

## 発表要旨

### これまでの研究業績

Fully Automated Cyclic Planning for Large-Scale Manufacturing Domains ≈ 1. **ICAPS14.**

Solving Large-Scale Planning Problems by Decomposition and Macro Generation ≈ 1. **ICAPS15.**

Tiebreaking Strategies for A\* Search: How to Explore the Final Frontier ≈ 1. **AAAI16.** (*JSAI*  
学生奨励賞)

Tie-Breaking Strategies for Cost-Optimal Best First Search ≈ 1. **JAIR 58 (2017): 67-121.**

Exploration Among and Within Plateaus in Greedy Best-First Search ≈ 1. **ICAPS17.**

Efficient Optimal Search under Expensive Edge Cost Computation ≈ 2. **IJCAI17.**

Classical Planning in Deep Latent Space: From Unlabeled Images to PDDL (and back) ≈ 1.  
**KEPS17.**

### 今後の研究計画、研究成果の産業応用への抱負

≈ 1 Masataro Asai, Alex Fukunaga

≈ 2 Masataro Asai, Akihiro Kishimoto, Adi Botea, Radu Marinescu, Elizabeth Daly M, and Spyros Kotoulas

# 1 背景 - AI プランニング



## 1.1 誰？



### 1.1.1 誰?

鉄腕アトム



### 1.1.2 誰?

鉄腕アトム



見て、聞いて、喋って、行動する  
自律行動 *Autonomous*



### 1.1.3 誰?

鉄腕アトム



見て、聞いて、喋って、行動する  
自律行動 *Autonomous*

T-52 援竜 @北九州消防局  
レスキュー・ロボット



#### 1.1.4 誰?

鉄腕アトム



見て、聞いて、喋って、行動する  
自律行動 *Autonomous*

T-52 援竜 @北九州消防局  
レスキュー・ロボット



自律行動 *Autonomous*  
?

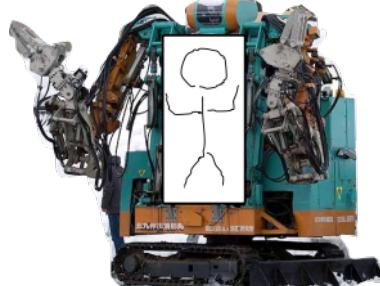
### 1.1.5 誰?

鉄腕アトム



見て、聞いて、喋って、行動する  
自律行動 *Autonomous*

T-52 援竜 @北九州消防局  
レスキュー・ロボット



自律行動 ~~X~~utonomous  
人間による操作  
でかいラジコンショベルカー

## 1.2 自律行動のための自動プランナ(≠ モータ制御)

プランナ=抽象行動列を求める人工知能プログラム



初期状態: 人が生き埋め!  
ゴール: 患者の搬送



出力: 被災者を助ける  
アクション列 = プラン  
出血をとめる  
→病院に電話  
→患者を助け出す  
→救急隊に引き渡し

### 1.3 自律行動のための自動プランナ(≠ モータ制御)

プランナ=抽象行動列を求める人工知能プログラム



初期状態: 人が生き埋め!

ゴール: 患者の搬送



出力: 被災者を助ける  
アクション列 = プラン

出血をとめる  
→病院に電話  
→患者を助け出す  
→救急隊に引き渡し

汎用性  
ゆえに  
様々な  
応用例

#### 一階述語論理で入力

##### 初期状態

出血(浅井)

on(木材, 浅井), on(岩, 木材)

##### アクション

電話(病院)

救出(浅井)

##### ゴール

at(浅井, 病院)

引き渡し(浅井, 救急車)

## 1.4 自律行動のための自動プランナ(≠ モータ制御)

プランナ=抽象行動列を求める人工知能プログラム

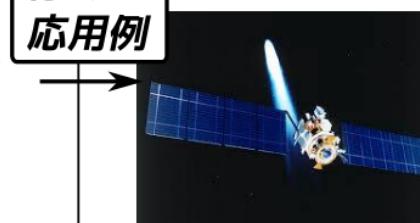


初期状態: 人が生き埋め!  
ゴール: 患者の搬送



出力: 被災者を助ける  
アクション列 = プラン  
出血をとめる  
→病院に電話  
→患者を助け出す  
→救急隊に引き渡し

汎用性  
ゆえに  
様々な  
応用例



Mars Exploration Rover  
(Bresina et al. '05)



プランニングの  
応用実績

Deep Space 1  
(Muscettola  
et al. '98)



ネットワークの自動ハッキング  
→脆弱性を報告

### 一階述語論理で入力

初期状態

出血(浅井)  
on(木材, 浅井), on(岩, 木材)

ゴール

at(浅井, 病院)

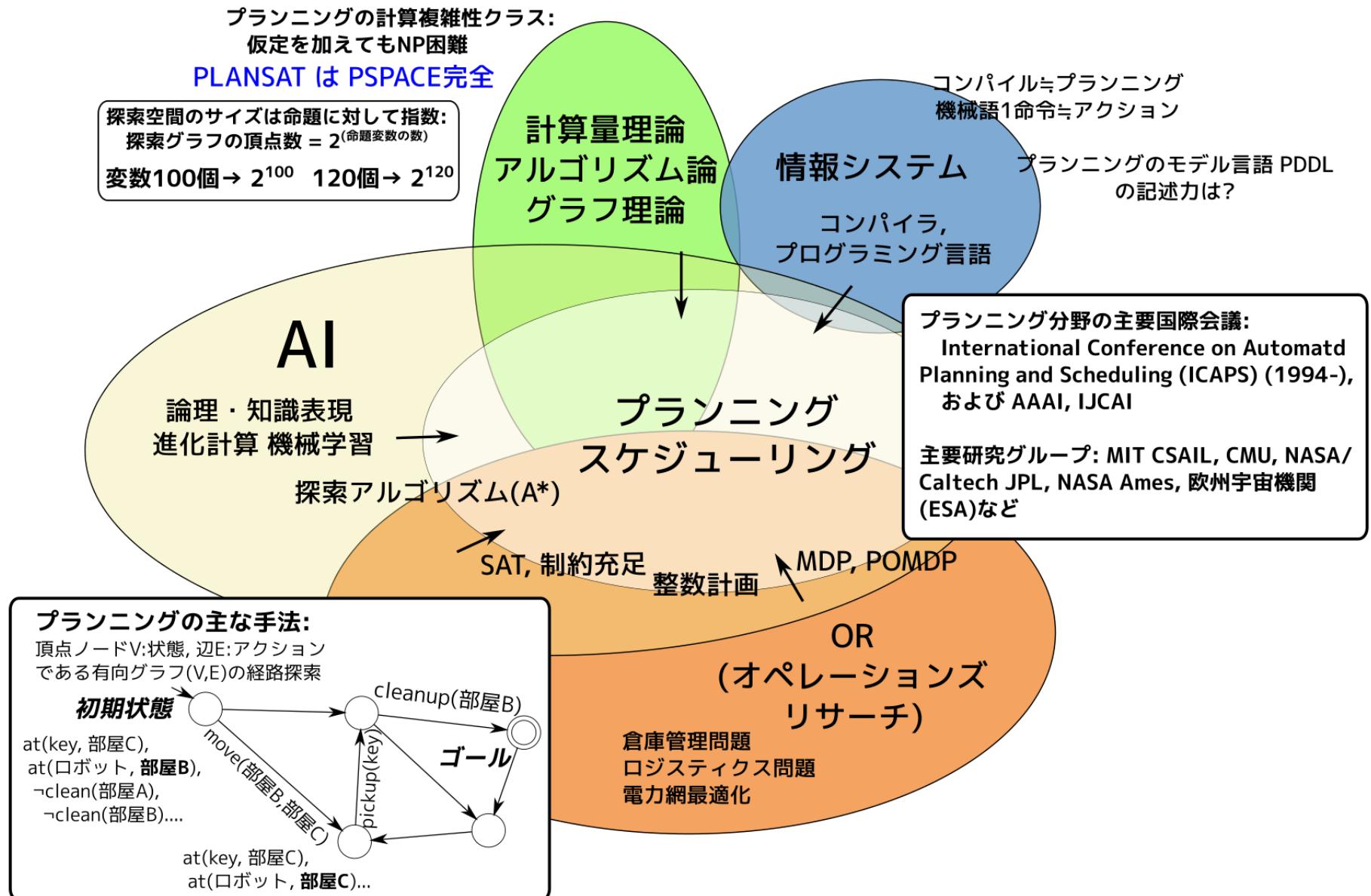
アクション

電話(病院)  
救出(浅井)

引き渡し(浅井, 救急車)

問題を論理で記述 →  
目的達成の手段を  
自動で計算可能

## 1.5 プランニング(自動行動計画)分野の位置づけ



## 1.6 業績1: 査読付き国際学会 ICAPS14 (採択率33%)

**セル生産方式:** 工場生産方式の一種 (対義語: ライン生産)

ロボットアームなどを含む小規模セルが複数協調して製造

**目的:** 並列の製造プランを自動計画 + 製作時間を最小化

一台の組み立て問題を解くのは ある程度複雑でも可能

**研究の対象**

✗ アクチュエータの角度制御 (例:  $\theta: 0\text{deg} \rightarrow 30\text{deg}, 1\text{deg/sec}$ )

◎ 行為の計画(例: 部品2をアーム1で塗装, ネジ3を締結 etc.)

**問題点:** 扱う対象の数が多くなる場合 (製品が4個程度以上の時)

最適解を返すソルバで直接解く → ✗ 探索空間が広大で探索不可

1つ分の製造手順を複数つなげて後処理最適化 → ✗ 生産時間が長い  
(makespan)

**手法:** 問題の繰り返し構造を「工場, セル生産, 製品」など前提なしに  
任意のプランニング問題の論理構造から自動検出

検出した繰り返し構造から効率よい (生産時間の短い) プランを生成

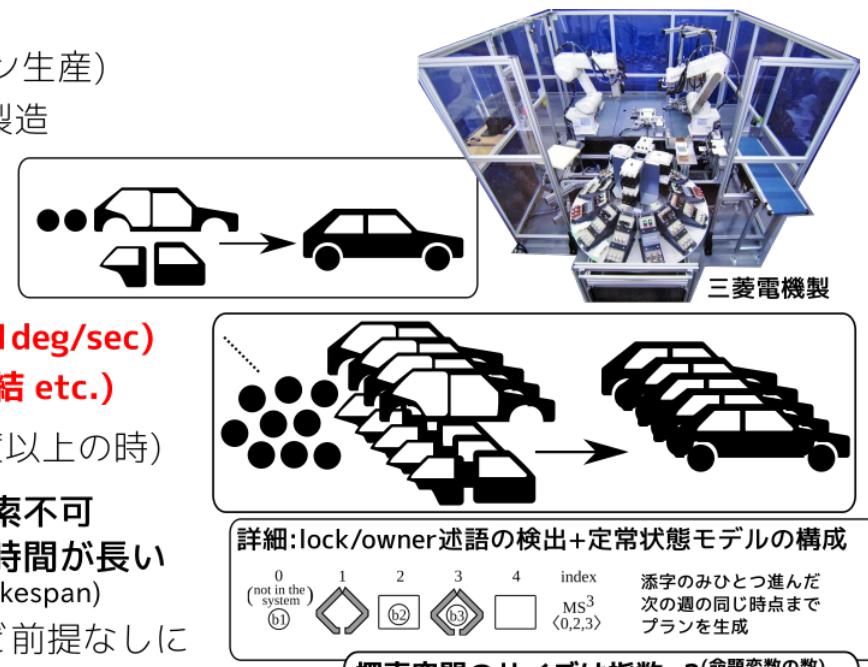
**結果: 1000台以上の製品を組み立てる問題を解けた**

+ 生産時間を最大3割短縮(後処理最適化と比較)

新規性: 繰り返し構造を自動で抽出するというアイディア+抽出技術

インパクト: 大きな高速化, プランニングの適用規模を広げた

株式会社IHIと共同研究(システムの先行評価として)



→ しかも探索空間はその指数で増加  
解ける探索空間のサイズ:

以前:  $10^6 \rightarrow$  提案手法:  $10^{274}$

## 1.7 業績2: 査読付き国際学会 ICAPS15 (採択率33%)

プランニング分野一般の目標:

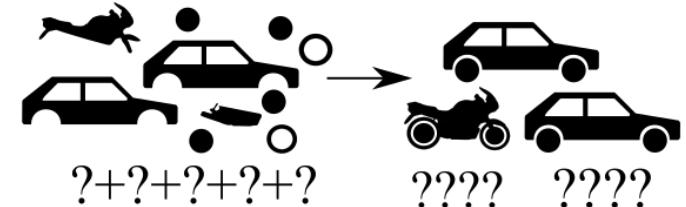
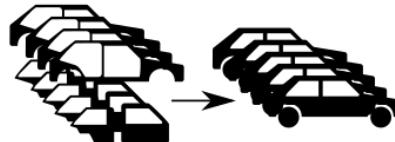
**汎用性を失わずにより複雑な問題を解く! → 高度なAI**

ICAPS14を一般化すればはるかに困難な問題が解けるハズ

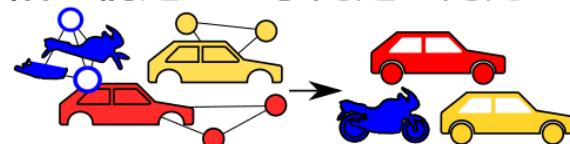
探索空間のサイズは指數:  $2^{(\text{命題変数の数})}$

変数100個 →  $2^{100}$  120個 →  $2^{120}$

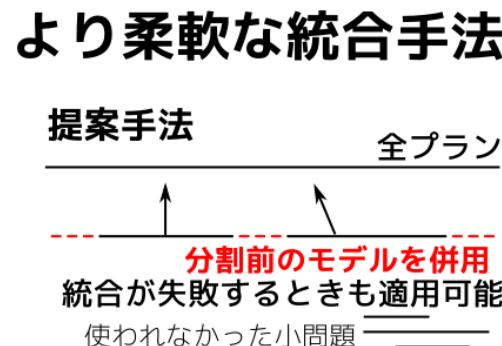
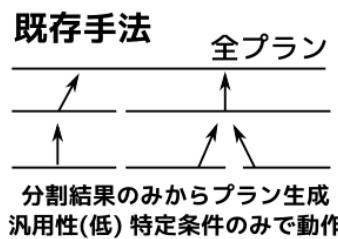
問題が少し大きくなるだけで百万倍難しくなる



手法: **問題の論理構造から小問題を発見する新手法**



分割なしでは解けない問題  
-----  
(探索失敗)



結果: **広い問題種別で**  
3-4倍変数の多い問題を解けた  
解ける探索空間のサイズ:

以前:  $\sim 10^7$  提案手法:  $\sim 10^{28}$

新規性: 小問題の発見手法 / 柔軟な統合手法  
インパクト: 高速化, 小問題分割の汎用性を実証

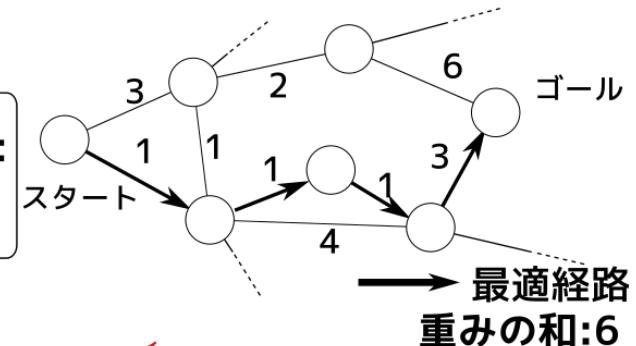
## 1.8 業績3: 査読付き国際学会 AAAI16 (採択率26%)

# グラフ探索一般に適用できる高速化手法の開発

グラフ探索: スタートからゴールまでの経路を探す  
+ 通る辺の重みの和を最小化する

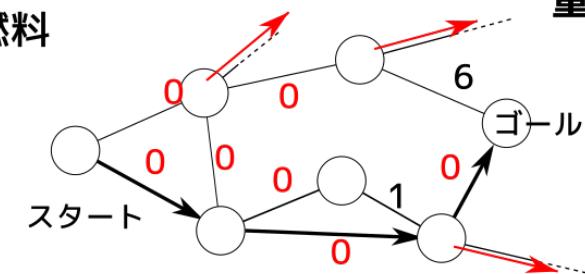
身近な例: カーナビ 但し、本研究の対象は  
メモリには収まらない規模のグラフを扱う  
例: アミノ酸の多重配列問題(MSA)など  
プランニング問題もグラフ探索で解く

著名なグラフ探索アルゴリズム:  
Dijkstra法 (1959), A\*(1968)



複雑/実用的な探索問題には重み0の辺が大量にある (指数的に多い)  
例: トラックの配達計画 (燃料を最小化) 辺=行為, 重み=燃料  
→ 荷物の積み替えには燃料を使わない

問題点: 重みの和が同じノードが大量に存在  
探索を進める方向を見失う → タイブレークの影響大



オリジナリティ: A\*(見積もり値を用いた最良優先探索(最良分枝限定法))にて

1. タイブレークに関する70年代からの定説を覆した
2. タイブレークの新手法を提案し性能向上

(A\*のTiebreakingには  
ゴールまでの距離の見積もり値  $h$  を  
流用すべきだとされてきた)

結果: 解ける問題規模が拡大 (ベンチマーク1104問:814問 → 867問 (+53問))

解ける問題の探索空間サイズ(例):  
(企業ネットワーク脆弱性の診断問題にて) 以前:  $10^6$  提案手法:  $10^{88}$

今後のインパクト: 重み0の辺を持つ広範なグラフ探索問題に対して性能向上

# AAAI16の一般化: タイブレーク≡非最適探索

問題点: 重みの和が同じノードが大量に存在 (**ゼロコスト問題**)

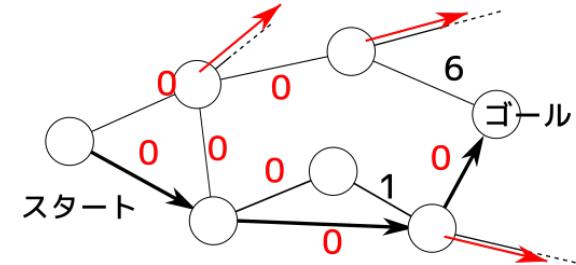
探索を進める方向を見失う → **タイブレーク**の影響大

ノードの下界関数  $h$  の種類

**許容的:** 常に真のコストより低く見積もる → 最適解

**非許容的:** 非最適解を見つけてしまう ← 従来は使われない

**Unit cost 関数:** 辺コストを全て1で代用, 非許容的関数の例



主な発見: タイブレーク とは **許容的関数  $h$**  で作ったプラトーの中で**非最適探索**を行うことである

新規性: **最適探索**で**非許容的関数**を効果的に使う方法を**初めて示した**

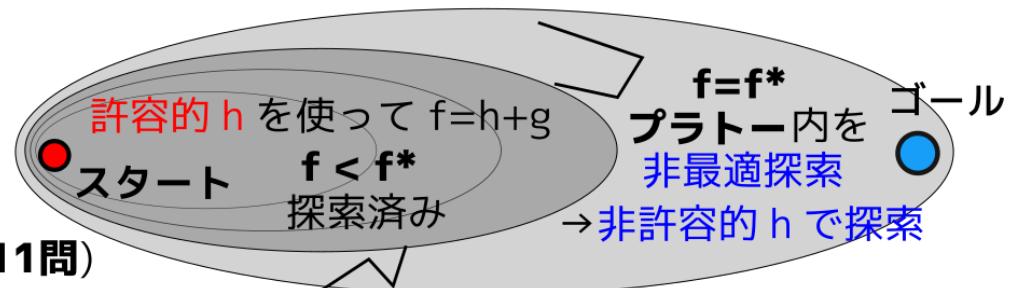
**性能評価:**

ゼロコスト問題のタイブレークに

**非許容的 Unit cost 関数**を使う

→ 指数的に高速化 (844問 → 906問)

→ AAAI16と同時使用でさらに高速 (→ 911問)



**理論的成果:**

無限グラフでのA\*の完全性を

様々なタイブレーク法ごとに証明

→ 非最適探索の理論的結果を援用

**非最適探索用の全技術を最適探索に援用可能**

(従来は非互換と考えられてきた)

→ 今後のさらなる発展が期待できる

# 非最適探索の 均一化手法 の理解と改善

非最適探索: GBFS (**貪欲最良** 優先探索) + **非許容的** 下界関数  $h$

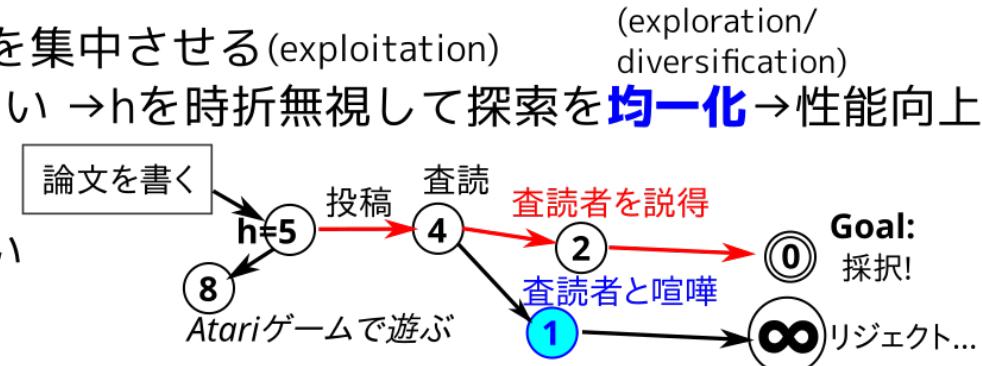
下界関数  $h$  を常に信頼して(**貪欲**)探索を集中させる(exploitation)

→**非許容的**  $h$  は常に正しいとは限らない →  $h$ を時折無視して探索を**均一化**→性能向上

問題点: アドホック手法が多い

→どう組み合わせればよいか解らない

目的: →クリーンに評価しなおす



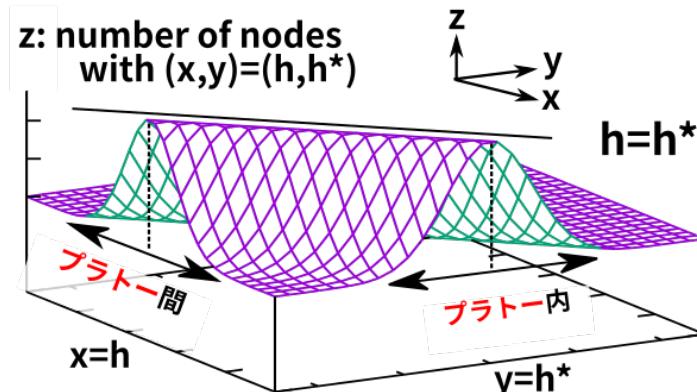
**発見1.** 2つの直行した均一化方法がある。

理由: 真の値  $h^*$  からの  $h$  の誤差は二次元的

→ **プラトー間**均一化と **プラトー内**均一化

→同じ均一化手法を二通りに利用出来る

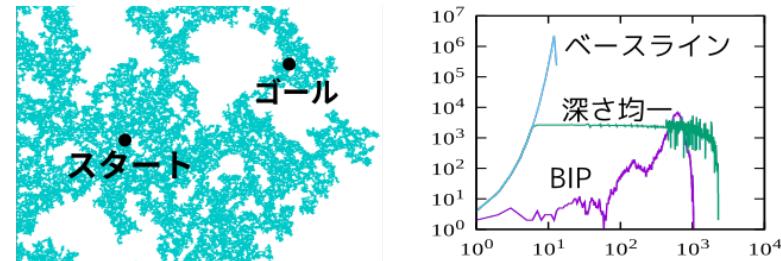
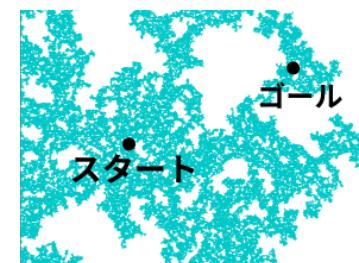
→直行だから同時利用してOK! → 性能向上



**発見2.** ボンドパーコレーション(BIP)

フラクタルを使った均一化手法

→ 性能向上の理由: 探索木の幅を削減



制限時間内で解けた問題数: 192 → 237.7

参考: [Xie et. al. 2014] 深さ均一化 223.9

- 未知の均一化アルゴリズムに汎用に適用できる法則
- フラクタルを用いてより良い探索が出来る可能性

## 1.11 業績6: 査読付き国際学会 IJCAI17 (採択率25%)

# 辺コスト動的計算が必要なグラフ探索の高速化

(IBM Research Ireland での研究)

### 例1: 辺コスト(都市間距離)が未知のTSP

都市数Nに対し  $O(N^2)$  回経路探索ソルバを呼び出す

### 例2: MWRP (複数ワーカー乗り換え問題):

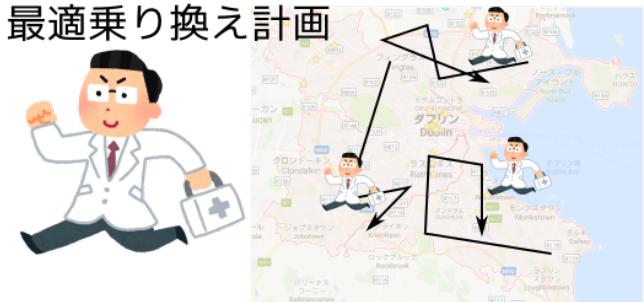
[地図上の全場所] $^2 \times$  [全出発時刻] 個の二点間乗り換え

→ 全ての辺コストを求めるのは現実的でない

→ コストを求めた後の問題自体もNP困難

### 在宅訪問診療のモデル問題

医者が患者を予約時間に訪問する最適乗り換え計画



問題点: 一回の辺コストを求める計算(外部ソルバ)が重い

→ 全コストを求めるのが現実的でない

→ A\*探索で必要になった辺コスト  $c_a$  のみを動的に計算

→ それでもなお実行時間の90%を辺コスト計算に使用

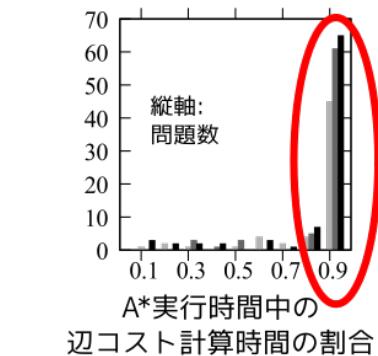
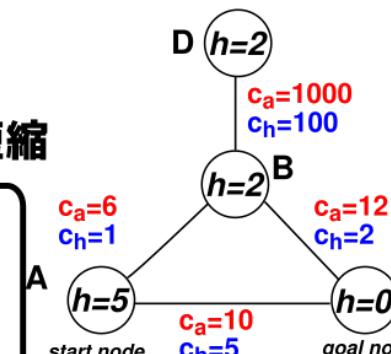
解決法:  $c_a$  の下界  $c_h$  で代用 必要になるまで  $c_a$  計算を遅延

$c_h$ : 外部ソルバの下界関数值,  $c_a$  より高速に計算可能

同じノードを  $c_h/c_a$  で二回展開・キューに挿入

→多くの  $c_a$  を計算せずにすむ → 探索時間を最大5倍短縮

多くの実問題は階層構造を持つ  
既存外部ソルバを用いて汎用に適用可能



e: expand	d: deleteMin	i: insert	DEA*
i A(5)	i A(5)	e B(8)	i D(106)
d A(5)	d A(5)	i C(8)	d C(8)
e A(5)	i A(5)	e A(5)	i C(18)
i B(8)	d A(5)	i B(3)	d C(10)
i C(10)	e A(5)	i C(5)	e C(10)
d B(8)	i B(3)	d C(5)	goal
e B(8)	i C(5)	i C(10)	
d C(10)	d C(5)	d B(8)	
e C(10)	i C(10)	goal	
goal	d B(8)		

## 2 産業技術総合研究所で行いたい研究

Neural-Symbolic複合システムによる

次世代AIシステム

の研究

## 2.1 Q. いまはやりの Deep Learningとの違いは?

A. レイヤが違う

機械学習・**Neural Networks** == 関数近似

for 認識・反射

- 入力は **Subsymbolic** (連続値)  
画像、音声、非構造化テキスト:
- 感覚的知能:  
反応、直後の行動の決定 パブロフの  
犬: 餌を認知 → よだれ  
自動運転: 赤信号、人 → 止まる。  
翻訳: 文章 → 文章
- 囲碁局面の評価関数: 局面 → 勝率 効率  
よく 1-to-1 mapping
- 単純作業

推論・探索

for プランニング・ゲーム・定理証明

- 入出力は **Symbolic**  
論理オブジェクトルール
- 論理・推論による知能:  
未来に渡る 戰略の決定  
(戦略 = 行動の列や木) レスキュー  
ボ: ゴール = 被災者生存  
証明器: ゴール = QED  
コンパイラ: 命令列の生成
- 囲碁、将棋: ゴール = 勝利 順序制約+複雑な作業

- AlphaGo = Subsymbolic (DLNNによる評価関数) + Symbolic (MCTSによる探索)

## 2.2 既存の有名システム

AlphaGo = Subsymbolic (NNによる評価関数) + Symbolic (MCTSによる探索)

- ただし ドメイン依存 – 囲碁に特化, ”マス目”や”石”といった概念をハードコード
- 膨大な棋譜が必要 — 運用データがない環境(e.g. 火星)には適用不能
- 人って模範解答がないと行動できませんか? 真の自律機械は前例無しでも行動可能

DQN = Subsymbolic (DLNN) + 強化学習 (DLNN)

様々な Atari Game につかえる汎用フレームワーク (Invader, Packman… ) だが

- RL の Acting: 学習した policy に従って greedy に行動
- Atari ゲームは 脊髄反射で生き残ることが可能 → 複雑な論理思考はいらない!

## 2.3 記号的AIによる論理推論の重要性

### f Conclusions

Y LeCun

#### ■ Deep Learning is enabling a new wave of applications

- ▶ Today: Image recognition, video understanding: **vision now works**
- ▶ Today: Better speech recognition: **speech recognition now works**
- ▶ Soon: Better language understanding, dialog, and translation

#### ■ Deep Learning and Convolutional Nets are being widely deployed

Deep Learning に  
推論技術を組合せないといけない。

#### ■ We need hardware (and software) for embedded applications

- ▶ For smart cameras, mobile devices, cars, robots, toys....

#### ■ But we are still far from building truly intelligent machines

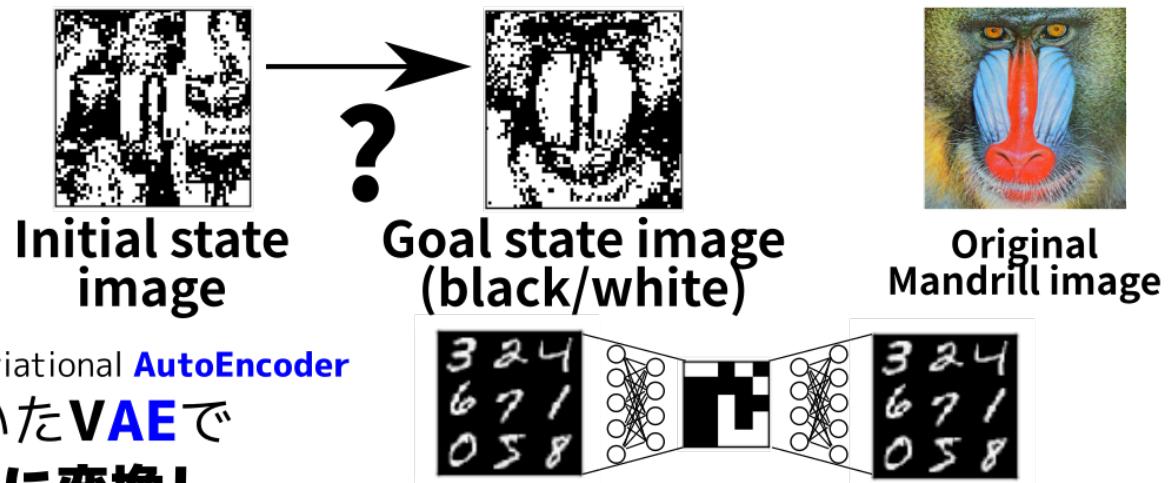
- ▶ We need to integrate **reasoning** with deep learning
- ▶ We need a good architecture for “**episodic**” (short-term) memory.
- ▶ We need to find good principles for unsupervised learning

## 2.4 業績7: 査読付きワークショップ KEPS (採択率 60%)

### 深層学習(非記号型AI)と行動計画(記号型AI)の融合

画像で示された問題を

「パネル」「9マス」「動く」などの情報は一切与えられない問題を解いた例も与えられない



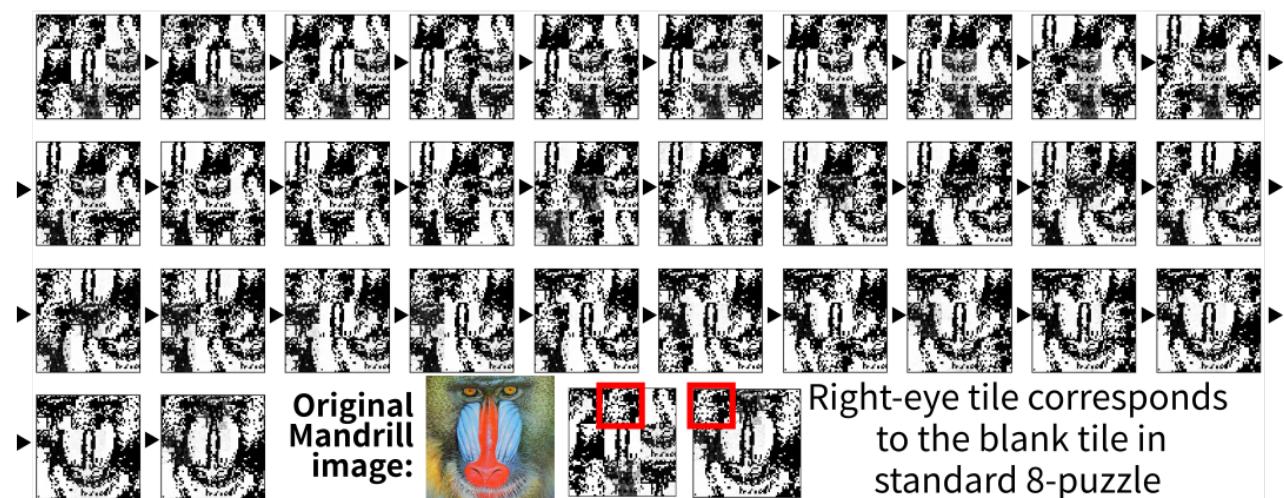
Gumbel-Softmax を用いたVAEで  
記号的命題表現に自動的に変換し  
記号的プランニングソルバで解く

ノードの数を減らしてから元のサイズに戻す NN  
入力画像と出力画像が一致するように学習  
かつ、真ん中の層は0/1値: 何らかの命題の真偽値に対応

結果を画像に戻して  
プランを返却する

システム

LatPlan

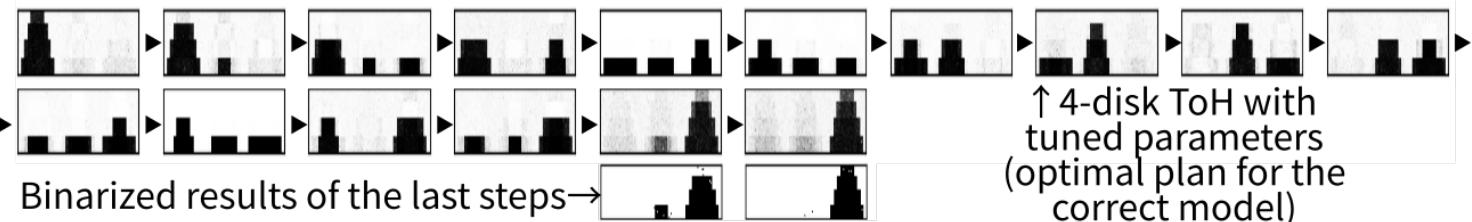


## 2.4.1 研究業績7：査読付きワークショップ KEPS

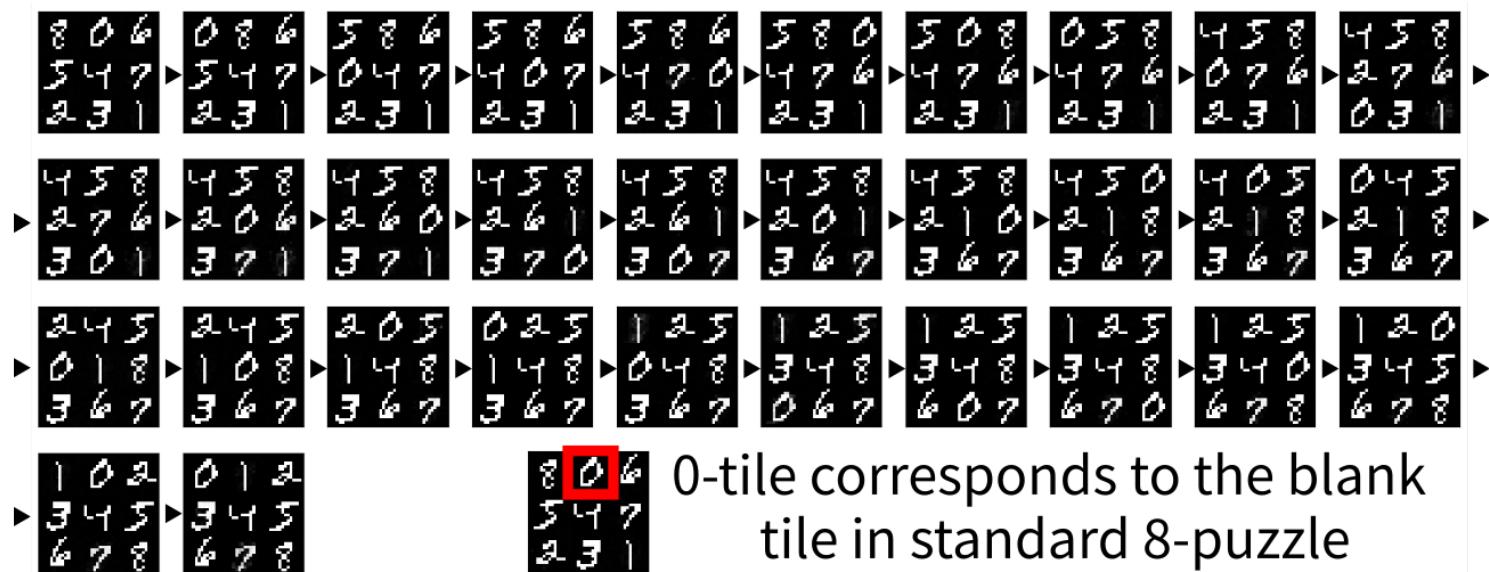
# LatPlan: 同じシステムで全く異なる問題を解ける

問題ごとにニューラルネットの学習は必要

ハノイの塔



数字版  
8-パズル



LightsOut

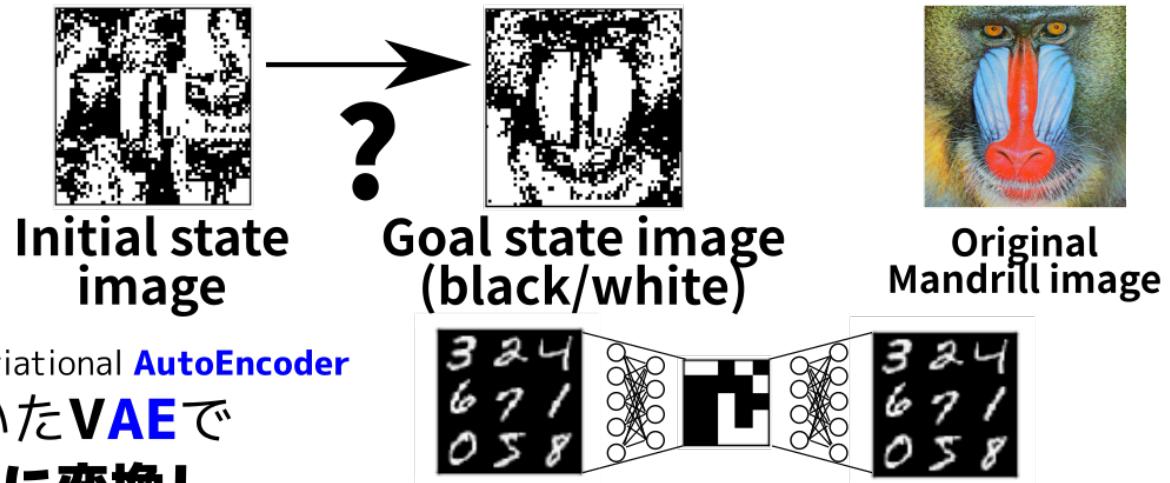


## 2.4.2 業績7: 査読付きワークショップ KEPS (採択率 60%)

# 深層学習(非記号型AI)と行動計画(記号型AI)の融合

画像で示された問題を

「パネル」「9マス」「動く」などの情報は一切与えられない問題を解いた例も与えられない



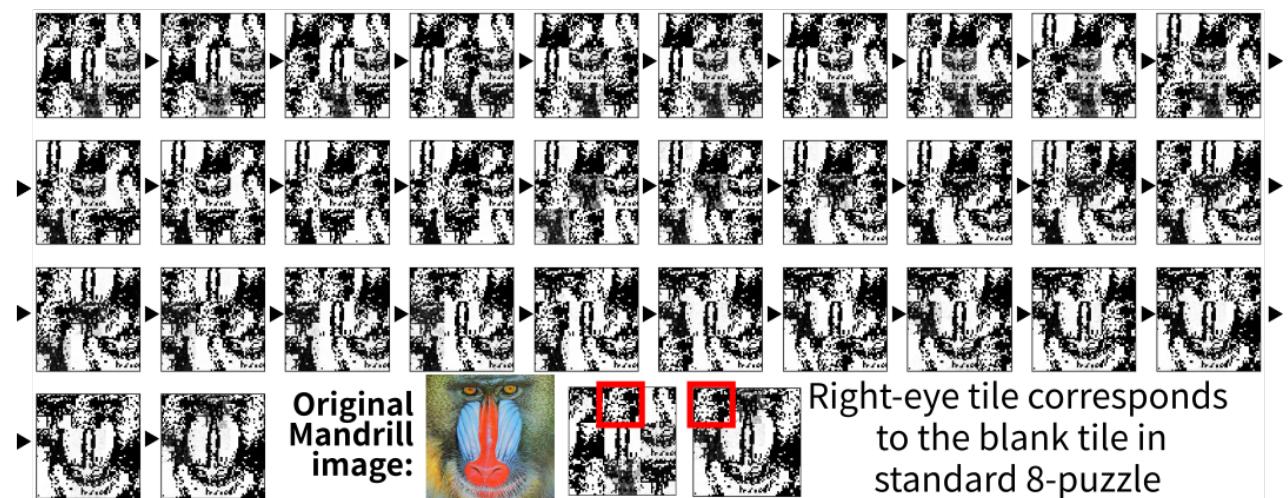
Gumbel-Softmax を用いたVAEで  
記号的命題表現に自動的に変換し  
記号的プランニングソルバで解く

ノードの数を減らしてから元のサイズに戻す NN  
入力画像と出力画像が一致するように学習  
かつ、真ん中の層は0/1値: 何らかの命題の真偽値に対応

結果を画像に戻して  
プランを返却する

システム

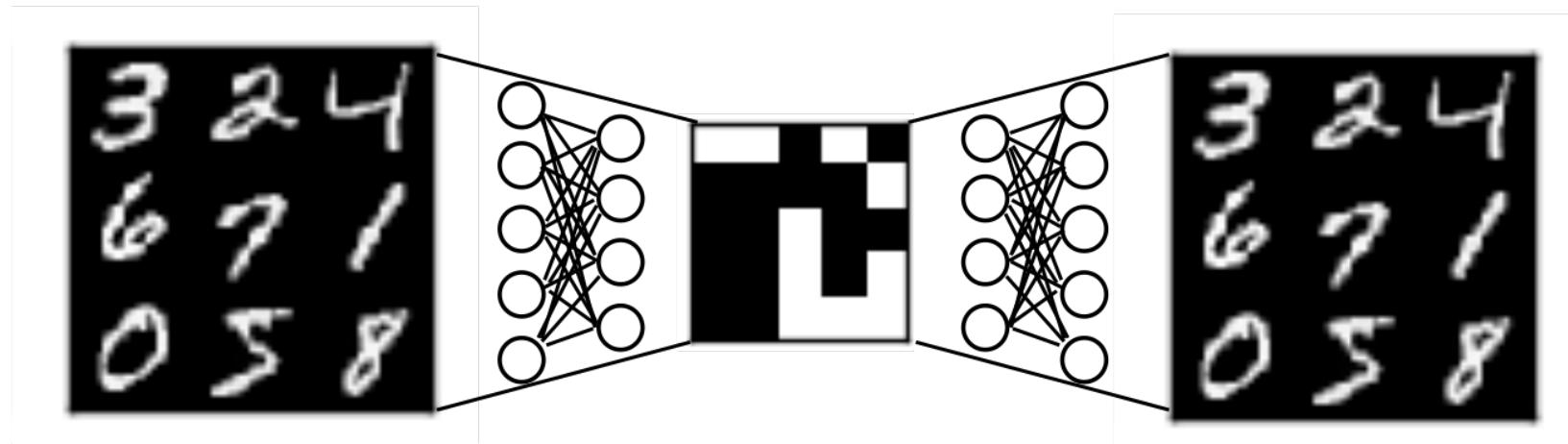
**LatPlan**



### 2.4.3 Gumbel-Softmax

AutoEncoder(AE): 入力と出力が一致するように学習するNN  
中間層: 記号的システムで扱えない実数値

↓  
Gumbel-Softmax AE: 1-hot カテゴリカル分布 を近似するAE  
中間層: 0/1値, 何らかの命題の真偽値に対応させて  
記号的システムで使える!



#### 2.4.4 研究業績7：査読付きワークショップ KEPS

## 強化学習とは異なり、優れた理論的性質

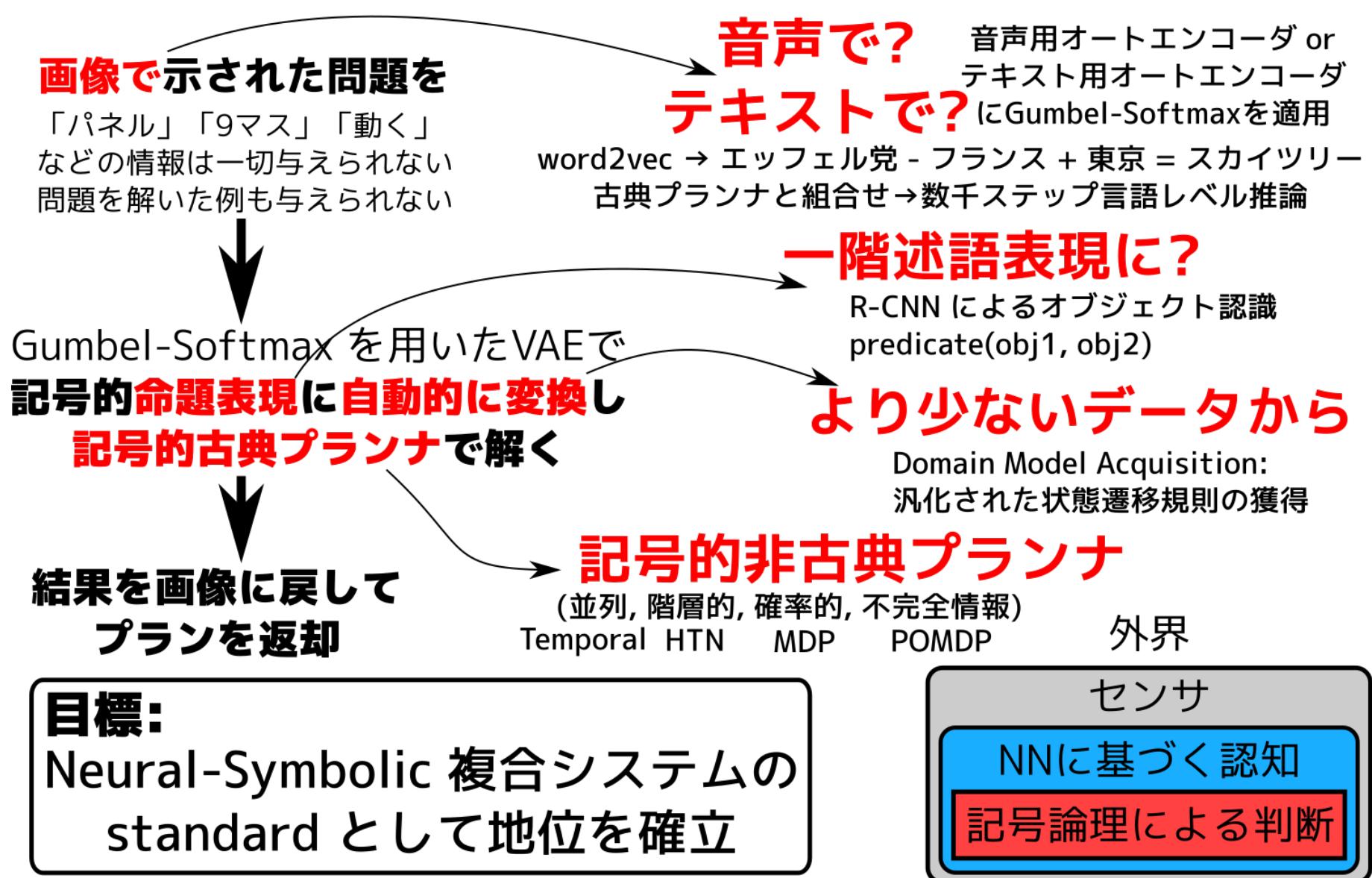
アルゴリズムの完全性：解が存在するときには必ず解を発見する  
解の最適性：理論的性質のわかっている下界関数を使うことで保証

	入力・認識	意思決定
DQN	画像・ニューラル (事前知識なし)	Greedy+NN強化学習 =反射的エージェント
AlphaGo	ハードコード ("石","マス目")	記号的 MCTS+NN強化学習 完全性 vs. イ・セドル 白74手 「読み違え」 学習結果に理論保証なし→非最適解 ( $t \rightarrow \infty$ での収束保証のみ)
LatPlan	画像・ニューラル (事前知識なし)	A*+許容的下界関数 (完全性+解の最適性)

**認識の誤り(NNに起因)はあれど 決断の誤りは無い**  
ニューラル認識と記号的意思決定(理論保証付き)を持つシステム

### 3 今後の研究計画

## LatPlanの様々な発展形を実現



## 4 研究成果の産業応用

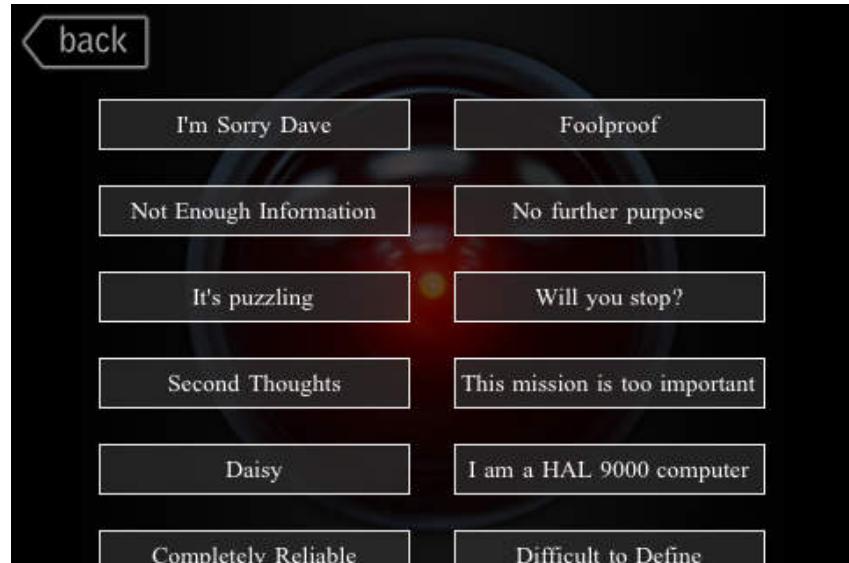
記号的推論により様々な分野にブレークスルーをもたらす

ピックアップロボ以上の産業用ロボ

- 画像 → 論理表現 → 記号的推論 → 行動決定

— 動的な環境や目標に自動で対応できる  
人工無能以上のチャットボット

- テキスト → 論理表現 → 記号的推論 → 意図推定 → 返答



## 5 まとめ

1. 難関国際会議(33%) Fully Automated Cyclic Planning for Large-Scale Manufacturing Domains. In ICAPS2014.
  - (a) 任意の問題から1種類の繰り返し構造を自動で検出
  - (b) 工場での製造スケジューリング(x1000高速化, 探索空間  $10^6 \rightarrow 10^{274}$ )
2. 難関国際会議(33%) Solving Large-Scale Planning Problems by Decomposition and Macro Generation. In ICAPS2015.
  - (a) 複数の繰り返し構造をより柔軟・汎用に組み合わせる手法
  - (b) ベンチマークセット全体で高速化(x3-4高速化, 探索空間  $10^7 \rightarrow 10^{28}$ )
3. 難関国際会議(26%) Tiebreaking Strategies for A\* Search: How to Explore the Final Frontier. In AAAI-2016. (JSAI学生奨励賞)
  - (a) コストゼロの辺がグラフ探索に引き起こす問題を解決(探索空間  $10^6 \rightarrow 10^{88}$ )
4. 難関論文誌(12%) Tie-Breaking Strategies for Cost-Optimal Best First Search. Journal of Artificial Intelligence Research 58 (2017): 67-121.
  - (a) (3.)に加えタイブレーキングと非最適コスト探索の関連性を指摘, さらに性能向上
5. 難関国際会議(33%) Exploration Among and Within Plateaus in Greedy Best-First Search. In ICAPS2017.
  - (a) 非最適コスト探索をフラクタルを用いて改善
  - (b) プラトー内均一化とプラトー間均一化の直交性を実証
6. 難関国際会議(25%) Efficient Optimal Search under Expensive Edge Cost Computation. In IJCAI-2017.
  - (a) 辺コストの動的計算が必要な問題に対して高速な最適アルゴリズム DEA\*
7. 国際ワークショップ(60%) Classical Planning in Deep Latent Space: From Unlabeled Images to PDDL (and back). Knowledge Engineering for Planning and Scheduling (KEPS) Workshop
  - (a) 画像から命題を自動生成して記号的AIで組合せ最適化問題を解き、画像で出力するシステム

## **6 付録**

## 6.1 既存システムとの違い (追加)

NNで直接問題を解くシステム

TSP [Hopfield and Tank, 1985], NeuroSolver [Bieszczad and Pagurek, 1998]

- NNで解くが、入力はシンボリック (それぞれのニューロンが人の与えた状態変数に対応)
- 完全性、最適性などの補償なし

NNを認識でなく探索の枝刈りの中で使うシステム

AlphaGo [Sievers 16], [Arfaee et al., 2011], [Satzger and Kramer, 2013]

- LatPlan は NN を探索の外で使う

## 6.2 Learning from Observation との違い

主にロボットの経路探索(ローレベル制御) [Argall et al., 2009]

ボードゲームの学習だが「マス目」など強い仮定 [Barbu et al., 2010; Kaiser, 2012; Kirk and Laird, 2016]

Action Segmentation problem がある

- 「映像の観察」を中心とするので、いつアクションが始まる/終わるのか解らない
- LatPlan には関係なし

### 6.3 AI プランニングの *Killer App*

人が高価 or 不可能な作業 原発, 宇宙空間,  
火星, 深海

正しさと最適性の理論保証が必要なミッション

製造システム、運送 (時間=お金)

人工衛星 (燃料使いきれば運用終了)

間違った解は許されない

思考過程を説明可能なシステム レス

キュー・宇宙船 (人間の安全がかかっている)

