# Situated Agents Can Have Goals

Pattie Maes

平成 16 年 6 月 29 日

この論文は自動エージェントの挙動選択問題について討論する。私達は「situated agent」とよばれる、古典的方法による限界を越えるために設計されたエージェントについて述べる。今までのエージェントでは、目標と同時進行する調停機構がないため、この問題に十分な解決法を与えなかった。私達は、環境とエージェントの間のループにおいて、すばやく、頑健な行動を処理する間、行動と目的の間をとりもつことができる斬新な行動選択理論を提案する。その理論は行動選択を、エージェントがとることが可能な行動と、行動と環境の間にある、活動的または抑制的な動的変化の新しい特徴としてモデル化する。ほんの少しの大域変数が、いろんな行動選択範囲を円滑化する。例えば、状況指向エージェントではなく目標指向エージェントでバランスをとるエージェントは、環境適応せず、きめられた計画に沿って、何も考えず目標に向かって調節をおこなう。

## 1　導入

人工知能研究の主要な関心の一つとして、次に何をすべきか判断できる自動エージェントの行動選択の機構の発展があります。その問題は二つの異なったアプローチで解かれようとしている。1970 年から、「熟慮思考」というパラダイムがこのような行動選択の理論とともに一般の AI 世界を支配した。その主要な論文は、知的処理は、記号化された内部モデル上で原因結果プロセスによって実装されていた。行動選択のために、より明示的に、これは、エージェントが行動、目的、事象の内部表現をもっていて、この表現は、何を行えば目的を達成できるかを判断する「プランニング」とよばれる原因結果プロセスで使用されているということを暗に示している。この「プラン」は実行プロセスを取り扱っており、多かれ少なかれ目に見えずとも、明示的な行動を成し遂げた。

わがままな考えのアプローチは、他の特定の処理では成功を収めたが、特に、動的にかわる実環境下においては、貧困な結果しか獲得できなかった。ほとんどのシステムがもろさや頑固さ、スローレスポンスなどの重要な欠点をもっていた。これらはまた、フレーム問題や単調原因問題などの解決できない理論的問題もかかえていた。より最近になって、ある研究者達は実時間で自己思念型機械による行動指向問題の解決は非現実的であることを証明した。このことは状況的自動機械、situated agent、相互システム、サブサンプションアーキテクチャ、振舞いベースアーキテクチャ、行動ネットワークなど、劇的に異なったアーキテクチャを発展させた。

これらのアーキテクチャで強調されていることは、行動認識、集中と分散、環境との動的インタラクションや資源の限界と不十分な知識を扱うための本能的メカニズムを統合することに着目することである。これらのアーキテクチャによって共通のおもしろい考えは「緊急機能」である。エージェントのその機能は動的な環境とやりとりするシステムの緊急特徴を示している。エージェントの振舞いの特徴だけではエージェントが処理を行うときに現れる機能を説明しない。代わりに、機能の大半は環境の静的、動的特徴に依存する。環境は動的に説明するばかりでなく、その特性が、システムの機能を与えることに利用される。この観点の重要な意味はだれも簡単にはエージェントがどのように目標を達成するのかを言えないことです。代わりに、望まれる目標に集中したシステムや環境をとりまく「ダイナミクス」や、インタラクションループ、サーボループなどがみつかります。目標を達成したときだけ、インタラクションループは休める。

これらの、行動選択の新しい理論 (この論文で

のべる situated agent) は実世界で信頼できる成功結果をすでに例証している。この論文では、現存する状況指向型システムが、明示的な目標、目標取り扱い機能がないため、またそれに関連してシステムが挙動選択について、あらかじめ組み立てたり、配線したりしなければならないため重要な限界を持っていることについて議論する。私達は行動のなかの活性的、抑制的ダイナミクスの緊急特性における斬新なより進んだアプローチを提案する。そのメカニズムは伝統的プランニングと situated agent の両方の特性を統合したものである。それは、環境とのループにおける迅速かつ頑健な行動を生みだすが、その一方で同時に予測やプランニングも行える。

私達のアプローチと今までの状況的行動システムとの重要な違いは、ハードワイヤー行動選択や、組み立てなし行動選択を用いないところです。行動間の調停は、自分自身がいる状況とシステムの目標によって異なる実時間実行課程である。それゆえに、それは問題に対してより単純で、より柔軟で、より一般的な解答を作る。それは、この実時間調停が組織されている方法を用いた古典的プランナーとは異なる。調停を普通に行う中央支配構造を進化させたプログラミングの代わりに、行動間と、行動と環境の間の並行的ローカルインタラクションによって分散された構造の中にこの新しいアプローチ支配構造は突然現れる。だから、situated agent の概念の限定のなかで、環境やシステムをとりまくインタラクションループやサーボ支配ループは、望まれる機能に集中するように設定される。そのダイナミクスは、行動選択を環境の状況やエージェントの内部目標への反応の中に現す。

この論文は行動選択の問題とプランニングアプローチと situated agent アプローチの限界についての議論から始まる。それから、その二つのよいところを統合した行動選択アルゴリズムを具体的にのべる。加えて、そのアルゴリズムは仕事環境によって行動選択をチューニングする大域変数も供給する。この方法で、目標達成指向なのか、状況適応指向なのか、といったいろんな行動選択の範囲を円滑に平滑化できる。このアルゴリズムによって得られた結果を詳しくのべる。この論文はその結果にたいしておもしろい討論を行って締めくくる。

# 2 行動選択の問題点

## 2.1 実時間調停の必要性

この章の主要目的は自動エージェントの行動理論として重要な限界がある古典的プランニングより比較的最近に提案された新しい行動選択アーキテクチャの例証を行うことである。これらのシステムが行動を組織する方法はコンパイル時に行動選択をハードワイヤリングすることである。エージェントは、状況と行動のルールの集合を実装した循環を装備している。異なったルール間での調停の問題がないように、状況表記が明示的にされたり、ルール間の調停が人手で行われることに注意を払う。ルールそのものは手で作成されてもいいし、エージェントの望まれるふるまいを宣言した表記に基づいて組立られたものでもよい。

手で作成された場合、結果は典型的によい結果を示すが、とても明示的なルールを書くためには膨大なコストがかかる。例えば、ペンジーシステムはペンゴゲームを遊ぶために、とても明確な「技巧」の集合で構成されている。逆に、プランニングにはプログラマーのかわりにエージェントとともにすべきことを理解するという負荷が残っている。プランナーは一般的に行動選択についての明示的な知識をもっている。すなわち、Y が X の前提条件を取り消すならば、行動 Y の前に行動 X をとらなければならない、などである。もし、古典的プランニングが自動プログラミング、自動プランニング問題を答えようとしたならば、現在の situated agent アプローチは全ての難しい仕事をこなすシステムの設計者に依存するだろう。

また、目標を表すための構成時間を基本に、行動選択を前構成する二番目のアプローチも限界をもっている。二つのアプローチは両方とも、人がすべての起こりうる事象の中でとるべき一番の行動が予測できる必要がある。私達は人が前構成できる範囲には限界があると信じている。より複雑な仕事の場合、とても少ない条件しか構成時にはわかっていない。理由は、行動選択は情報に基づいて起きるからである。ある情報は構成時には利用できないし、ある情報は実行中にわかる。複雑な環境下における複雑なエージェントにおいては、後者の情報が支配してる。これは、前構成の解答はたくさんの状況を含んでいるということを暗に示している。

二番目の限界は二つのアプローチは両方とも、前構成の結果えられたエージェントが明示的な目標をもっていないということである。明示的な目標はたくさんの理由から、自動知的エージェントには重要である。

- 目標が新しくなったり、変わったりしたとき、再プログラミングせずほかのエージェントと通信できることは重要である。例えばエージェントが動けなくなったとき、助けたり、助言したりするために行動選択にバイアスをかけて、振舞いに影響させれたらよいと思う。私達はエージェントが、調停された目標を与えられることを要求しないということを注意しなければならない。

- たとえ、人がエージェントに対して目標を通知することをしなくても存在する。「内部」の目標が重要でもある。どんな複雑なエージェントでも、違ったモデルと駆動原理を持っている。その振舞いと行動選択はこれらの内部目標とそれらがどれだけ重要であるかによって異なる。例えば、あなたが本当に喉が喝いているとき、あなたはカップに手を伸ばす。あなたが喉が喝いていなかったら、カップに手を伸ばさないだろう。あなたがしたいことをすることができる。

- 特定の目標を持つことは可能な選択を制限するため、行動選択をより簡単にする。目標が果たしうる機能のひとつには、人が仕事のローカルな細かいことをしている間それを忘れている場合の、人が計算しているグローバル画像の記憶のようなものである。エージェントが明示的な目標をもっていなかったら、全ての状況は次に行うべき一番よいことを決めるため、完全な情報をもたなければならない。たくさんの仕事のためこれは非現実的な要求である。しばしばエージェントはグローバルに求められる結果を得るため、ローカルな知覚より悪いと思われる状況に走らなければならない。

- 複雑なエージェントは複雑な目標をもつ。まず初めに、それはたくさんの目標をもっている。第二に、目標は常に変化する。第三に、異なった優先順位をもっている。そして第四に、その優先順位は状況と相互関係によって異なる。だから、自動エージェントが目標の中間値をとること、それらの競合を取り扱うこと、実時間で達成できるよう最適化するように、自分の相互関係を開発することは重要である。

- 最後に、目標は自己意識のために重要な材料である。目標は、エージェントが、自分自身のパフォーマンスを改善したり、学んだりすることができるために特に重要である。エージェントは、(1) いつそれを行うのがよいのかを知るため、(2) 学ぶコストを知るため、何がエージェントの目標となるのかという明示的な概念が必要である。明示的な目標をもつことのさらなる利点は、一つの目標を達成する行動の背景をいったん学習したら、異なる目標の背景にも適用することができる。

だから、知的自動エージェントは完全に作りあげられた目標と目標を取り扱う能力が必要である。situated agent の研究者たちによって提案された解法は状況の表記中に抽出された状況として目標をもっていることである。しかし、この解法は十分ではない。なぜならば (1) その解法が、ルールを表記した全ての状況に対して、行動がとりうる全ての目標の大まかな分離を書くことを表していて、(2) これはまだ、エージェントが貢献しなければならないたくさんの目標の基本となる行動の間の調停とエージェントの相対的な重要性を許さないからである。もう一つの提案は、古典的プランナーと反応的システムとを統合することである。この考えは続けてプランをうみだそうとするプラン部品をもつことである。もし、プランを無効にする環境の変化の前に、プランニングが成功したら、そのプランは、反応的、状況的システムによって考慮される。この解法がどれぐらい満足いくものかはまだ、明らかではないが、私達が予測した不利益がいくつかある。(1) 結果的振舞いがとても不連続に現れる。(2) プラニングのために利用できる時間の量や必要とされるプランの質は調整できない。理想的に、プランをうみだすのに費す時間とプランの質はトレードオフの関係になりうるでしょう。

### 2.2 異なる範疇におけるトレード オフ

複雑で動的な環境において、たくさんのグローバルな目標を達成しなければならない自動エージェントを想像してください。例は、火星で土壌のサンプルをとらなければならなかった、「ローバー」でしょう。実際に、何が「最も適切」な、エージェントがその特定の瞬間にとるべき次の行動であるのだろうか?行動の最もよい選択のはっきりした定義を示すのはとても難しい。にもかかわらず、行動選択メカニズムが例証すべき性質ははっきりしている。

- それは目標に到達する行動を好む。特にいくつかの目標に一度に貢献する行動を好む。
- それは現在の状況に適切な行動を好む。特に、機会を開発し、予測不可能で変化する状況に対応するものである。
- それは現在目指している目標やプランへ貢献する行動を好む。すなわち、それは、異なる何かをするべきよい理由がないかぎり、特定の目標に「固定」される。
- それは、先を見る。特に、危険な状況を回避したり、相互的な目標や競合する目標を取り扱うことを行う。
- それは頑健である。
- それは反射的であり、迅速である。

行動選択メカニズムのための重要な制約は、エージェントが世界の不完全な、時には誤ってさえいる知識を、限られた計算資源と時間で処理しなければならないことである。これは、行動選択が完全に「合理的」、最適になりうらないということを暗に示している。加えて、上で挙げた望まれる性質は矛盾がある。例えば、エージェントは進行中のプランにバイアスがかけられるのと同様に、環境の変化に同時に完全に対応することはできない。上記性質の正しい「混ぜかた」は、その環境がどれぐらい危機的なものか、変化のペースはどれぐらいなのか、決定の時間はあるのか、環境がどれぐらい予測可能なのか、その仕事にどれぐらいの正確さが求められるのか、などといった、環境と仕事の性質に依存する。理想的には、私達は、環境と仕事の性質への適応を異なった次元において円滑に中庸化できる行動選択メカニズムを求めている。

## 3 動的行動選択

### 3.1 仮説

この論文の残りの章では、古典的な人工知能プランナーの落とし穴を避ける一方、(目標から遠ざかるなどの) 代替的「プランニング」を許す、situated agent の行動選択メカニズムについてのべる。それは、機能や、分散、平滑化された内部モデルの役割、緊急機能に沿うかわりに、仕事に沿っての分解のような、新しいアーキテクチャの美しさと、古典的プランナーのすばらしい方法による、目標取扱い能力を統合する。それは、環境の性質と合うようにある次元に沿った行動選択振るまいを変化させることを可能とする。

私達が検証した仮説は、行動選択は、エージェントがとることができる異なる行動の中の活性的、抑制的ダイナミクスの緊急特性としてモデル化されうるのかどうかということである。私達は、(それ一つでなんでもこなせるような)「官僚的」モジュールを用いることを避け、コントロールの大域的形式を持つことにする。その考えは、いろんな方法、古典的プランナーにおいて起こる、故意的なコントロールによって供給されるものと同様の機能を得ることです。私達の行動選択メカニズムの「故意」とは暗示的です。それは、動的な処理の緊急特性である。合理的なコントロールでは、単純なモジュールの集合間で並行かつ局所的に相互作用することで分散された構造のなかで緊急事態を計画する。

私たちは、これらの仮説の十分性について研究し、活性的抑制的ダイナミクスのどちらが適切であるかを決定しようとしている。この論文の終りに、私達は一連のアルゴリズムを発展させ、コンピュータシミュレーションにおいてそれらを検証する。ある人は、そのようなアルゴリズムを討論していた。この論文は、より単純で、より面白い結果を出す、たくさんのアルゴリズムを表記する。実験は、いくつかのアプリケーションによって行われた。システムが出した結果は、目標指向、状況指向、適応性、頑健性、予測性などの望まれる特性を示す。加えて、便利な帯域変数のおかげで、

これらの行動選択範疇の間を円滑に中庸化することが可能である。ある人が、データ指向に対して、目標指向とのlバランスをとることができるように、慣性に対する適応性、速さに対する思いやり、目標に対する感度の調整が競合する。

## 3.2 アルゴリズム

この章は、アルゴリズムについて書かれており、とても簡単な例でそれを説明する。そして、数学モデルについて述べ、複雑な例を紹介する。

このアルゴリズムは、自動エージェントをそれぞれが、自分自身の明確な能力を持っているモジュールの集合であると見る。これらの competence モジュールは、古典的プランニングシステムのオペレータににています。competence モジュール i は $(c_i, a_i, d_i, \alpha_i)$ の組み合わせによって記述される。$c_i$ は、その competence モジュールが活性化する前に、満たされなければならない前状態のリストである。$a_i$ と $d_i$ はリストを加えたり、削除したした時点での、competence モジュールの行動結果の影響の予測を表す。加えて、それぞれの competence モジュールは、活性化レベル $\alpha_i$ をもっている。時刻 t ですべての前条件が真になったら、そのモジュールは時刻 t で実行可能になる。活性化レベルが閾値を越えた実行可能な competence モジュールが、選択される。つまり、実世界で行動を行う。competence モジュールの処理 (どのような行動を行うかなど) は、明示的には表されない。つまり、competence モジュールは内部でハードワイヤーでつながれ、論理インターフェースやその他のことを行う。図 1 は competence モジュールの例である。

competence モジュールは、三つの型を持つリンクでネットワークとしてつながれる。三つのリンクは、後任者リンク、前任者リンク、競合者リンク、である。前状態リスト、リスト挿入、リスト削除における、自動エージェントの competence モジュールの記述は、完全にこのネットワークで定義される (これはこのリストの目的でしか無い)。

- x の挿入リストのメンバーでありまた、y の前状態リストのメンバーでもある、すべての命題 p のために、competence モジュール x から competence モジュール y に向けて後任者リンクがある (つまり、2 つのモジュール間に前後任者リンクが複数存在する可能性があることを表す)。つまり、competence モジュール $x = (c_x, a_x, d_x, \alpha_x)$ と competence モジュール $y = (c_y, a_y, d_y, \alpha_y)$ が、与えられたとき、すべての命題 $p \in a_x \cap c_y$ のために、x から y への後任者リンクがある。

- モジュール x からモジュール y への前任者リンクは、y から x へのすべての後任者リンクのために存在する。つまり、competence モジュール $x = (c_x, a_x, d_x, \alpha_x)$ と competence モジュール $y = (c_y, a_y, d_y, \alpha_y)$ が、与えられたとき、すべての命題 $p \in c_x \cap a_y$ のために、x から y への前任者リンクがある。

- モジュール x からモジュール y への競合者リンクは、y の削除リストのメンバーであり、x の前状態リストのメンバーである、すべての命題 p のために存在する。つまり、competence モジュール $x = (c_x, a_x, d_x, \alpha_x)$ と competence モジュール $y = (c_y, a_y, d_y, \alpha_y)$ が、与えられたとき、すべての命題 $p \in c_x \cap d_y$ のために、x から y への競合リンクがある。

図 2 にネットワークの例を示す。

提案するアイデアは直観的には、現在の状況と目標において、「もっとも良い」行動をとれるモジュールの中で活性化エネルギーがしばらく後に蓄積されるように、モジュールが、お互いに活性化、抑制化するためにこれらのリンクを用いる。いったんそのようなモジュールの活性化レベルがある閾値を越えて、モジュールが実行可能になったら、それは、活性化し、実世界において行動をとる。モジュール間の活性化の広がり方ばかりでなく、新しい活性化エネルギーをネットワークに入力するのも、現在の状況と現在のエージェントのグローバル目標によって決定される。

- 現在の状況による活性化

  部分的に現在の状況に合う、モジュールに対して、観察された状況からくる活性化エネルギーの入力がある。もし、competence モジュールが現在の状況において正しいと観察される前状態を一つでももっているなら、そのモジュールは現在の状況と部分的に合うとよばれる。私達が、大域的に続く、更新された世界モデルがあるという仮定をおいていないことに注意してもらいたい。その代わりに、実世界で

のロボットにおいて、それぞれの命題は、仮想のセンサーによって運ばれてくるが、それは、ある命題が正しいと考えられるべきか否かを実際のセンサーデータに基づいて決定されるモジュールである。「現在に観察された状況」は関連する仮想センサーによって、現在観察されるすべての命題の和集合から構成される。

- 目標による活性化

  活性化エネルギーの第二の資源は、エージェントのグローバル目標である。それらは、グローバル目標の一つを達成するモジュールの活性化レベルを増加させる。目標の一つが、たくさんの competence モジュールの追加リストのメンバーであるならば、モジュールは、グローバル目標の一つを成し遂げると言われる。私達が二つの異なるタイプの目標を作っていたことを注意して欲しい。「一度限り」の目標はたった一度成し遂げられれば良くて、「永久的な」目標は常に達成されなければならない。一つ目の例は「車にスプレーペイントを行う」といったもので、二つ目の例は、「半分の電池を積んでおけ」といったものである。

- 守られた目標による抑制

  加えて、すでにエージェントが達成した、または守られるべきグローバル目標による、外部抑制がある。これらの「守られた目標」はそれらをもとに戻そうとするモジュールの活性エネルギーを奪う。モジュールは、守られた目標の一つが、あるモジュールの削除リストのメンバーであったとき、そのモジュールは、守られた目標の一つをもとに戻すと言う。

これらの処理は続く。部分的に現在の状況に合うモジュールやグローバル目標の一つを実現するモジュールに対する、活性化エネルギーの途絶えることの無い流れがある。守られた目標をもとに戻すモジュールの活性化エネルギーの終り無い減少もある。これは、環境やグローバル目標が予期せず突然変わることを意味している。もし、これが起こったら、他の competence モジュールへ活性化の外部入力は自動的に流れる。

- 後任者の活性化

  実行可能な competence モジュール x が活性化を前方に広げている。それは、命題「$p \in a_x \cap c_y$ が正しくない」を共有する、後任者 y の活性化レベルを増加させる。直観的に、私達は、これらの後任者モジュールに、それらは「既に実行可能」であるので、より多くのそれらの前状態が、competence モジュールが活性化になった後に満たされるでしょうから、より、活性化してもらいたい。形式的に、実行可能である、competence モジュール $x = (c_x, a_x, d_x, \alpha_x)$ が与えられたら、「$p \in a_x$ は間違いである」で定義される命題を後任者リンクで持つモジュールに前向きに広がって行く。

- 前任者による活性化

  実行可能でない competence モジュール x は活性化エネルギーを後向きに広げる。それは、命題「$p \in c_x \cap a_y$ が正しくない」を共有する、それらの前任者 y の活性化レベルを増加させる。直観的に、実行不可能な competence モジュールは、後で実行可能になっても良いように、まだ真でない前状態を満たすことを「約束する」モジュールへ広がっている。形式的に、実行可能でない、competence モジュール $x = (c_x, a_x, d_x, \alpha_x)$ が与えられたら、「$p \in c_x$ は間違いである」で定義される命題を前任者リンクで持つモジュールに後ろ向きに広がって行く。

- 競合による抑制

  すべての competence モジュール x は、命題「$p \in c_x \cap d_y$ が真である」を共有する、それらの競合者 y の活性化レベルを減少させる。どんなモジュールでも、活性化されて、真になっている前状態をもとに戻すモジュールを妨げようとする。私達は、モジュールが自分自身を抑制することを許していないと言うことに注意してもらいたい。モジュール間の相互競合があったとき、高い活性化レベルにいるもののみが、他方を抑制できる。これは、もっとも適切なモジュール同士が打ち消し合う現象を避ける。形式的に、実行可能でない、competence モジュール $x = (c_x, a_x, d_x, \alpha_x)$ が与えられたら、「$p \in c_x$ は真である」で定義される命題を前任者リンクで持つモジュー

ルすべてに広がって行く。ただし、より強い矛盾した競合リンクが存在する場合は除く。

例として、図 3 に示すネットワークを考えてみて欲しい。環境の状態が、$S(0) = \{sprayer - in - hand, sander - on - table\}$ であると想定してください。目標は、$G(0) = \{board - sanded\}$ である。活性化の外部広がりは次のようになる。両方の条件が満たされたので、現在の状況が、put-down-sprayer にたくさんの新しい活性化エネルギーを与える。加えて、pick-up-sander の条件の一つが満たされたので、それに活性化エネルギーを与える。sand-board が活性化すると目標が達成されることが約束されるので、エージェントの目標が sand-board の活性化レベルを増加させる。内部の活性化の広がりは以下のようになる。put-down-sprayer が実行可能状態である。よって、その後任者である pick-up-sander へ、前向きに活性化エネルギーを広げている。sand-board と、pick-up-sander,put-down-sander は、実行可能状態ではない。したがって、それらは、それらの前任者に後向きに活性化エネルギーを広げている。特に、sand-board は、自分自身の活性化レベルのかけらとともに pick-up-sander の活性化レベルも増加させる。pick-up-sander は、put-down-sprayer の活性化レベルを増加させる。put-down-sander は、pick-up-sander の活性化レベルを増加させる。(しかしながら、ここで上がるレベルはあまりに小さいので、あまり影響が無いかも知れないが)。同じ、活性化の広がり方が、(状況が変わらない限り) つづいて、put-down-sprayer に活性化エネルギーを蓄えさせる。これが、ある閾値に到達したとき、活性化され、行動を実行する。すなわち、スプレーが置かれるでしょう。

まとめて、環境の状態は変わり、外部と内部の活性化パターンも変わるでしょう。特に、pick-up-sander は、その後十分な活性化エネルギーを蓄え、活性化するでしょう。その後、sand-board と put-down-sander が実行可能になるでしょう。しかし、sand-board は、第一に、それが、目標から活性化エネルギーを受けるから、第二に、put-down-sander が、命題「sander-in-hand」のため、sand-board の競合者になるため、put-down-sander から活性化エネルギーが逃げるから、put-down-sander より速く活性化エネルギーを蓄える。sand-board が活性化された後、目標は達成される。

グローバルアルゴリズムは competence モジュールのすべてと代わる以下の計算を毎ステップごとに行うループを持っている。

1. モジュールの活性化レベル上の現在の状況や、目標、守られた目標の衝撃が計算される。
2. competence モジュールが活性化し、抑制される、前任者リンク、後任者リンク、競合者リンクを通してモジュールと関連づけられる方法が、計算される。
3. 遅れた機能が、全ての活性化レベルが定数であるかを確かめる。
4. 以下の 3 つの条件を満たす competence モジュールが活性化される。(1) それは、実行可能である。(2) その活性化レベルはある閾値を越えている。(3) その活性化レベルは他の (1)、(2) の条件を満たすどの competence モジュールよりも高いものを持っていなければならない。二つの competence モジュールがこれらの条件を満たしたとき、それらのうちの一つがランダムに選ばれる。活性化されたモジュールの活性化レベルは再び 0 に初期化される。もし、どのモジュールも (1) と (2) の条件を満たさなければ、閾値が何%か引き下げられる。

これらの 4 つのステップは無限に繰り返される。面白いグローバルな観察できる特徴は、活性化された一連の competence モジュールであったり、この一連の最適解であったり、それが獲得される速さであったりする。

4 つの大域変数が広がる活性化ダイナミクスとエージェントの行動選択の振るまいを「調整する」ために用いられる。

1. $\theta$ は、活性化するための閾値であり、活性化に関係がある。$\pi$ は、活性化の中庸レベル。$\theta$ は、一定時間どのモジュールも活性化しなければ 10%減少させ、モジュールが選択されたらもとの値に戻す。
2. $\phi$ は真であると観測された命題がネットワークに導入される活性化エネルギーの量。
3. $\gamma$ は目標がネットワークに導入される活性化エネルギーの量

4. $\delta$ は守られた目標がネットワークから外される活性化エネルギーの量

これらのパラメータの例えばの値として、$\theta = 45, \pi = 20, \phi = 20, \gamma = 50, \delta = 40$ となる。これらのパラメータは、モジュールが前向き、後向き、取り除く、などで広げる活性化エネルギーの量も決定する。より正確に、その前状態リストの偽となる命題のそれぞれのために、実行不可能モジュールが、その前任者に $\alpha$ を広げる。追加リストの中のそれぞれの偽となる命題のために、実行可能モジュールはその後任者に $\alpha(\phi/\gamma)$ を広げる。前状態リストの真となる命題のそれぞれのためにモジュールは競合者から $\alpha(\delta/\gamma)$ を取り除く。これらの要素は、内部の活性化の広がりが、現在の状況と目標によって、意味、影響と入力、出力を同じにすべきであるからこの方法により選ばれる。状況からの入力と、目標からの入力と、守られた目標による出力の比が、前任者からの入力と、後任者からの入力と、そのモジュールが競合するモジュールによる出力との比と同じである。直観的には、私達は、まだ真でない前条件を副目標とみなし、それは、まもなく真になる「前条件」であり、前条件は守られた副目標によって真になるとみなす。

これまでに述べてきたようなアルゴリズムは、扱わなければならない欠点を持っている。前状態リストと、追加リスト、削除リストの長さはモジュールへの活性化の入力と出力に影響する。特に、追加リストと前状態リストの中にたくさんの命題を持っているモジュールは、少ししか持っていないモジュールより、より多くの活性化エネルギーを持っている。それゆえに、全てのモジュールへの活性化の入力やモジュールから取り除かれる活性化は、1/n に重みづけされる。n は、(1) 前状態リストの命題の数 (2) 追加リストの命題の数 (3) 削除リストの命題の数である。

最後に、私達は、お互いに競争しあって、同じ前条件を使うモジュールや、同じ目標を達成するモジュールに活性化して欲しい。それゆえに、特定の命題のために取り除かれたり、広げられたりした活性化の量が影響を受けるモジュールで分けられる。例えば、その状況で真となる特定の命題 p がモジュールの前条件リストの中に、前条件を持っている全てのモジュールの間で $\phi$ に分けられる。おなじものは、目標と守られた目標の影響ばかりでなく、活性化の内部広がりも持っている。例えば、たくさんのモジュールがモジュール m の前状態を達成するとき、その命題のために後向きに m が広めた活性化 $\alpha_m$ はこれら全てのモジュールの間で分配される。一方で、この前状態が真となる他のモジュールが一つしか無かったとき、モジュール m は、それ自身の活性化レベル $\alpha_m$ によってモジュールの活性化レベルを増加させる。これが基本とする、一つの明示的な仮定は、前状態が、接続的な普通の型であるということである。二つの前条件の引き離しはたった一つの命題によって表されるが、それは、命題を真にする二つの competence モジュールような命題である。

# 4 結果

この論文で紹介されているアルゴリズムは、異なる数式のシステムでモデル化されている。しかし、このシステムは、行動選択の結果について、正確な予測ができず、あまりに複雑なので答えることができない。にもかかわらず、重要な質のある結果が、獲得されている。例えば、結果の論理出力や、ネットワークの大きさなどのパラメータの成長に伴う可能な遷移段階において、獲得される。私達は、いくつかの例となるアプリケーションを使う広い視野での実験を行うことにより、経験的にアルゴリズムを評価した。図 4 はシミュレーション環境のビットマップです。ネットワークはとても「広い」、とても「長い」、循環を含んでいる、リンクの局所的に高い接続、サブネットワークとの切断、壊れたモジュール、競合していたり相互に競合するモジュールなどの特徴があった。示される全ての問題は、パラメータの広い範囲で答えられる。最初の結果はとても約束されたものであり、それゆえに、よりシステマチックな研究が後に続いている。以下の節は、詳細に観察される結果を議論している。

## 4.1 プランニング能力

シミュレートされたネットワークによって選択される行為は、「jump-first think-never」となる振るまいを示すとはいわれない。ネットワークは、プランニング能力を示す。ネットワークは、実際に実

行に移す前に、一連の動作の影響をある程度「考慮」する。もし、現在の状況を、目標状況に変える、一連の competence モジュールが、存在するならば、前向き広がり (現在の状況から始まる) と後向き広がり (目標から始まる) による、蓄積の影響を通して、この一連の動作は、高く活性化される。もし、この一連の動作が、潜在的に、否定的な影響を暗に示していたら、抑制ルールによって、それは弱められるでしょう。

より明確に、行動選択の適切目標は、目標からの入力と、活性化の後向き広がりを通して、獲得される。状況の適切さと、適合的な振舞は、現在の状況と活性化の前向き広がりを通して獲得される。競合したり、相互作用する目標は、守られた目標と、競合するモジュール間の抑制による抑制を通して考慮される。加えて、行動選択に置ける局所的な最大値は、避けられ、活性化の広がりは、十分長く広がり続けることができるので、ネットワークは、より良い行動パターンに向けて進化することができるということを導く。最後に、計画が、現在の状況と目標の間の距離をより短くしようとしがちで、過去の活性化の広がりかたをなるべく残そうとするので、アルゴリズムが自動的に実行中の計画に偏る。加えて、大域変数は、人が、これらの異なる行動選択性質の間を円滑に平滑化できるように用いる、コントロールを与える。これらの結果の全ては、後の節で詳細に述べる。

ここでのプランの概念は人工知能分野の古典的なものとはとても異なる。ネットワークは、明示的な一つのプランによって表されないけれども、代わりに、相互作用するモジュールの活性化レベルが高くなることによりある行動をとるための「意図」や、「促進」を表現する。もう一つの重要な違いは、集中的にプログラムされた探索処理が無いということである。代わりに、オペレータ (competence モジュール) 自身が、活性化された一連のオペレータを選択する。これは、非階層的であり、良く分散配置された方法である。探索木は構成されていない、すなわち、ある行動を行った後の状況変化を構築する明示的な表現が無いということである。

まとめて、システムは、次のような探索木の欠点に苦しまされることは無い。その欠点とは、情報が、木のいくつかの部分に分散してしまうこと、木の大きさが指数的に増加してしまうこと、可能な計画だけの堅い表現しかできないなどである。加えて、活性化処理の広がりは、よりチープな処理だけである。もちろんこれらの利点はないわけではない。生産された行動選択は、人工知能分野で構築されたするどく恣意的なプランナーの行動選択よりも「合理的」ではない。一方で、後者のシステムは、自動エージェントに適応されたとき、もろさとのろさに苦しむ。ここで示すアルゴリズムについてとくにおもしろいことは、アルゴリズムが、一方の、適応性、速さ、適切性ともう一方の、思慮、合理性の間を調停するパラメータを与えることである。

## 4.2 目標指向

アルゴリズムは、エージェントのグローバル目標に貢献する行動を選択する。ネットワークのグローバル目標 g が与えられたとき、新しい活性化エネルギー $\gamma$ はこの目標を達成するモジュールに与えられる。これらのモジュールは、交互に副目標一つにつき、その副目標を真にしたりするモジュールの活性化レベルを増加させる。この活性化の後向きの広がりは、目標 g に貢献するモジュールが貢献しないモジュールより多くの活性化がなされるように注意する。加えて、異なる目標に貢献するモジュールはこれらの目標と意志のそれぞれのために活性化されそれゆえに、一つにだけ貢献するモジュールを好む。

もしエージェントが一つ以上のモジュールを持っていたら、「最も近い」目標に貢献するモジュールが好まれる。ここでいう「最も近い」とは、目標達成モジュールと状況適合モジュールの距離が一番短いことをいう。アルゴリズムは、ほとんど競合の無いモジュールも好む。例えば、もしエージェントが g1 と g2 という二つの目標を持っていて、g1 を達成するモジュールが一つ、g2 を達成するモジュールが二つあったならば、アルゴリズムは g1 を達成するモジュールを好み、それゆえに、g1 が初めに達成される確率は高い。これらのコメント全ては、副目標が目標と同様に扱われるので、副目標と目標の両方に適用できる。

その振舞は、$\gamma$ と $\phi$ の比を変化させることによって、選択に置いて多かれ少なかれ目標指向でなされる。例えば、もし $\phi = 0$ ならば、古典的な後向き連鎖が行われる (すなわち選択は完全に目標指向である)。一方で、システムは現在の状況からよ

り少ない情報しかえず、あまり反応的ではないし、現在に観察されているものや、近い将来に真になるようなものによってあまり影響を受けない。加えて、現在の環境状態が、行動選択に影響しないから、システムが遅くなる。理想的に、私達は、システムに主として目標指向であって欲しいが、面白い機会を利用して欲しい。これが、$\gamma > \phi$ を選ぶことによって獲得される。最も良い比率は、もちろん問題依存である。

## 4.3 状況適応

アルゴリズムは、現在の状況に、より適したモジュールを活性化する。これに責任がある処理は、環境の状態と、前任者 (次に何が真になるかの予測、計画を実装している前任者) に対して実行可能であるモジュールによって広げられる活性化エネルギーの広がりからくる活性化エネルギーの入力である。前の節ですでに述べたように、利点は、(1) システムが探索を軽視していて行動選択が速いこと (2) システムが機会を開発できること (環境によってより大きく左右される行動選択) の二つである。自動エージェントのための (2) の重要性はリアクティブシステムと呼ばれ、発展し、注目され、人工知能分野でも最近になって認識されるようになってきた。状況指向の性質は、多かれ少なかれ、パラメータ $\phi$ を変化させることによって、開発される。図 5 はパラメータ $\gamma$ と $\phi$ の比を変化させたときの実験結果である。

前向きに広がるルールは、モジュールが現在の環境の状況を与えられ、どのくらい「近い」か、実行可能かに比例して、現在の状況から活性化を受け取るということを行います。もしモジュールが本当に実行可能ならば (すなわち、全ての前条件が満たされていれば)、そのモジュールは実行可能に一番近い。実行可能でないモジュールのため、「近さ」は、実行可能なモジュールから、そのモジュールの前条件のそれぞれに向かうモジュール自身との道の距離の重みの合計に反比例する。例えば、これは、モジュールが p1 と p2 の二つの前状態を持っていたとします。そのうちの一つ、例えば p1 が与えられた現在の状況で真にならないので、このモジュールは現在の状態から比較的少ない活性化をもらう。したがってこのモジュールは「プラン」の一部になる可能性はより低くなる。

## 4.4 適応性

行動選択処理は、完全に「開かれて」いる。目標と同様に環境も、実行中に変化する。結果として、内部の活性化/抑制パターンと同様に、外部の入力/出力も修正された状況に反映して変化する。さらに、「プランニング」や活性化の広がりの間中の外部の影響はとても重要なので、プランは環境と目標からの影響や入力/出力が存在するかぎり、形成されるだけである。

この絶えま無い「再評価」のため、行動選択振舞は、簡単に予知しなかったり変化する状況に適応する。例えば、「pick-up-board」モジュールが活性化した後ならば、ボードはロボットの手の中には無い (ボードが滑べったから)。同じ competence モジュールはまだロボットの手にボードを持たせたいという competence モジュールからたくさんの活性化を受け取るから、もう一度活性化される。または、もし条件が真になるような二番目のモジュールがあれば、それが選択される (「pick-up-board」の活性化レベルが 0 にリセットされてるから)。掘り出し上手は、適応能力のもう一つの例である。もし、目標や副目標が突然満たされたら、この目標に貢献するモジュールはもはや活性化されない。これらの実験の全ては、成功でシミュレートされた。そのような予期しない事象は、システムが実行中のプランを「捨て」新しいものを「構築」しなければならないということを意味しないということに注意してもらいたい。実際にシステムは絶えま無く、異なる相互のものを比較する。いくつかの状況が変わったとき、これは、現在のプランよりももう一方のプランの方が魅力的 (活性化的) になるかも知れない。

それが、システムが毎ステップごとにプランを新しくしているという場合ではないことも注意してもらいたい。活性化の広がりの「歴史」も、活性化レベルが毎ステップごとにリセットされないから、行動選択の振舞に重要な役割を果たす。だから、ちょうど目標指向と状況指向の関係がトレードオフであったように、私達はここで、適応と実行中のプランの偏りの間のトレードオフを持つ。人は、パラメータ $\gamma$ と $\phi$ と $\pi$ の特定の比の選択によって二つの流れを調停できる。

図 6 のモジュールの例を考えてください。初期に観察される状況は $(a, x)$ であり、目標は f であ

る。モジュール「one」が活性化された後、私達は、w をグローバル目標に追加する。$\gamma$ と $\phi$ が $\pi$ と比較して比較的小さいとき、内部の活性化の広がりは環境の状態と、グローバル目標からの影響より大きな衝撃がある。それゆえに結果である行動選択振舞は、より適応されなくなる。明確に、それは、状況から目標までの道は目標 w のほうが短いけれども、システムは目標 f を計算し続け、f が達成されたら、目標 w を計算し始める。再び、適切な解答が中間のどこかに横たわる。パラメータは、システムが常に異なる目標の間を飛ばず、機会を開発し、変化する状況に対応するように選択される。

最後に、間違った我慢とよばれる、アルゴリズムが見せる適応のもう一つの型に注目しよう。これは、アルゴリズムの分配された自然のまとまりである。一つのモジュールが他のよりも重要であるので、ネットワークはまだあまり嬉しくない前状態下で成し遂げることができる。まだ可能である competence モジュールやネットワークを削除することは可能である。

## 4.5 実行中のプランへの偏り

アルゴリズムは明示的な偏りのメカニズムを例証する。それは十分緊急なので何か違うことを始めなければならない時を除いて、実行中の目標と副目標に貢献するモジュールを好む。偏りが現れる主要な理由は、モジュールが活性化されるごとに活性化レベルが初期化されないということである。結果として、過去の活性化の広がりの歴史は、行動の選択において役割を果たす。特に、状況と目標の影響が中間の活性化レベルに対して比較的小さいときに役割を果たす。しかし、たとえそのような場合でなくても、アルゴリズムは、実行中のプランに対して偏りを示す。より明確には、それは、二つの偏りの型 (水平方向、縦方向) を例証する。

1. 水平的偏り

   行動選択アルゴリズムによって示される最初の偏りの型は、現在の目標に貢献する行動を好むということである。モジュールの集合で表される図 8 を見てください。初期状態は $S(0) = (a, x)$ であり、グローバル目標は $G(0) = (f, r)$ である。one から five が目標 f を達成するために必要なモジュールで、一方 one' から five' は目標 r に貢献するモジュールである。

   シミュレートするとき、このネットワークは f に貢献するモジュールと r に貢献するモジュールの間の 4 つを飛び越えたり、もどったりしない。代わりに一つの目標を計算し始め、それが終れば、他の目標を計算し始める。この場合は、one か one' のどちらかのモジュールが選ばれるときであるから、目標への道の距離は、他の道の距離よりも短い。それゆえに、後向きの活性化の広がりはより大きな影響を持ち、始めた道が最初に終ることを確かめる。現在の状況から目標への道が、長く成長するにつれて、閾値は、この影響を獲得するため、増加されなければならない。

2. 垂直的偏り

   偏りの二つ目の型は「悩ませる」目標 (同じ目標を含む副目標) に貢献する行動を好む。図 10 のモジュールを見てください。環境の初期状態は、$S(0) = (a1, c1, e1, g1, a2, c2, e3, g2)$ であり、目標は $G(0) = (k1, k2)$ である。

   再び閾値が十分大きかったならば、このネットワークは最初に二つの目標のうちの一つに貢献する全ての行動を実行し、次に二つ目の目標をやる。この理由は、一旦モジュールの前任者が活性化し、ノード自身が環境の状態からより多くの活性化エネルギーを受け取る。それゆえに、そいつは、そいつの残っている前任者により多くの活性化を広げる。

   前の節で述べたように、大域変数の比を適切に選ぶことによって環境と目標を変える点では、多かれ少なかれある程度の慣性を与える。特にとても動的な環境に置いて、システムがゆっくり適応することは必要かも知れないし、そうでなければ、システムは何もなさないかも知れない。

## 4.6 目標競合を避ける

行動の悪い順序は目標を達成するために必要な行動の数が劇的に増えるものや、解答が発見されるのを妨げるものである。それゆえに、どんな行動選択アルゴリズムでもある程度競合行動の間の

調停を行うことができる。守られた目標をもとに戻してしまうネットワークのモジュールが要素 $\delta$ によって弱められる。もし $\delta$ が (特に $\gamma$ や $\phi$ に比べて) 十分大きければ、行為の結果はグローバル目標を守る。

副目標 (やモジュールの前状態) に置いても同じである。全てのモジュールは真である状態をもとに戻すモジュールの活性化レベルを下げる。再び、行動選択振舞に置いて「副目標」が守られているから目標が競合するという結果は避けられる。これがどのように起きるのか例証するために、私達は古典的なブロック世界の異例を再実装しましょう。図 12 はこの問題を挿絵で示したもの。図 13 はいくつかの competence モジュールがこの例に巻き込まれているところである。

図 14 と 15 は得られた結果を示す。最初の実験に置いて $\delta$ は $\phi$ よりはるかに大きいが、$\gamma$ と同じ値にした。結果、「clear-b」状態のための「stack-b-on-c」による「stack-a-on-b」の抑制が現在の状況による活性化よりもかなり重要であるということになった。このため、モジュール「take-a-from-b」は「stack-a-on-b」が目標であるにも関わらず、それを消してしまった。もし $\delta$ が十分に大きくなかったら、システムが a が b に積まれているところから始まるので目標「a-on-b」を満たす性急度が「clear-b」を避ける性急度に勝る。しかし、それはまだ状況をもとに戻し、二つの目標を獲得する事が可能である。なぜならば、守られた目標からの影響があまり高くないので、システムが達成された目標「a-on-b」をもとに戻すのを避けられないからである。再び、全く目標競合に対的にしないものとあまりに堅すぎて達成した目標をもとに戻せないでデッドロックになるものとの間で平衡が見られなければならない。

### 4.7 思慮深さ

ネットワークは閾値 $\theta$ によって決まる局所的な近傍しか先を見ない。振舞は閾値 $\theta$ を増加させることによって多かれ少なかれ「思慮深く」なりうる。これは、明確な行動が選択される前に活性化を広げる処理を長い間続ける。そのように閾値は、ネットワークが先を読むことを助け、それゆえに、活性化レベルの局所的な最大値を避ける。例えば、上記に述べたブロックの世界に置いて、最初にモジュール「stack-a-on-b」が最も高い活性化レベルを持っている (現在の目標と状況の両方から入力を直接受け取るから)。閾値は、ネットワークがモジュール間の競合を考慮し続けられるように、すぐにこのモジュールが選択されることを避けるため、十分に高くしておかなければならない。

理想的に、私達は閾値をとても高い値に設定しておきたい (例えば、全ネットワークの活性化エネルギーの総和にするなど)。これは、「最適な」行動が選択されるように、活性化の広がりの処理が十分長い間続けられることを保証するでしょう。閾値を高くすることの問題は、まず始めに行動選択処理にとても長い時間がかかるということ、二つ目に、エージェントが遠い先の未来を考慮しようとして泥沼にはまるという結果があり得るということである。これは、予期できない環境に置いては最も無駄になるものである。それゆえに私達は、エージェントに近い未来だけを見てもらう。特定のアプリケーションで先を読んでもらいたい量は閾値に固有の値を選ぶことによって獲得されるでしょう。

### 4.8 速さ

思慮深いことと対極の位置にあるのが速さである。行動選択振舞は上記で説明した閾値 $\theta$ を変化させることで速くなるでしょう。しかし、行動選択の結果は「思慮深い」ものではなくなり、思慮深くなくなるということは、目標指向でも、状況指向でもなくなり、競合目標をあまり考慮しなくなり、実行中のプランに偏らなくなるということを意味している。にもかかわらず、素早く反応し、思慮深くしないことが時々重要になる。好運なことに、アルゴリズムがあまり複雑でないので、思慮深さをあまり犠牲にせずに、速さを獲得することができる。

## 5 討論

### 5.1 アルゴリズムの複雑さ

アルゴリズムは探索処理と似たような計算を行っているので、古典的 AI 探索と同じ問題を持っているのではないかと考えるかも知れない。より明確には、たくさんのモジュールがプランが大きく

なるのに巻き込まれるにつれて、効率が悪くなるのは必然的である。以下は、私達が、この問題にぶち当たらないと信じさせるコメントである。

- アルゴリズムが成し遂げる計算はそんなにコスト高ではない。実際に、AIの探索の概念よりも、マーカーパッシングアルゴリズムに似ている。システムが一つの行動を選択するためにとるステップの最大個数は、閾値の線形関数によって多様化されるネットワークの「広さ」(実行可能モジュールと目標達成モジュールの距離) に拘束される。システムは探索木を構築せず、現在の仮説的な状態と部分的なプランを維持しない。加えて、それは、一つの道が解を見付けられなくても一つのプランからもう一つのプランへ円滑に移行するので迷う必要が無いから同時に異なる道を評価する。

- 全てのタイムステップでシステムは、完全には「プランの再構築」をしない。アルゴリズムは行動がとられたときはいつでも、活性化レベルを0に初期化するわけではない。これは実行する最初の行動を決定するのにある程度時間がかかるかも知れないが、それ以降はネットワークが特定の状況と目標の集合に偏るということを暗に示している。これは、後続の行動が選ばれるのにはあまり時間がかからないという意味であり、特に目標や現在の状況の点でほとんど同時に変わらないときは時間がかからない。

- 私達は実際の自動エージェント (ロボット) にとって、ネットワークは「広く」というか「深く」成長すると信じている。なぜならば、典型的にエージェントは、とるべき行動がより多く要求されるという仕事/目標を持つ代わりにより多くの仕事/目標をもつであろうからです。接続されない大きな部分集合もネットワークに存在するかも知れない。結果として、システムの効率はあまり影響しない。たとえ状況に合うものから目標を達成するものまでのいくつかの道が、とても長くても、システムはまだ行動についてくる。なぜならば、それは、活性化レベルの収束を待たず、時間とともに閾値を減少させるからである。しかし、選ばれた行動が最適でないかも知れない。

- 同じ、単純な活性化の広がりのルールは、それぞれのモジュールに適応される。加えて、モジュールの間には局所的で固定されたリンクしか無い。これは、並列実装のために、面白い機会を開きますが、それは、思慮深いスピードアップを暗に意味します。

## 5.2 変数の問題

アルゴリズムは古典的な変数と変数通過と協力しない。実際のところ、もしそれらが導入されたら、たくさんの利点が失われるでしょう。例えば、古典的なアルゴリズムには変数が無いから、確実にたくさんの探索が失われるというのが一つの理由です。変数が無いことによるはじめの影響は、人が変数を使って目標を明確化できないということです (例えば go-to-location(x,y))。二つ目の影響は領域の全てのモジュール/オペレータを前もって見積もっておかなければならないということです。

私達は「対象指示の表現」を使うことによってともに変数の必要性を避けようとしている。すなわち、瞬時の環境の適切な特性を表記するために指示機能の側面だけを用いるのである。環境におけるものは、内部に表現された目的との関連とエージェントの事情である。例えばモジュール「spray-paint-self」は「the-sprayer-I-am-holding-in-my-hand」という一つのパラメータに実証されなければならない。これのために、世界に導入される全ての新しいものが新しいオペレータ/モジュールを生み出す必要は無い。オペレータとオブジェクトの使い果たされた組み合わせは無い。

指示的関数の側面の考えは、特に自動エージェントにとって面白いものとなる。なぜならば、それは、どの知覚が配られ得るかということについてのより現実的な仮定を作るからである。特に、それは、認識システムがオブジェクトのIDと正確な位置を生み出すということを要求しない。変数が無いことは、人がシステムとやりとりするために用いることができる言語を束縛するが、とても強い方法に置いてはそうでもない。それが求める全てのことは、エージェントに何をすべきかを伝える方法について考える新しい方法である。より明確に、特定の目標について人間はオブジェクトに唯一の名前を用いない。代わりに、関係するオブジェクト上で、指示的な拘束や機能的な拘束の点

に置いて目標は明確化される。例えば、人はエージェントに位置 (x、y) に行けとはいわないが、人はエージェントに目標は玄関の位置にあるという。

にもかかわらず、現在のアルゴリズムに変数を取り入れることが必然的に不可能というわけではない。例えば、competence モジュールの複製は毎時間ごとに生み出されるわけではないし、私達は変数値の異なる集合を備えたモジュールを実証しなければならない。しかし、これは今より機械的なものを求めるでしょう。私達がとっているアプローチは、私達が指示的で機能的な表現だけを用いてどこまで行けるかを見ることです。アルゴリズムは私達が変数の必要性を感じたとき、完全に育てられた変数に拡張するだけでしょう。

## 5.3 緊急コントロールの流れ

この論文で議論されているアルゴリズムと古典的なプランイングとの間の主要な違いは、それらがそれぞれ一連の行動を生み出す仕事を組織する方法にある。古典的なプランナーは前もってプログラムされた、一連の集中した探索処理を用いる。一連の行動は実行されたものがシステムを目標状態にするとき内部モデルにしたがって探索される。代わりに、このアルゴリズムでは、通常の支配構造は、特定の行動が達成されたとき現れる。つまり、環境の状態に反応において、完全に分配された方法で選択された一連の動作を設立する動作 (モジュール) と、グローバル目標の間の相互作用のダイナミクスである。

situated agent の個々の competence モジュールの発展の下にある緊急の機能の同じ「哲学」はここで、動的支配構造の問題に適用される。その 2 つは、複雑な環境において機能しなければならないシステムの設計のしかたのより一般的な観点の例であり、ときどき「ボトムアップ」構造と言われる。キーアイデアはシステムを構成する部品の間のいくつかの局所的で激しい相互作用のグローバルな副作用として機能が現れることである。グローバルシステムの機能はどのように獲得されるべきなのかを明確に示すルールやプログラムはない。代わりに、ダイナミクスは環境の状態の反応の中で構造を形づくるものと交換する。複雑なシステムの設計に対するボトムアップアプローチの利点は、

- 結果がより自然で優美である。その理由は、ボトムアップシステムにおいて、とってかわったダイナミクスは、なんらかの要求をする環境の状態に反応する点において機能の緊急性を作るということである。もうひとつの理由はボトムアップシステムは、より連続的で濃い振舞いをみせるということである。それらは、環境を、(激しく) 異なる行動が求められる状況の分離した型に分類しない。

- 結果がより柔軟である。階層的に組織された機能は、典型的にとても堅い。一つの理由は状況や処理の状態が変わるときはいつでも、この変化は、これらの変化が現実の違いを暗に示すトップノードから、すべての抽象的な階層を通して、階層のボトムへと落ちるように広がらなければならない。もうひとつの理由は、どのような行動をシステムがとるべきかということをプログラムできるように、すべての可能な状況があらかじめ予想されなければならないということである。緊急機能に頼るシステムにおいて、環境との相互作用は直接的に、すべてのボトムレベルの部品を通して、並行に起こる。

- 結果はより頑健である。緊急機能をもつシステムは誤りに強い。システムは、システムが処理している環境が変化したり、システムにおける部品が失敗するとき、よりもろくなる。ひとつの理由は、システムが適切に機能していないとき、すべてのシステムを落とすような中心的、危機的部品がないことです。もうひとつの理由は、もしひとつかそれ以上の部品が壊れたとき、システムは円滑な実行をみせるということである。そして最後にこれらのシステムはしばしば重複した部品を持っているので、より頑健な結果になる。上記に述べられているアルゴリズムにおいて、たとえば、人は新しい competence モジュールを追加したり、存在するモジュールを修正したり、消去したりさえできる。ダイナミクスは、自動的に、新しい状況に適応されるし、まだ可能性があるどんなことでもできる。対照的に、古典的プランナーは大きく、すべての部品が完璧に動くという事実に依存している。

緊急機能をもったシステムの不利益点は結果と

なる振る舞いがあまり予期されず、これに関して、望まれるグローバル機能がどのように獲得されるかあまりわからないということである。

### 5.4 限界と拡張

今まで見てきたアルゴリズムはたくさんの限界がある。ひとつは、行動選択のループが突然現れるかもしれないことである。それらは、それらは、とてもめったに起こることはないが、それは、システムが過去に行ったことの歴史を維持しないという事実から広がる。そうして、それは、行動選択において再びどんどん同じ「間違い」を行う。それらの行き詰まりに対する解答が作れるかどうかは疑わしい。仮説は、実環境において、状況と目標がとても小さい経過時間 $\Delta t$ 時間後になんらかに変化するということが適用されうる。これは、活性化の広がり方を変化させ、それゆえにネットワークを行き詰まりから解放する。もし、私たちが行き詰まりを回避することを主張したら、これは、注意深いパラメータの選択によって保証されない。しかしひとつのとても単純な解答システムのいくつかの不規則性によって導入される。もうひとつの解答は、一番目のネットワークの可能なループを監視するために二つ目のネットワークを使い、これが起こったらいつでも行動をとることかもしれない。最後に私たちは、いくつかか、すべてのモジュールにいくらかの抑制機構を実装することができる。この機構はモジュールが活性化されるごとに、そのモジュールは近い将来に活性化されることはないだろう（すなわち、いつも異なるローカル閾値をもつ）ということに注意する。

もうひとつの問題は、明確なアプリケーションが与えられたとき、人が、望む行動選択振る舞いを生産するグローバルパラメータの値をどのように与えたらよいかが明確でないことである。パラメータは、行動選択振る舞いの効果や性格を大きく決定する。それらは、全ての問題の範囲が、目標指向、状況指向、速さ、適応性など、で異なる程度を求めるからということばかりでなく、ネットワークの大きさや構造もこれらの性格を決めるからということにも依存する問題である。例えば、とても大きなネットワークに伴うアプリケーションにおいて、閾値は、同じ結果を得るために、より高くしなければならない。その瞬間、私たちは、一連の経験中パラメータを手動調整する。私たちは初めのネットワークの結果を見て、ユーザによって明確化される行動選択の性格を獲得する為にパラメータを調整する competence モジュールの二つ目のネットワークを構築することによってこの仕事を自動化するようにする。

最後に、現在の理論の拡張は、役に立つように思えるし、現在の研究の注目点である。一つは、行動選択の並列実行の導入で、もう一つは、異なった抽象レベル上で行動選択を導入することです。まだもう一つは、より、現実的でおそらくノイズだらけのセンサーデータを取り扱う。発展している解答は、現在の哲学と、それが持っているメリットと共鳴する。

## 6 結論

私たちは、古典的プランニングも現在人気のある状況行動システムも自動エージェントの行動選択問題に満足行く結論を出していないということを討論した。私たちは、後者の、分散配置、物理的グラウンディング、相互作用といった「美しさ」と、前者の目標を扱う一般的能力を組み合わせるという問題に対する新しいアプローチを示した。