Study Group Workshop 2026　成果報告集

課題提供：村田機械株式会社

SGW2026

自動物流倉庫における搬送性能の限界予測

編集者）
重富 尚太（九州大学マス・フォア・インダストリ研究所）
武内 太貴（九州大学マス・フォア・インダストリ研究所）
髙倉 科瑞紀（九州大学大学院マス・フォア・イノベーション連係学府）
一木 俊佑（九州大学大学院数理学府）

この成果報告集について

- 成果報告集は複数の成果報告書をまとめたものです
  - ✓ 成果報告書の下線付きの著者は、成果報告会（SGW5日目）で登壇したことを表します
  - ✓ 登壇の有無に関わらず、成果報告書を成果報告集に採用することがあります

- 著作権および利用条件について
  - ✓ 含まれる成果物の著作権は、原則として各成果物の創出者に帰属します
  - ✓ 閲覧および閲覧・参照を目的としたダウンロードは自由です
  - ✓ これらを超える転載、複製、改変、再配布、公衆送信等の自由利用の直接の対象は、原則としてSGW構成員です
    - 第三者による利用は、著作権法上認められる場合、または権利者・SGW構成員等の許諾がある場合に限られます
    - 製品開発等の商業的利用は、自由利用の対象外です
  - ✓ 記載されたSGW2026由来のアイデアやノウハウそのものは、第三者も無償で自由に利用できます
  - ✓ 詳細は以下のSGW規約をご確認ください
    - 国立大学法人九州大学マス・フォア・インダストリ研究所 スタディグループ・ワークショップ規約

© 2026 Respective authors / creators.

Contents

- 衝突を無視したAGV経路計画のLP緩和による下界評価
  - ✓ 一木 俊佑（九州大学大学院数理学府）
- 衝突を考慮したAGV経路計画へのアプローチ
  - ✓ 川根 大輝（九州大学大学院マス・フォア・イノベーション連係学府）
  - ✓ 細谷 泰稚（九州大学大学院マス・フォア・イノベーション連係学府）
  - ✓ 政田 智秋（九州大学大学院マス・フォア・イノベーション連係学府）
- 衝突を無視できるようなAGV経路計画手法
  - ✓ 許 智祐（九州大学大学院数理学府）
- 完全グラフを用いた作業順序の決定
  - ✓ 髙倉 科瑞紀（九州大学大学院マス・フォア・イノベーション連係学府）

課題提供者、モデレーター等

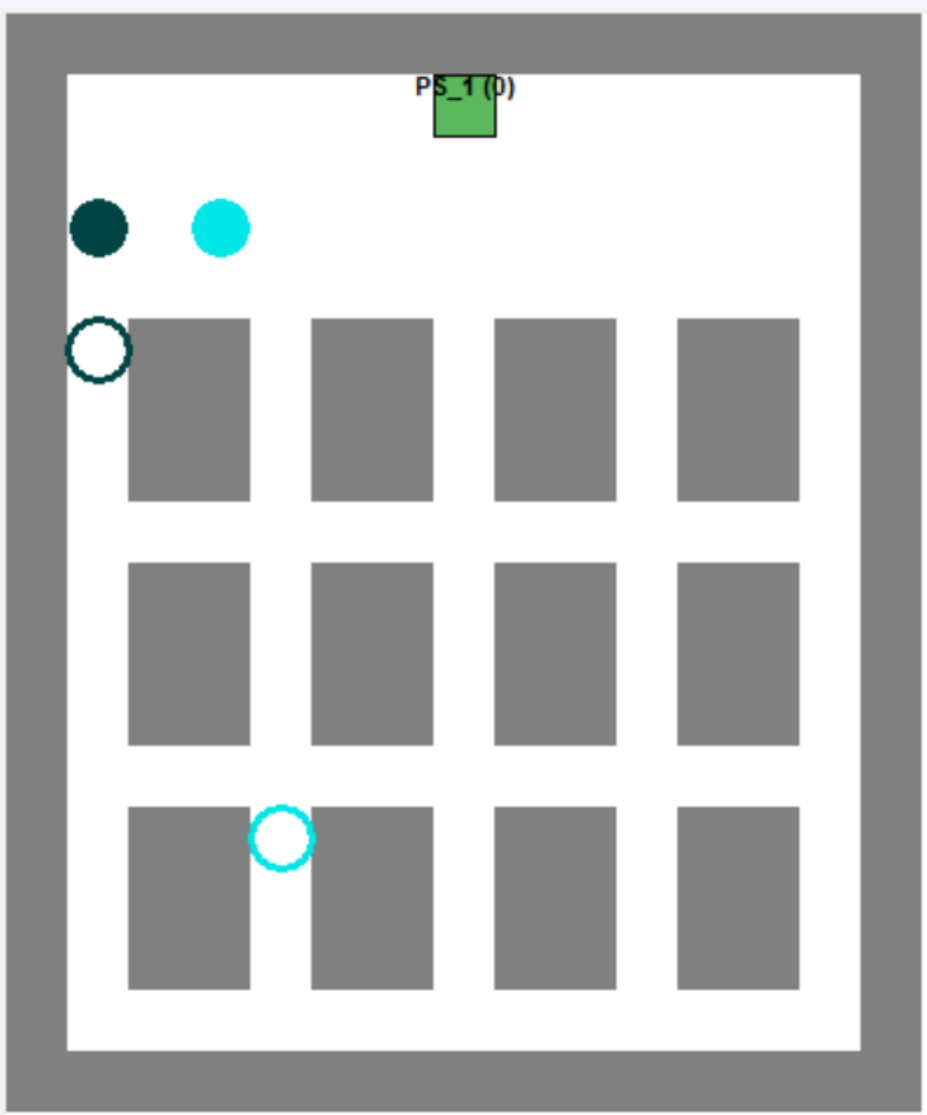
- 課題提供者・サポーター
  - ✓ 木村 藍貴 様（村田機械株式会社）
  - ✓ 藤根 大己 様（村田機械株式会社）
- コーディネーター
  - ✓ 脇 隼人（九州大学マス・フォア・インダストリ研究所）
  - ✓ 穴井 宏和（九州大学マス・フォア・インダストリ研究所）
  - ✓ 吉良 知文（九州大学マス・フォア・インダストリ研究所）
- モデレーター
  - ✓ 重富 尚太（九州大学マス・フォア・インダストリ研究所）
  - ✓ 武内 太貴（九州大学マス・フォア・インダストリ研究所）
- 学生モデレーター
  - ✓ 髙倉 科瑞紀（九州大学大学院マス・フォア・イノベーション連係学府）
  - ✓ 一木 俊佑（九州大学大学院数理学府）

SGW2026

衝突を無視した AGV経路計画のLP緩和による下界評価

著者）
一木 俊佑（九州大学大学院数理学府）

問題設定ー複数のAGVで全てのオーダーを処理する



提供の可視化フローを使用

- 1つのオーダーの処理
  - ✓ AGV初期位置
  - ---
  - ✓ 商品位置で荷つかみ(5秒)
  - ✓ PSへ搬送・処理(5秒)
  - ✓ 商品位置へ戻り荷おろし(5秒)
  - ✓ 次の仕事へ

  繰り返し

- 目的
  - ✓ 全てのオーダーが終了する時刻を最小化

- 課題
  - ✓ 理論的に達成可能な搬送性能の限界を予測すること

アイデア

AGV経路計画

↓ 衝突・干渉を無視

配送計画問題

↓ 定式化可能

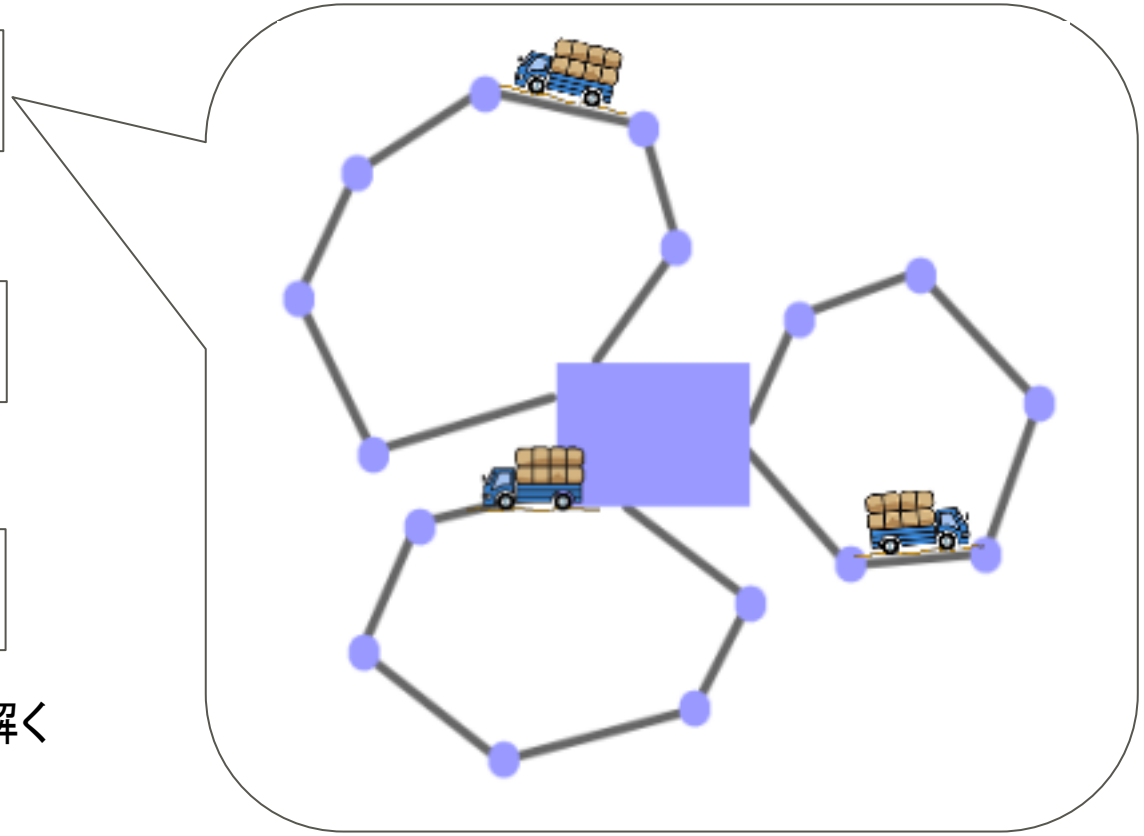
0-1 整数線形計画問題

↓ LP緩和

線形計画問題

↓ LPソルバーで解く

解きたい問題の下界



数値実験でLP下界の精度を評価する

実験条件

- ✓ 倉庫レイアウト：8条件
- ✓ アイテム数：5または10
- ✓ AGV数：1～5、PS数：1～3
- ✓ オーダー構成を複数パターンで比較
  （すべて異なる注文、すべて同じ注文など）

評価指標

- ✓ 企業から渡された最適値が出ているケースについて計算し、緩和ギャップを計算。

平均緩和ギャップは19.52%、ケースによって大きく変化

- ケース数61（アイテム数5個が9ケース、ほか10個のケース）
- 平均緩和ギャップ19.52%
- 最大34.95%
- AGVが2個以上で最小の誤差は
  - ✓ アイテム数5個だとAGV2個PS1個で3.99%
  - ✓ アイテム数10個だとAGV4個PS2個で8.93%

LPが理想的になるケースほど下界が弱くなる

- アイテム数5個での平均緩和ギャップは9.84%、10個だと21.19%
- 注文構成がすべて異なると7.54%、重複ありだと21.07%

- LPによる過度に理想的な負荷の平準化（分数割り当て、分数経路）
- 重複注文時、同じ位置、経過時間を均等化
- 多数AGVだと細かく負荷を分けられる

- 最大誤差例：10仕事・5AGV・2PS、注文が2種類に集中
- 誤差35%弱
- 実際には仕事は整数個、PSや通路では衝突や競合する。
- 現在はその待ち時間を十分に表せていない。

まとめ

- できたこと
  - ✓ LP緩和による基本的な下界を構築
    - 平均緩和ギャップは19.52%
    - 重複注文、多数AGVで誤差が大きい傾向
    - 分数割り当て、分数経路、衝突や混雑を無視

- 今後取り組みたいこと
  - ✓ 計算量を抑えて計算できる、より強い下界を得ること
  - ✓ 現実の制約を少し反映させたもとでの下界
    （経路や割り当て、通路や混雑領域の制約）

衝突を考慮したAGV経路計画へのアプローチ

著者）
川根 大輝（九州大学大学院マス・フォア・イノベーション連係学府）
細谷 泰稚（九州大学大学院マス・フォア・イノベーション連係学府）
政田 智秋（九州大学大学院マス・フォア・イノベーション連係学府）

目次

- アプローチ1（川根） 幅優先探索
- アプローチ2（細谷） 整数計画問題としての定式化
- アプローチ3（政田） マルチエージェント強化学習
- 課題点

アプローチ1（川根）幅優先探索

- AGV数・注文数・盤面サイズが小さい場合
  - ✓ すべての状態を管理して、幅優先探索によって最適解を求める
  - ✓ 持つべき状態は以下の通り:

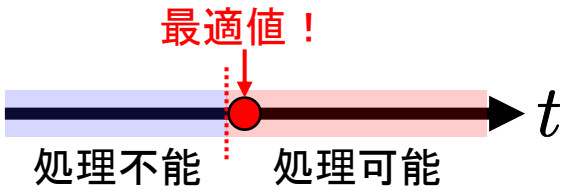
H: 盤面の高さ, W: 盤面の幅

状態	状態数
各AGVの位置	$(H \times W)^{(\#\text{ of AGVs})}$
注文の処理状態	$2^{(\#\text{ of orders})}$
各AGVが請けている注文	$((\#\text{ of orders})+1)^{(\#\text{ of AGVs})}$

    - すべての状態数は、これらの積となる (計算回数は、少なくともこれぐらい必要)
  - ✓ 各状態において、それぞれのAGVは (上下左右＋何もしない or 特別なマスでの作業) の動作を行える
  - ✓ なお、PSや商品棚での作業は通過に5ステップかかると解釈し、頂点を適切に追加することで対処
- 計算結果
  - ✓ H=15,W=18 AGV 2台,注文数 5個 のケースで1分ほどで最適解が得られた
  - ✓ H=27,W=15 AGV 2台,注文数 5個のケースで4分ほどで最適解が得られた
  - ✓ 解が(現実的な時間で)求まるのは 注文数 5個以下、AGV 2台以下

アプローチ2（細谷）整数計画問題としての定式化

- 最適解の構成を目標に据え，整数計画問題として定式化
  - ✓ 時空間グラフ上で全 AGV の経路を設計する問題として捉える
    - 衝突の制約が扱いやすくなる
    - 部分巡回路（独立した巡回路）を除去する制約が不要
  - ✓ 最適値は未知だが，「指定した時刻内で全注文を処理できるか？」という判定問題を考えれば，扱う時空間グラフのサイズを有限に収められる
    - この判定問題が真であるような時刻の下限が最適値（makespan の最小値）
  - ✓ この判定問題を0-1整数計画問題として定式化し，MIP ソルバー（Gurobi）に解かせる
    - 変数の数（≒時空間グラフの辺数）は爆発的に増えない



- 計算結果
  - ✓ H=15, W=18, AGV 2台，注文数 2個の場合: 最適値が得られることを確認
  - ✓ 現実的な時間で計算できるのは（確認済みの範囲で）Hは18以下，Wは18以下，注文数 2個以下
    - 変数の数の多さと0-1整数計画問題であるという二つの問題点によるものだと推察
    - LP 緩和（バイナリ制約を緩和）によって最適値の下限評価を与えることが可能
  - ✓ H=15, W=18, AGV 2台，注文数 5個の場合: LP緩和の最適値は71（緩和前の最適値は118）

アプローチ3（政田）マルチエージェント強化学習

- マルチエージェント強化学習による「もっともらしい値」の探索
  - ✓ Deep Q Network（DQN）を採用 （1エピソードの上限 300ステップ，全2000エピソード）
  - ✓ 生成AI（Google Gemini 3.1 Pro，Claude Sonnet 5）でコーディング
- 報酬・ペナルティの設定

タスク	作業待機中	荷つかみ完了	PSで作業完了	空コン返却
報酬	+0.5	+2.0	+2.0	+10.0

目的地との距離	増加	減少
報酬	-0.2	+0.2

その他	衝突	時間経過（タスク完了AGVは除外）
報酬	-1.0	-0.1

- 実行結果（H=15，W=18，AGV 2台）
  - ✓ 注文数 2，3個 ⇨最適値が得られた
  - ✓ 注文数 4個 ⇨失敗
  - ✓ 学習したモデルで推論を行うとなぜかうまくいかない…
    - 特定の2か所の行き来をひたすら繰り返す台車が出てきてしまう（移動に関するペナルティの影響？）

課題点

- アプローチ1
  - ✓アプローチ3と同様に強化学習による探索に持ち込めば，大きなケースにも対応できる可能性あり
- アプローチ2
  - ✓定式化した問題の最適値とその LP 緩和の最適値の間に大きなギャップがある
  - ✓変数の数が多く，求解に多大な時間を要する
    - グラフを時間方向に拡大しているため（実はさらにその時空間グラフを注文の数だけ複製している）
    - 何らかの緩和方法で対処できないか？（LP 緩和を行うと所要時間は大きく短縮）
  - ✓整数計画問題のモデリングにも比較的長い時間を要する
    - 使用言語の特性，多重の繰り返し処理，複雑な分岐処理など
- アプローチ3
  - ✓AGV・注文数・盤面サイズが小さいケースですら困難
    - 報酬やペナルティの与え方，メソッドに工夫が必要？
  - ✓実行時間が長い（AGV 2台・注文数 3個で 1時間程度）
    - 実用に堪える計算スピードではないかも…？

SGW2026

衝突を無視できるような AGV経路計画手法

著者）
許 智祐（九州大学大学院数理学府）

The contributions of this work.

- We summarize the difficulties in Step 2 of the AGV problem and propose a framework for extending the algorithm developed for Step 1 to Step 2.

From the Original AGV Problem to Graph Language

Original problem in words.

- Several AGVs start from fixed warehouse positions. Each order requires one AGV to:

  go to the container's position -> carry it to a picking station -> return it

- The goal is to assign and sequence the orders among AGVs so that the last finishing AGV finishes as early as possible.

The collision-free (step 1)

- We separate our discussion into two step. The step 1 contains the following assumptions:
  - No collision between AGVs.
  - AGVs don't spend time at container's position or picking station.
- ✓ We will relex the first assumption until the step 2.

From the Original AGV Problem to Graph Language

- **Grid-point formulation.**
  - Represent the warehouse floor as traversable grid points

$$\Omega \subseteq \mathbb{Z}^2$$

  - Two grid points are adjacent if they differ by one horizontal or vertical step:

$$F = \left\{ \{u, v\} \subseteq \Omega \mid \|u - v\|_1 = 1 \right\}$$

  - Thus, the warehouse becomes a grid (undirected) graph:

$$H = (\Omega, F)$$

From the Original AGV Problem to Graph Language

Shortest-distance idea.

- Instead of describing the full movement path cell by cell, define

$$d_H(u, v) = \text{length of a shortest path from } u \text{ to } v \text{ in } H$$

- Following our assumption, we can:
  - Assume that during each AGV movement, the AGVs only consider how to reach the next position and do not spend any time at intermediate grid points.
  - Hence, we can reformulate the movements of AGVs as travel on a graph.

Reduced graph language.

- Keep only containers' positions, PS positions, and AGV start positions:

$$V_R = \{x_i\}_{i \in \mathcal{I}} \cup \{p_j\}_{j \in \mathcal{P}} \cup \{s_a\}_{a \in \mathcal{A}}$$

- The reduced weighted (undirected) graph is

$$G = (V_R, E_R, d), \qquad d(u, v) = d_H(u, v)$$

Final Mathematical Formulation on the Reduced Graph

- Here we summary what should be the input and out of the ideal algorithms
- **Input of ideal algorithms**
  - The positions of containers, PSs and AGVs: $G = (V_R, E_R, d),$
  - The orders and the indices of their corresponding containers:

$$\mathcal{O} = \{1, \ldots, K\}, \quad \iota : \mathcal{O} \to \mathcal{I}, \quad q_k = x_{\iota(k)}$$

- **Output.**
  - For each AGV *a*, an ordered sequence of assigned orders:

$$R_a = (k_{a,1}, \ldots, k_{a,m_a}) \in \mathcal{O}^{m_a}$$

  - The sequences R_a must partition $\mathcal{O}$, so every order is assigned and completed exactly once.
  - The output further contains each AGV's completion time C_{R_a} and the final makespan:

$$T = \max_{a \in \mathcal{A}} C_{R_a}$$

From Collisions-free (Step 1) to Collisions (Step 2)

- Our ideal algorithm must solve the following optimization problem:

$$\arg\min_{\{R_a\}_{a\in\mathcal{A}}} T = \arg\min_{\{R_a\}_{a\in\mathcal{A}}} \max_{a\in\mathcal{A}} C_{R_a} \tag{1}$$

- If $|\mathcal{A}|$ >1, then each C_{R_a} still depends on the assigned order R_a of other AGVs.

➢ One important observation is that, because we do not consider collisions in step 1,

we can freely search over all possible orders R_a to optimize our goal function $\max_{a\in\mathcal{A}} C_{R_a}$.

➢ Conversely, if we consider collisions, we encounter the following difficulties:

- It is difficult to characterize the candidate solutions of the optimization problem (1) , because we add constraints on (1) . It is also hard to find bounds of an optimization problem with constraints.
- For example, we need to check orders R_a of different AGVs to ensure no collision in any timesteps;
  the collision-free assumption we used before let us just need to focus on the static graph structures.

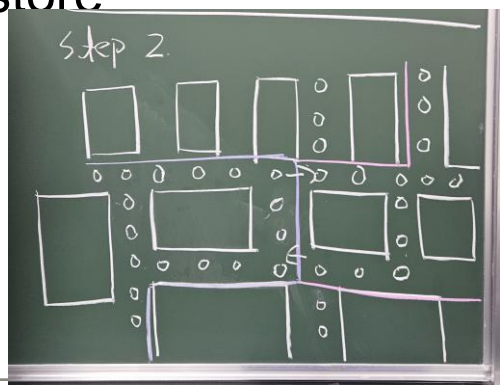
From Collisions-free (Step 1) to Collisions (Step 2)

- Because of the benefit of collisions-free condition, we want to generalize what did we do in Step 1 into Step 2.
  Let us use the idea of roundabouts, or circles in graph theory.

Definition

A circle C_k in a undirectied graph H is a sequence $v_1, v_2, \cdots, v_{b_k}, v_1$, where every consecutive pair is connected by an edge.

- For the grid graph H of a warehouse, a circle C_k provide a free space that can store at most b_k -AGVs, so let they can stay in this circle without collision, and each still can go to their next position freely in some sense (by spend time on the rotation).



From Collisions-free (Step 1) to Collisions (Step 2)

- Our idea is:
  - Partition the original grid graph *H* to a set of subgrids

    $$\{G_i \subseteq H\}$$

    of *H*, where each G_i contains at least circle C_{k_i} of H.
  - Calculate the storage of each G_i , and the average moving time $\tau((G_i, G_j))$ of AGVs fram a subgrid G_i to another subgrid G_j.
  - Combine G_i with the information we calculated above to make a **weighted directed graph**

    $$\tilde{H} = (\tilde{V}, \tilde{E}, \tau)$$

    where $\tilde{V} = \{G_i\}$ and the directed edges $\tilde{V}$ depends on how we cut the subgrids from *H*.
- The reason of considering directed graph here is because:
  - We not only need to ensure that no collisions in one subgrid G_i , but also need to ensure that no collisions when AGVs move between subgrids.
  - We use one-way street to limit the moving directions of AGVs between subgrids.

Mathematical Formulation of the Cutting-Grid Algorithm

- **Input of the ideal algorithm.** An undirected graph $H = (\Omega, F)$.

- **Output.**
  - First, an index set $\mathcal{S}$ and a partition

$$\Omega = \dot{\bigcup_{i \in \mathcal{S}}} \Omega_i, \qquad G_i = H[\Omega_i],$$

where every connected subgraph G_i of vetices Ω_i contains a roundabout cycle

$$C_i = (c_{i,1}, \ldots, c_{i,b_i}, c_{i,1}) \subseteq G_i.$$

  - Second, the weighted directed multigraph

$$\tilde{H} = (\tilde{V}, \tilde{E}, \tau), \qquad \tilde{V} = \{G_i : i \in \mathcal{S}\}, \qquad \tau : \tilde{E} \to \mathbb{Z}_{\geq 1}.$$

where each edge $(G_i, G_j) \in \tilde{E}$ of $\tilde{H}$ corresponds to an edge $\{v_i, v_j\} \in F$ of H such that $v_i \in G_i, v_j \in G_j$.

Final Mathematical Formulation of Order Distribution on Subgrids

- Note that the cutting-grid algorithm abstracts the warehouse graph

$$H = (\Omega, F)$$

into the weighted directed multigraph

$$\tilde{H} = (\tilde{V}, \tilde{E}, \tau) \ .$$

- We then optimize the AGVs' completion time on $\tilde{H}$, similarly to our first algorithm on *H*.
Since the AGV and container positions are vertices of *H*, they must first be mapped to their corresponding subgrids in $\tilde{H}$.

Final Mathematical Formulation of Order Distribution on Subgrids

- **Input of the ideal algorithm.**
  - An undirected graph $H = (\Omega, F)$.
  - The weighted directed multigraph $\tilde{H} = (\tilde{V}, \tilde{E}, \tau)$ obtained from H .
  - The directed shortest-time distance on $\tilde{H}$:

$$d(G_i, G_j) = \min_{P: G_i \rightsquigarrow G_j} \sum_{e \in P} \tau(e)$$

  - The subgrid-membership map

$$\rho : V_R \longrightarrow \widetilde{V}, \qquad \rho(v) = G_i$$

    means that the physical position $v \in H$ belongs to $G_i \in \tilde{H}$.
  - The orders and corresponding container indices:

$$\mathcal{O} = \{1, \ldots, K\}, \qquad \iota : \mathcal{O} \to \mathcal{I}, \qquad q_k = x_{\iota(k)}.$$

Final Mathematical Formulation of Order Distribution on Subgrids

- **Output.**
  - For each AGV *a*, an ordered sequence of assigned orders:

$$R_a = (k_{a,1}, \ldots, k_{a,m_a}) \in \mathcal{O}^{m_a}$$

The sequences R_a must partition $\mathcal{O}$, so every order is assigned and completed exactly once:

$$\mathcal{O} = \dot{\bigcup_{a \in \mathcal{A}}} \{k_{a,1}, \ldots, k_{a,m_a}\}$$

  - The output further contains each AGV's completion time C_{R_a} and the final makespan between subgrids:

$$T = \max_{a \in \mathcal{A}} C_{R_a}$$

Postive side effects of our idea

- **Remark.**
  The makespan we calculated depends on the bound of time by our algorithm depends on the assigned orders of AGVs and the time that AGVs spent on traveling between subgrids.

- There are two postive side effects of our idea:
  - For the case that containers and AGVs are sparse in a large warehouse, comparing to the traditional method, the subgrids can also reduce the calculation on solving our optimization problem.
  - After we use the algorithm to decide the moving paths of AGVs between subgrids, we can use GPU, because in this phase we can parallel calculate different subgrids' optimization problem, which is also a bound of the total spend time.

完全グラフを用いた作業順序の決定

著者）
髙倉 科瑞紀（九州大学大学院マス・フォア・イノベーション連係学府）

※SGW 5日目は学会参加のため、コーディネーターが代理で登壇

完全グラフを用いた作業順序の決定（STEP1）

- 渋滞・衝突を無視したモデル
  - ✓ 作業時間の最大値が最小になるようにオーダーをふりわける
  - ✓ 各オーダーは必ず行われる必要がある
- 完全グラフを(台数)本のパスによって被覆する問題ととらえることができる
  - ✓ 頂点：各オーダー
  - ✓ オーダーA/B間の辺重み：それぞれの作業・移動時間の半分の和
  - ✓ 外部に各オーダーへの辺をもつ頂点s/tを置き、(台数)本のs-tパスで完全グラフを被覆したとき、作業の割当例が得られる
  - ✓ パス重みの最大値が最小となる割当を見つけることで、渋滞・衝突を無視した場合のよい作業順序が得られる
- 今回の例ではモデルが大きく厳密解を求めるのが厳しいためヒューリスティックを用いた
  - ✓ プログラムはSlackの課題チャンネルにて配布済み

作業順序からの経路計画の構成アイデア（STEP2）

- 各AGVに優先度をつけ、一台ごとに経路計画をつくる
  - ✓ 一台の計画を一度に作りきるのではなく、何らかの単位で作ってから改めて優先度を計算する
- 優先度は（残り作業時間）-f（猶予時間）とする
  - ✓ 猶予時間は計算時における(全作業終了予定時刻)-(その台車の作業終了予定時刻)とする
  - ✓ 関数fについては定数0や単調増加関数などでよいものを実験的に決定する
- 各AGVの通路における停止作業時刻には、他のAGVは通路を通らないようにする
  - ✓ 干渉を避けるため
  - ✓ 時刻は優先度を加味して延長することも考えられる
- 利用するPSは待ち時間に応じて柔軟に変化することを考える