The Clever Shopper Problem

Laurent Bulteau Danny Hermelin **Anthony Labarre** Stéphane Vialette

The 13th International Computer Science Symposium in Russia

CSR 2018

June 6th, 2018

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

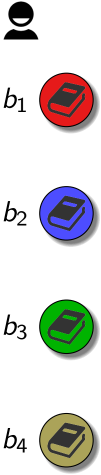
Gaspard - Monge Laboratoire d'informatique

אוניברסיטת בן-גוריון בנגב

Introduction

Clever Shopper

Introduction



Clever Shopper

Input:

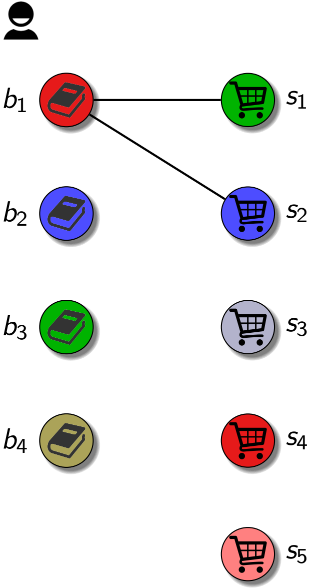
a set of books B,

Introduction

Clever Shopper

Input:
a set of books B,
a set of shops S,

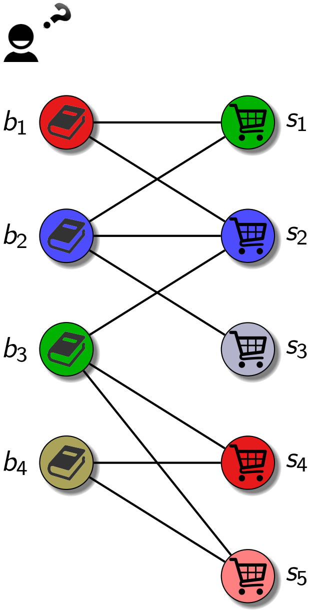
Introduction



CLEVER SHOPPER

Input:
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,

Introduction

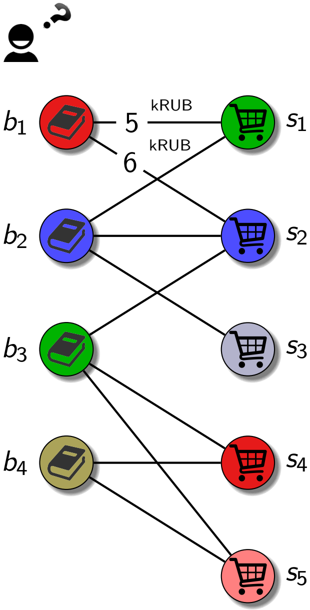


CLEVER SHOPPER

Input:

a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,

Introduction

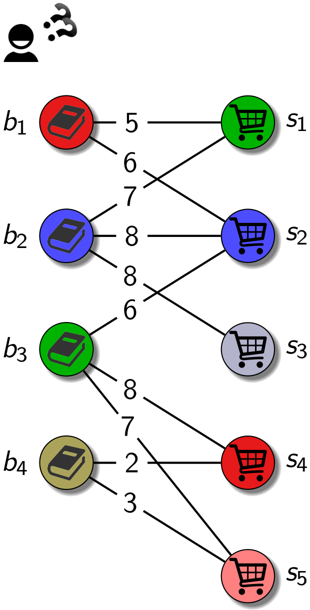


Clever Shopper

Input:

a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights $w \colon E \to \mathbb{N}^+$,

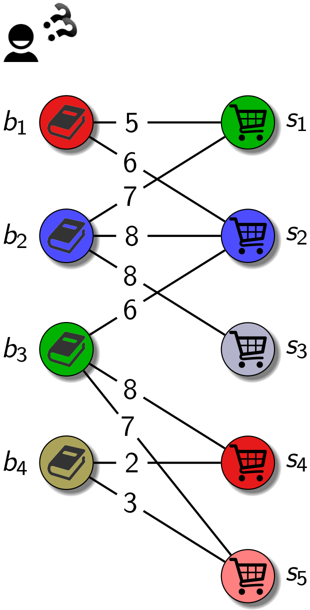
Introduction



CLEVER SHOPPER

Input:
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights w: $E \to \mathbb{N}^+$,

Introduction



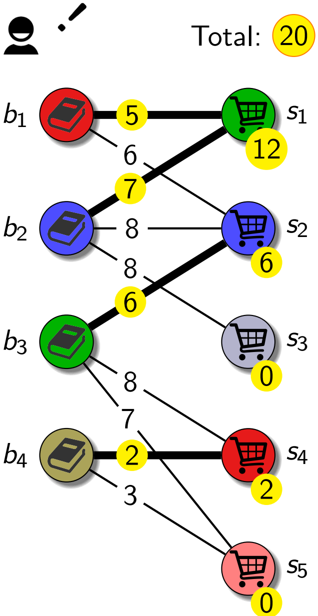
CLEVER SHOPPER

Input:
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights $w\colon\ E \to \mathbb{N}^+$,

Output:
$E' \subseteq E$ such that:

- each $b \in B$ has 1 incident edge
- minimum total cost

Introduction

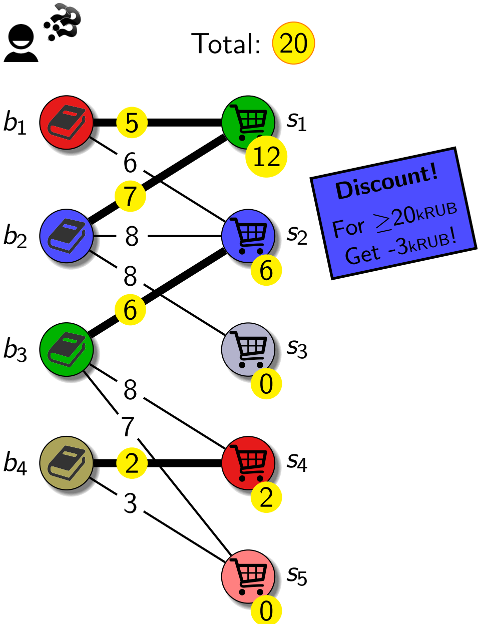


Clever Shopper

Input:
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights w: $E \to \mathbb{N}^+$,

Output:
$E' \subseteq E$ such that:
- each $b \in B$ has 1 incident edge
- minimum total cost

Introduction



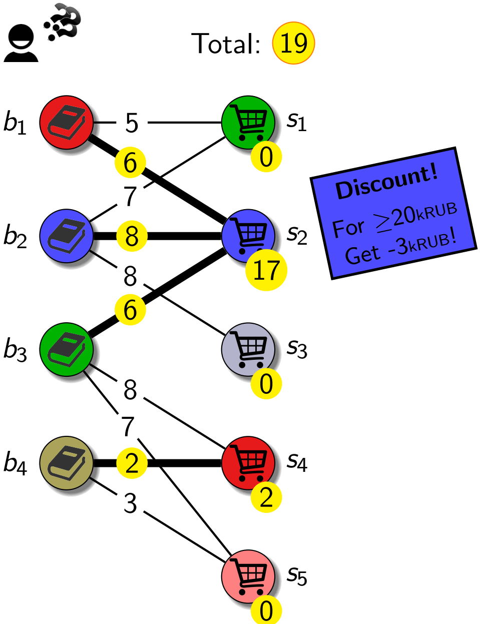
Clever Shopper

Input:
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights w: $E \to \mathbb{N}^+$,
a discount function $\mathcal{D}$,

Output:
$E' \subseteq E$ such that:

- each $b \in B$ has 1 incident edge
- minimum total cost

Introduction

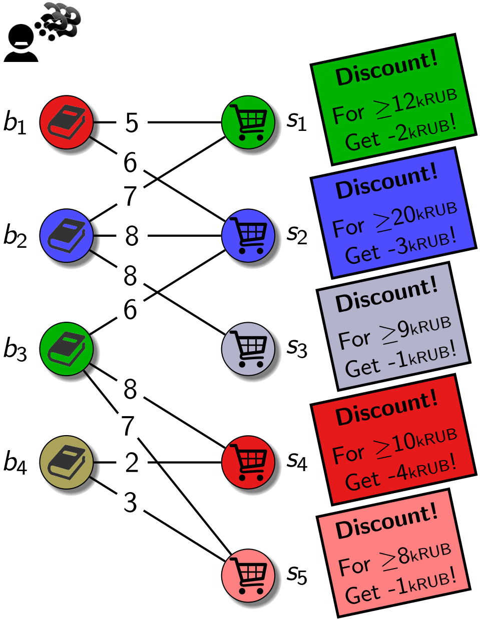


Clever Shopper

Input:
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights w: $E \to \mathbb{N}^+$,
a discount function $\mathcal{D}$,

Output:
$E' \subseteq E$ such that:
- each $b \in B$ has 1 incident edge
- minimum total cost

Introduction



Clever Shopper

Input:

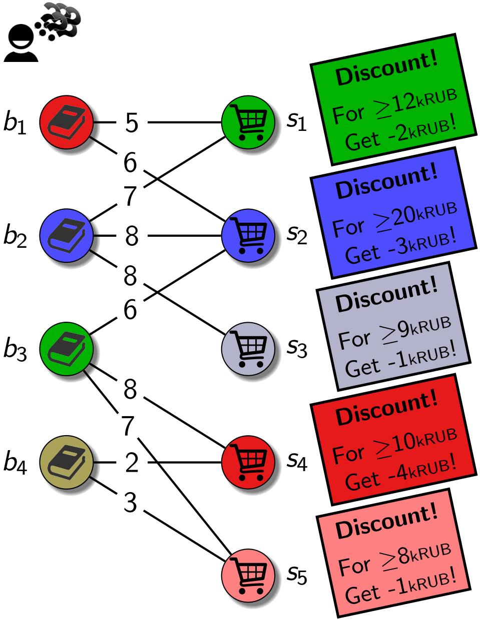
a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights $w\colon E \to \mathbb{N}^+$,
a discount function $\mathcal{D}$,

Output:

$E' \subseteq E$ such that:

- each $b \in B$ has 1 incident edge
- minimum total cost

Introduction



Clever Shopper

Input:

a set of books B,
a set of shops S,
edges $E \subseteq B \times S$,
with weights w: $E \to \mathbb{N}^+$,
a discount function $\mathcal{D}$,
a budget $K \in \mathbb{N}^+$

Output:

$E' \subseteq E$ such that:

- each $b \in B$ has 1 incident edge
- total price $\leq K$

Introduction

- Variant of INTERNET SHOPPING problem
  [Blazewicz et al., 2010]
  - Discounts $\leftrightarrow$ free shipping depending on specific sets of purchased items
  - Strongly NP-hard, even with free items and unit shipping costs

Introduction

- Variant of INTERNET SHOPPING problem
  [Blazewicz et al., 2010]
  - Discounts $\leftrightarrow$ free shipping depending on specific sets of purchased items
  - Strongly NP-hard, even with free items and unit shipping costs

- We seek a complete picture of the tractability of CLEVER SHOPPER, with respect to:
  - Number of books (n)
  - Number of shops (m)
  - Price range
    Constant? Polynomially bounded? Unconstrained?
  - Degree
    Few books per shops ? Few shops selling each book?

Introduction

- Variant of INTERNET SHOPPING problem [Blazewicz et al., 2010]
  - Discounts $\leftrightarrow$ free shipping depending on specific sets of purchased items
  - Strongly NP-hard, even with free items and unit shipping costs
- We seek a complete picture of the tractability of CLEVER SHOPPER, with respect to:
  - Number of books (n)
  - Number of shops (m)
  - Price range
    Constant? Polynomially bounded? Unconstrained?
  - Degree
    Few books per shops ? Few shops selling each book?
- Any approximation algorithm?

Results

Sparse instances:

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices 3-SAT

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices 3-Sat
- Polynomial if shop degree ≤ 2 *Matching*

Few shops (parameter m):

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-Sat
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — Partition
- XP for m with polynomial prices — *dynamic programming*

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for “selected shops” with unit prices — PERFECT CODE

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for “selected shops” with unit prices — PERFECT CODE

Approximation:

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — PERFECT CODE

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — PERFECT CODE

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — MAX 3-SAT

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-Sat
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — Partition
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — Bin-Packing
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — Perfect Code

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — Max 3-Sat
- ... but k-approximable for shop-degree k — *greedy*

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-Sat
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — Partition
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — Bin-Packing
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — Perfect Code

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — Max 3-Sat
- ... but k-approximable for shop-degree k — *greedy*

Few books (parameter n):

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-Sat
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — Partition
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — Bin-Packing
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — Perfect Code

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — Max 3-Sat
- ... but k-approximable for shop-degree k — *greedy*

Few books (parameter n):

- FPT — *dynamic programming*

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — PERFECT CODE

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — MAX 3-SAT
- ... but k-approximable for shop-degree k — *greedy*

Few books (parameter n):

- FPT — *dynamic programming*
- No polynomial kernel — OR-*composition of* X3C

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — PERFECT CODE

Approximation:

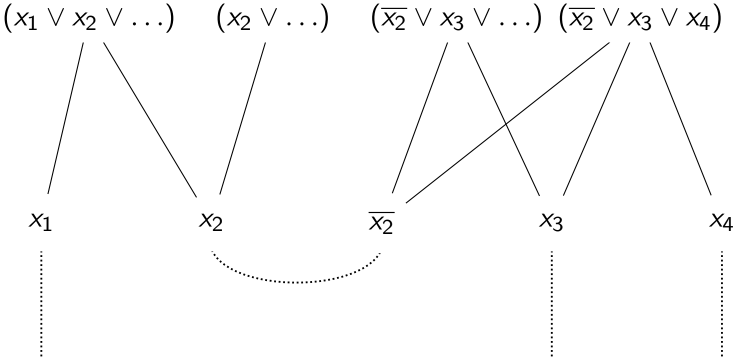
- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — MAX 3-SAT
- ... but k-approximable for shop-degree k — *greedy*

Few books (parameter n):

- FPT — *dynamic programming*
- No polynomial kernel — OR-*composition of* X3C

NP-hard with book-degree 2, shop-degree 3, unit prices

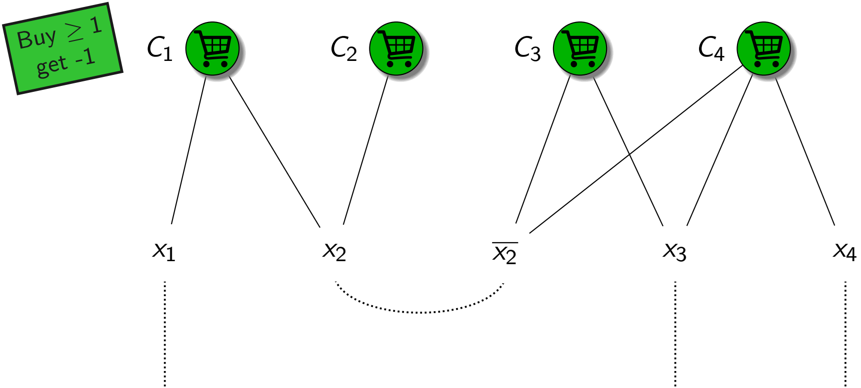
Reduction from $\textsc{Max 3-SAT}$, with 2 occurrences per literal



m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

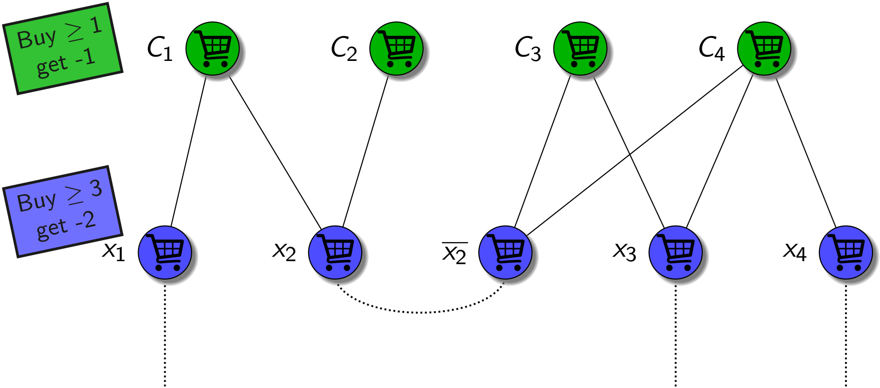
Reduction from MAX 3-SAT, with 2 occurrences per literal



m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

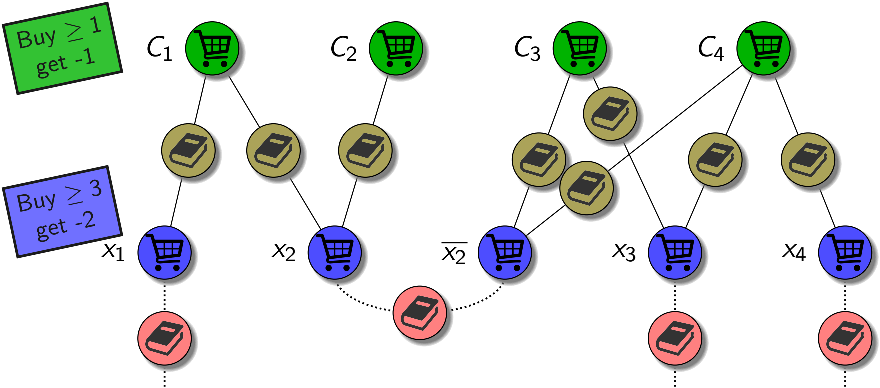
Reduction from MAX 3-SAT, with 2 occurrences per literal



m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

Reduction from MAX 3-SAT, with 2 occurrences per literal

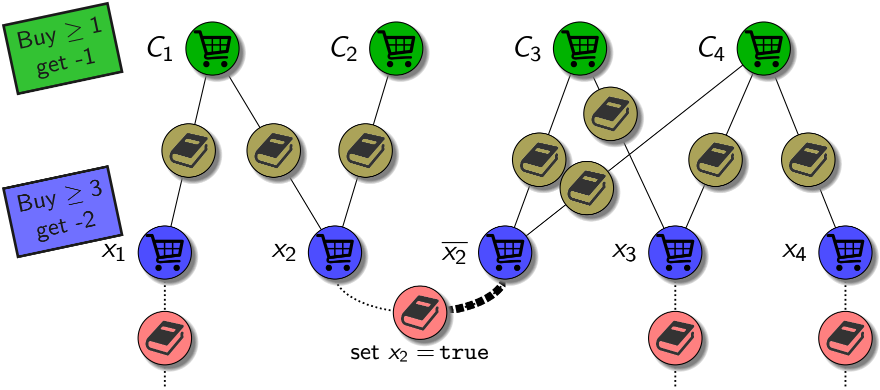


All prices = 1

m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

Reduction from MAX 3-SAT, with 2 occurrences per literal

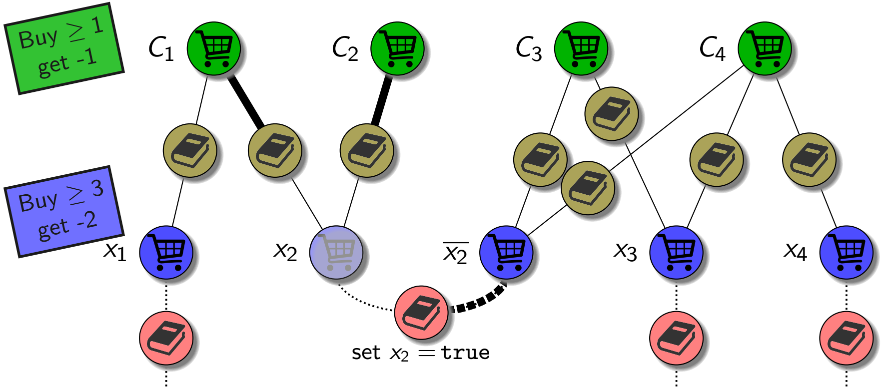


All prices $= 1$

m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

Reduction from MAX 3-SAT, with 2 occurrences per literal

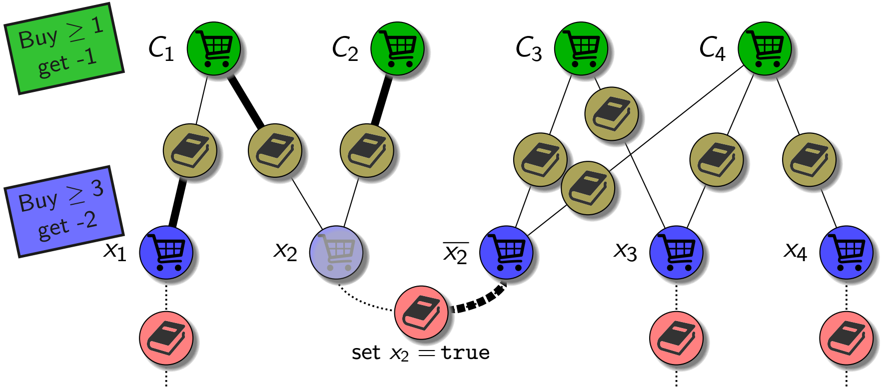


All prices = 1

m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

Reduction from MAX 3-SAT, with 2 occurrences per literal

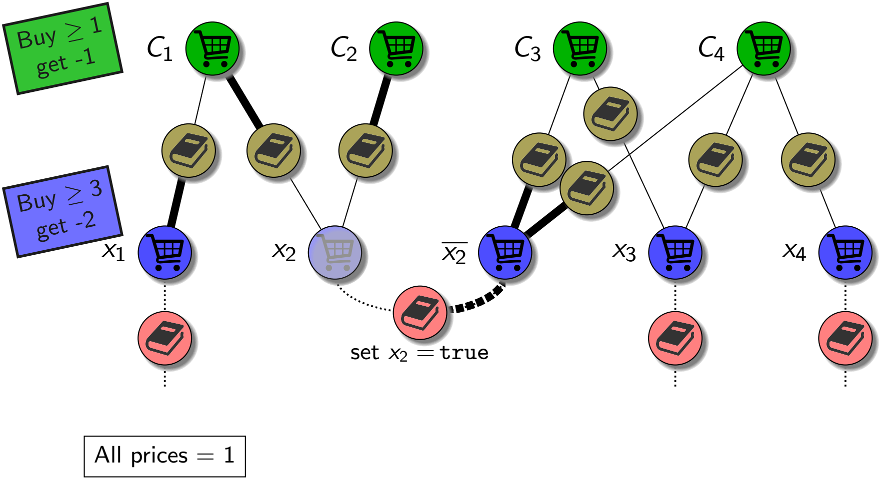


All prices = 1

m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

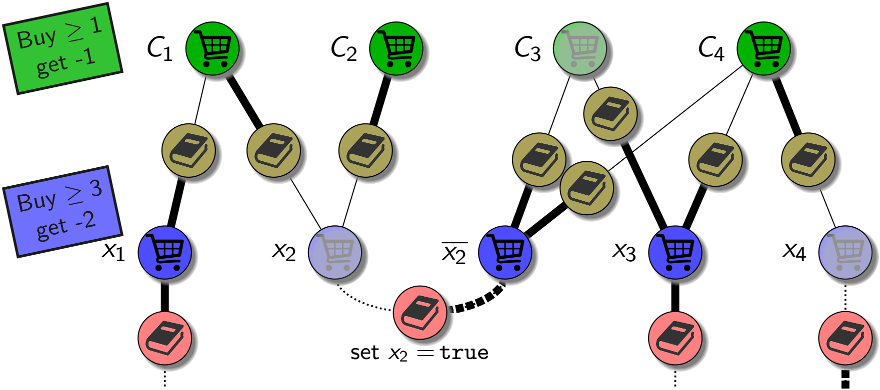
Reduction from MAX 3-SAT, with 2 occurrences per literal



m clauses, n variables

NP-hard with book-degree 2, shop-degree 3, unit prices

Reduction from MAX 3-SAT, with 2 occurrences per literal



All prices = 1

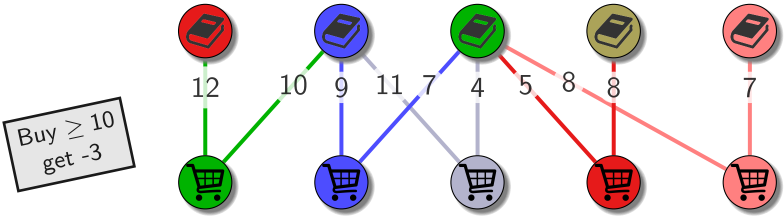
Total discount:
$2n$ + (1 per satisfied clause)

m clauses, n variables

Polynomial for shop-degree ≤ 2

Algorithm

- Subtract minimum price for each book from its incident edges

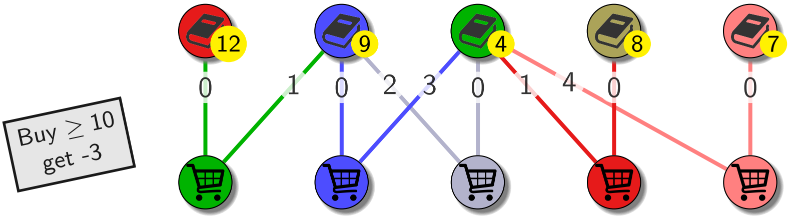


greedy solution $= (12 - 3) + 9 + 4 + 8 + 7 = 37$

Polynomial for shop-degree ≤ 2

Algorithm

- Subtract minimum price for each book from its incident edges
- Connect books sold by same shop with remaining cost

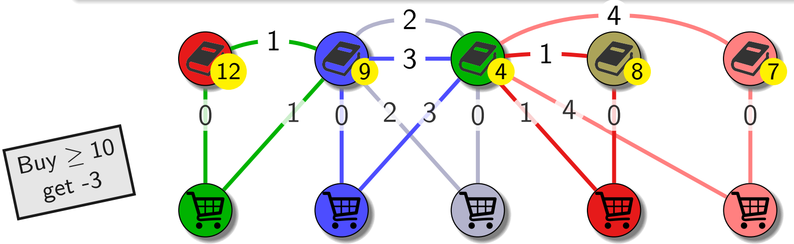


greedy solution $= (12-3)+9+4+8+7=37$

Polynomial for shop-degree ≤ 2

Algorithm

- Subtract minimum price for each book from its incident edges
- Connect books sold by same shop with remaining cost
- Subtract discount wherever threshold is reached

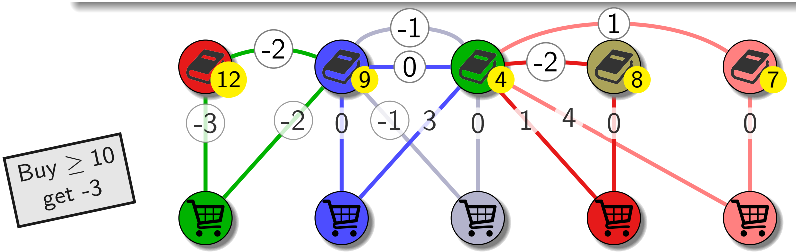


greedy solution $= (12 - 3) + 9 + 4 + 8 + 7 = 37$

Polynomial for shop-degree ≤ 2

Algorithm

- Subtract minimum price for each book from its incident edges
- Connect books sold by same shop with remaining cost
- Subtract discount wherever threshold is reached
- Find max weight matching (on graph with opposite weights)

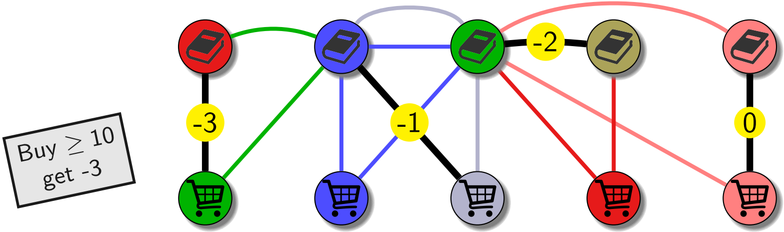


greedy solution $= (12 - 3) + 9 + 4 + 8 + 7 = 37$

Polynomial for shop-degree ≤ 2

Algorithm

- Subtract minimum price for each book from its incident edges
- Connect books sold by same shop with remaining cost
- $\boxed{\text{Subtract discount}}$ wherever threshold is reached
- Find max weight matching (on graph with opposite weights)
- Matched edges yield a solution

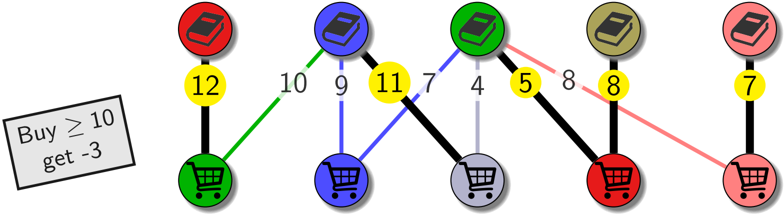


greedy solution $= (12 - 3) + 9 + 4 + 8 + 7 = 37$

Polynomial for shop-degree ≤ 2

Algorithm

- Subtract minimum price for each book from its incident edges
- Connect books sold by same shop with remaining cost
- Subtract discount wherever threshold is reached
- Find max weight matching (on graph with opposite weights)
- Matched edges yield a solution



greedy solution $= (12 - 3) + 9 + 4 + 8 + 7 = 37$

optimal solution $= (12 - 3) + (11 - 3) + (5 + 8 - 3) + 7 = 34$

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter *m*):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for *m* with polynomial prices — *dynamic programming*
- W[1]-hard for *m* with polynomial prices — BIN-PACKING
- FPT for *m* with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — PERFECT CODE

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — MAX 3-SAT
- ... but *k*-approximable for shop-degree *k* — *greedy*

Few books (parameter *n*):

- FPT — *dynamic programming*
- No polynomial kernel — OR-*composition of* X3C

NP-hard with 2 shops, unbounded prices

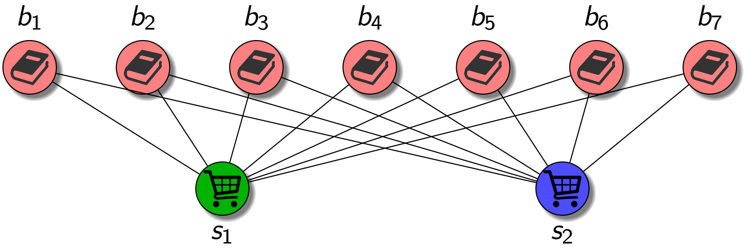
weakly NP-hard

Reduction from PARTITION

Input: $X = \{x_1, \ldots, x_n\}$ with $\sum_{x_i \in X} x_i = 2A$.

Question: $\exists ? X' \subseteq X$ such that $\sum_{x_i \in X'} x_i = A$

- 2 shops, n books
- book i has cost x_i (in both shops)
- discounts: buy A get -1
- budget: $2A - 2$



NP-hard with 2 shops, unbounded prices

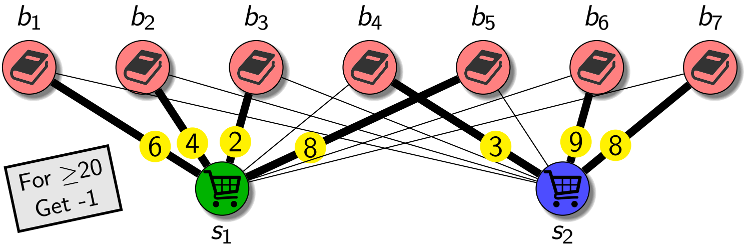
weakly NP-hard

Reduction from PARTITION

Input: $X = \{x_1, \ldots, x_n\}$ with $\sum_{x_i \in X} x_i = 2A$.

Question: $\exists ? X' \subseteq X$ such that $\sum_{x_i \in X'} x_i = A$

- 2 shops, n books
- book i has cost x_i (in both shops)
- discounts: buy A get -1
- budget: $2A - 2$



NP-hard with 2 shops, unbounded prices

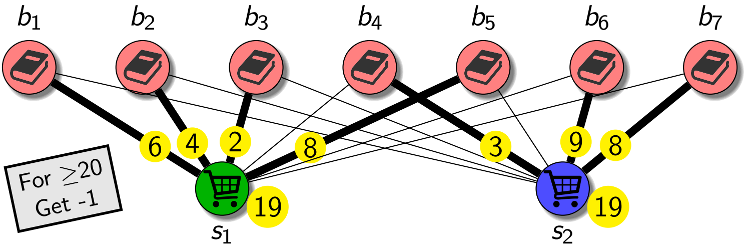
Reduction from PARTITION

Input: $X = \{x_1, \ldots, x_n\}$ with $\sum_{x_i \in X} x_i = 2A$.

Question: $\exists ? X' \subseteq X$ such that $\sum_{x_i \in X'} x_i = A$

weakly NP-hard

- 2 shops, n books
- book i has cost x_i (in both shops)
- discounts: buy A get -1
- budget: $2A - 2$



W[1]-hard for m shops, polynomial prices

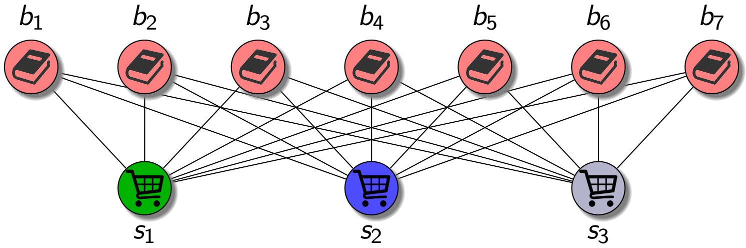
Reduction from BIN-PACKING

strongly W[1]-hard

Input: $X = \{x_1, \ldots, x_n\}$ with $\sum_{x_i \in X} x_i = mB$.

Question: $\exists ? (X_1, \ldots, X_m)$ partition of X with $\sum_{x_i \in X_j} x_i = B$.

- m shops, n books
- book i has cost x_i (in all shops)
- discounts: buy B get -1
- budget: $m(B-1)$



W[1]-hard for m shops, polynomial prices

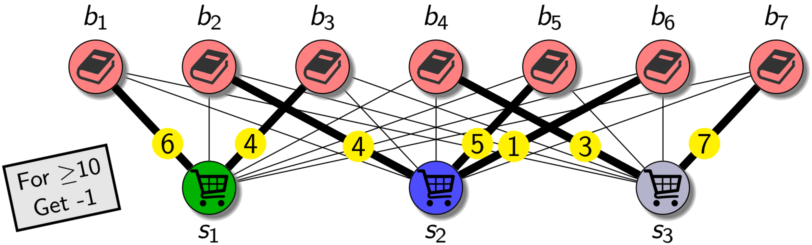
Reduction from BIN-PACKING

strongly W[1]-hard

Input: $X = \{x_1, \ldots, x_n\}$ with $\sum_{x_i \in X} x_i = mB$.

Question: $\exists ? (X_1, \ldots, X_m)$ partition of X with $\sum_{x_i \in X_j} x_i = B$.

- m shops, n books
- book i has cost x_i (in all shops)
- discounts: buy B get -1
- budget: $m(B-1)$



W[1]-hard for m shops, polynomial prices

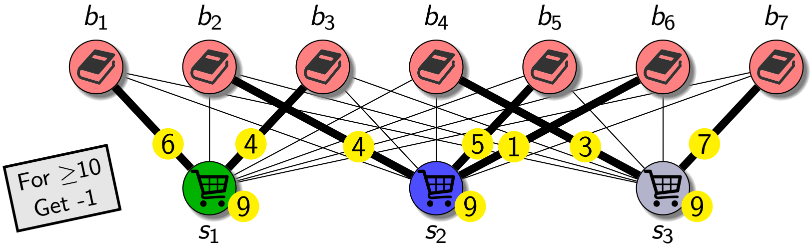
strongly W[1]-hard

Reduction from Bin-Packing

Input: $X = \{x_1, \ldots, x_n\}$ with $\sum_{x_i \in X} x_i = mB$.

Question: $\exists?(X_1, \ldots, X_m)$ partition of X with $\sum_{x_i \in X_j} x_i = B$.

- m shops, n books
- book i has cost x_i (in all shops)
- discounts: buy B get -1
- budget: $m(B-1)$



Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for "selected shops" with unit prices — PERFECT CODE

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount... — MAX 3-SAT
- ... but k-approximable for shop-degree k — *greedy*

Few books (parameter n):

- FPT — *dynamic programming*
- No polynomial kernel — OR-*composition of* X3C

The problem with approximations

When minimising the total cost: no approximation is possible.

The problem with approximations

When minimising the total cost: no approximation is possible.

- Take the (commercially questionable) discount function:

$$\boxed{\begin{array}{c}\text{Buy} \geq A \\ \text{Get } -A\end{array}}$$

- Partition, Bin-Packing or Perfect Code reductions yield:

  Clever Shopper is NP-hard, even with $K = 0$

The problem with approximations

When minimising the total cost: no approximation is possible.

- Take the (commercially questionable) discount function:

$$\boxed{\begin{array}{c}\text{Buy} \geq A \\ \text{Get } -A\end{array}}$$

- Partition, Bin-Packing or Perfect Code reductions yield:

  Clever Shopper is NP-hard, even with $K = 0$

Other optimisation strategy: maximise the total discount

- Meaningful only if each book has a uniform price
- Max 3-SAT reduction yields APX-hardness.

Results

Sparse instances:

- NP-hard with book-degree 2, shop-degree 3 and unit prices — 3-SAT
- Polynomial if shop degree ≤ 2 — *Matching*

Few shops (parameter m):

- NP-hard with 2 shops and unbounded prices — PARTITION
- XP for m with polynomial prices — *dynamic programming*
- W[1]-hard for m with polynomial prices — BIN-PACKING
- FPT for m with unit prices — *f-star subgraphs*
- W[1]-hard for “selected shops” with unit prices — PERFECT CODE

Approximation:

- No approximation is possible — NP-*hard with* $K = 0$
- APX-hard to maximise the total discount… — MAX 3-SAT
- … but k-approximable for shop-degree k — *greedy*

Few books (parameter n):

- FPT — *dynamic programming*
- No polynomial kernel — OR-*composition of* X3C

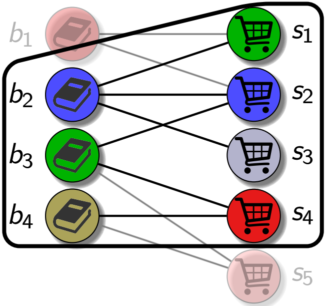
FPT for number of books n

Dynamic Programming Table:

$\forall B' \subseteq B,\ j \leq m,\ p_{\leq j}(B') :=$ Lowest possible price when buying books of B' from shops $\{s_1, \ldots, s_j\}$.

Recurrence:

$$p_{\leq j}(B') := \min_{B'' \subseteq B'} \left\{ p_{\leq j-1}(B' \setminus B'') + \text{cost for books } B'' \text{ in } s_j \right\}$$



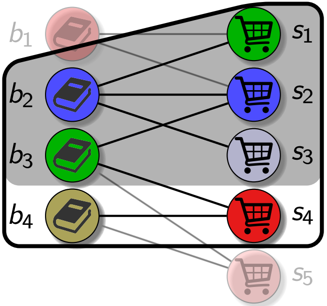
FPT for number of books n

Dynamic Programming Table:

$\forall B' \subseteq B,\ j \leq m,\ p_{\leq j}(B') :=$ Lowest possible price when buying books of B' from shops $\{s_1, \ldots, s_j\}$.

Recurrence:

$$p_{\leq j}(B') := \min_{B'' \subseteq B'} \left\{ p_{\leq j-1}(B' \setminus B'') + \text{cost for books } B'' \text{ in } s_j \right\}$$



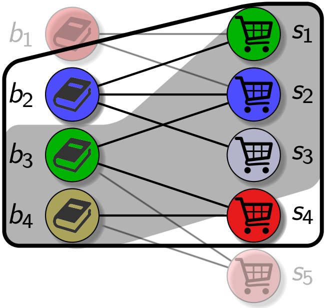
FPT for number of books n

Dynamic Programming Table:

$\forall B' \subseteq B,\ j \leq m,\ p_{\leq j}(B') :=$ Lowest possible price when buying books of B' from shops $\{s_1, \ldots, s_j\}$.

Recurrence:

$$p_{\leq j}(B') := \min_{B'' \subseteq B'} \left\{ p_{\leq j-1}(B' \setminus B'') + \text{cost for books } B'' \text{ in } s_j \right\}$$



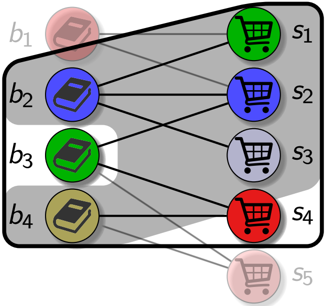
FPT for number of books n

Dynamic Programming Table:

$\forall B' \subseteq B,\ j \leq m,\ p_{\leq j}(B') :=$ Lowest possible price when buying books of B' from shops $\{s_1, \ldots, s_j\}$.

Recurrence:

$$p_{\leq j}(B') := \min_{B'' \subseteq B'} \left\{ p_{\leq j-1}(B' \setminus B'') + \text{cost for books } B'' \text{ in } s_j \right\}$$



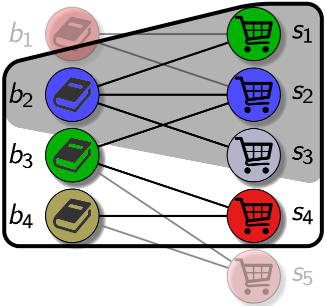
FPT for number of books n

Dynamic Programming Table:

$\forall B' \subseteq B,\ j \leq m,\ p_{\leq j}(B') :=$ Lowest possible price when buying books of B' from shops $\{s_1, \ldots, s_j\}$.

Recurrence:

$$p_{\leq j}(B') := \min_{B'' \subseteq B'} \left\{ p_{\leq j-1}(B' \setminus B'') + \text{cost for books } B'' \text{ in } s_j \right\}$$



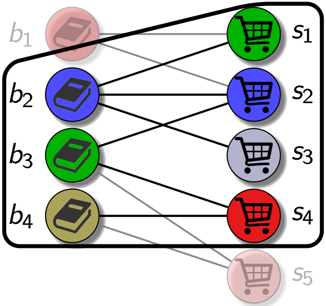
FPT for number of books n

Dynamic Programming Table:

$\forall B' \subseteq B,\ j \leq m,\ p_{\leq j}(B') :=$ Lowest possible price when buying books of B' from shops $\{s_1, \ldots, s_j\}$.

Recurrence:

$$p_{\leq j}(B') := \min_{B'' \subseteq B'} \{p_{\leq j-1}(B' \setminus B'') + \text{cost for books } B'' \text{ in } s_j\}$$



For each j: enumerate every $B'' \subseteq B' \subseteq B$

$\longrightarrow \mathcal{O}(m3^n)$

?? Open questions

- Constant-factor approximation (maximising total discount)?
- Kernel for parameter m with unit prices?
- FPT for number of shops + max. price?
- What if all books are available everywhere at constant price?

Thank you!