Successive Shortest Path (SSP) Algorithm with Multipliers

Birgit Engels

ZAIK
University of Cologne

13th Combinatorial Optimization Workshop
January 11th-17th, Aussois

Motivated by a joint project of:

A freight car disposition problem

Complexity of integral flow with multipliers 1,2

Instances with fractional vs. halfintegral solutions

A modified SSP algorithm

Obtaining an integral solution

basic problem formulation

input:

- known supplies/demands of empty freight cars ($10^3 - 10^4$)

supply

demand

basic problem formulation

input:

- known supplies/demands of empty freight cars ($10^3 - 10^4$)
- both of different *types* at different *locations* and *times*

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00

22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation

input:

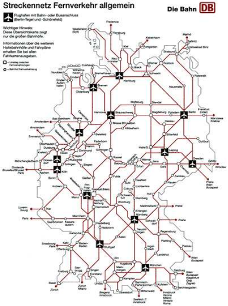
- known supplies/demands of empty freight cars ($10^3 - 10^4$)
- both of different *types* at different *locations* and *times*
- timetable (time constraints, costs)

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation

input:

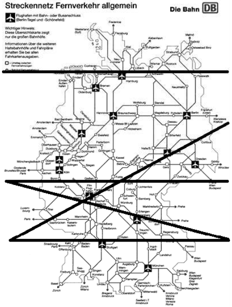
- known supplies/demands of empty freight cars ($10^3 - 10^4$)
- both of different *types* at different *locations* and *times*
- timetable (time constraints, costs)
- type subtitution rules (1:1,1:2)

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation

input:

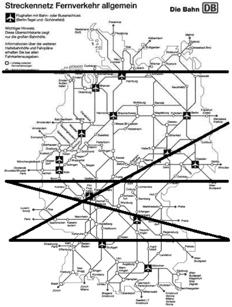
- known supplies/demands of empty freight cars ($10^3 - 10^4$)
- both of different *types* at different *locations* and *times*
- timetable (time constraints, costs)
- type subtitution rules (1:1,1:2)

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation

input:

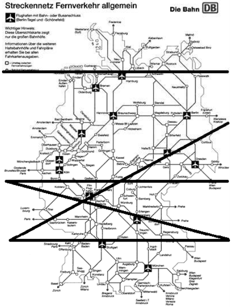
- known supplies/demands of empty freight cars ($10^3 - 10^4$)
- both of different *types* at different *locations* and *times*
- timetable (time constraints, costs)
- type subtitution rules (1:1,1:2)
- side constraints (storage, priority, etc)

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation (1)

output: optimal disposition, i.e.

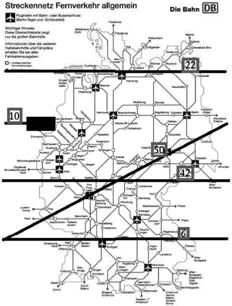
- allocation of all supply to as much demand as possible

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation (1)

output: optimal disposition, i.e.

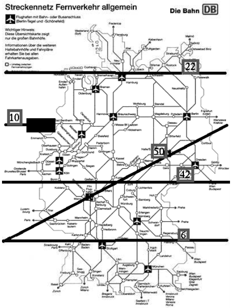
- allocation of all supply to as much demand as possible
- respecting all rules/constraints, integrality

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation (1)

output: optimal disposition, i.e.

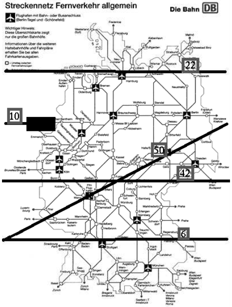
- allocation of all supply to as much demand as possible
- respecting all rules/constraints, integrality
- minimal costs

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation (1)

output: optimal disposition, i.e.

- allocation of all supply to as much demand as possible
- respecting all rules/constraints, integrality
- minimal costs

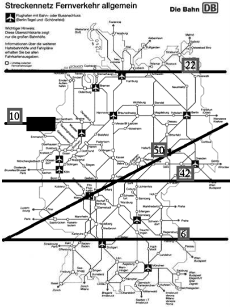
  disposition can 'almost' be modelled as flow problem.

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

basic problem formulation (1)

output: optimal disposition, i.e.

- allocation of all supply to as much demand as possible
- respecting all rules/constraints, integrality
- minimal costs

  disposition can 'almost' be modelled as flow problem.

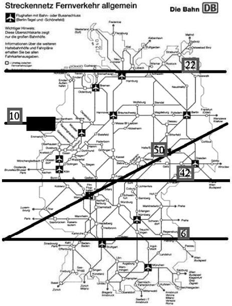
But: some features cannot, e.g. 1:2 substitution

22, Berlin,
Jan 11th, 10:00

10, Berlin,
Jan 12th, 10:00

42, Munich,
Jan 16th, 13:00

56, Cologne,
Jan 13th, 18:00



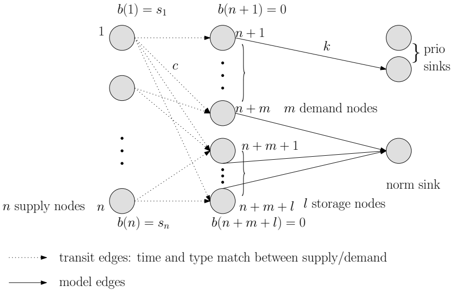
22, Berlin,
Jan 15th, 11:00

50, Munich,
Jan 15th, 14:00

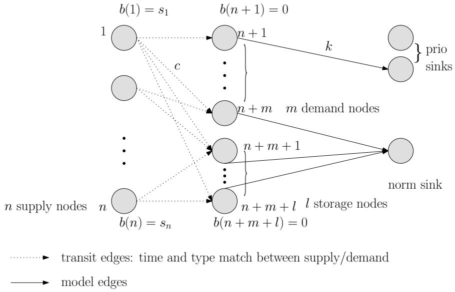
42, Munich,
Jan 16th, 16:00

6, Cologne,
Jan 17th, 18:00

network model $N = (V, A)$ with 1:1 substitution

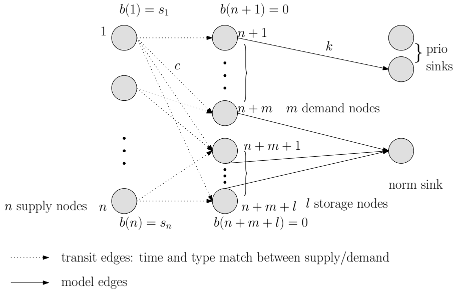


network model $N = (V, A)$ with 1:1 substitution



- Obtain disposition as solution of min-cost flow on $N = (V, A)$ in polynomial time (e.g. by SSP).

network model $N = (V, A)$ with 1:1 substitution



- Obtain disposition as solution of min-cost flow on $N = (V, A)$ in polynomial time (e.g. by SSP).
- All input values integral
  $\Rightarrow$ flow solution integral as demanded!

1:2 substitution and flow multipliers

Definition (flow $f(A)$ in $N = (V, A)$)

- $\forall a_{ij} \in A : l_{ij} \leq f(a_{ij}) \leq u_{ij}$
- $\forall v_i \in V : \sum_{a_{li}=(v_l,v_i)\in A} f(a_{li}) - \sum_{a_{ik}=(v_i,v_k)\in A} f(a_{ik}) = b(v_i)$

1:2 substitution and flow multipliers

Definition (flow $f(A)$ in $N = (V, A)$)

- $\forall a_{ij} \in A : l_{ij} \leq f(a_{ij}) \leq u_{ij}$
- $\forall v_i \in V : \sum_{a_{li}=(v_l,v_i)\in A} f(a_{li}) - \sum_{a_{ik}=(v_i,v_k)\in A} f(a_{ik}) = b(v_i)$

1:2 substitution $\Rightarrow$ Use same model enriched by multipliers:

1:2 substitution and flow multipliers

Definition (flow $f(A)$ in $N = (V, A)$)

- $\forall a_{ij} \in A : l_{ij} \leq f(a_{ij}) \leq u_{ij}$
- $\forall v_i \in V : \sum_{a_{li}=(v_l,v_i)\in A} f(a_{li}) - \sum_{a_{ik}=(v_i,v_k)\in A} f(a_{ik}) = b(v_i)$

1:2 substitution $\Rightarrow$ Use same model enriched by multipliers:

Definition (flow $f_m(A)$ in $N = (V, A)$ with multipliers)

- $\forall a_{ij} \in A : l_{ij} \leq f_m(a_{ij}) \leq u_{ij}$
- $\forall v_i \in V$:
  $\sum_{a_{li}=(v_l,v_i)\in A} \mu(a_{li}) f_m(a_{li}) - \sum_{a_{ik}=(v_i,v_k)\in A} f_m(a_{ik}) = b(v_i)$

network model with 1:2 substitution

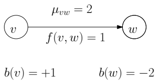
Example

$$v \xrightarrow[f(v,w)=1]{\mu_{vw}=2} w$$

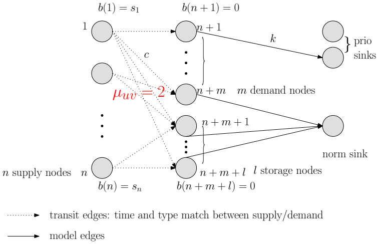
$b(v) = +1$ $\qquad$ $b(w) = -2$

network model with 1:2 substitution

Example



Network N:



disposition network

We can model all instances as:

Definition (disposition networks)

- network $N = (V = X \cup Y, A)$ is bipartite digraph
- $\forall a \in A : \mu_a \in \{1, 2\}$
- $\forall a = (u, v) \in A$ with $\mu(a) = 2 : u \in X, v \in Y$.
- Every path from a supply to a sink has *either* path multiplier 1 or 2.

Definition (path multiplier)

Let path $\pi_{u_1 u_n} = u_1, u_2, \ldots, u_n$, then

$$\mu(\pi_{u_1 u_n}) = \prod_{i=1}^{n-1} \mu_{u_i u_{i+1}}$$

complexity of integral flow

Theorem (S.Sahni,'74)

Integral maximum flow with multipliers is NP-hard.

Proof.

Reduction from subset sum, using general multipliers □

complexity of integral flow

Theorem (S.Sahni,'74)

Integral maximum flow with multipliers is NP-hard.

Proof.

Reduction from subset sum, using general multipliers □

Proof does not hold for multipliers 1 and 2. Problem easier?

complexity of integral flow

Theorem (S.Sahni,'74)

Integral maximum flow with multipliers is NP-hard.

Proof.

Reduction from subset sum, using general multipliers □

Proof does not hold for multipliers 1 and 2. Problem easier?

No, we can even proof:

Theorem

Integral maximum flow on disposition networks is NP-hard.

Proof.

Reduction from 3SAT by construction of disposition network. □

halfintegral and fractional solutions

- Integral solution: hard to guarantee.

halfintegral and fractional solutions

- Integral solution: hard to guarantee.
- Guarantee certain fractional solutions?

halfintegral and fractional solutions

- Integral solution: hard to guarantee.
- Guarantee certain fractional solutions?

- Theorem

  Optimal solutions for disposition networks are halfintegral.

  Proof.

  - Circulation: Extend network N to special circulation.
  - Induction: Flow increases only (half)integral on certain arcs.

  $\square$

halfintegral and fractional solutions

- Integral solution: hard to guarantee.
- Guarantee certain fractional solutions?

- **Theorem**
  Optimal solutions for disposition networks are halfintegral.

  Proof.
  - Circulation: Extend network N to special circulation.
  - Induction: Flow increases only (half)integral on certain arcs.

  $\square$

- Not in general!

halfintegral and fractional solutions

- Integral solution: hard to guarantee.
- Guarantee certain fractional solutions?

- Theorem
  Optimal solutions for disposition networks are halfintegral.

  Proof.

  - Circulation: Extend network N to special circulation.
  - Induction: Flow increases only (half)integral on certain arcs.

  $\square$

- Not in general!
- We can construct other instances with $3n$ nodes and $\frac{1}{2^n}$ fractional solutions.

motivation for modified SSP

We...

- started with simple flow model
- obtained disposition solution by integral min-cost flow solution with SSP
- introduced flow multipliers for 1:2 substitution
- lost polynomially achievable integral solution
- guaranteed halfintegral solution for disposition network
- want to keep SSP application (easy incorporation of other side constraints)

$\Rightarrow$ Modify SSP algorithm.

original SSP

SSP

Init.
while $(b(s) > 0)$ and $(b(t) < 0)$ and $(\exists \pi_{st})$
    Find shortest s-t-path π_{st} in N'
    Augment max. poss. flow δ along π_{st}
    Update res. network N'
end while

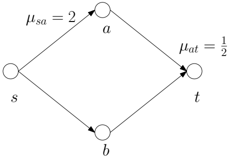
modifying original SSP

3: Find shortest s-t-path π_{st} in N'

modifying original SSP

3: Find shortest s-t-path π_{st} in N'

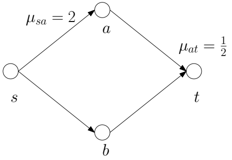
With usual path costs $c'(\pi_{uv}) = \sum_{i=2}^{n} c((i-1)i)$:



modifying original SSP

3:	Find shortest s-t-path π_{st} in N'

With usual path costs $c'(\pi_{uv}) = \sum_{i=2}^{n} c((i-1)i)$:



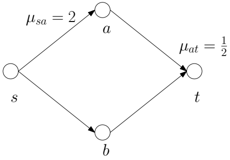
$\Rightarrow$ Define new path costs with multipliers:

$$c'(\pi_{uv}) = \sum_{i=2}^{n} \prod_{j=1}^{i-1} \mu_{j(j+1)} \cdot c((i-1)i).$$

modifying original SSP

3: Find shortest s-t-path π_{st} in N'

With usual path costs $c'(\pi_{uv}) = \sum_{i=2}^{n} c((i-1)i)$:



$\Rightarrow$ Define new path costs with multipliers:

$$c'(\pi_{uv}) = \sum_{i=2}^{n} \prod_{j=1}^{i-1} \mu_{j(j+1)} \cdot c((i-1)i).$$

Compute with Dijkstra:
Multiplier $\mu_i = \prod_{j=1}^{i-1} \mu_{j(j+1)}$ for each node i.

modifying original SSP (1)

4: Augment max. poss. flow δ along π_{st}

modifying original SSP (1)

4:	Augment max. poss. flow δ along π_{st}

Usual:

$$\delta := min\{b(s), b(t), min_{a=(uv)\in\pi_{st}} cap_r(a)\}$$

modifying original SSP (1)

4:	Augment max. poss. flow δ along π_{st}

Usual:

$$\delta := min\{b(s), b(t), min_{a=(uv)\in\pi_{st}} cap_r(a)\}$$

Here:

$$\delta_m := min\{b(s), -\frac{b(t)}{\mu_t}, min_{a=(uv)\in\pi_{st}} \frac{cap_r(a)}{\mu_u}\}$$

modifying original SSP (2)

Update res. network N'

modifying original SSP (2)

Update res. network N'

In residual network N'_m for each residual arc $\overline{a} = (u, v)$:

- capacity $cap(\bar{a}) = f(a)$, cost $c(\bar{a}) = -c(a)$, flow 0
- $\mu_{\bar{a}} = \frac{1}{\mu_a}$

multiplier SSP

	mSSP
Init.	
while	$(b(s) > 0)$ and $(b(t) < 0)$ and $(\exists \pi_{st})$
	Find shortest multiplier s-t-path π_{st} in N'_m
	Augment max. poss. flow δ_m along π_{st}
	Update res. network N'_m
end	while

multiplier SSP

	mSSP	
1:	Init.	
2:	while	$(b(s) > 0)$ and $(b(t) < 0)$ and $(\exists \pi_{st})$
3:		Find shortest multiplier s-t-path π_{st} in N'_m
4:		Augment max. poss. flow δ_m along π_{st}
5:		Update res. network N'_m
6:	end	while

Correctness:

- Analougously to SSP (reduced cost criterium)
- Based on modified path and reduced costs

multiplier SSP

mSSP	
Init.	
while	$(b(s) > 0)$ and $(b(t) < 0)$ and $(\exists \pi_{st})$
	Find shortest multiplier s-t-path π_{st} in N'_m
	Augment max. poss. flow δ_m along π_{st}
	Update res. network N'_m
end	while

Correctness:

- Analougously to SSP (reduced cost criterium)
- Based on modified path and reduced costs

Running time:

- Generally depends on δ_m (lower bound?)
- Disposition application: $\delta_m \in \{\frac{1}{2}, 1\}$
  $\Rightarrow$ (pseudo)polynomial running time

Obtaining an integral disposition solution

Apply simple rounding heuristic:

Rounding

while ($\exists$ halfintegral flow f)
  Find cheapest f from s to t via u
  if ($f = f + \frac{1}{2}$ violates $cap(u, t)$ by at most $\frac{1}{2}$)
    Round f up and round most expensieve halfintegral $s - t$-flow f' down.
  else
    Round f down and round next cheapest halfintegral $s - t$-flow f' up.
end while

Obtaining an integral disposition solution

Apply simple rounding heuristic:

	Rounding		
1:	while	($\exists$	halfintegral flow f)
3:		Find	cheapest f from s to t via u
4:		if	($f = f + \frac{1}{2}$ violates $cap(u,t)$ by at most $\frac{1}{2}$)
5:			Round f up and round most expensieve halfintegral $s-t$-flow f' down.
6:		else	
7:			Round f down and round next cheapest halfintegral $s-t$-flow f' up.
8:	end	while	

Remark:

If no flow f' can be found in line

5 Decrease rest supply (initial integral supplies!).

7 Increase rest supply.

quality of integral solution

The simple heuristic ...

- applies to halfintegral solutions for disposition networks
- ends (in polynomial time) with integral solution and no capacities violated
- increases total costs by factor of 2 in the worst case

quality of integral solution

The simple heuristic ...

- applies to halfintegral solutions for disposition networks
- ends (in polynomial time) with integral solution and no capacities violated
- increases total costs by factor of 2 in the worst case

But:

Actual flow value of resulting solution can be decreased!

quality of integral solution

The simple heuristic ...

- applies to halfintegral solutions for disposition networks
- ends (in polynomial time) with integral solution and no capacities violated
- increases total costs by factor of 2 in the worst case

But:

Actual flow value of resulting solution can be decreased!

Therefore:

We work on better heuristics...

quality of integral solution

The simple heuristic ...

- applies to halfintegral solutions for disposition networks
- ends (in polynomial time) with integral solution and no capacities violated
- increases total costs by factor of 2 in the worst case

But:

Actual flow value of resulting solution can be decreased!

Therefore:

We work on better heuristics...

...and...

on running time results for the mSSP on general instances.

Thank you for your attention!

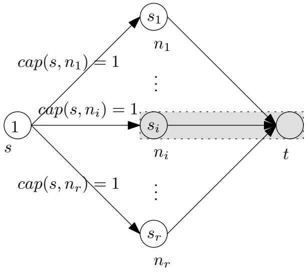
flow with general multipliers

Theorem (S.Sahni,'74)

Integral maximum flow with multipliers is NP-hard.

Reduction from subset sum:

Given an instance $I = [S = \{s_i, 1 \leq i \leq r\}, M]$ of subset sum: Demand of $-M$ at t can be satisfied by an integral flow $\Leftrightarrow$ I is solvable (or vice versa). $\square$



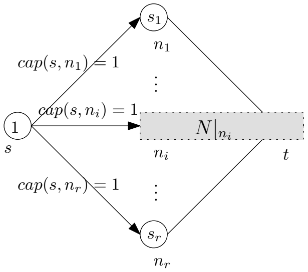
flow with multipliers 1 and 2

Theorem

Integral maximum flow with multipliers $1, 2$ *is NP-hard.*

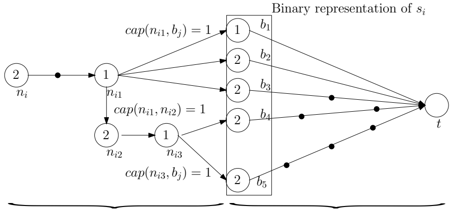
Proof.

Replace n_i with subgraph $N|n_i$ with inflow 1, outflow s_i, only multipliers $1, 2$ (binary encoding s_i). □



subgraph $N|_{n_i}$

for $s_i = 31$:



Amplification of one flow unit to z_i units at nodes $b_j, 1 \leq j \leq z_i$.

Amplification of each flow unit at b_j to the number of units resembling the valency of bit j.

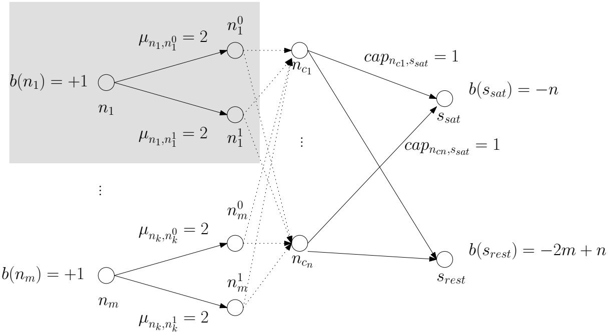
flow with multipliers 1 and 2

Theorem

Integral maximum flow in disposition networks is NP-hard.

Reduction:

Given a boolean formula α in CNF with *n* clauses and *m* variables (limited occurance!): Demands can be satisfied by an integral flow $\Leftrightarrow$ α is satisfiable. $\square$



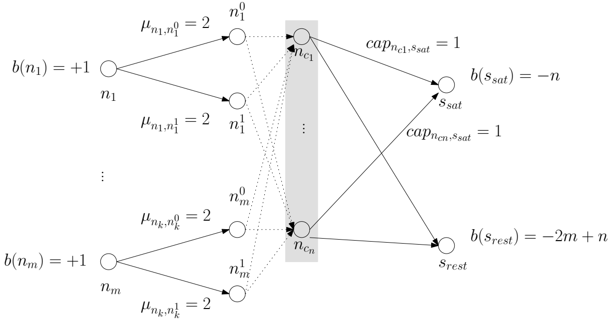
flow with multipliers 1 and 2

Theorem

Integral maximum flow in disposition networks is NP-hard.

Reduction:

Given a boolean formula α in CNF with *n* clauses and *m* variables (limited occurance!): Demands can be satisfied by an integral flow $\Leftrightarrow$ α is satisfiable. $\square$



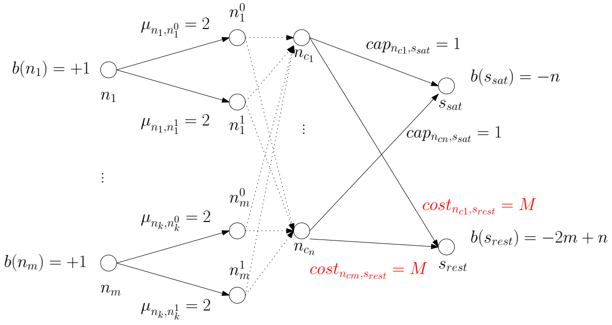
flow with multipliers 1 and 2

Theorem

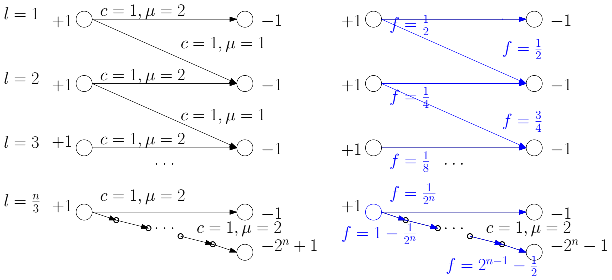
Integral maximum flow in disposition networks is NP-hard.

Reduction:

Given a boolean formula α in CNF with n clauses and m variables (limited occurance!): Demands can be satisfied by an integral flow with cost $2m \cdot M - n \Leftrightarrow \alpha$ is satisfiable. □



fractional solution



halfintegral solution

Theorem

Optimal solutions for disposition networks are halfintegral.

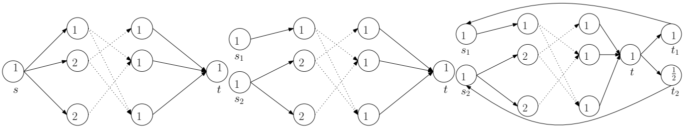
halfintegral solution

Theorem

Optimal solutions for disposition networks are halfintegral.

Proof.

Extend network N to circulation with only unit gain cycles.



□

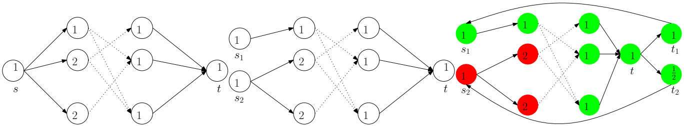
halfintegral solution

Theorem

Optimal solutions for disposition networks are halfintegral.

Proof.

Induction: Flow increases only halfintegral on red-green and green-green arcs and integral on red-red and green-red arcs.



□