

Balancing Slacks

Stephan Held

Research Institute for Discrete Mathematics
University of Bonn

NATO Summer School in Combinatorial Optimization
June 2006

- 1 Clock Scheduling in Practice
 - Late, Early and Window Optimization
 - Graph-Models
 - Slack Stealing
- 2 Netlength Bounds
 - Problem Description
 - General Slack Balance Problem
 - Link to Minimum Balance Problem

Slack Targets

- Large instances have up to 10 Mio nodes and 50 Mio edges.
- The runtime of $O(nm + n^2 \log n)$ sounds high for VLSI-instances.
- The algorithm is stopped when a certain **slack target** is reached.
- This way reasonable runtimes can be reached.

Late, Early and Time Window Optimization

- Early mode slacks can also be removed by inserting delay buffers ([early mode padding](#)).
- [Late slacks should be prioritized!](#)
- Remaining slack (after reaching the target) should be used to compute [clock arrival time windows](#):
 - Makes clock tree construction easier.
 - Enables more power efficient trees.

Late & Early Optimization

Late Mode Optimization

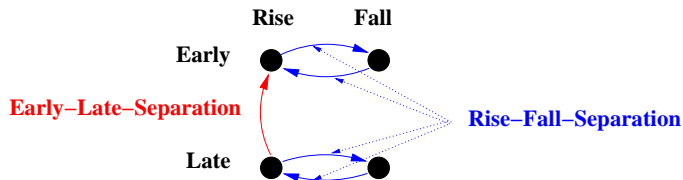
- Remove early mode test arcs.
- Balance slacks until a late slack target l_{target} is reached.

Early Mode Optimization

- Keep optimized late slacks below l_{target} fixed.
- Reduce costs of remaining late slack arcs by l_{target} and remove them from E_w .
 - These late slacks are guaranteed to stay above l_{target} .
 - All cycles have non-negative total costs now.
 - $E_w = \emptyset$.
- Add early mode test arcs to $E(G)$ and E_w .
- Balance slacks until an early slack target e_{target} is reached.

Time Window Optimization

- Optimize and fix late and early slacks below their slack targets.
- Fix remaining slack such that targets are guaranteed.
- Use the **early-late-separation** edges as slack edges E_w :



- The time windows for a clock pin c are given by the balanced early and late ATs:

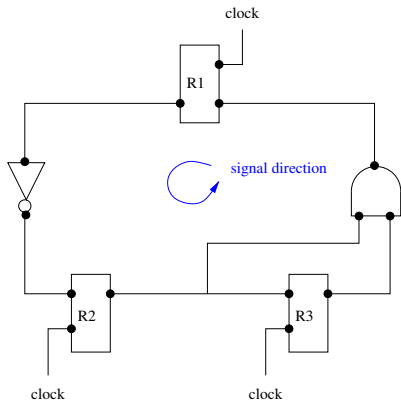
$$[\underline{AT}_{rise}(c), \overline{AT}_{rise}(c)] \text{ and } [\underline{AT}_{fall}(c), \overline{AT}_{fall}(c)]$$

3 runs with different objectives:

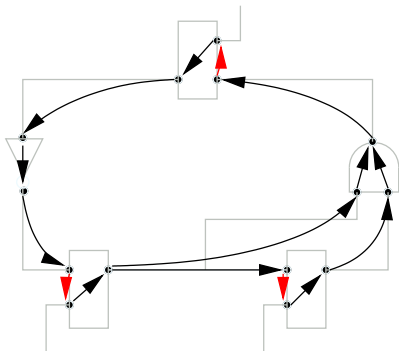
- ① **Late mode slacks (Setup)**
Slacks that **cannot** be removed by padding
- ② **Early mode slacks (Hold)**
Slacks that **can** be removed by padding
Preserve Late Slacks from 1.
- ③ **Time windows for each sink**
Enables power saving clocktrees
Preserve Slacks from 1. & 2.

- The **slack balance graph** G_B represents endpoint constraints from static timing.
- In the introduction we used the **register-graph**, here longest (late-mode) and shortest (early-mode) paths between registers are represented.
- The arcs of the register graph can be computed by propagating ATs individually from every start point (register or PI).

CycleOpt Graph Models

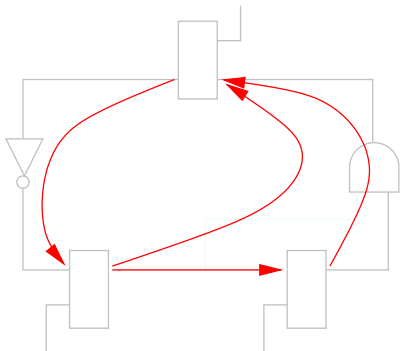


Timing Graph Model



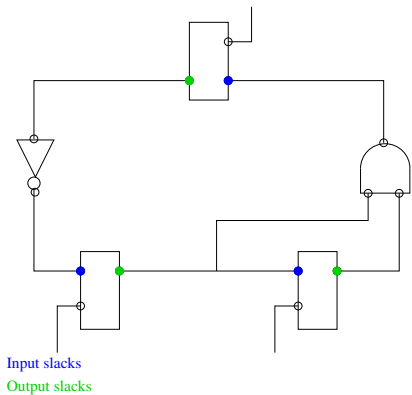
- Endpoint slacks are optimized.
- Any path that ends at the worst endpoint will get the same worst slack.
- Graph size is linear in netlist size.

Register Graph Model



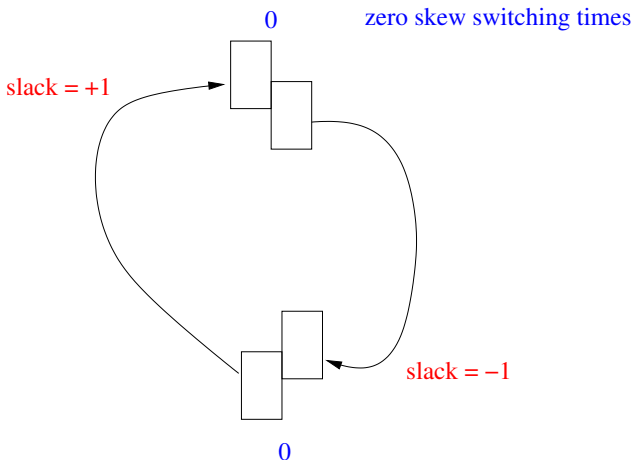
- Register to register slacks are optimized.
⇔ Path slacks are optimized.
- Less critical paths than in the timing graph model
- Graph size is bounded quadratically in netlist size
In practice: number of edges $\approx 20 - 60 \times$ number of registers.

Slack Stealing



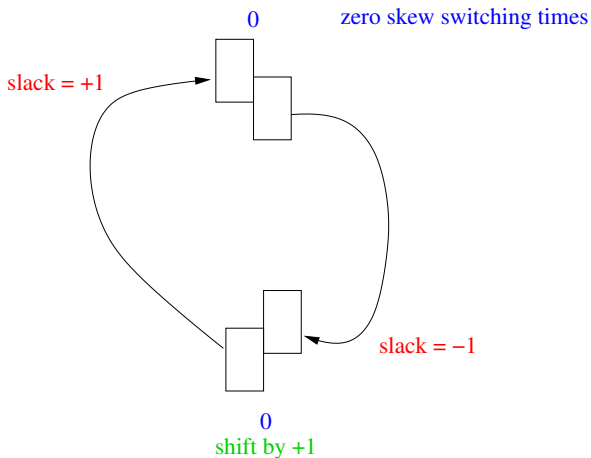
- Clock-ATs are shifted locally according to input and output slacks

Principle



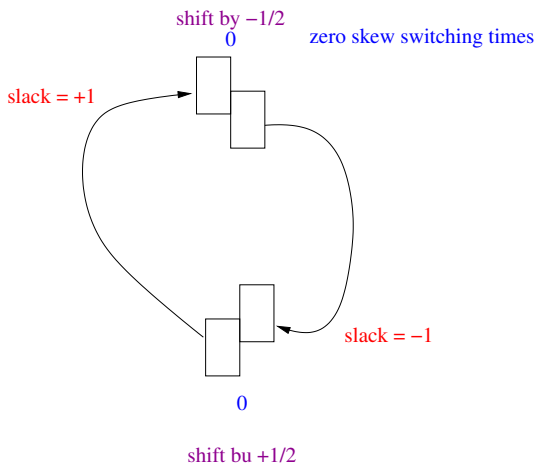
The locally optimum shift for a register is $(\text{out_slack} - \text{in_slack})/2$.

Shift one Register



Here the cycle would be balanced when shifting the lower register locally.

Shift all Registers



Shifts of other registers must be anticipated when all registers are shifted simultaneously.

⇒ Perform half the locally optimum shifts.

Alternative: Do shift all registers with same direction only.

Efficiency of Slack Stealing

- The worst slack is maximized by this iterative Jacobi-like method.
- In practice this happens after 4-10 iterations.
- No guarantee but good results in practice for less critical slacks.
- In contrast to the graph-models the **ATs in the critical cycles are fixed instead of the slacks.**
- Works directly on top of timing engine. No external graph needed.
- Overnight runtimes with the graph model could be reduced to 20 minute runs without measurable degeneration of quality.
- Results can even be better than in the timing-graph model.

Hybrid Balancing Model

Main disadvantages of Slack-Stealing

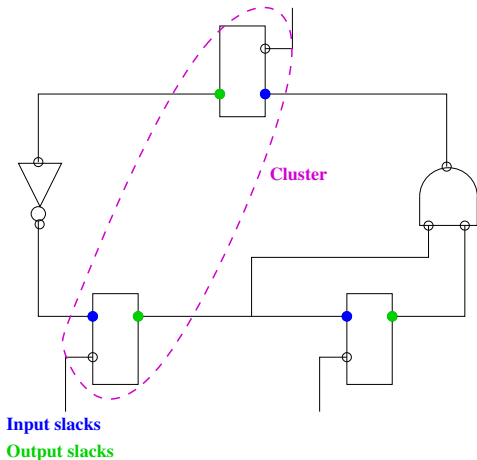
No simultaneous shifting of cycles. \Rightarrow

- 1 No lexicographically maximum solution
- 2 Poor early mode optimization on some instances

Hybrid Graph Model

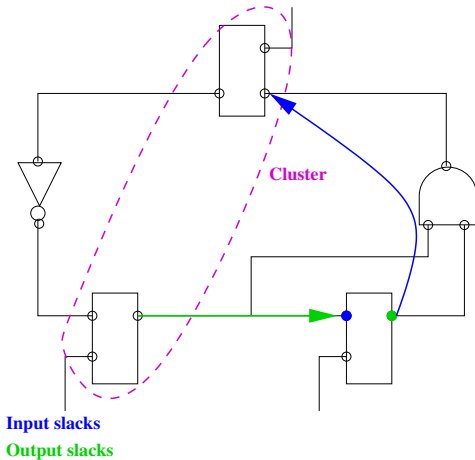
- Combine **efficiency** of **Slack-Stealing** with **quality** of **Register-Graph Model**.
- Create Register-Graph Model on critical parts.

Clustering



- AT-relations inside cluster must be fixed
- Slacks between Cluster and other latches have to be optimized

Cluster Implementation



- Timing point slacks within cluster are ignored
- Slack arcs to exterior registers (or clusters) are introduced

- 1 Clock Scheduling in Practice
 - Late, Early and Window Optimization
 - Graph-Models
 - Slack Stealing
- 2 Netlength Bounds
 - Problem Description
 - General Slack Balance Problem
 - Link to Minimum Balance Problem

Generation of Netlength Bounds

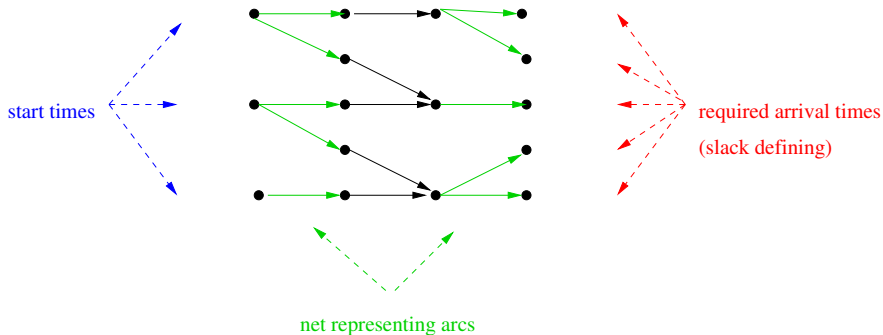
- Assume all slacks to be positive.
- Final placement or routing usually require detours of nets,
 - ⇒ netlength increases
 - ⇒ delay increases
 - ⇒ timing restrictions may become violated
- We are interested in bounds for detours that guarantee timing requirements.
- The delay of a net N is (approximately) a linear function in the additional detour.

$$\overline{\text{DELAY}}(N) \approx \overline{\text{DELAY}}_{fix}(N) + \alpha_N \text{detour}(N).$$

with

$$\alpha_N = \frac{\partial \overline{\text{DELAY}}}{\partial \text{detour}}(N)$$

- Consider an acyclic graph G representing combinatorial logic



- Timing requirement on bounds:

$$\overline{\text{SLACK}}(P) \geq \sum_{N \in P} \alpha_N \overline{\text{detour}}(N) \quad \forall \text{ paths } P \in G \quad (1)$$

- Requirements on proper detour bounds [Shragowitz, Youssef, Lin, Lu 1992,2003]:

- On an individual path P distribute the detour bounds equally

$$\overline{\text{detour}}(N) = \frac{\overline{\text{SLACK}}(P)}{\sum_{N \in P} \alpha_N} \quad \forall N \in P. \quad (2)$$

- And in a network:

$$\overline{\text{detour}}(N) = \min_{P \text{ path containing } N} \frac{\overline{\text{SLACK}}(P)}{\sum_{N \in P} \alpha_N}. \quad (3)$$

- This problem turns out to be \mathcal{NP} -complete (path enumeration).
- Only a heuristic having exponential worst case running time is known.
- On examples the heuristic does not converge to the correct solution.

Problem Relaxation

- We can relax the problem accepting larger bounds (these are better bounds).

Worst Path Constraints:

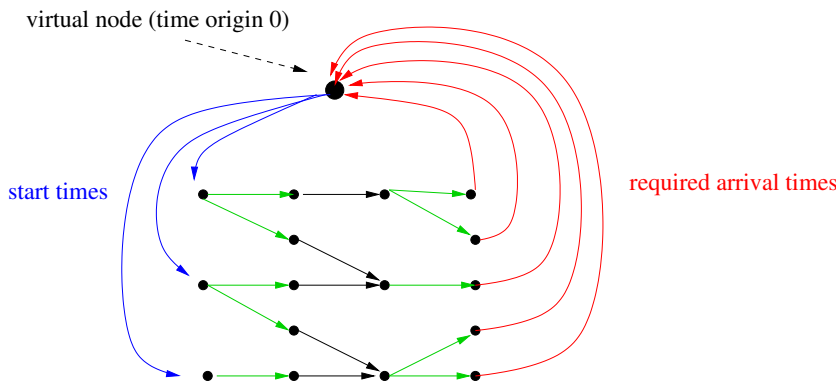
$$\overline{\text{detour}}(N) \geq \min_{P \text{ path containing } N} \frac{\overline{\text{SLACK}}(P)}{\sum_{N \in P} \alpha_N} \quad \forall \text{ nets } N. \quad (4)$$

Timing constraints:

$$\overline{\text{SLACK}}(P) \geq \sum_{N \in P} \alpha_N \overline{\text{detour}}(N) \quad \forall \text{ paths } P \in G. \quad (5)$$

- This problem can be solved in strongly polynomial time (S.H.).

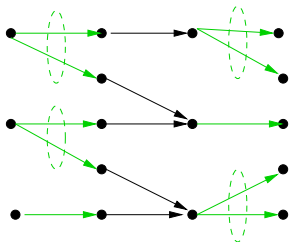
Model



This way IO constraints are modeled for clock scheduling, too.

Further Improvement

- Some nets are represented by multiple edges.



- If a net is represented by multiple edges only the most critical edge is of interest.

General Slack Balance Problem (GSBP)

Instance:

- directed graph G , $c : E(G) \rightarrow \mathbb{R}$
- a set $F_0 \subseteq E(G)$ and a partition \mathcal{F} of $E(G) \setminus F_0$
- weights $w : E(G) \setminus F_0 \rightarrow \mathbb{R}_{>0}$

Task:

Find a potential $\pi : V(G) \rightarrow \mathbb{R}$ with

$$c_\pi(e) := c(e) + \pi(x) - \pi(y) \geq 0 \text{ for } e = (x, y) \in F_0$$

such that the vector

$$\left(\min \left\{ \frac{c(e) + \pi(x) - \pi(y)}{w(e)} \mid e = (x, y) \in F \right\} \right)_{F \in \mathcal{F}}$$

(after sorting entries in non-decreasing order) is lexicographically maximal.

Lemma

Let $C \subset G$ be a cycle with $w(C) > 0$ and *minimum weighted costs* $c(C)/w(C)$ and let π^* be an optimal solution of GSBP. Then

$$\frac{c(C)}{w(C)} = \min \left\{ \frac{c_{\pi^*}(e)}{w(e)} \mid e \in E(G) \setminus F_0 \right\}.$$

Proof.

Analog to SBP. □

Solving the SDP II

```
1 while  $\mathcal{F} \neq \emptyset$  do  
2   Compute Minimum Ratio Cycle  $C$  in  $(G, c, w)$ ;  
3    $\lambda \leftarrow \frac{c(C)}{w(C)}$ ;  
4   /* Fix costs of all involved partition sets: */  
5   foreach  $F \subset \mathcal{F}$  with  $E(C) \cap F \neq \emptyset$  do  
6     foreach  $f \in F$  do  
7        $c(f) \leftarrow c(f) - \lambda w(f)$ ;  
8        $w(f) \leftarrow 0$ ;  
9     end  
10  end  
11 end
```

Algorithm 1: Cost Fixation

Solving the SDP II

```
1 while  $\mathcal{F} \neq \emptyset$  do
2   Compute Minimum Ratio Cycle  $C$  in  $(G, c, w)$ ;
3    $\lambda \leftarrow \frac{c(C)}{w(C)}$ ;
4   /* Fix costs of all involved partition sets: */
5   foreach  $F \subset \mathcal{F}$  with  $E(C) \cap F \neq \emptyset$  do
6     foreach  $f \in F \setminus E(C)$  do
7        $c(f) \leftarrow c(f) - \lambda w(f)$ ;
8        $w(f) \leftarrow 0$ ;
9     end
10  end
11  Contract  $C$  and adopt costs of  $\delta^-(C) \cup \delta^+(C)$ ;
12 end
```

Algorithm 2: Cycle Contraction

Computing the Minimum Ratio Cycle

Parametric Shortest Path Algorithm

- Works also for general weights $w : E(G) \rightarrow \mathbb{R}_{\geq 0}$.
- Runtime: $O(w_{\max}(nm + n^2 \log n))$,
where $w_{\max} = \max\{w(e) | e \in E(G)\}$.

Megiddo's Minimum Ratio Cycle Algorithm

- The Minimum Ratio Cycle Problem is solvable in $O(n^3 \log n + \min(nm, n^3) \log^2 n \log \log n + nm \log m)$
- This gives a runtime for GSBP:
 $O(n(n \log n + \min(nm, n^3) \log^2 n \log \log n + nm \log m))$
- The memory demand is bounded by $\Omega(n^2)$ and $O(n^3)$.
- Linear memory consumption would be possible at higher runtime.

Main Idea by Megiddo

Theorem (Megiddo 1979)

Let $X \subset \mathbb{R}^n$, $a = (a_0, a_1, \dots, a_n)$, $b = (b_0, b_1, \dots, b_n) \in \mathbb{R}^{n+1}$ and $c = (c_1, \dots, c_n) \in \mathbb{R}^n$.

Let there be two optimization problems on D
(Problem A:)

$$\text{Minimize} \quad c_1 x_1 + \dots + c_n x_n, \quad (6)$$

$$\text{s.t.} \quad x = (x_1, \dots, x_n) \in D, \quad (7)$$

and

(Problem B:)

$$\text{Minimize} \quad \frac{a_0 + a_1 x_1 + \dots + a_n x_n}{b_0 + b_1 x_1 + \dots + b_n x_n}, \quad (8)$$

$$\text{s.t.} \quad x = (x_1, \dots, x_n) \in D. \quad (9)$$

If Problem A is solvable with $O(q_v(n))$ comparisons and $O(q_a(n))$ additions, then Problem B is solvable in $O(q_v(n)(q_a(n) + q_v(n)))$. Here $q_a, q_v : \mathbb{N} \rightarrow \mathbb{N}_{\geq 0}$.

Main Idea by Megiddo

Notation

We call an algorithm for *Problem A* an A-algorithm and an algorithm for *Problem B* an B-algorithm.

Lemma

Given *Problem B* define parameterized costs by

$$c_i(\lambda) = a_i - \lambda b_i, \quad i = 1, \dots, n \text{ for a parameter } \lambda.$$

If $\text{opt}_A(\lambda) = \min(c_1(\lambda) \cdot x_1 + \dots + c_n(\lambda) \cdot x_n)$ is a solution of *Problem A* w.r.t costs $c_i(\lambda)$ we can differentiate 3 cases:

- 1 $\text{opt}_A(\lambda) = \lambda b_0 - a_0 \implies \text{opt}_B = \lambda.$
- 2 $\text{opt}_A(\lambda) < \lambda b_0 - a_0 \implies \text{opt}_B < \lambda.$
- 3 $\text{opt}_A(\lambda) > \lambda b_0 - a_0 \implies \text{opt}_B > \lambda.$

Proof.

Let λ^{opt} be optimum solution for Problem B.

Proof.

Let λ^{opt} be optimum solution for Problem B. Then:

$$\lambda^{opt} \leq \frac{a_0 + a_1x_1 + \cdots + a_nx_n}{b_0 + b_1x_1 + \cdots + b_nx_n}, \quad \forall x = (x_1, \dots, x_n) \in D.$$

Proof.

Let λ^{opt} be optimum solution for Problem B. Then:

$$\lambda^{opt} \leq \frac{a_0 + a_1x_1 + \cdots + a_nx_n}{b_0 + b_1x_1 + \cdots + b_nx_n}, \quad \forall x = (x_1, \dots, x_n) \in D.$$

$$\begin{aligned} \Leftrightarrow c_{\lambda^{opt}}(x) &= a_0 - \lambda^{opt} \cdot b_0 + \sum_{i=1, \dots, n} a_i \cdot x_i - \lambda^{opt} \sum_{i=1, \dots, n} b_i \cdot x_i \\ &\geq 0, \quad \forall x = (x_1, \dots, x_n) \in D. \end{aligned}$$



Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.

Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.
- Instead of exact λ and costs c_λ perform the algorithm with parameterized linear functions c_λ , $\lambda \in [l, u]$ where the initial $[l, u]$ is large enough.

Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.
- Instead of exact λ and costs c_λ perform the algorithm with parameterized linear functions c_λ , $\lambda \in [l, u]$ where the initial $[l, u]$ is large enough.
- The algorithm decreases the size of $[l, u]$ s. t. all temporarily computed functions stay linear within $[l, u]$.

Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.
- Instead of exact λ and costs c_λ perform the algorithm with parameterized linear functions c_λ , $\lambda \in [l, u]$ where the initial $[l, u]$ is large enough.
- The algorithm decreases the size of $[l, u]$ s. t. all temporarily computed functions stay linear within $[l, u]$.
- An addition within the A-algorithm results in a linear function again.

Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.
- Instead of exact λ and costs c_λ perform the algorithm with parameterized linear functions c_λ , $\lambda \in [l, u]$ where the initial $[l, u]$ is large enough.
- The algorithm decreases the size of $[l, u]$ s. t. all temporarily computed functions stay linear within $[l, u]$.
- An addition within the A-algorithm results in a linear function again.
- Whenever the A-algorithm makes a comparison, two functions f_1, f_2 which are linear in $[u, l]$ are compared.

Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.
- Instead of exact λ and costs c_λ perform the algorithm with parameterized linear functions c_λ , $\lambda \in [l, u]$ where the initial $[l, u]$ is large enough.
- The algorithm decreases the size of $[l, u]$ s. t. all temporarily computed functions stay linear within $[l, u]$.
- An addition within the A-algorithm results in a linear function again.
- Whenever the A-algorithm makes a comparison, two functions f_1, f_2 which are linear in $[u, l]$ are compared.
- The result depends on a potential intersection $\lambda' \in [l, u]$ of f_1, f_2 .

Main Idea by Megiddo

- Simulate a run of the A-algorithm as if the optimum λ^{opt} would be known.
- Instead of exact λ and costs c_λ perform the algorithm with parameterized linear functions c_λ , $\lambda \in [l, u]$ where the initial $[l, u]$ is large enough.
- The algorithm decreases the size of $[l, u]$ s. t. all temporarily computed functions stay linear within $[l, u]$.
- An addition within the A-algorithm results in a linear function again.
- Whenever the A-algorithm makes a comparison, two functions f_1, f_2 which are linear in $[u, l]$ are compared.
- The result depends on a potential intersection $\lambda' \in [l, u]$ of f_1, f_2 .
- Now: call the A-algorithm with exact $c_{\lambda'}$, to decide whether to continue with the interval $[l, \lambda']$ or $[\lambda', u]$.

Special Cases of GSBP

- $\mathcal{F} = \{E(G)\}$: Minimum Ratio Cycle Problem
- $\mathcal{F} = \{\{e_1\}, \{e_2\}, \dots, \{e_{|E(G)|}\}\}$ and $w : E(G) \rightarrow \{0, 1\}$:
Slack Balance Problem
- $\mathcal{F} = \{\{e_1\}, \{e_2\}, \dots, \{e_{|E(G)|}\}\}$ and $w \equiv 1$:
Minimum Balance Problem (Schneider & Schneider)

Minimum Balanced Problem

Schneider & Schneider

Instance:

- directed graph G , $c : E(G) \rightarrow \mathbb{R}$

Task:

Find a potential $\pi : V(G) \rightarrow \mathbb{R}$, s.t.

$$\min\{c_\pi(e) \mid e \in \delta^-(X)\} = \min\{c_\pi(e) \mid e \in \delta^+(X)\} \quad \forall X \subset V(G)$$

General Slack Balance Problem – Alternative Formulation

Theorem (Vygen 2003)

Let (G, c, w, \mathcal{F}) and instance of the General Slack Balance Problem.

Let $\pi : V(G) \rightarrow \mathbb{R}$ with $c_\pi(e) \geq 0$ for $e \in F_0$.

Let

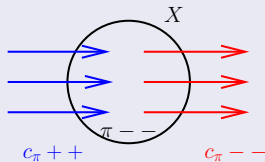
$$E_\pi := \left\{ f \in F \in \mathcal{F} \mid \frac{c_\pi(f)}{w(f)} \text{ minimal in } F \right\}.$$

Then π is an optimum solution with E_π minimal iff for each $X \subset V(G)$:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)}$$

or

$$\min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0.$$



General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.
- Chose f such that $\frac{\min\{c_\pi(f), c_{\pi'}\}}{w(f)}$ is minimum.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.
- Chose f such that $\frac{\min\{c_\pi(f), c_{\pi'}(f)\}}{w(f)}$ is minimum.
- W.l.o.g. $\pi(p) - \pi(q) < \pi'(p) - \pi'(q)$.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.
- Chose f such that $\frac{\min\{c_\pi(f), c_{\pi'}\}}{w(f)}$ is minimum.
- W.l.o.g. $\pi(p) - \pi(q) < \pi'(p) - \pi'(q)$.
- Define Q as the set of vertices reachable from q by edges $e \in F_0$ with $c_\pi(e) = 0$ or edges $e \in \bigcup \mathcal{F}$ with $\frac{c_\pi(e)}{w(e)} \leq \frac{c_\pi(f)}{w(f)}$.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.
- Chose f such that $\frac{\min\{c_\pi(f), c_{\pi'}\}}{w(f)}$ is minimum.
- W.l.o.g. $\pi(p) - \pi(q) < \pi'(p) - \pi'(q)$.
- Define Q as the set of vertices reachable from q by edges $e \in F_0$ with $c_\pi(e) = 0$ or edges $e \in \bigcup \mathcal{F}$ with $\frac{c_\pi(e)}{w(e)} \leq \frac{c_\pi(f)}{w(f)}$.
- By (\star) $p \in Q$.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (\star)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.
- Chose f such that $\frac{\min\{c_\pi(f), c_{\pi'}(f)\}}{w(f)}$ is minimum.
- W.l.o.g. $\pi(p) - \pi(q) < \pi'(p) - \pi'(q)$.
- Define Q as the set of vertices reachable from q by edges $e \in F_0$ with $c_\pi(e) = 0$ or edges $e \in \bigcup \mathcal{F}$ with $\frac{c_\pi(e)}{w(e)} \leq \frac{c_\pi(f)}{w(f)}$.
- By (\star) $p \in Q$.
- \exists q - p -path P that consists of edges $e = (x, y)$ with $c(e) + \pi(x) - \pi(y) \leq c(e) + \pi'(x) - \pi'(y)$.

General Slack Balance Problem – Alternative Formulation

Proof.

To show: π is optimum solution of GSBP if $c_\pi(e) \geq 0$ for $e \in F_0$ and:

$$\min_{e \in E_\pi \cap \delta^-(X)} \frac{c_\pi(e)}{w(e)} \geq \min_{e \in E_\pi \cap \delta^+(X)} \frac{c_\pi(e)}{w(e)} \text{ or } \min_{e \in \delta^+(X) \cap F_0} c_\pi(e) = 0. \quad (*)$$

- Let $\pi, \pi' : V(G) \rightarrow \mathbb{R}$ two such potentials.
- Assume $\exists f = (p, q) \in \bigcup \mathcal{F}$ with $\pi(p) - \pi(q) \neq \pi'(p) - \pi'(q)$.
- Chose f such that $\frac{\min\{c_\pi(f), c_{\pi'}\}}{w(f)}$ is minimum.
- W.l.o.g. $\pi(p) - \pi(q) < \pi'(p) - \pi'(q)$.
- Define Q as the set of vertices reachable from q by edges $e \in F_0$ with $c_\pi(e) = 0$ or edges $e \in \bigcup \mathcal{F}$ with $\frac{c_\pi(e)}{w(e)} \leq \frac{c_\pi(f)}{w(f)}$.
- By (*) $p \in Q$.
- \exists q - p -path P that consists of edges $e = (x, y)$ with $c(e) + \pi(x) - \pi(y) \leq c(e) + \pi'(x) - \pi'(y)$.
- Summation yields $\pi(q) - \pi(p) \leq \pi'(q) - \pi'(p)$!

