e \in E, w(e) < d\}$ and $E_M = \{e \mid e \in E, w(e) \geq d\}$.

(b) Use *Routine 1* to find a maximal matching M in G_M.

(c) Use *Routine 2* to calculate the weight of the matching to be removed. Remove the corresponding parts of the edges.

(d) Add edges to M from E_m to maximize its cardinality and remove them form E_m.

(e) From the induced graph move all edges of weight less than d to E_m.

(f) Repeat until $E_M = \varnothing$.

(g) Calculate Δ_m matchings in G_m, where Δ_m is the maximum degree.

The Spilt-Graph algorithm 2/3

Routine 1

(a) $M = \varnothing$

(b) Sort all nodes in decreasing order according to their work $(W(u))$. Let L be the induced list.

(c) Let x be the first node in L. Find his first neighbour in L, say y, and increase $M \leftarrow M \cup \{x, y\}$. Remove x, y from L.

(d) Repeat until M is maximal.

The idea behind the above routine is to reduce as much as possible the workload of G_M.

The Spilt-Graph algorithm 3/3

Routine 2

(a) For each $e \in M$, with corresponding weight $w(e)$, calculate what the value $W(G')$ would be if all the edges in the matching were to be reduce be $w(e)$. Edges with weight less than $w(e)$ are completely removed. Then set:

$$r(e) = \begin{cases} w(e) & \text{if } W(G') = W(G) - w(e) \\ 0 & \text{otherwise} \end{cases}$$

(b) Calculate $R = max\{r(e) \mid e \in M\}$.

(c) For each edge $e \in M$ set its new weight

$$w'(e) = \begin{cases} w(e) - R & \text{if } w(e) > R \\ 0 & \text{otherwise} \end{cases}$$

A polynomial solution to the problem 1/2

Definition

A weighted bipartite graph will be called *unvarying* if the number of edges of any specific weight w incident to any node u is less than or equal to the number of edges of the same weight, which are adjacent to the node with the maximum degree.

A-PBS-UN

(a) Split the graph in $|W|$ graphs $G_1, G_2, \ldots, G_{|W|}$, each having edges of the same weight, where $|W|$ is the number of different edge weights appearing in G.

(b) Find exactly $\Delta(G_i)$ matchings in G_i so that the union of these matchings covers all edges of G_i.

(c) The union of the $|W|$ matchings found is an optimal solution.

A polynomial solution to the problem 2/2

A-PBS-UN is optimal

- Let $G_1, G_2, \ldots, G_a$ be the unique weight graphs.
- Let u with $d(u) = \Delta(G)$. G unvarying $\implies$ there are exactly $\Delta(G_i)$ edges of weight i incident to u. Thus
$$\Delta(G) = \sum_{i=1}^{a} \Delta(G_i) \text{ and } W(G) \geq W(u) = \sum_{i=1}^{a} i \cdot \Delta(G_i).$$
- The cost of each graph is $d \cdot \Delta(G_i) + i \cdot \Delta(G_i)$.
- The total cost is $\sum_{i=1}^{a} (d \cdot \Delta(G_i) + i \cdot \Delta(G_i)) \leq d \cdot \Delta(G) + W(G)$

Experimental results