\Orchestrating a brighter world NEC

ERATO感謝祭 SeasonIV 2017.8.3@NII

Large-Scale Price Optimization via Network Flow

Shinji Ito, Ryohei Fujimaki

Our goal: profit maximization by optimizing prices

What is the best pricing strategy?

Strategy 1		Strategy 2
: $ 1.3	Price	: $ 1.0
	Sales quantity	
?	Profit	?

Our goal: profit maximization by optimizing prices

What is the best pricing strategy?

	Strategy 1	Strategy 2
Price	: $ 1.3	: $ 1.0
	×	×
Sales quantity		
	=	=
Profit	$ 5.2	$ 6.0

<

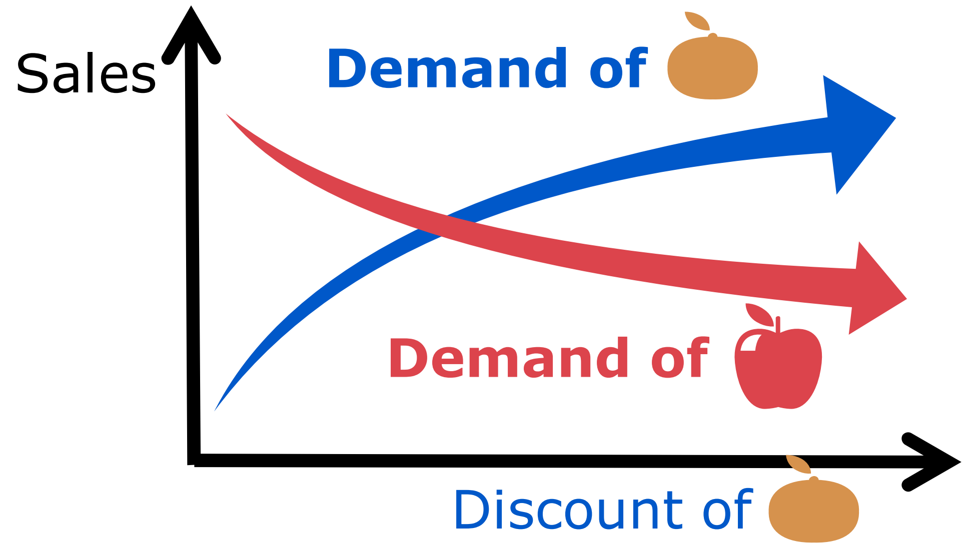
Complicated structure in price optimization

Changing the price of one product affects other's sales

- Cannibalization:

Growing the sales of

makes the sales of down

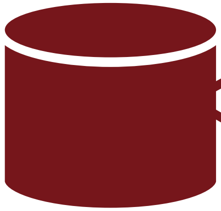


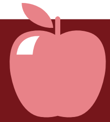
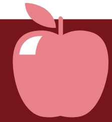
	Price	Quantity	Profit
Product 1	**$1.3 → $1.0**	**+200**	**+ $80**
Product 2	**$1.2**	**-100**	**- $120**
Gross profit	**- $40**		

Predictive price optimization and its difficulty

- Recent advanced ML reveals relationship between prices and sales quantities

Input:
pos data



	 Price	 Price	 Sales	 Sales	...
Day 1	**$1.3**	**$1.0**	**2**	**2**	...
Day 2	**$1.2**	**$1.0**	**4**	**2**	...
⋮	⋮	⋮	⋮	⋮	

Machine learning

Predictive model: $\text{sales} = f(\text{prices})$

Predictive price optimization and its difficulty

Recent advanced ML reveals relationship between prices and sales quantities

Input:
pos data

	Price	Price	Sales	Sales	...
Day 1	**$1.3**	**$1.0**	**2**	**2**	...
Day 2	**$1.2**	**$1.0**	**4**	**2**	...
⋮	⋮	⋮	⋮	⋮	

Machine learning

Predictive model: $\text{sales} = f(\text{prices})$

Optimization (NP-hard) **[This work]**

Output:
optimal prices

Our contribution

Scalable algorithm for price optimization

Based on:

1. Submodularity behind pricing
2. Network flow algorithm
3. Supermodular relaxation

Our contribution

Scalable algorithm for price optimization

Based on:

1. Submodularity behind pricing
2. Network flow algorithm
3. Supermodular relaxation

Achieved:

- Can deal with thousands of products
- High accuracy for real-data problem

Table of contents

1. Introduction

2. Problem definition

3. Scalable price optimization algorithm

4. Experiments

Objective function of price optimization

Want to maximize is the gross profit ℓ

Gross profit:

$$\ell(p_1, p_2, \dots, p_M) = \sum_{i=1}^{M} (p_i - c_i) q_i$$

product id

price

cost

sales quantity

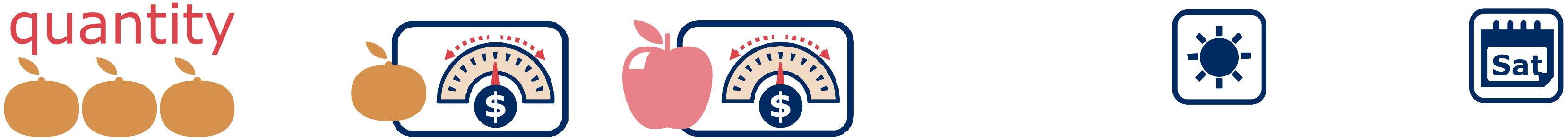
Unknown, but predictable

Predictive model for sales quantity

Sales quantity q_i is a function in prices p_i

$$\underline{q_1(p,r)} = f_{11}(\underline{p_1}) + f_{12}(\underline{p_2}) + \cdots \quad + g_{11}(\underline{r_1}) + g_{12}(\underline{r_2}) + \cdots$$

sales quantity | price | price | weather | calendar



Use historical data to infer $f_{i,j}, g_{i,j}$

Ex: $f_{i,j}(p_j) = a_{i,j} p_j^2 + b_{i,j} p_j + c_{i,j}$ (polynomial model)

$f_{i,j}(p_j) = \exp(\alpha_{i,j} p_j + \beta_{i,j})$, (generalized linear model)

Predictive model for sales quantity

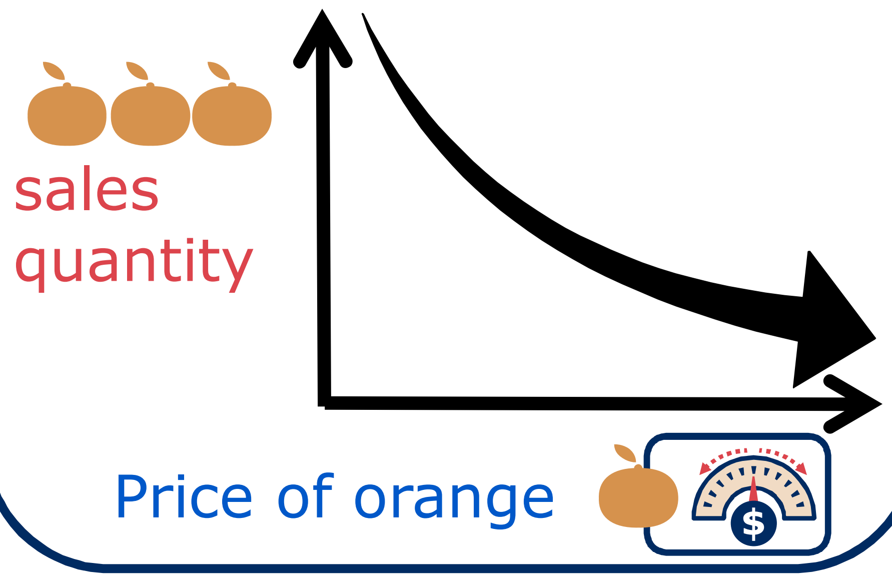
Sales quantity q_i is a function in prices p_i

$$\underline{q_1}(p,r) = f_{11}(\underline{p_1}) + f_{12}(\underline{p_2}) + \cdots \quad + g_{11}(\underline{r_1}) + g_{12}(\underline{r_2}) + \cdots$$

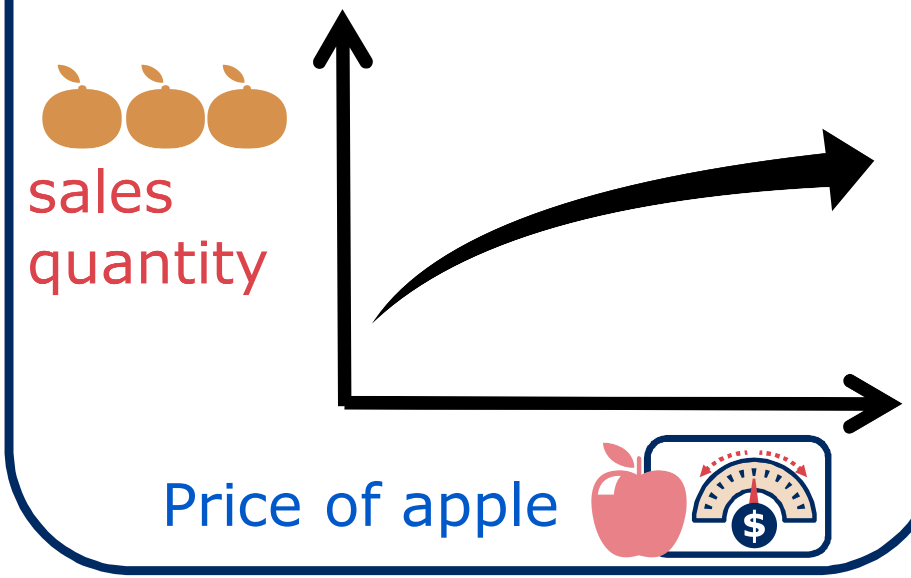
sales quantity | price | price | Weather | calendar

Sat

f_{11}: Price elasticity of demand



f_{12}: Cross price effect



Substitute goods in price optimization

Many substitute goods in price optimization

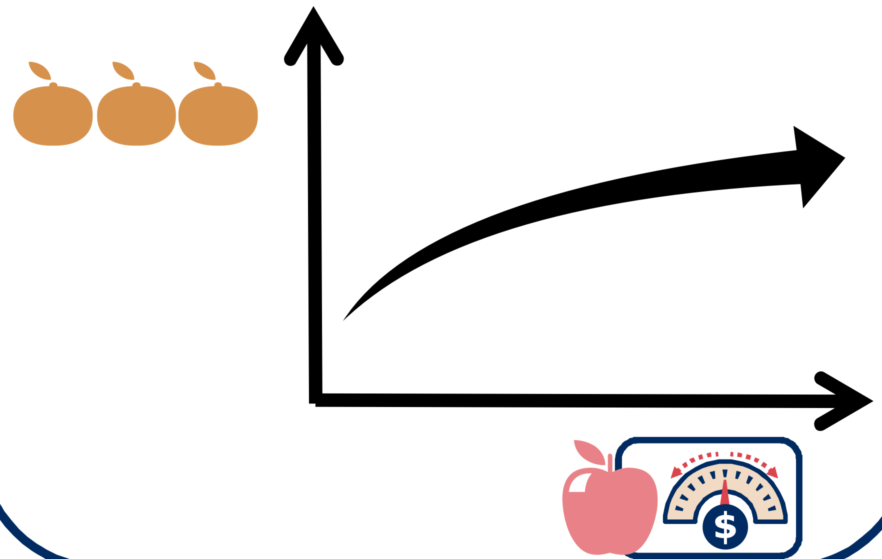
$$\underline{q_1(p,r)} = f_{11}(\underline{p_1}) + f_{12}(\underline{p_2}) + \cdots \quad + g_{11}(\underline{r_1}) + g_{12}(\underline{r_2}) + \cdots$$

and : *Substitute goods*

$\Updownarrow$ def

Discounting apple makes the sales of orange down (cannibalization)

f_{12}: Cross price effect



Optimization problem

Optimization is NP-hard

Gross profit

Maximize $$\ell(p) = \sum_{i=1}^{M} (p_i - c_i) q_i(p)$$

Subject to $$p_i \in \{P_{i1}, P_{i2}, \ldots, P_{ik}\}$$

Discrete price candidates

A commercial solver takes >24[h] for 50 products

Table of contents

1. Introduction

2. Problem definition

3. Scalable price optimization algorithm

4. Experiments

Our contribution

Scalable algorithm for price optimization

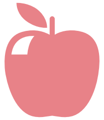
Based on:

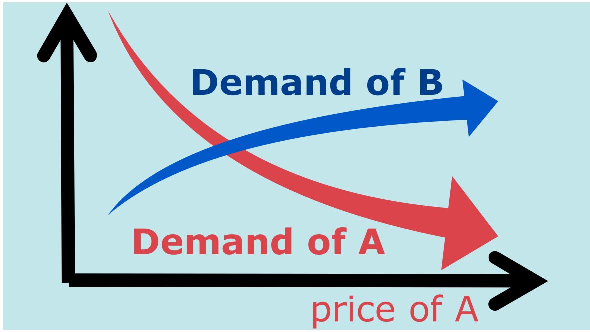
1. Submodularity behind pricing
2. Network flow algorithm
3. Supermodular relaxation

Idea 1: Substitute goods and supermodular

Connection between substitute goods and submodular

Substitute goods:

Discounting  ⇒ less sales of 



Theorem [Substitute goods ⇒ supermodular]

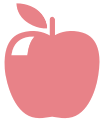
If all pairs of products are substitute goods or independent

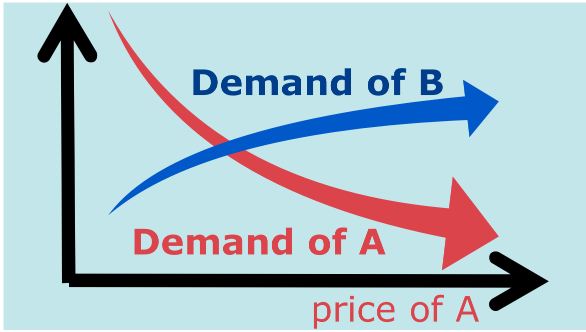
⇒ $\ell(p)$ is a <u>supermodular function</u>

Idea 1: Substitute goods and supermodular

Connection between substitute goods and submodular

Substitute goods:

Discounting  ⇒ less sales of 



Theorem [Substitute goods ⇒ supermodular]

If all pairs of products are substitute goods or independent

⇒ $\ell(p)$ is a <u>supermodular function</u>

maximized in polynomial time

[Iwata, Fleischer, Fujishige (2001)]

Two issues in Key idea 1

- If all products are substitute goods or independent, profit can be maximized in polynomial time

Two issues in Key idea 1

- If all products are substitute goods or independent, profit can be maximized in polynomial time

- This approach is still impractical because of two issues

Issue 1: General supermodular maximization is slow

$\sim O(n^5)$

Issue 2: Substitute goods assumption is too restrictive

Two issues in Key idea 1

- If all products are substitute goods or independent, profit can be maximized in polynomial time

- This approach is still impractical because of two issues

Issue 1: General supermodular maximization is slow $\sim O(n^5)$

➡ Resolved by 2. network flow: $O(n^2) \sim O(n^3)$

Issue 2: Substitute goods assumption is too restrictive

➡ Resolved by 3. supermodular relaxation

Applicable for non-substitute

Two issues in Key idea 1

- If all products are substitute goods or independent, profit can be maximized in polynomial time

- This approach is still impractical because of two issues

Issue 1: General supermodular maximization is slow $\sim O(n^5)$

⇒ Resolved by 2. network flow:

$O(n^2) \sim O(n^3)$

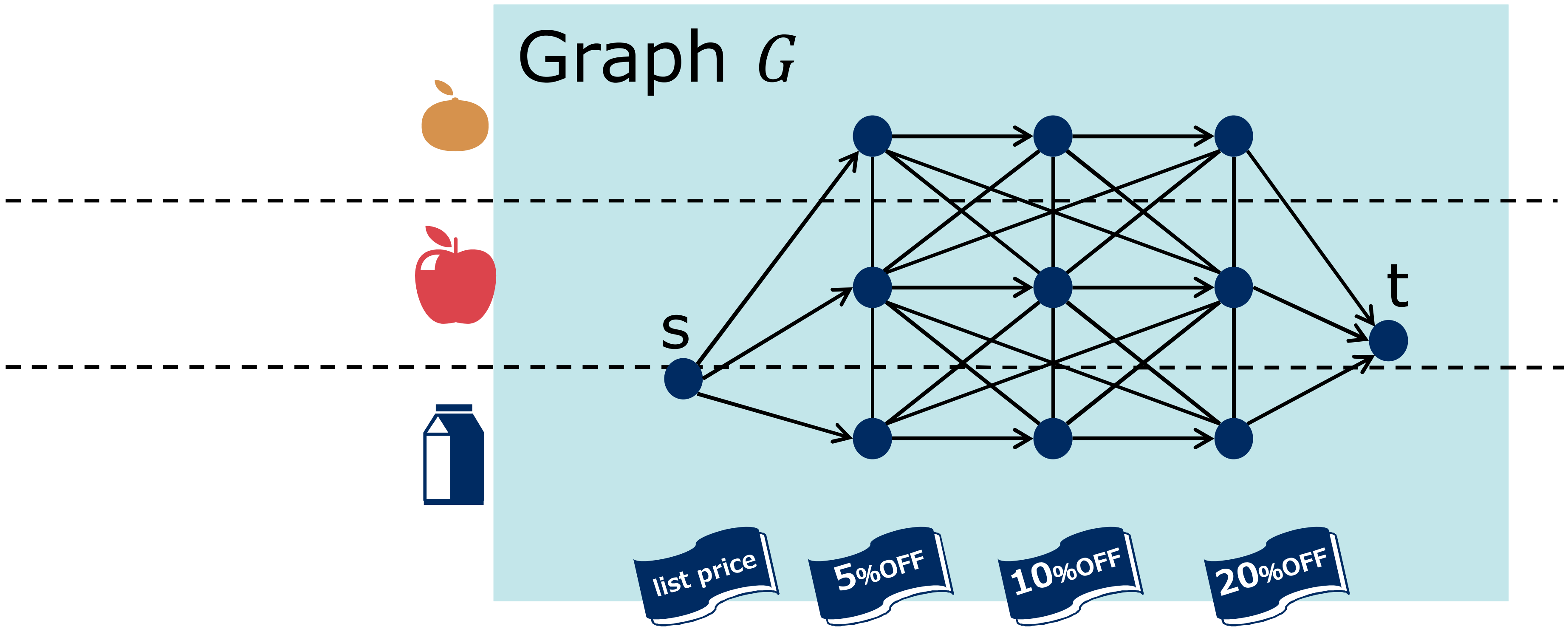
Issue 2: Substitute goods assumption is too restrictive

⇒ Resolved by 3. supermodular relaxation

Applicable for non-substitute

Substitute goods price optimization ⇔ Minimum Cut

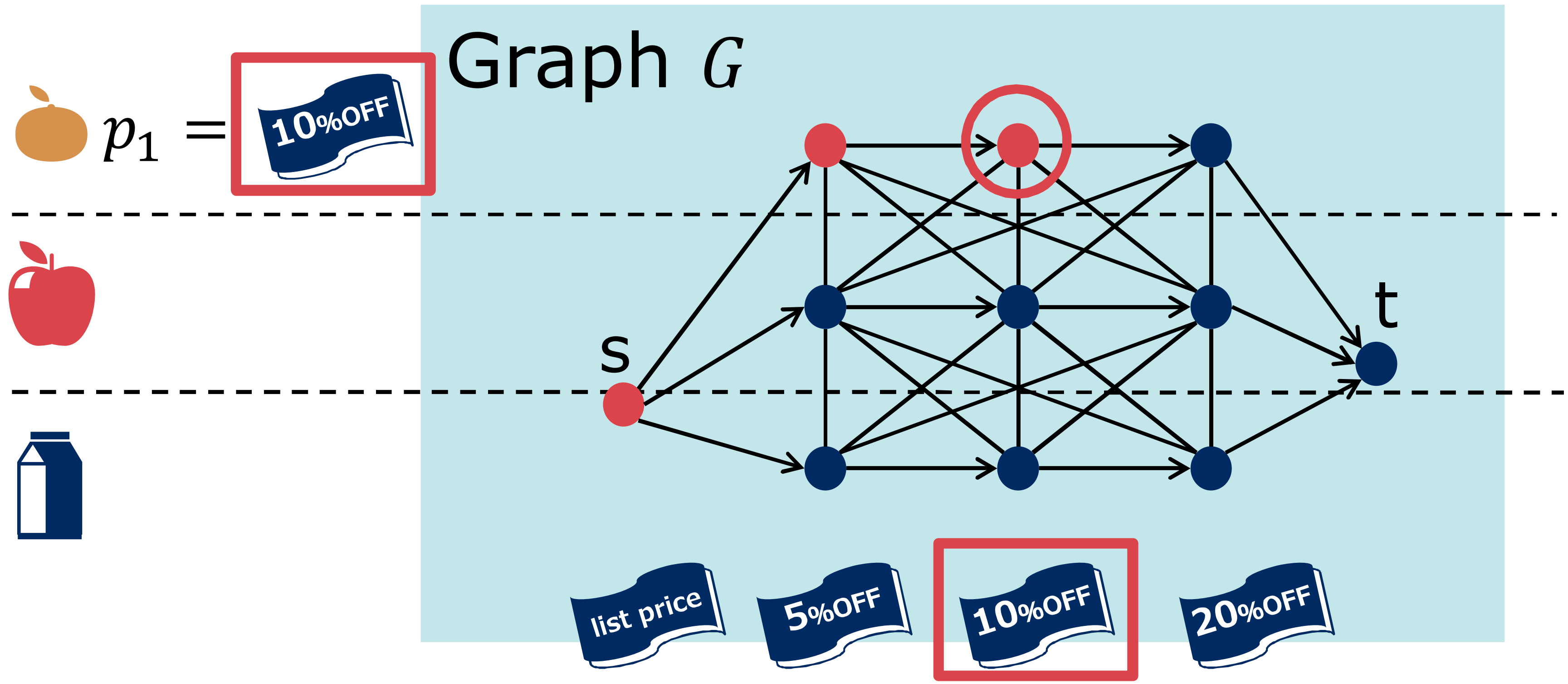
- Maximizing $\ell(p)$ ⇔ Finding minimum s-t cut
  :solved efficiently by network flow
  [Ford, Fulkerson (1956)], [Orlin (2013)]



Substitute goods price optimization ⇔ Minimum Cut

Maximizing $\ell(p)$ ⇔ Finding minimum s-t cut

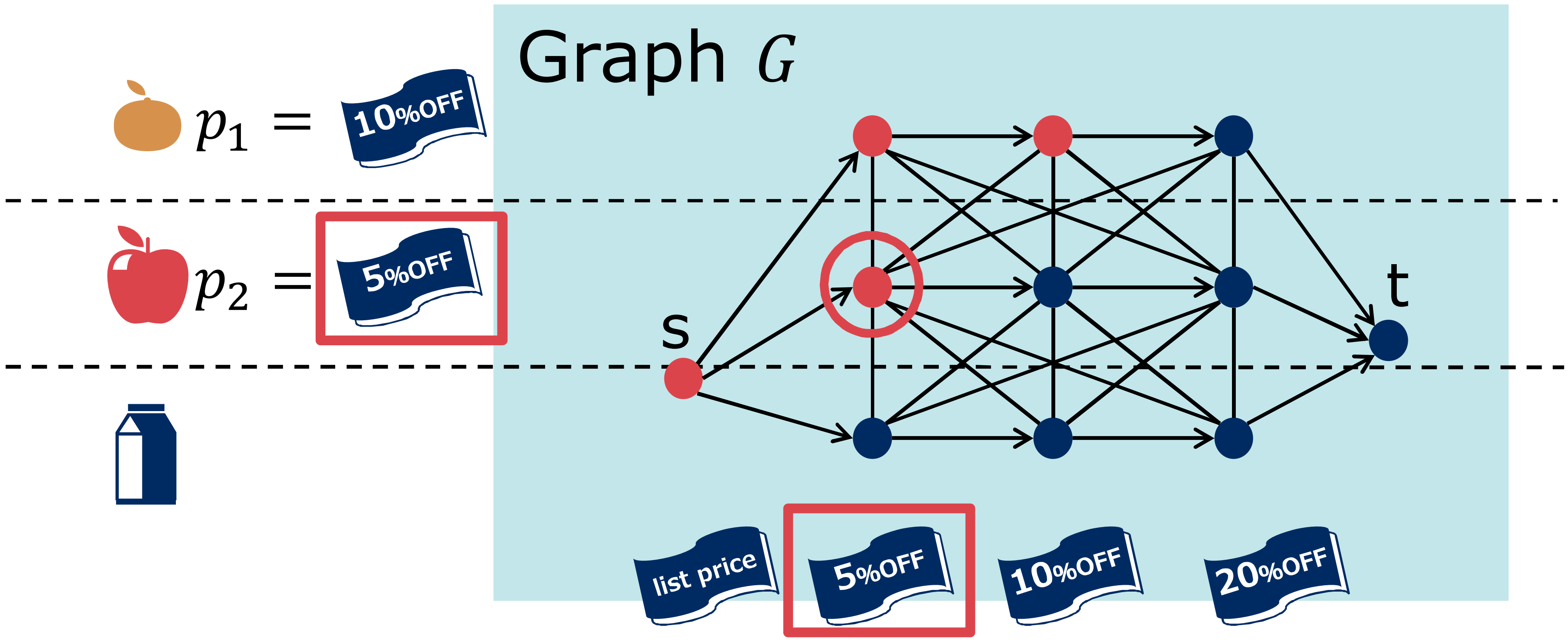
:solved efficiently by network flow



Substitute goods price optimization ⇔ Minimum Cut

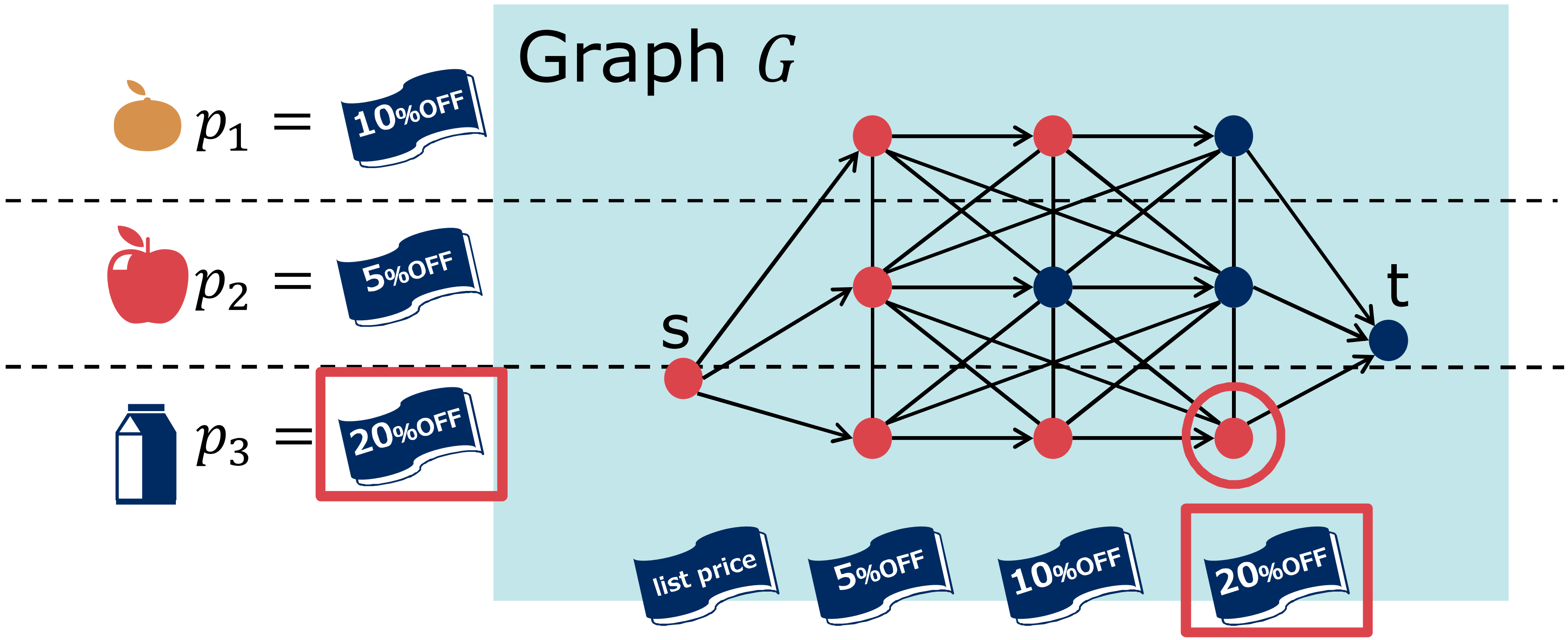
- Maximizing $\ell(p)$ ⇔ Finding minimum s-t cut

  :solved efficiently by network flow



Maximizing $\ell(p)$ ⇔ Finding minimum s-t cut

:solved efficiently by network flow

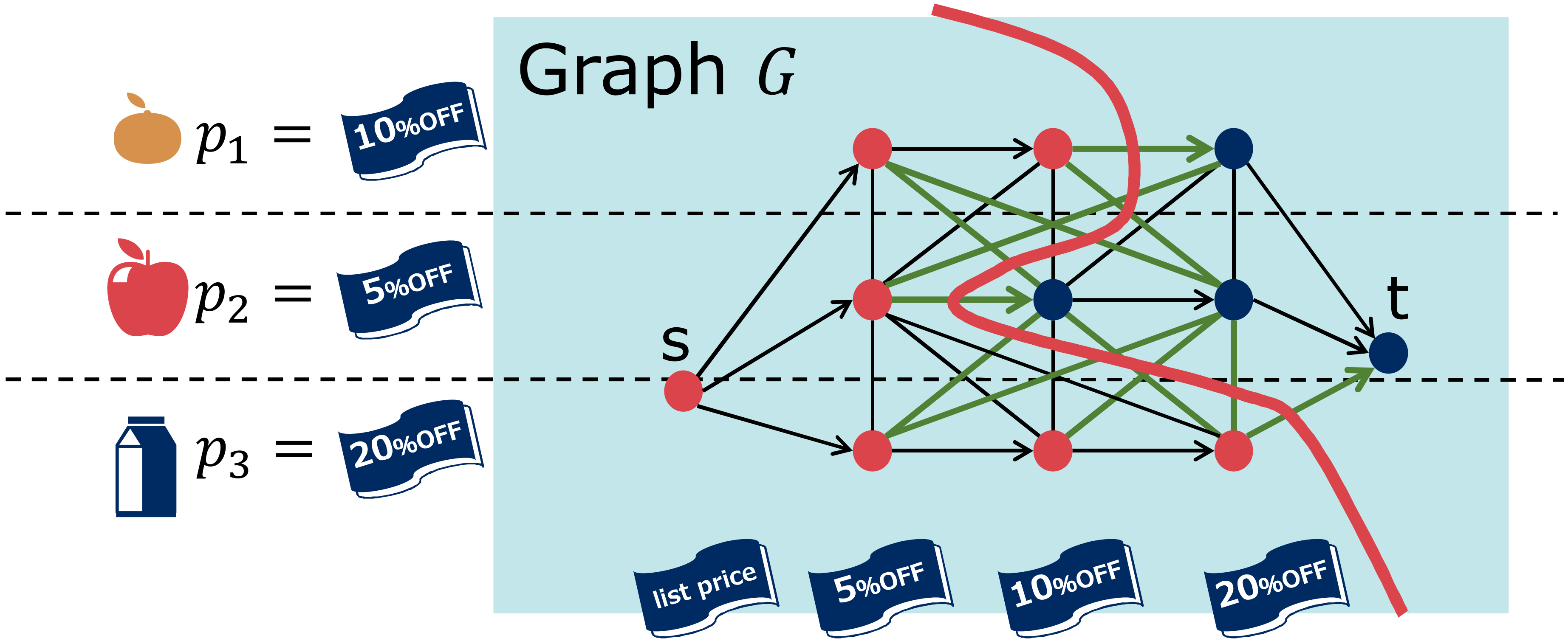


Substitute goods price optimization ⇔ Minimum Cut

Maximizing $\ell(p)$ ⇔ Finding minimum s-t cut

:solved efficiently by network flow

$\ell(p)$ = constant – (capacity of st-cut of graph G)



Two issues in Key idea 1

- If all products are substitute goods or independent, profit can be maximized in polynomial time

- This approach is still impractical because of two issues

Issue 1: General supermodular maximization is slow $O(n^5)$

➡ Resolved by 2. network flow: $O(n^2) \sim O(n^3)$

Issue 2: Substitute goods assumption is too restrictive

➡ Resolved by 3. supermodular relaxation

Applicable for non-substitute

Non-SGP case: supermodular relaxation

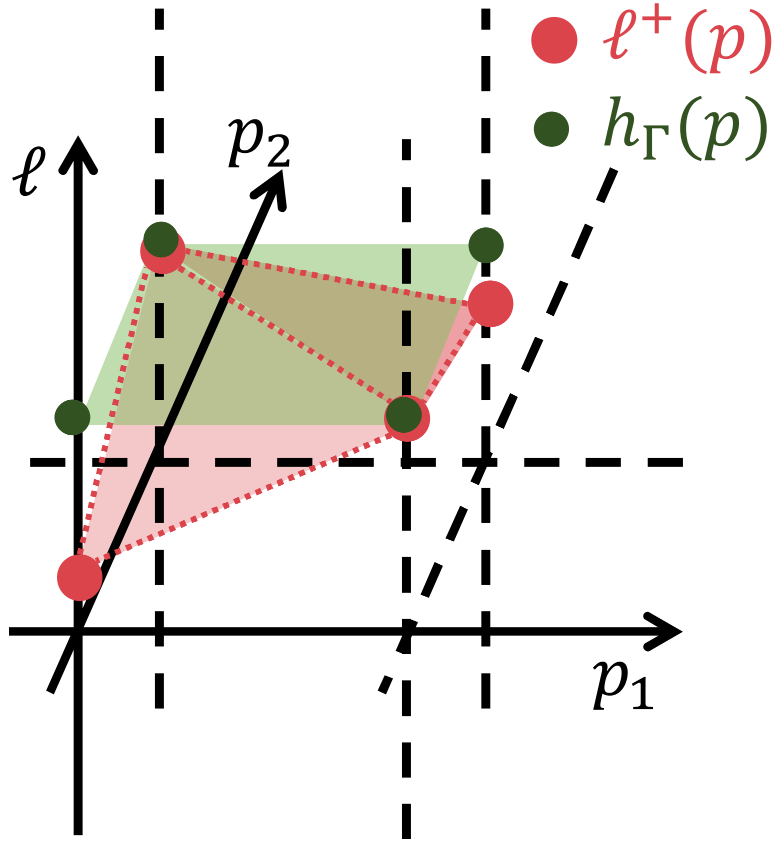
∃ non-substitute goods

⇒ approximate ℓ by supemodular function

$$\ell(p) = \underbrace{\ell^{-}(p)}_{\text{supermodular}} + \underbrace{\ell^{+}(p)}_{\text{submodular}}$$

$$\leq \underbrace{\boxed{\ell^{-}(p) + \underbrace{h_{\Gamma}(p)}_{\text{modular}}}}_{\text{supermodular}}$$

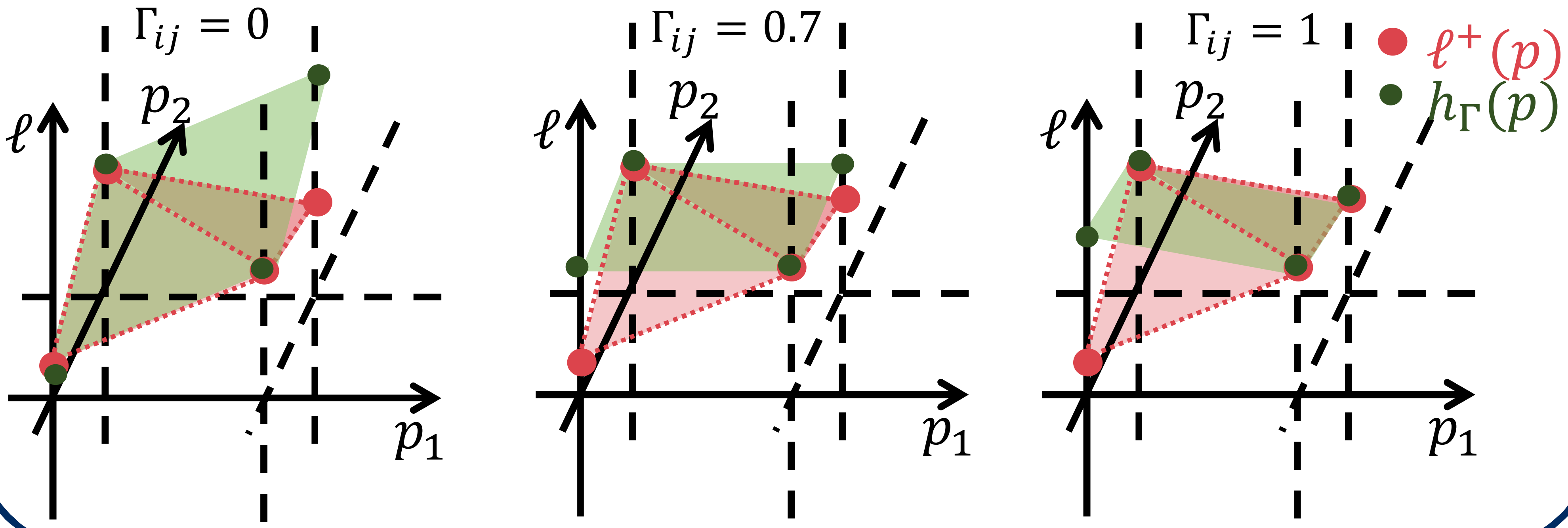
⇒ maximized via network flow



Many possibilities of relaxation function

Better relaxation is chosen automatically

Relaxation changes depending on $\Gamma \in [0,1]^{n \times n}$



Simulation experiments

Proposed approximation algorithm:
fast and give solution with only <1% loss

Existing methods

	Past data	Proposed	QPBO	Others
Computing Time		**36 [s]**	**964 [s]**	**> 1day**
Achieved Profit	**1.40 M**	**1.88 M**	**1.25 M**	**Nan**
Upper bound		**1.90 M**	**1.89 M**	**Nan**

- Real-world retail data* of a supermarket

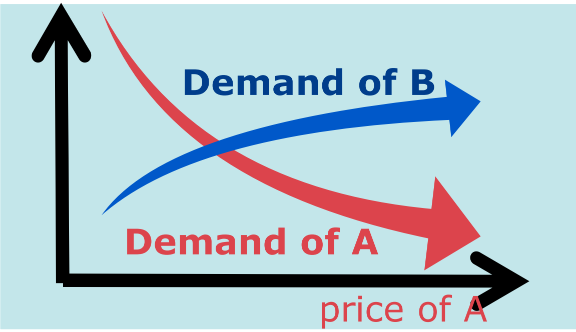
*provided by KSP-SP Co., LTD.

- 1000 products
- Compared with SDP, QPBO, QPBOI [Rother et.al (2007)]

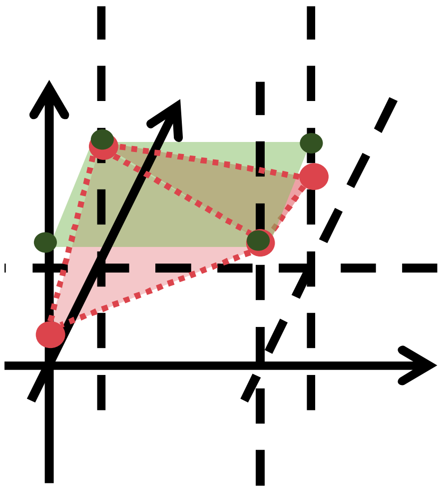
Conclusion

- Constructed a scalable price optimization algorithm by

  - Associating substitute goods with supermodularity



  - Using network flow and supermodular relaxation



- Future work: how to cope with the errors in objective?

  e.g., by robust optimization?

\Orchestrating a brighter world

NEC