

情報理工学PBL II (A-18-2)

サービスチェイニング問題に対する整数線形計画問題の定式化

大規模システム管理研究室

はじめに

- 実習の概要

- NFVネットワークの資源割当問題の一つであるサービスチェイニング問題を整数線形計画問題として定式化する。数値実験より、NFVや整数線形計画問題の基本知識の理解を深める。

- 本講義の目的

- 自身の研究活動で活用できるように、整数線形計画問題の定式化方法とその解法を理解する
 - NFVネットワークにおける資源割当問題の一つであるサービスチェイン問題を理解する

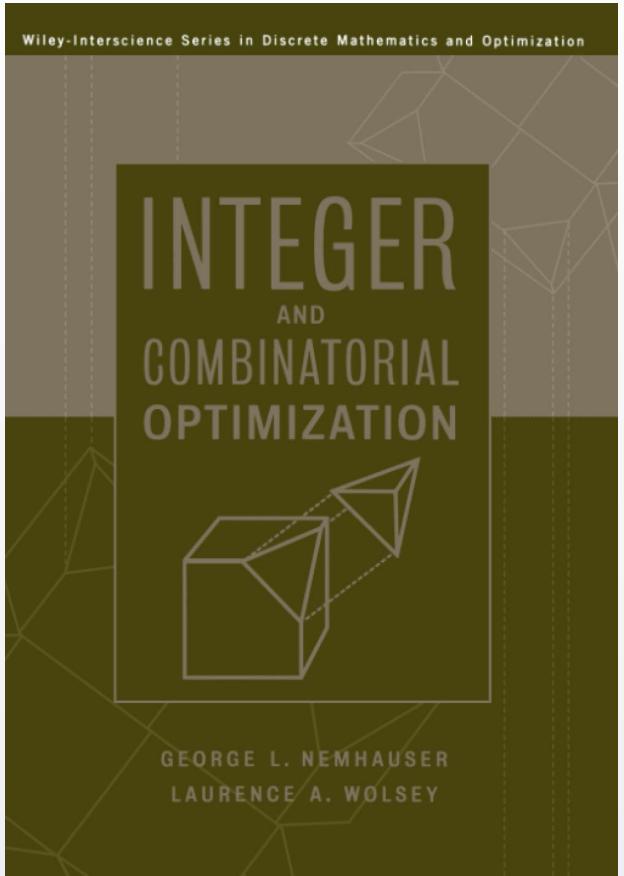
- PBLのプログラム

- <https://github.com/oakeshott/21pbl-a18-2>
 - 下記コマンドでDLして利用してください。
 - \$ git clone <https://github.com/oakeshott/21pbl-a18-2>

課題の提出方法

- PBLのすべての講義修了後1週間以内にDataboxで提出してください
 - URL:
 - 提出時のディレクトリ名: 学籍番号
 - Databoxに提出後 hara.takanori@is.naist.jp にメールしてください
 - 件名: [氏名, 学籍番号]PBL2の課題提出
- 課題
 - 課題1: Shortest path problemの定式化と最適解の導出
 - 課題2: CSPTPに基づくサービスチェイニングの定式化と最適解の導出
 - 発展課題: バッチ処理可能なCSPTPに基づくサービスチェイニングの定式化と最適解の導出
 - 課題3: TUMの特性の確認
- [Required]: 課題1-3
- [Optional]: 発展課題

参考文献



Wolsey, Laurence A., and George L. Nemhauser. *Integer and combinatorial optimization*. Vol. 55. John Wiley & Sons, 1999.



福島雅夫, 新版 数理計画入門, 朝倉書店, 2011

目次

- 最適化問題
 - 線形計画問題
 - 整数線形計画問題
 - 組合せ最適化
 - 分枝限定法
 - 緩和問題
 - CPLEX + pulpによる最適化問題の定式化と解法
- ネットワーク仮想化と最適化問題
 - サービスチェイニング問題とShortest path tour problem (SPTP)
 - 容量制約付きSPTP
- Totally unimodular matrix

III 最適化問題

■ 最適化問題

- 入力: 解の候補集合 \mathcal{X} と目的関数 $f : \mathcal{X} \rightarrow \mathbb{R}$
- 目的: $f(x)$ を最小化する $x \in \mathcal{X}$ を見つける

■ 最適化問題の形式的な表現

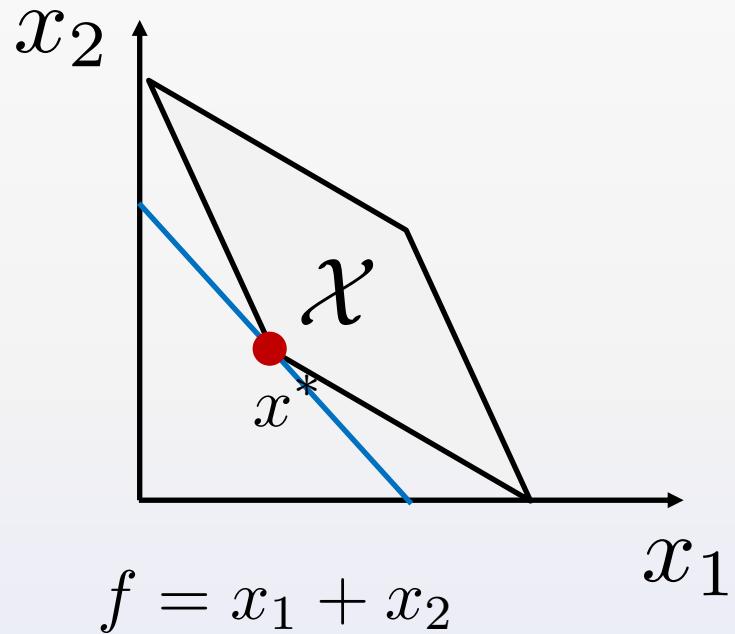
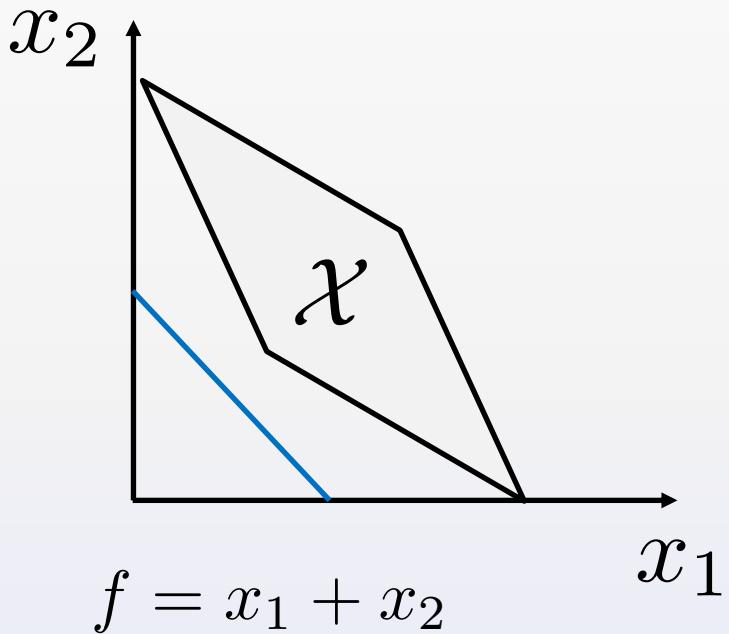
$$\min f(x)$$

$$\text{s.t. } x \in \mathcal{X}$$

- $f(x)$ の最小値を実現する $x^* \in \mathcal{X}$ を最適解と呼ぶ
- 最適解 x^* に対して $f(x^*)$ を最適値と呼ぶ

線形計画問題

- 目的関数と制約条件がすべて線形である最適化問題
 - $f(x)$ や $x \in \mathcal{X}$ が線形である最適化問題



線形計画問題の性質

- $x \in \mathbb{R}^n$ が線形計画問題の実行可能解 $\Leftrightarrow x \in \mathbb{R}^n$ かつ $Ax \leq b$ を満たす
- 定理
 - 任意の線形計画問題について以下が成立する
 - 実行可能解が存在しない
 - 実行可能解が存在し、目的関数の値に下限がない
 - 実行可能解が存在し、目的関数の値に下限があるさらに下限を達成する実行可能解が存在する

$$\min c^T x$$

$$\text{s.t. } Ax \leq b$$

$$x \in \mathbb{R}^n$$

III 整数線形計画問題

- 線形計画問題において解の各要素を整数に限定した問題
 - 線形計画問題: 目的関数と制約条件がすべて線形である最適化問題
 - 解ベクトルを0, 1に限定したものを0-1整数線形計画問題と呼ぶ

$$\min \quad c^T x$$

$$\text{s.t.} \quad Ax \leq b$$

$$x \in \mathbb{Z}^n$$

整数線形計画問題の例

$$\min \quad c^T x$$

$$\text{s.t.} \quad Ax \leq b$$

$$x \in \{0, 1\}$$

0-1整数線形計画問題の例

組合せ最適化問題

- 組合せ最適化 (離散最適化)
 - 解が組合せ的な構造を持つ最適化問題
 - 例: 巡回セールスマン問題, ナップサック問題
- 最適解を現実的な時間で求めることが難しい
 - すべての組合せを列挙すれば解ける→組合せ爆発
 - →解の全列挙の無駄を可能な限り省略する
 - 動的計画法: 同一の部分問題を繰り返し解かない
 - 分枝限定法: ある部分問題から最適解が得られないことが確認できた場合, その部分問題を無視する
- 現実的な時間で問題を解くには近似解法(ヒューリスティクス)が必要

分枝限定法

- 組合せ問題の最適解を求める汎用アルゴリズム
 - 分枝操作: 組合せ問題を部分問題に分解
 - 限定操作: 解く必要がない部分問題を検出できたら分岐操作を停止
 - 暫定解と緩和問題により無駄な部分問題を確認
 - 暫定解: 分枝限定法による計算により、得られた最良の許容解
 - 緩和問題: 元の組合せ問題の制約を一部緩和して得られる問題
- 部分問題を解く必要のないケース
 - 既に最適解が得られている
 - 現在の暫定解より良い許容解が得られる可能性がない
 - 許容解が存在しない (Infeasible)

分枝限定法

$$\max \quad 10x_1 + 2x_2 + 8x_3$$

x_1 を0に固定

部分問題

s.t. $3x_1 + 8x_2 + 10x_3 \leq 50$

$x_j \in \{0, 1\}, \quad j = 1, 2, 3$

x_1 を1に固定

部分問題

$$\max \quad 2x_2 + 8x_3$$

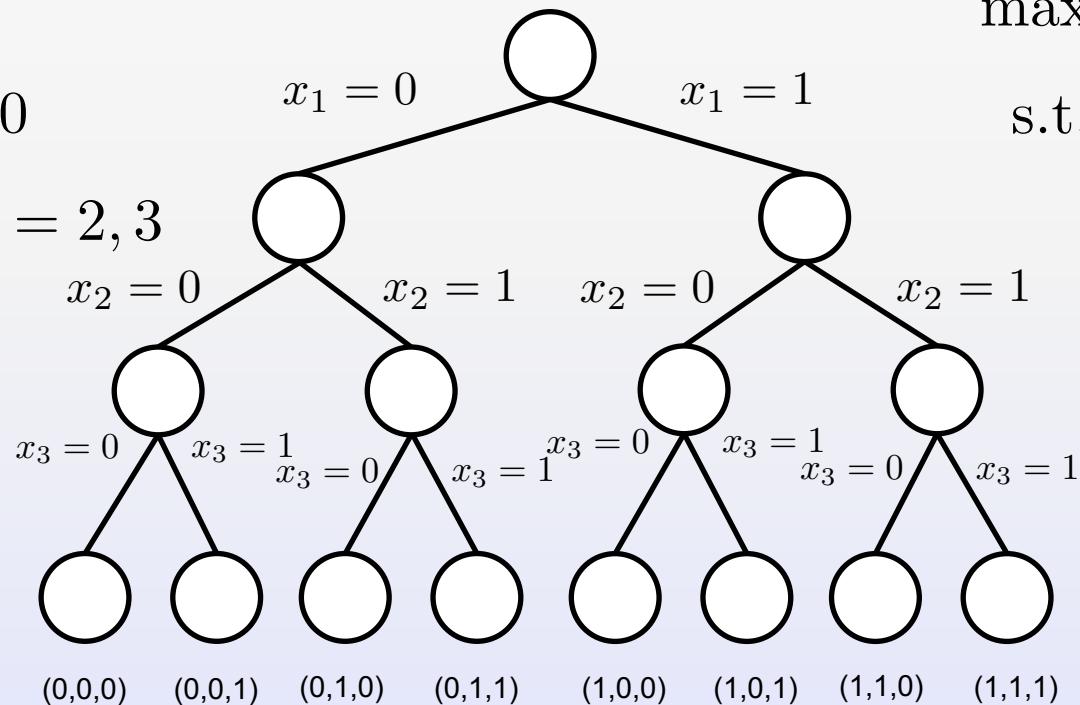
s.t. $8x_2 + 10x_3 \leq 50$

$x_j \in \{0, 1\}, \quad j = 2, 3$

$$\max \quad 10 + 2x_2 + 8x_3$$

s.t. $3 + 8x_2 + 10x_3 \leq 50$

$x_j \in \{0, 1\}, \quad j = 2, 3$



分枝限定法のフロー

- 部分問題の集合: L , 暫定解: x^* , 暫定値: y

1. 初期化

- $L = \{\text{元問題}\}, z = \infty$

2. 探索

- L が空なら計算終了, x^* が最適解 (x^* が未定義の場合, Infeasible)
- L が非空なら L から部分問題 P を選択し, P を削除 $L = L - \{P\}$

3. 限定操作

- P がInfeasibleならステップ1へ
- P で最適解が得られた場合, x^* , z を更新(必要ならば)し, ステップ1へ
- P の緩和問題を解き, 暫定解より良い許容解が得られない場合, ステップ1へ

4. 分枝操作

- 問題 P を部分問題 $\{P_1, P_2, \dots, P_k\}$ に分解し, L に追加した後 ($L = L \cup \{P_1, P_2, \dots, P_k\}$),
ステップ1へ

緩和問題

- 元の最適化問題の制約条件を一部緩和して得られる問題
 - e.g., 線形緩和
- 線形問題は多項式時間で解ける
 - 緩和問題の実行可能解集合は元の問題の実行可能解集合を含む
 - 緩和問題の最適値の上界は元の問題の最適値
 - 緩和問題の最小値 \leq 元の問題の最小値

$$\min \quad c^T x$$

$$\text{s.t.} \quad Ax \leq b$$

$$x \in \{0, 1\}$$



$$\min \quad c^T x$$

$$\text{s.t.} \quad Ax \leq b$$

$$0 \leq x \leq 1$$

解空間(実行可能解集合)を緩和

例: 最小費用流問題 (Minimum cost flow problem)

■ 最小費用流問題

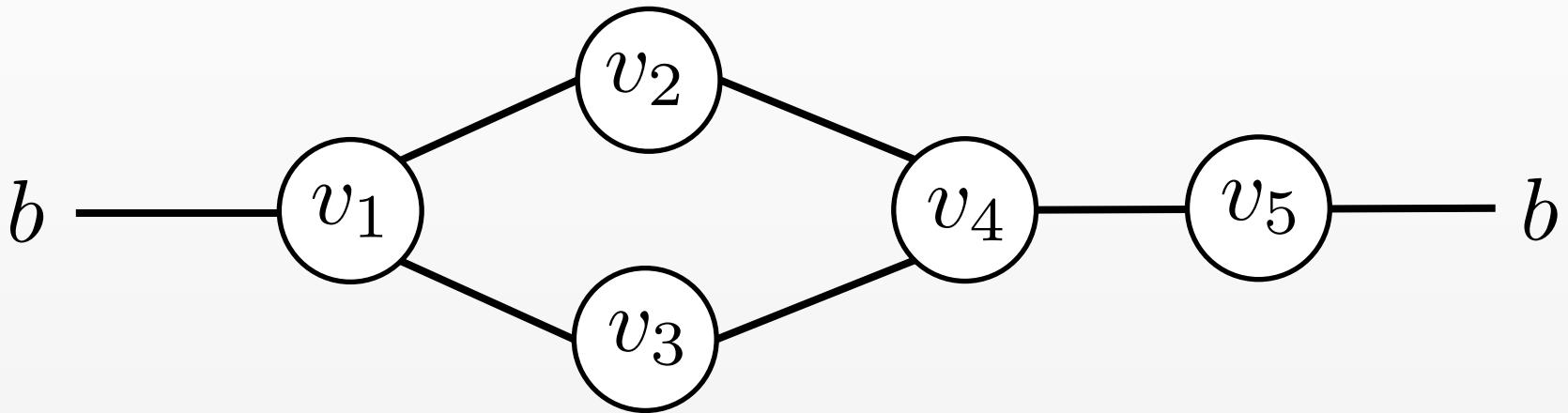
- 入力: 有向グラフ $G = (\mathcal{V}, \mathcal{E})$ $\mathcal{E} = \mathcal{E}^{\text{in}} \cup \mathcal{E}^{\text{out}}$
 - 各頂点 $i \in \mathcal{V}$ の供給・需要量 b_i
 $b_i > 0$ の場合 i は供給点, $b_i < 0$ の場合 i は需要点, $b_i = 0$ の場合 i は通過点
 - 各辺 $(i, j) \in \mathcal{E}$ の容量 $u_{i,j} \geq 0$, 費用 $c_{i,j}$
- 出力: 総費用が最小となる需要供給を満たすフロー

■ 定式化

$$\begin{aligned} \min \quad & \sum_{i,j \in \mathcal{E}} c_{i,j} x_{i,j} \\ \text{s.t.} \quad & 0 \leq x_{i,j} \leq u_{i,j}, \quad (i, j) \in \mathcal{E} \\ & \sum_{i,j \in \mathcal{E}^{\text{in}}} x_{i,j} - \sum_{j,i \in \mathcal{E}^{\text{out}}} x_{j,i} = b_i, \quad i \in \mathcal{V} \end{aligned} \quad \left. \begin{array}{l} \text{目的関数 (最小化)} \\ \text{制約} \end{array} \right\}$$

補足: フロー保存則

- フロー b を始点(v_1)から終点(v_5)に流す



$$x_{v_1,v_2} + x_{v_1,v_3} - x_{v_2,v_1} - x_{v_3,v_1} = b$$

$$x_{v_1,v_2} + x_{v_2,v_4} - x_{v_2,v_1} - x_{v_4,v_2} = 0$$

$$x_{v_1,v_3} + x_{v_3,v_4} - x_{v_3,v_1} - x_{v_3,v_2} = 0$$

$$x_{v_3,v_4} + x_{v_2,v_4} + x_{v_5,v_4} - x_{v_4,v_3} - x_{v_4,v_2} - x_{v_4,v_5} = 0$$

$$x_{v_4,v_5} - x_{v_5,v_4} = b$$

III 定式化に利用するツール

- CPLEX

- 商用ソルバの一つ
 - 学生の場合、無料で利用可能
 - ダウンロード・インストール方法は後に示す

- Pulp

- PythonによるLPモデル作成ライブラリ
 - <https://github.com/coin-or/pulp>
 - LPファイルの生成、LPファイルの実行するためのラッパー

- Networkx

- グラフを扱うライブラリ
 - <https://networkx.org/>

- Pandas

- テーブルデータを扱うライブラリ

CPLEX + pulpによる最適化問題の解法

- Pulpを利用してLpファイルを生成する
 - 最適化問題(目的関数と制約条件)を定義する
 - 制約条件は行列式として表現可能なためNetworkxやPandasを利用して効率的に定義できる
- Lpファイルを入力としてCPLEXを用いて最適化問題を解く
 - Pulpを利用することで、Pythonプログラム上でCPLEXを呼び出せる



例: 最小費用流問題 (Minimum cost flow problem)

- 例: Pulp + Networkxを利用した定式化

- https://github.com/oakeshott/21pbl-a18-2/blob/master/examples/mcfp/minimum_cost_flow_problem.py

- 例: LPファイル

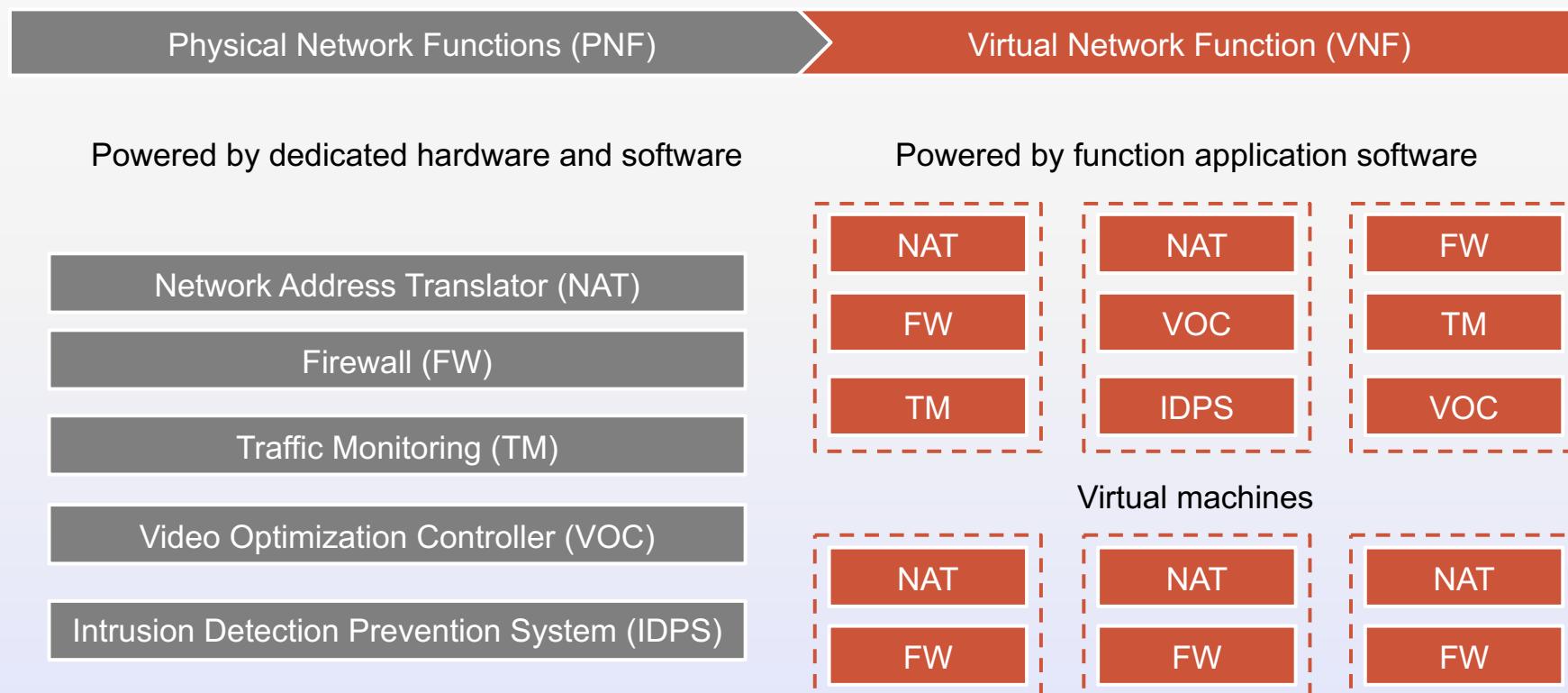
```
1  /* Minimum_cost_flow_problem */
2  Minimize
3  OBJ: 3 x_0,1 + 7 x_0,2 + x_1,2 + 2 x_1,3 + 9 x_2,3 + x_2,4 + 7 x_3,4 + 8 x_3,5
4  + 4 x_4,5
5  Subject To
6  capacity_constraint_of_link_0,1: x_0,1 <= 5
7  capacity_constraint_of_link_0,2: x_0,2 <= 2
8  capacity_constraint_of_link_1,2: x_1,2 <= 6
9  capacity_constraint_of_link_1,3: x_1,3 <= 4
10 capacity_constraint_of_link_2,3: x_2,3 <= 2
11 capacity_constraint_of_link_2,4: x_2,4 <= 3
12 capacity_constraint_of_link_3,4: x_3,4 <= 3
13 capacity_constraint_of_link_3,5: x_3,5 <= 1
14 capacity_constraint_of_link_4,5: x_4,5 <= 9
15 flow_constraint_of_node_0: x_0,1 + x_0,2 = 6
16 flow_constraint_of_node_1: - x_0,1 + x_1,2 + x_1,3 = 0
17 flow_constraint_of_node_2: - x_0,2 - x_1,2 + x_2,3 + x_2,4 = 0
18 flow_constraint_of_node_3: - x_1,3 - x_2,3 + x_3,4 + x_3,5 = 0
19 flow_constraint_of_node_4: - x_2,4 - x_3,4 + x_4,5 = 0
20 flow_constraint_of_node_5: - x_3,5 - x_4,5 = -6
21 End
```

課題1

- Shortest Path Problemを整数線形計画問題として定式化し、
CPLEXを用いて最適解を導出せよ
 - examples/mcfp/dataのデータを利用した結果を示してください
- 最適解とDisjktra algorithmで得られた解を比較せよ
 - Dijkstra algorithm:
https://networkx.org/documentation/stable/reference/algorithms/shorest_paths.html

ネットワーク機能仮想化 (Network functions virtualization: NFV)

- ネットワーク機能を専用機器から分離
汎用機器上で仮想ネットワーク機能 (Virtual network function: VNF) として実行
- 多様なユーザ要求に対して、迅速かつ柔軟にネットワークサービスを展開可能
- 導入・運用コストの削減を実現



サービスチェイニング問題

- サービスチェイン
 - ネットワークサービスは複数のVNFからなる列 (サービスチェイン) として構成
 - ネットワークサービス: e.g., Video streaming
- サービスチェイニング問題
 - 資源制約の下で、所望の順序で中間ノード上にあるVNFを実行しながら始点から終点までのサービスパスを構築する問題
 - VNFの実行順序を保証しなければならない
 - NP困難であることが知られている



Network service (service chain): Video streaming

サービスチェイニングとShortest Path Tour Problem

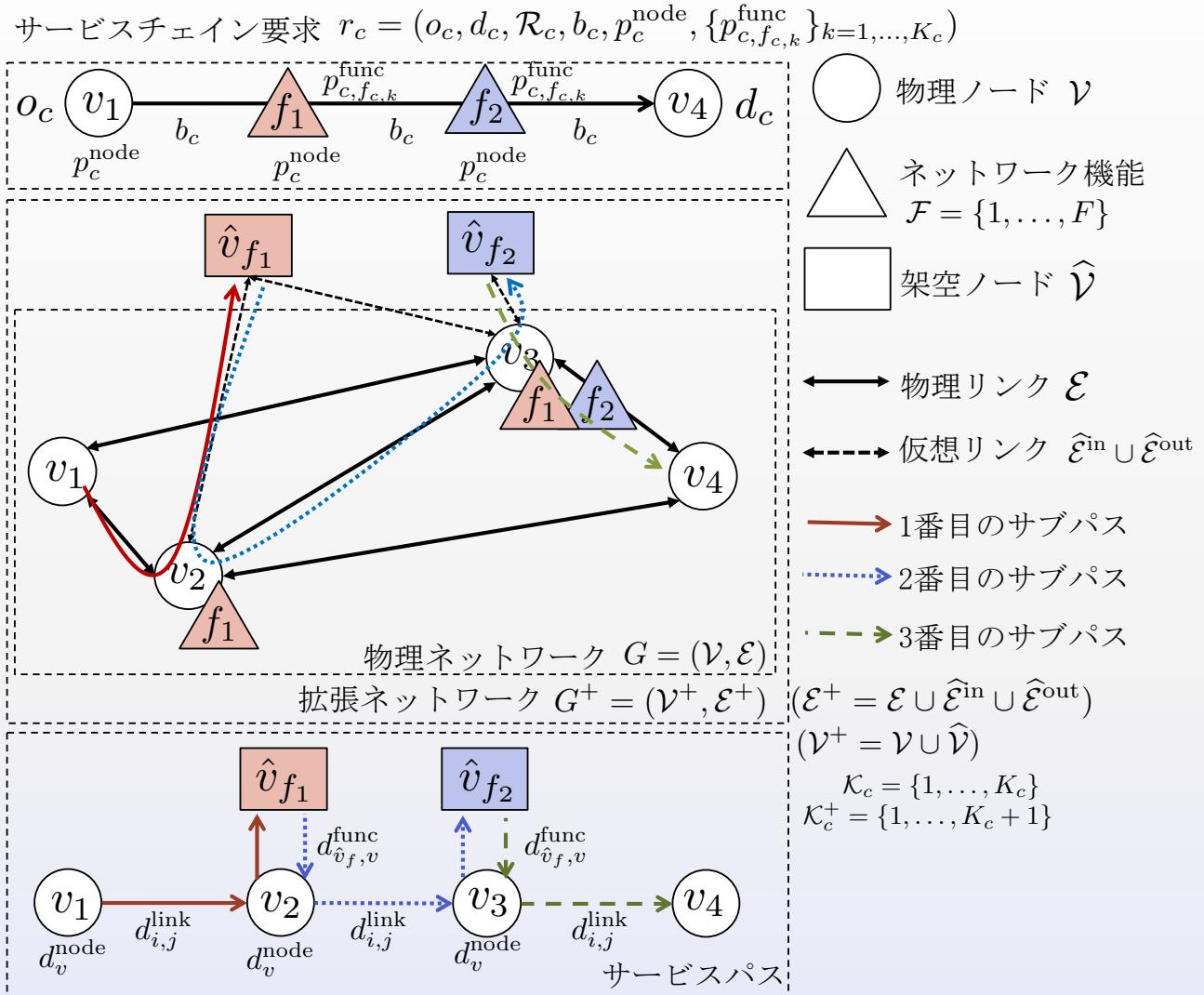
- サービスチェイニング問題とSPTPの類似性 [Bhat+17]
 - Shortest Path Tour Problem (SPTP):
 - 素なノード集合に含まれるノードを所望の順序で経由しながら、始点から終点までの最短路を計算する問題
- サービスチェイニング問題とSPTPの相違点
 - 資源制約（ノードやリンクに容量制約）が含まれる
- 制約付きSPTP (Constrained SPTP) [Ferone+16]
 - 同一リンクの利用を高々1回に制限したSPTP
- 容量制約付きSPTP (Capacitated SPTP: CSPTP) [Sasabe+21]
 - 制約付きSPTPに対してノードやリンクの容量制約を実数として与えることでモデル化
 - サービスチェイニング問題をCSPTPに基づくILPとして定式化

[Bhat+17] S. Bhat and G.N. Rouskas, “Service-Concatenation Routing with Applications to Network Functions Virtualization,” Proc. of 26th International Conference on Computer Communication and Networks (ICCCN), pp.1–9, July 2017.

[Ferone+16] D. Ferone, P. Festa, F. Guerriero, and D. Lagan`a, “The Constrained Shortest Path Tour Problem,” Computers & Operations Research, vol.74, pp.64–77, Oct. 2016.

[Sasabe+21] M. Sasabe and T. Hara, “Capacitated Shortest Path Tour Problem Based Integer Linear Programming for Service Chaining and Function Placement in NFV Networks,” IEEE Transactions on Network and Service Management, vol.18, no.1, pp.104–117, March 2021.

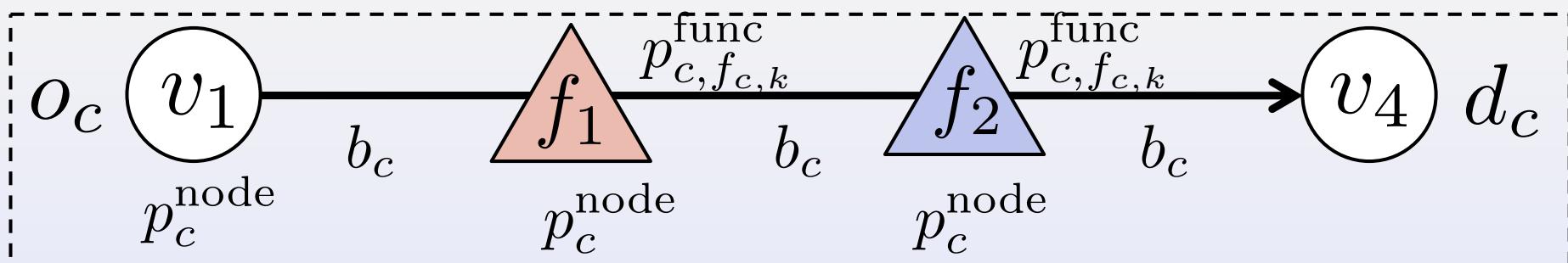
システムモデル



システムモデル: サービスチェイン要求

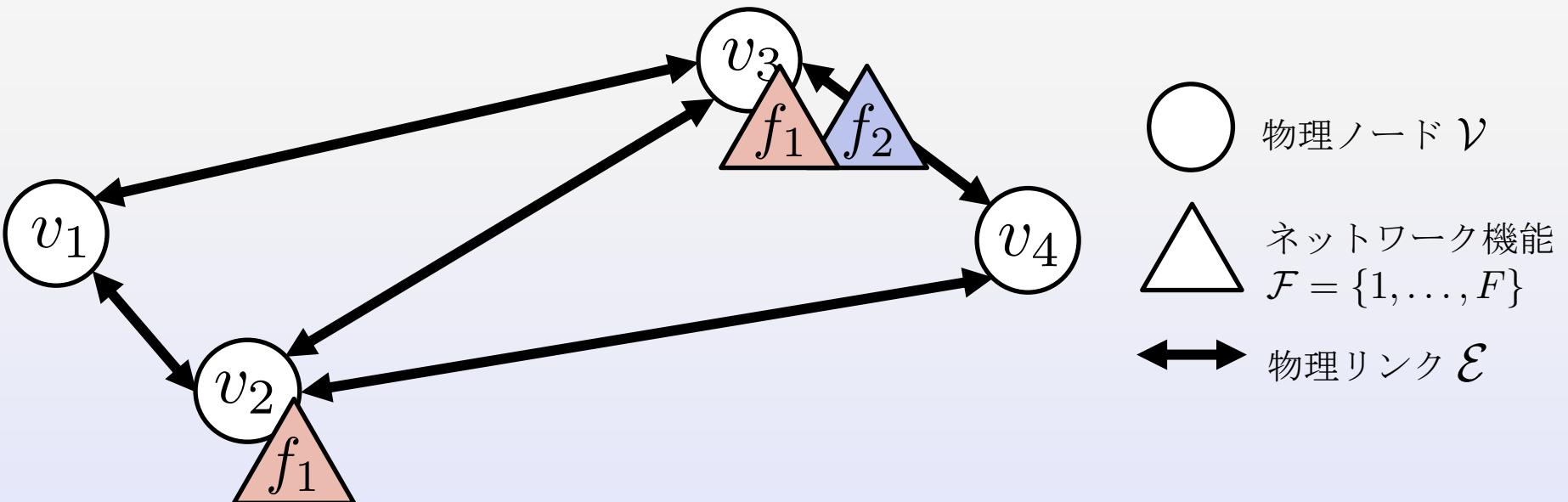
- サービスチェイン要求はサービス要件 r_c を持つ
- NFV orchestratorはサービス要件が満たされるようにサービスパスを確立する
 - VNFの実行順序を保証

$$r_c = (o_c, d_c, \mathcal{R}_c, b_c, p_c^{\text{node}}, \{p_{c,f_{c,k}}^{\text{func}}\}_{k=1,\dots,K_c})$$



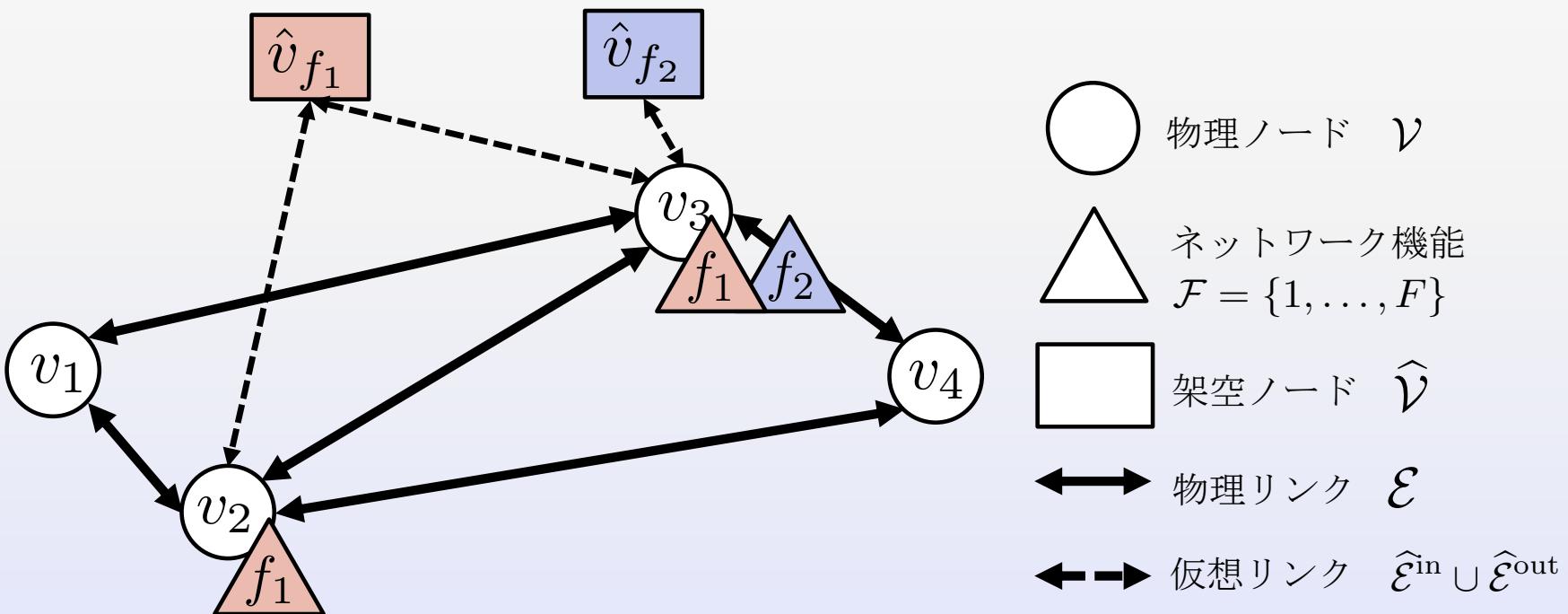
III システムモデル: 物理ネットワーク

- 物理ネットワーク
 - 有向グラフ $G = (\mathcal{V}, \mathcal{E})$ として表現
 - ノード: 物理ノード, 残余処理容量 P_i
 - リンク: 物理リンク, 残余帯域容量 $B_{i,j}$
- NFVネットワークは F 個のネットワーク機能を提供する
 - 各機能 $f \in \mathcal{F}$ は N_f 個の物理ノードによって提供されている



III システムモデル: 拡張ネットワーク

- 物理ネットワークを架空ノード, 仮想リンクにより拡張したネットワーク
 - $G^+ = (\mathcal{V}^+, \mathcal{E}^+)$
 - 架空ノード $\hat{v}_{f_{c,k}} \in \hat{\mathcal{V}}$ は k 番目のネットワーク機能 $f_{c,k}$ に対応している
 - 仮想リンク $(i, \hat{v}_{f_{c,k}})$ は物理ノード i がネットワーク機能 $f_{c,k}$ をサポートしている
 - 仮想リンク $(i, \hat{v}_{f_{c,k}})$ の選択は物理ノード i がネットワーク機能 $f_{c,k}$ を実行することに対応

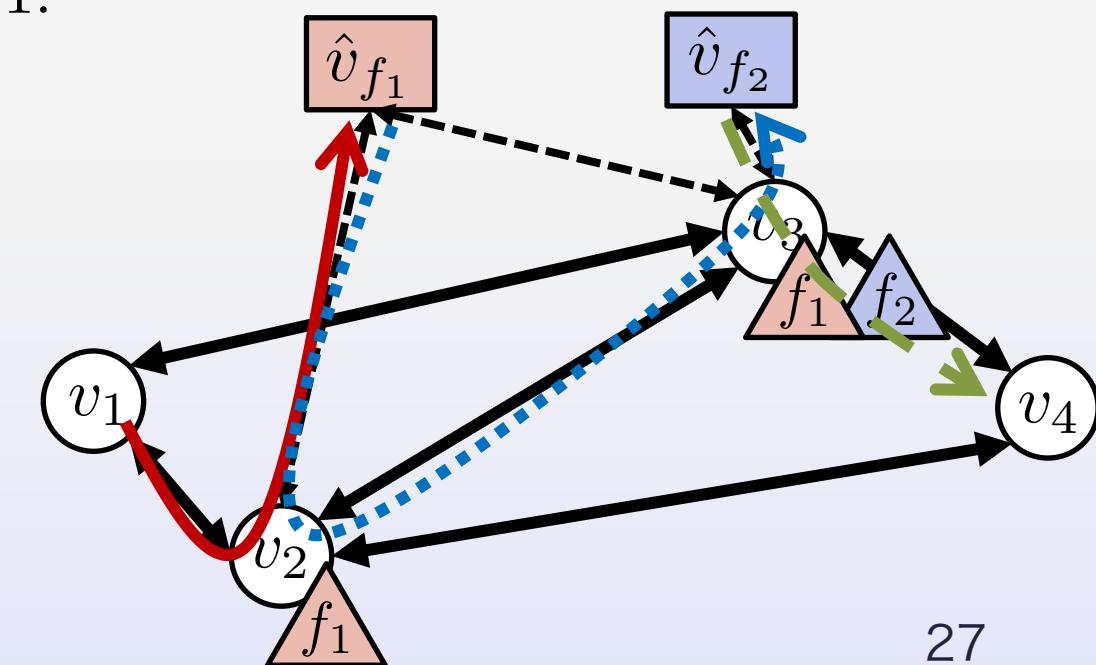
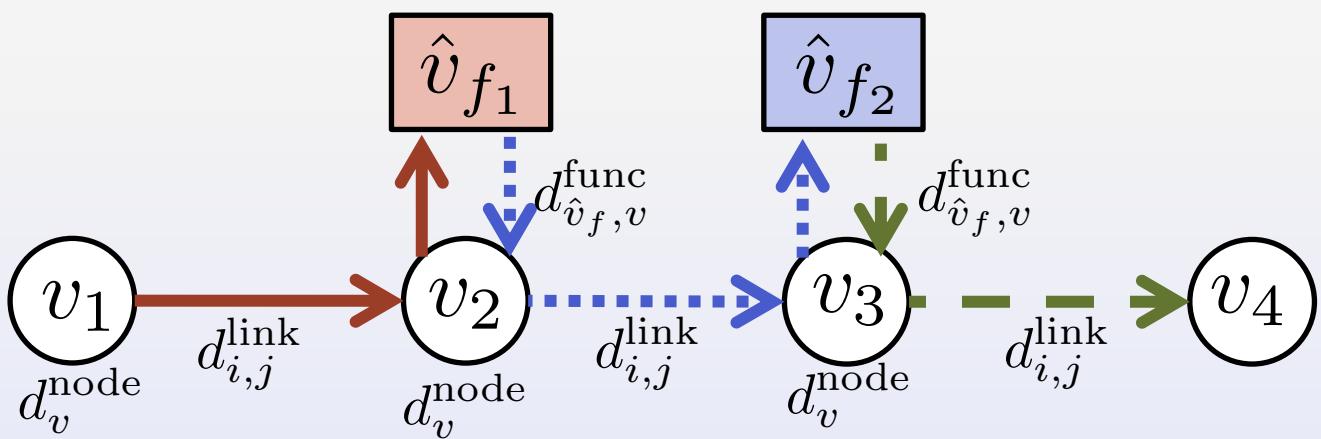


システムモデル: サービスパス

- サービスパスは $K_c + 1$ 個のサブパスに分解できる
 - 各サブパスの始点・終点

$$(a_{c,k}, b_{c,k}) = \begin{cases} (o_c, \hat{v}_{f_{c,1}}), & k = 1, \\ (\hat{v}_{f_{c,k-1}}, \hat{v}_{f_{c,k}}), & k = 2, \dots, K_c, \\ (\hat{v}_{f_{c,K_c}}, d_c), & k = K_c + 1. \end{cases}$$

サブパスはループを含まない
サービスパスはループを含むかもしれない



サンプルプログラム

- 物理ネットワーク・拡張ネットワークの構築に関するサンプルプログラム
 - https://github.com/oakeshott/21pbl-a18-2/blob/master/examples/system-model-for-SC/network_model.py
 - 拡張ネットワーク
 - 架空ノード
 - 架空ノードをネットワーク機能のID+物理ノード数として定義している
 - 架空ノードのIDからネットワーク機能のIDを参照できる

.....

各数値は以下のネットワーク機能に対応する
便宜上、数値で表現している。

# NAT: Network Address Translator.	1
# FW: Firewall	2
# TM: Traffic Monitoring	3
# IDPS: Intrusion Detection Prevention System.	4
# VOC: Video Optimization Controller	5

.....

容量制約付きSPTP (CSPTP: Capaciated SPTP)

$$\min_x \sum_{(i,j) \in \mathcal{E}^+} d_{i,j} \sum_{k \in \mathcal{K}_c^+} x_{i,j}^{c,k}, \quad (1)$$

$$\text{s.t. } x_{i,j}^{c,k} = \{0, 1\}, \quad (i, j) \in \mathcal{E}^+, k \in \mathcal{K}_c^+ \quad (2)$$

$$\sum_{j \in \mathcal{V}_i^+} x_{i,j}^{c,k} - \sum_{j \in \mathcal{V}_i^+} x_{j,i}^{c,k} = \begin{cases} 1 & \text{if } i = a_{c,k} \\ -1 & \text{if } i = b_{c,k} \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

$i \in \mathcal{V}^+, k \in \mathcal{K}_c^+$

$$x_{i, \hat{v}_{f_{c,k}}}^{c,k} = x_{\hat{v}_{f_{c,k}}, i}^{c,k+1}, \quad (4)$$

$$(i, \hat{v}_{f_{c,k}}) \in \hat{\mathcal{E}}^{\text{in}}, (\hat{v}_{f_{c,k}}, i) \in \hat{\mathcal{E}}^{\text{out}}, k \in \mathcal{K}_c \quad (5)$$

$$x_{i, \hat{v}_{f_{c,m}}}^{c,k} = 0, \quad (i, \hat{v}_{f_{c,m}}) \in \hat{\mathcal{E}}^{\text{in}}, k \in \mathcal{K}_c^+, m \neq k \quad (6)$$

$$b_c \sum_{k \in \mathcal{K}_c^+} x_{i,j}^{c,k} \leq B_{i,j}, \quad (i, j) \in \mathcal{E} \quad (7)$$

$$p_c^{\text{node}} \sum_{(v,j) \in \mathcal{E}} \sum_{k \in \mathcal{K}_c^+} x_{v,j}^{c,k} + \sum_{(\hat{v}_f, v) \in \hat{\mathcal{E}}^{\text{out}}} p_{c,f}^{\text{func}} \sum_{k \in \mathcal{K}_c^+} x_{\hat{v}_f, v}^{c,k} \leq P_v, \quad (8)$$

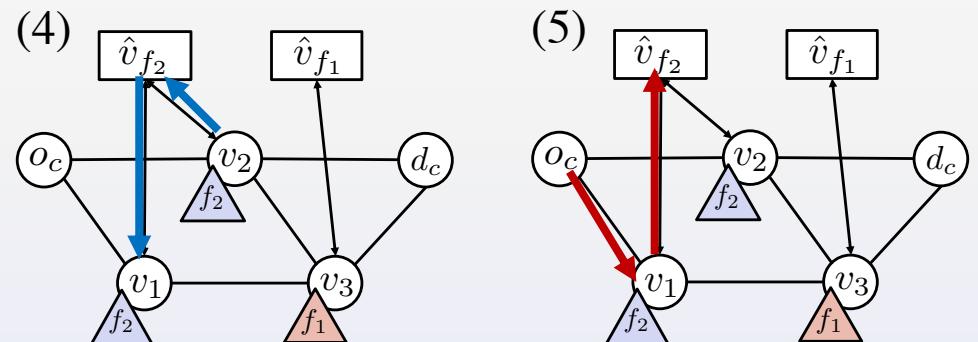
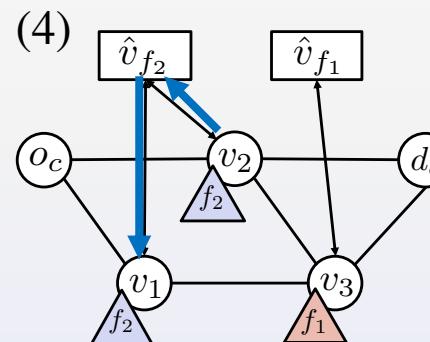
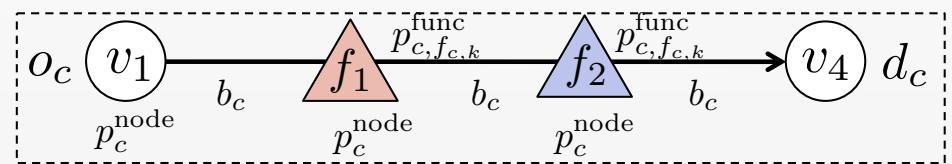
$v \in \mathcal{V}$

SPTPに
関連する
制約

資源制約

$$d_{i,j} = \begin{cases} d_i^{\text{node}} + d_{i,j}^{\text{link}}, & \text{if } (i, j) \in \mathcal{E} \\ d_{i,j}^{\text{func}}, & \text{if } (i, j) \in \hat{\mathcal{E}}^{\text{out}} \\ 0, & \text{Otherwise} \end{cases}$$

$$r_c = (o_c, d_c, \mathcal{R}_c, b_c, p_c^{\text{node}}, \{p_{c,f_{c,k}}^{\text{func}}\}_{k=1,\dots,K_c})$$



課題2

- CSPTPのILPを定式化するプログラムを作成してください。
 - CSPTP-SC/csptp_based_ilp.pyを編集してください。
 - LPファイルを提出してください。
- 提供データの最適解と最適値を導出してください。
 - CSPTP-SC/dataのデータを利用した結果を示してください。
- ノード数の増加により、 ILPの実行時間がどのように変化するのかを評価してください。
 - レポートでは評価環境や評価結果の情報を示してください。
 - dataset/random_network_V[# of nodes]のデータを利用した結果を示してください。

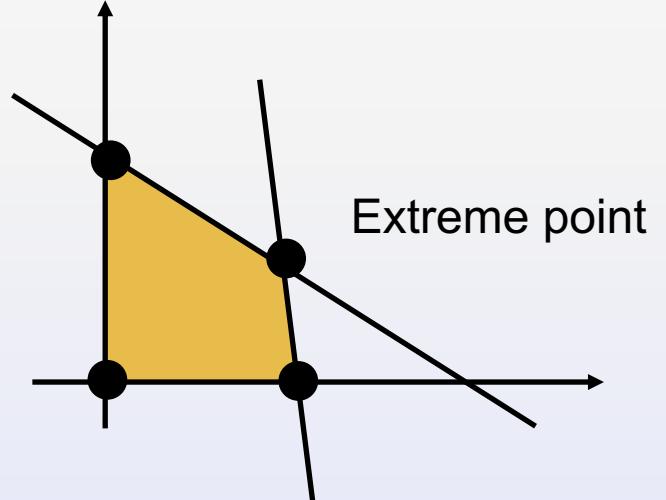
発展課題

- 複数のサービスチェイン要求に対応可能なCSPTPに基づくSFCのILPを定式化するプログラムを作成してください
 - CSPTP-SC/csptp_based_ilp_for_batch_processing.pyを編集してください.
 - LPファイルを提出してください.
- 提供データに対する最適解と最適値を導出してください.
 - CSPTP-SC/dataのデータを利用した結果を示してください.
- サービスチェイン要求数の増加に対して、 ILPの実行時間の推移を示してください.
 - dataset/random_network_C[# of requests]のデータを利用した結果を示してください.

Totally unimodular matrix (TUM, 完全単模行列) [Wolsey+99]

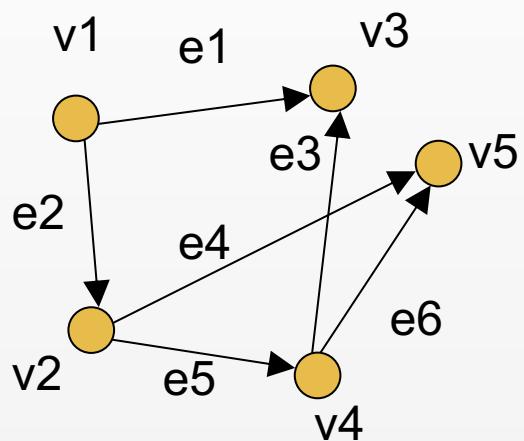
- 任意の部分正方行列の行列式が0, 1, -1のいずれかになる行列
- TUMの特徴

- 定理: $Ax \leq b$ で表現される多面体をPとする。
AがTUMでbの成分がすべて整数の場合,
Pのすべてのextreme pointsはすべて整数である.
- 行列AがTUMである
 - 整数線形計画問題を線形緩和した問題の解は元の問題の解と等しくなる
- 定理の逆も成立する
 - すべての整数ベクトルbについて,
Pのすべてのextreme pointsが整数である場合,
AはTUMである.



III 有向グラフと頂点・辺接続行列 [Wolsey+99]

- 次に示す有向グラフと頂点・辺接続行列を考える



	e1	e2	e3	e4	e5	e6
v1	1	1	0	0	0	0
v2	0	-1	0	1	1	0
v3	-1	0	-1	0	0	0
v4	0	0	1	0	-1	1
v5	0	0	0	-1	0	-1

辺の始点に1, 終点に-1
列には1と-1が一つずつ並ぶ

- 頂点・辺接続行列はTotally unimodular (TU) である
- TUMの条件
 - 任意の部分正方行列の行列式が0, 1, -1のいずれかになる行列

Totally unimodularの証明 [Wolsey+99]

- 任意の部分正則行列Nは次の3つの内1つの可能性を持つ
 - ある列(行)のすべてが0なら $\det(N)=0$
 - 各列(行)に2つの非零な要素がある場合、一つは1、もう一つは-1
 - Nのある列がi番目で非零の要素を持つ

	e1	e2	e3	e4	e5	e6
v1	1	1	0	0	0	0
v2	0	-1	0	1	1	0
v3	-1	0	-1	0	0	0
v4	0	0	1	0	-1	1
v5	0	0	0	-1	0	-1

Totally unimodularの証明 [Wolsey+99]

- 任意の部分正則行列Nは次の3つの内1つの可能性を持つ
 - ある列(行)のすべてが0なら $\det(N)=0$
 - 各列(行)に2つの非零な要素がある場合、一つは1、もう一つは-1
 - すべての行の和が零ベクトルになる→線形独立
 - Nのある列がi番目行で非零の要素を持つ

	e1	e2	e3	e4	e5	e6
v1	1	1	0	0	0	0
v2	0	-1	0	1	1	0
v3	-1	0	-1	0	0	0
v4	0	0	1	0	-1	1
v5	0	0	0	-1	0	-1

Totally unimodularの証明 [Wolsey+99]

- 任意の部分正則行列Nは次の3つの内1つの可能性を持つ
 - ある列(行)のすべてが0なら $\det(N)=0$
 - 各列(行)に2つの非零な要素がある場合、一つは1、もう一つは-1
 - すべての行の和が零ベクトルになる→線形独立
 - Nのある列がi番目で非零の要素を持つ
 - 帰納法より導出($n=1$ のときは明らか)

	e1	e2	e3	e4	e5	e6
v1	1	1	0	0	0	0
v2	0	-1	0	1	1	0
v3	-1	0	-1	0	0	0
v4	0	0	1	0	-1	1
v5	0	0	0	-1	0	-1

帰納法

Nの部分正則行列Mを考える

ここでMはNからi番目の行、m番目の列を取り除いたもの

行列式の展開より $\det(N)=\pm 1 \det(M)$

$\det(M)$ は0,-1,1であるため、 $\det(N)$ も0,-1,1となる

課題3

- 最小費用流問題の制約行列はTUMである。
最小費用流問題のLPとILPの解が等しくなることを確認してください。
 - ただしベクトルbの各成分はすべて整数であること
 - https://github.com/oakeshott/21pbl-a18-2/blob/master/examples/mcfp/minimum_cost_flow_problem.py

まとめ

- 最適化問題の定式化・解法について理解を深めた
 - 分枝限定法
 - 線形緩和
- NFVネットワークにおけるサービスチェイニング問題を定式化した
- 完全単模行列と解の整数性の関係について理解を深めた

III CPLEXのダウンロード

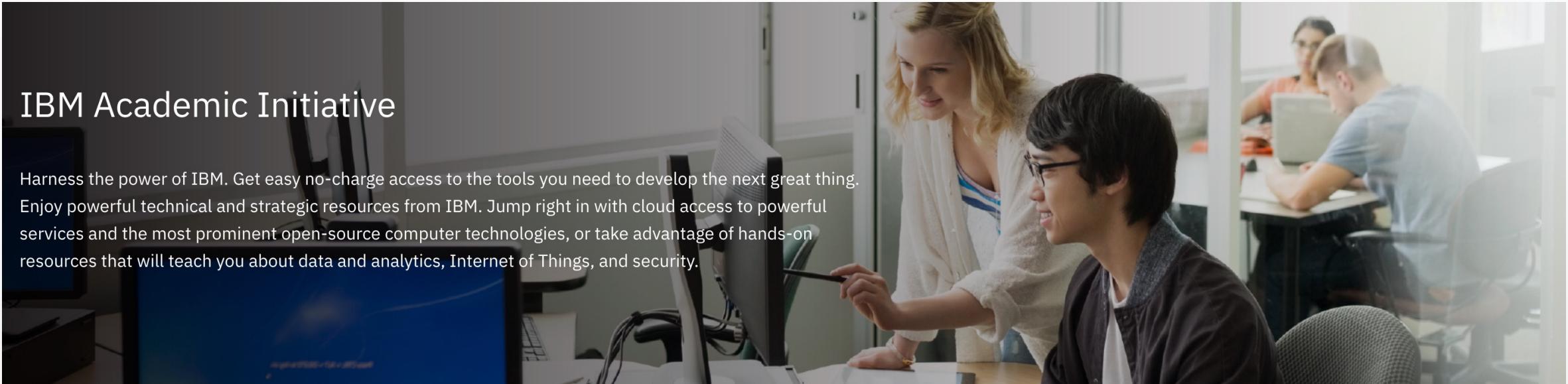
- <https://www.ibm.com/jp-jp/products/ilog-cplex-optimization-studio>



クリック

CPLEXのダウンロード

- 所定の手続きに従いアカウントを作成



IBM Academic Initiative

Harness the power of IBM. Get easy no-charge access to the tools you need to develop the next great thing. Enjoy powerful technical and strategic resources from IBM. Jump right in with cloud access to powerful services and the most prominent open-source computer technologies, or take advantage of hands-on resources that will teach you about data and analytics, Internet of Things, and security.

Use your skills to take on climate change and join a community of developers and innovators using open source technology in the [2021 #CallforCode challenge](#).

Get access today to the same tools used in the enterprise.

[Register now →](#)

Already registered? [Log in →](#)

A red circle highlights the "Register now →" button, and a red arrow points to it from the Japanese text "クリック" (Click) at the bottom right.

クリック

CPLEXのダウンロード

- <https://content-eu-7.content-cms.com/b73a5759-c6a6-4033-ab6bd9d4f9a6d65b/dxsites/151914d1-03d2-48fe-97d9-d21166848e65/academic/topic/data-science>

The screenshot shows a web page from the IBM Academic Initiative. The top navigation bar includes links for IBM, IBM Academic Initiative, Topics, Usage terms, Additional Resources, FAQ, a search bar, and a user icon. On the left, a sidebar lists Courseware, Software (which is circled in red), and Resources. The main content area displays three software options:

- Watson Studio Desktop**: Integrated SPSS Modeler functionality to drag-and-drop your way to ML, model building, and data exploration. Data science students can use Watson Studio Desktop to exercise their R and →
- ILOG CPLEX Optimization Studio**: Analytical decision support toolkit for rapid development and deployment of optimization models using mathematical and constraint programming. It combines an integrated → (this link is also circled in red)
- SPSS Modeler Premium**: SPSS Modeler provides an intuitive graphical interface to help visualize each step in the data mining process. Automatically transforms data into the best format for the most accurate predictive modeling. →

Below these three options is a "View All" button on a dark background.

CPLEXのインストール

- CPLEXをダウンロードした後にインストールする
- インストール後、CPLEXのパスを設定

```
export CPLEX_PATH=/Applications/CPLEX_Studio201  
export PATH=$PATH:$CPLEX_PATH/cplex/bin/x86-64_osx  
export PYTHONPATH=$PYTHONPATH:${CPLEX_PATH}/cplex/python/3.8/x86-64_osx/
```

- パスが設定されたことを確認

```
$ cplex
```

```
Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 20.1.0.0  
with Simplex, Mixed Integer & Barrier Optimizers  
5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21  
Copyright IBM Corp. 1988, 2020. All Rights Reserved.
```

```
Type 'help' for a list of available commands.  
Type 'help' followed by a command name for more  
information on commands.
```

```
CPLEX>
```

III CPLEXがインストールできない場合

- MacのAppearanceをDarkモードからLightモードに変更