Toward Defining Domain Complexity Measure Across Domains

Katarina Doctor[1], Christine Task[2], Eric Kildebeck[3], Mayank Kejriwal[4], Lawrence Holder[5], Russell Leong[2]

[1] Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, D.C.
[2] Knexus Research Corporation; Springfield VA; [3] University of Texas at Dallas, TX;
[4] USC Information Sciences Institute, CA; [5] Washington State University, WA;

Abstract

AI systems planned for deployment in real world applications are frequently researched and developed in closed simulation environments where all variables are controlled and known to the simulator. Transition from these simulators and testbeds to more open world domains poses significant challenges to AI systems, including significant increases in the complexity of the domain and the inclusion of real world novelties where the open world environment contains numerous out of distribution elements that are not part in the AI systems' training set. We propose here a path to a general, domain-independent measure of domain complexity level. We distinguish two aspects of domain-complexity: intrinsic and extrinsic. The intrinsic complexity is the complexity that exists by itself without any action or interaction from an AI agent, who is performing a task on that domain. This is an agent-independent aspect of the domain complexity. The extrinsic domain complexity is agent- and task-dependent. Intrinsic and extrinsic elements combined capture the overall complexity of the domain. We frame the components that define and impact domain-complexity levels in a domain-general light. Domain-general measures of complexity could enable quantitative predictions of the difficulty posed to AI systems when transitioning from one testbed or environment to another, when facing out of distribution novelties in open world tasks, and when navigating the rapidly expanding solution and search spaces encountered in open worlds.

1. Introduction

When designing AI that can operate in an open-world setting, it is important to be aware of the complexity level of the domain for which the AI system is built and the complexity level of the domain where it will be applied or transitioned.

As one example of an AI agent transitioning from a closed to an open-world domain, (Wilson, McMahon, and Aha 2014) developed a motion and task planning system for autonomous underwater vehicles (AUV) using a common simulation platform. In general, agents in simulated environments navigate a much smaller set of possible states and perform deliberative-reasoning search tasks over a much smaller set of possible-state action paths than what happens in the open world of a non-simulated, real environment. Wilson et al. then moved from development to deployment. If one takes the AUV agent, which has been built within a less complex simulated environment, installs it on a robot, and lowers it off the side of a ship into the real world, then it will have significant problems correctly navigating in that real-world state space.

In the ever-changing open-world domains, there are a plethora of novelties that will have effects on an AI agent that's been trained in a closed-world setting. The overall complexity level also changes when transitioning from a closed to an open world. Transitioning from a closed to an open world does not necessarily mean transitioning from a simple to a complicated or complex system. However, understanding the boundaries of the domains we are facing will help with adapting to novelties in an open world. Knowing the complexity level of the domain will enable robustness in an AI agent when performing a task in the domain of interest.

The core concept of complexity level transcends the boundaries of a domain. In this paper, we define a framework for estimating the level of complexity of domains in a transdisciplinary, domain-general light.

2. Motivation and Background

The Science of Artificial Intelligence and Learning for Open-world Novelty (SAIL-ON) Defense Advanced Research Projects Agency (DARPA) program (Senator 2019) is designed to inspire the AI science community to develop robust agents that are ready for the challenges of open-world domains where a plethora of novelties will be present. The SAIL-ON program is divided into two groups: A) Those that generate novelty for their domains; and B) Those that develop agents that detect and characterize novelty and adapt to domain distributions with novelty present. The setup of the program is designed to encourage developing domain-independent, robust agents that are ready for the open world. The novelties are generated for testing scenarios outside of the training set distribution.

During the 24-month period of the program thus far, every six months, there has been a test evaluation of the agents for detecting, characterizing, and adapting to novelty distributions. In each test period (Phases 1, 2, and 3), the generation

of novelties for each domain in the program has been designed to incrementally increase their complexity level by adding increasingly more complex attributes to the novelties. Further, each novelty level has three difficulty levels: easy, medium, and hard. These difficulty levels refer to novelty detection and adaptation to it. The novelties are categorized in a hierarchy representing the fundamental elements that make up an open world, and every element within an open world must, by definition, have characteristic attributes and be represented in some way. The open-world novelty hierarchy levels are: object, agent, actions, relations, interactions, rules, goals, and events (Table 1). We are treating these hierarchy levels as part of the components to consider when estimating the complexity level of domains, which we describe in Section 5 of this paper.

Open World Novelty Hierarchy			
Single Entities	Phase 1	1	**Objects:** New classes, attributes, or representations of non-volitional entities.
	Phase 2	2	**Agents:** New classes, attributes, or representations of volitional entities.
		3	**Actions:** New classes, attributes, or representations of external agent behavior.
Multiple Entities		4	**Relations:** New classes, attributes, or representations of static properties of the relationships between multiple entities.
		5	**Interactions:** New classes, attributes, or representations of dynamic properties of behaviors impacting multiple entities.
Complex Phenomena	Phase 3	6	**Rules:** New classes, attributes, or representations of global constraints that impact all entities.
		7	**Goals:** New classes, attributes, or representations of external agent objectives.
		8	**Events:** New classes, attributes, or representations of series of state changes that are not the direct result of volitional action by an external agent or the SAIL-ON agent.

Table 1: Open world novelty hierarchy levels

Complexity levels of domains are estimated in different ways for different domains, and it is challenging to generalize complexity estimation across multiple diverse domains. The structure of the SAIL-ON program highlights this need, where we would like to be able to compare both domains and their complexity levels as well as estimate the complexity level of generated novelties.

Complexity versus difficulty Complexity and difficulty are different mental operations. The complexity level of a domain will affect the difficulty of detecting novelty, characterizing it, and adapting to it. It will also influence the difficulty of generating novelty. "Complexity" is the description of a state, the space of possibilities, whereas "difficulty" relates to the challenge posed by a specific novelty within this space of possibilities.

3. Benefits of knowing domain complexity

In the Wilson et al. example, the simulated environment assumed that there was a 1-1 mapping between perception and the state space; i.e., that a change in perception was due to a meaningful change in state. Instead, in real-world contexts with noise and complex factors related to environment, there may be many different perception values that might all indicate effectively the same state for the goal reasoner. The real-world perception state space is effectively much larger than the space over which goal reasoning is actually performed. An AI that was developed and tested on a simple, simulated perception space will tend to fail when one puts it in the open sea, incessantly reporting discrepancies between its sensors and its model of the world, potentially getting trapped in a loop of constant replanning. As a result, the robot can freeze, drift off course, and lose contact with its humans. This can be a very expensive problem. Addressing this issue required the development of new logic for dealing with the true complexity level of the perception space. Perception information was processed with bounding boxes to reduce its complexity sufficiently so that the existing goal-reasoning and planning logic could operate over it effectively. This measure allowed the robot to operate successfully in the open water. It is important to note that this issue was task-independent. There was effectively no non-trivial task that the robot could execute successfully before its algorithm had been modified to address the increase in domain complexity as compared to the simulator domain.

Outside of the context of real-world robotics with deliberative reasoning (Ingrand and Ghallab 2013), the domain complexity problem arises in other application areas as well. In game AI, the performance of Monte-Carlo Tree Search (MCTS) will depend on the size and complexity of the game tree; if the space of possible states and actions becomes excessively large, then the probabilistic exploration of the tree will have a higher probability of failing to sample the optimal paths, and then the agent may select moves which are sub-optimal, poor, or even absurd. If the complexity of the game is correctly understood during AI development, then modifications can be made to improve the algorithm performance, such as using domain knowledge to bias search in games with large branching factors (Chaslot et al. 2008).

More dangerously, the problem arises in self-driving vehicles. (Claussmann et al. 2017) exhaustively surveys, categorizes, and evaluates diverse approaches to autonomous driving, but limited to only highway environments, and with respect to only 8 simple maneuvers (such as change lanes or exit). This analysis does not cover important complexities in the highway domain, such as encountering an obstacle in the road or another vehicle merging into your lane without seeing you. Furthermore, it will obviously not apply to city street environments, which have far more states and complex transitions. Techniques that perform well in the simple, limited, highway domain may have very different properties and limitations than techniques that excel in a more open, real-world environment, and failing to understand the impact of domain complexity on algorithm choice could lead to sub-optimal decision-making and potentially fatal consequences.

The issue of correctly addressing real-world domain complexity also arises in data science contexts. In medical applications, explainable machine learning (ML) systems are increasingly deployed to assist test analysis and decision-making. (Holzinger et al. 2017) Systems that were designed for simple data distributions over one or two educational-benchmark datasets will fail or become extremely sensitive to data preparation and parameter tuning; as the feature set grows more complex, evaluation becomes more rigorous, or the data distribution becomes more diverse and heterogeneous. Again, systems developed on simple, toy research domains, when moved to much larger open-world domains without consideration for the change in complexity of the problem definition, will have failures with real-world consequences. (Holzinger et al. 2017)

4. Perspectives on complexity from different disciplines

We briefly review existing approaches to understanding and measuring complexity as it applies to relevant computational disciplines: Classical AI, Data Science, Computational Complexity, and Systems Research. These existing perspectives are important to take into consideration, but none provide the comprehensive, domain-general complexity level necessary to support the development and transition of AI to open-world domains.

Classical AI (Hernández-Orallo and Dowe 2010) uses the term "environment complexity" to refer to increasingly complex classes of domain/tasks pairs. Their thesis is that more intelligent agents are able to succeed in more complex environments. Their complexity level increases dependent on the domain's number of possible states, number of transitions (agent actions that change the state), and the difficulty of reaching the objective (in general, winning the game). The more possible states and the more difficult-to-select actions to navigate optimally through them, the more complex the problem.

Relating domain complexity to the state transition graph is a common approach in classical AI systems. (Ingrand and Ghallab 2013), (Chaslot et al. 2008), and (Claussmann et al. 2017) give examples of agents whose performance is, in fact, dependent on the environment complexity as used in Hernandez-Orallo. These systems are sufficiently intelligent to solve problems of a certain complexity, but may fail as that complexity increases. (Pereyda and Holder 2020) proposed a theory for measuring complexity by taking a resource-requirements approach, focusing on the three spaces: task, solution, and policy, relating complexity to the minimum description length of agents necessary to achieve different levels of performance on the task.

While these approaches are relevant to domain complexity in our context, they focus on task-specific complexity and do not explicitly address intrinsic domain properties, which may impact solution performance independently of the task. A domain-general complexity level needs to take into account a wider set of components in order to fully support development for open-world domains.

Algorithmic and Computational Complexity Whereas Classical AI considers the difficulty of agents navigating iteratively to a desired state, Computational Complexity theory explores the fundamental limits of efficiently solving for a correct solution to a well-defined computational problem. Some problem classes cannot be solved with guaranteed correctness under a certain asymptotic time limit (with respect to the problem size). In general, computational complexity is not suitable for addressing the issue of domain complexity in AI contexts; it considers problems that can be defined as questions with a single correct answer and the asymptotic time complexity of finding that answer in the worst case. AI systems that operate in real-world contexts often address problems with no clear, single solution and need to perform "well enough," rather than optimally, on pathological cases, and their practical performance is best measured in clock time rather than asymptotically ($O(n) = 3n$ minutes may be acceptable, whereas $0(n) = 3n$ days may not be). As an example, (Stephenson, Renz, and Ge 2020) explores the computational complexity of the Angry Birds game under different contexts. However, the question they address is: "Does there exist a single strategy that always results in all pigs being killed?" This is a hard problem with exponential time complexity, but it does not need to be solved in order to build an AI agent that can play the game reasonably well. In general, computational-complexity classes do not capture domain-complexity levels at a granularity relevant for supporting an AI agent navigating a specific, real-world environment while meeting objectives sufficiently well, in real time.

Data Science Data science focuses on fitting conceptual models to input data, allowing users to make predictions about new data points. Unlike classical AI, there is no agent that can take actions to interact with its environment; instead, the domain consists of a feature set and a distribution of data points across the feature space, defined by that feature set. Complexity is determined by the difficulty of fitting meaningful, accurate models to these distributions, such that they support correct predictions on new data. Properties such as the size of the feature space, the sparsity of the data, and the shape of the distribution impact the difficulty of this problem. (Remus and Ziegelmayer 2014) considers the context of computational linguistics. They consider four task-independent domain complexity measures, focusing on the sparsity (word rarity), feature set size (word richness), and the distribution of the input data (entropy and homogeneity). They find a strong correlation between domain complexity and the performance of a standard ML classifier on the data. Similarly to Hernández-Orallo's observations on Classical AI, as complexity increases, performance decreases. These general ideas are broadly applicable as sources of complexity in problem solving as we will discuss in Section 6. However, by themselves, they do not comprise a general definition for domain-complexity levels.

Systems Research Systems research considers complexity in the form of challenges that arise when organizing

multiple interacting components, whether those components are team members, organizations, industrial production systems, software modules, and/or even elements of programming languages as interpreted by a compiler. In these contexts, patterns of dependencies between components are a key factor in the complexity of the problem. Large webs of inter-dependencies require significantly more computational time, or human cognitive load, to consider fully and develop optimal solutions. If the inter-dependencies can be arbitrarily complex, then the problem, in general, may be computationally hard (e.g., the knapsack problem). However, most real-world problem instances are tractable.

Various tools have been developed to help visualize, measure, and address complexity in system design contexts (Štuikys and Damaševičius 2009) (Jung, Sinha, and Suh 2020) (Prnjat and Sack 2001) (Kim 2014).

This work relates to our problem of domain complexity, specifically in the case of domains that contain multi-component systems. For example, when an agent must navigate a multi-agent environment with a large number of external entities, it can cause a combinatorial explosion in the set of possible states and transitions necessary to capture the interactions of those entities, resulting in a dramatically larger domain. This can be an important facet of domain complexity; however, there are many other sources of complexity that do not originate from interactions with multi-component systems. Additional components are necessary to capture the complexity of scenarios, such as single agents navigating inclement environments, ambiguous perception data, or challenging game objectives. System complexity metrics do not, by themselves, comprise a general-domain complexity level.

In this paper, we frame the components that define and impact domain-complexity levels in a domain-general light. We organize these components into groups. We then describe methods for representing these components in a measurable form for estimating domain-complexity level.

5. Framework for the components that define domain-complexity level

This framework is for defining the components to consider when estimating domain-complexity level. There are two parts when determining the complexity of a domain: intrinsic and extrinsic. The intrinsic complexity is the complexity that exists by itself without any action or interaction from an AI agent, who is performing a task on that domain; it is an agent-independent aspect of the domain complexity. The intrinsic components that define and impact the domain-complexity levels are further grouped into ”environment space” and ”task solution space.” The extrinsic complexity of a domain is agent-dependent and contains the ”performance space,” the ”planning space,” and the ”skills space.”

Considering the complexity level only from an intrinsic, agent-independent perspective, or only from an agent-dependent perspective, where we only observe the complexity level of what an agent is doing, would result in a skewed metric of the complexity level. Therefore, we need to look at both the intrinsic and extrinsic parts of the domain complexity in order to get a balanced metric. We took into consideration the open-world novelty hierarchy levels described earlier in the ”Motivation” section as part of the components to consider when estimating complexity level. These hierarchy levels are: objects, agents, actions, relations, interactions, rules, goals, and events (Table 1). We differentiate between novelty theories, ontologies, and categories, where theories can describe anything that is conceptually possible, ontologies classify elements that are realistic, and categories represent a subset of elements that are practically important and scientifically useful for open-world domains. Different tasks in different domains may or may not be affected by all the hierarchy levels in the same way, as shown in Figure 1.

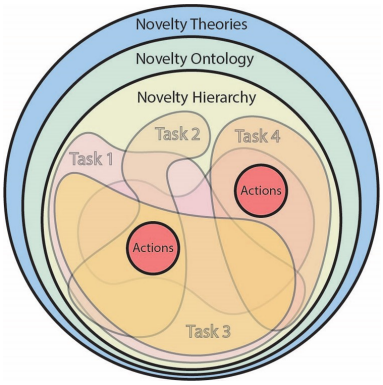

Figure 1: Venn diagram for open world novelty

We list these components into groups that define the intrinsic and extrinsic domain-complexity level. Each of these components has task-dependent and task-independent parts that we do not list separately here.

Note that these components will not be present for every domain. Also, some tasks will not interact with some of these components, even if they exist in the domain.

5.1 Components that define intrinsic domain complexity

The intrinsic, agent-independent complexity is structural and can have many relevant levels of description for computer simulations and for the natural world. There are task-independent and task-dependent parts of the intrinsic domain complexity. The task-independent part is the input, and it is defining the structure of the domain complexity. The task-dependent part is based on which aspects of the available complexity in the domain are relevant to, or utilized in, a given task. The elements contributing to intrinsic domain complexity can be divided into the 'Environment Space' that includes all elements of the task environment and the 'Task Solution Space' that includes only those elements relevant to completing a given task within that environment.

Environment space

The environment may be composed of parameters/variables, data schema, tokens/observations, scale size, objects, states,

or agents internal to the system. The categories of the novelty hierarchy (Table 1) represent the elements of an open-world environment space. The environment is composed of discrete entities (objects and agents), and these entities can have static relationships to each other (relations) and dynamic interactions (interactions), which in turn can combine to create complex phenomena based on multiple relations and interactions, such as events. The complexity of the environment increases as the number of elements in each category, and the number of distinct attributes and representations for each element, increases.

Single Entities: The fundamental elements for an environment are the single, discrete entities present in the space. In action domains, this would be the number of unique objects and agents in a task environment, (e.g., blocks, tools, and enemy units in Minecraft). In perception domains, this would be the number of classes (supervised learning) or number of clusters (unsupervised learning). Each of these categories can be expanded to include more discrete classes, and instances of each class can have an increasing number of attributes and representations. In the AUV example, an underwater simulator may have only one type of fish with a single color and single swim forward action that must be avoided by a submersible. A more complex simulator may have hundreds of types of marine life that each perform multiple actions and have variable visual appearances. Note that these agents are internal to the domain's system and not to the AI agent, who is performing a task on the domain. The agents that are internal in the domain are (e.g., other players of a game and other drones in a swarm). Sensors that are internal in the domain would be, for example, a radar sensor that detects the AI agent's presence or movement.

Multiple Entities: When multiple entities are present, new elements of the domain emerge, including static relationships and dynamic interactions. In the AUV example, in a less complex simulator, all fish may be the same size (relationship) and swim in a straight line (interaction). In a more complex simulator, marine life may include very small and very large animals, and fish may swim in complex schooling and swarming patterns. These elements are, by definition, only present and observable when multiple entities are present in the environment. Similar to the single-entity categories, the relationship and interaction categories can have a growing number of discrete classes, attributes, and representations as domain complexity increases. For instance, new interactions can be added, such as animals following each other, eating each other, and fighting with each other, etc. The attributes of existing relationships and interactions can also increase in complexity, such as the speed and intricacy of schooling behavior.

Complex Phenomena: When multiple relations and interactions are present, complex phenomena can emerge within environments. Three examples of such complex phenomena are events, goals, and rules. When multiple interactions occur in series based on entity relationships, events can take place (e.g., a fire spreading across trees based on their physical proximity relationships). The goals of agents in the environment also become discernible through the sequence of interactions the agent pursues within that environment. In real-world domains, and extensively in games, rules can be applied globally to define which relationships and interactions are permissible in the environment. Rules and constraints in the environment space reduce the size of the environment. These include the rules of a game, number of states constrained by symmetry, number of possible agent interactions, and the set of state transitions that are constrained by rules.

Examples of rules: The rules of driving restrict which lanes can be used when driving in certain directions; the rules of monopoly constrain spending by the amount of money the player has (0 money = 0 spending); and the rules of tic-tac-toe prevent a player from placing an X on a spot that already has an O. In AI simulations, real-world interactions or complex phenomena (e.g., a submersible requiring fuel to generate power) may be simulated as domain rules (e.g., agents cannot issue move commands if their fuel level = 0).

For each category of elements that emerges with single entities, multiple entities, and complex phenomena, the number of distinct classes, distinct attributes, and distinct representations can be increased. Collectively, the scale and diversity of these elements determine the environment space.

Task Solution Space

The task solution space is composed of the number and diversity of paths that can be taken to complete a task, whether in a real, open world or in a simulator, where the available world states and action space is defined. This space increases in complexity as the set of possible state transitions increases and as the available paths for success becomes more complex (see example state transition graphs in Figures 3 and 4). In perception domains, the task solution space would also include the set of data classification classes. The complexity of the task solution space is not dependent on the complexity of the environment, and the complexity of the environment may increase, decrease, or not impact the task solution space. For example, consider the domains of chess and a self-driving car in the real world. The environment of chess is quite limited with a small number of unique objects and interactions, whereas the real world of the car may have thousands of unique objects, external agents, relationships, and interactions. If the task for the self-driving car is to move forward 1 yard, then this expansive environmental complexity has a minimal impact on the task. Furthermore, if the available action space includes a command to 'move forward 1 yard,' then this task becomes trivial. Winning a game of chess against a challenging opponent, in comparison, would require much more computation and strategy and have a lower success percentage than the self-driving car task.

The number of possible paths, set of possible agent interactions, and restrictions on successful paths to achieve a goal are the primary drivers of complexity in the task solution space. We can consider a maze task in a grid environment. If the number of available paths increases from 2 to 10, then the complexity of the task solution space will increase if there is only one correct path, but will not become more

challenging if every available path leads to the goal. Further, the addition of a new object class, such as a boulder, may increase the complexity of the maze environment, while decreasing the complexity of the task solution space, by blocking off incorrect paths and reducing the search space.

The task solution space defines the number of possible actions and the distribution of paths (number of paths, number of intersections) through the state transition graph, the number and degree of dependencies and connections between agents and state transitions, and the degree of available strategies (defined as the set of all possible decision sequences through the environment, without violating any constraints). This set of paths can be represented as a state transition graph as explored in Section 6.

5.2 Components that define extrinsic domain complexity

The components of extrinsic domain complexity are defining the AI agent that is performing a task on the domain of interest and the skills needed to performing these tasks. The agent dependent or extrinsic complexity is a mental model of an agent that has structural, observational, and planning components and is a subset of full domain complexity.

Performance space

Performance space is the agent-policy scoring function and the performance of the agent acting on the domain. These are the scoring of games, such as win/loss, win/loss/draw, or the range of score numbers. This would also be the reward function in reinforcement learning. For an AUV, the solution space would be, for example, the time taken to get from point A to point B.

Planning space: This is the number of elements or the size of the possible goals to have in the domain. Some domains have only one goal; e.g., to win, to kill, or to move from point A to point B. Other domains have more possible goals; e.g., win with the highest amount of money, move from point A to B while having x amount of fuel left, and avoiding actions m, n (e.g., car: crashing, toll roads, flooded roads; aircraft: storm cloud or all clouds in VFR flight conditions). Set of possible strategies; the set of all possible paths through the state transition graph that end in a goal state. Number of goal states.

Skills space

These are the components that define the skills that an AI agent would need to have to perform a task in a domain of interest. They consist of physical and mental skills.

Physical skills (hardware): These are the physical abilities of the AI agent; the hardware. If a big rock is in the moving agent's path, then it is insignificant if the agent is a bulldozer robot, but it is significant if the agent is a vacuum-cleaner robot. Other characteristics include GPU, CPU, ROM/RAM, control problem, type, number, accuracy, and precision of sensors or activators. The robot AUV has sensors, such as camera, pressure sensor, and sound sensor, etc. The camera might have three (RGB) channels and a certain resolution.

Mental skills (software): These are the software or knowledge, cognitive skills, and intuitive skills that an AI agent needs for performing the task in the domain of interest. This would be the ability for prediction, planning, setting up, or following goals. Other characteristics include the size and difficulty of interactions, actions, perception, goal states, and events.

Note that these components will have additional dimensions when we are considering a dynamic domain where the complexity level becomes non-stationary. As the complexity level increases, the time scale will become more significant.

These components will be represented and estimated using the suggested methods in Section 6.

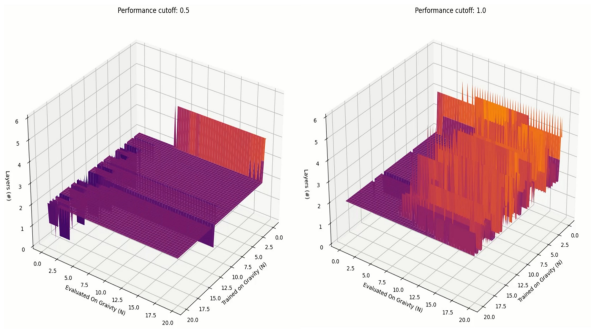

Figure 2: Minimum number of layers needed by DQN agent to achieve 50% (left) performance (pole balanced for 100 seconds) and 100% (right) performance (pole balanced for 200 seconds) on tasks where the agent is trained on one value of gravity, but tested on another value.

Agent-based Extrinsic Measures The above dimensions define the spaces in which an agent resides in order to solve tasks in the domain. One way to identify relationships across these spaces is to define a class of agents from the skills space to solve a task and compare the minimal complexity of an agent from the class that is capable of achieving different points in the performance space. Holder and Pereyda (Pereyda and Holder 2020) proposed a measure of task complexity, which is defined as the sum of complexities of minimal-complexity agents capable of achieving the possible range of performance scores for the task. For example, Figure 2 depicts the complexity of a deep Q-learning agent in terms of the number of layers necessary to achieve half (0.5) and full (1.0) performance on the Cartpole task (balance a pole on a cart by pushing the cart left and right), where the agent is trained on one setting of gravity and tested on another (to mimic the open-world novelty of a change in gravity). As the figure shows, more complex agents are needed to achieve higher performance, especially on tasks where gravity increases, but less so when gravity decreases. These agent-based extrinsic results allow us to evaluate the complexity of tasks and provide insights into the intrinsic domain complexity.

6. Representation and measures

The perception and action/planning parts of the domains, because of their high difference, require different methods for assessing the components related to them, mentioned in the framework in Section 5. For the action and planning domains, we use state transition graphs, and for perception domains, the feature space provides a geometric representation for assessing the size of the domain and the significant components of the data distribution in the domain.

Domain Representations

In this section, we briefly review the two fundamental domain representations for problems in Classical AI action and planning domains and Data Science perception domains. We will then discuss several key factors of domain complexity that can be understood to hold over both of these problem contexts and domain representations.

State Transition Graph Space A state transition graph provides a conceptual representation of the agent's domain, including possible world states, possible actions or transitions in each state, and the consequences of those actions on the state. In addition, the graph can indicate the initial state that the agent begins in at the start of execution, the set of states which satisfy the current task objective (goal states), and the possible paths that lead from the initial state to the goal state. State transitions may occur as the result of the agent's own actions, due to opponents' or allies' actions, or external environmental actions.

This method of representing the domain allows us to characterize an AI action and planning task as identifying a potential path through the state space to the goal, successfully taking actions at each decision point to remain on an efficient, low-cost path, which leads to the goal. This becomes more challenging if the complexity of the state transition graph increases: more states, more possible actions, very heterogeneous states and transitions, and sparser or more inaccessible goal paths all increase the difficulty of recognizing and maintaining an efficient, successful path to a goal state.

Feature Space Figure 5 depicts a small, 3-dimensional feature space for a simple data science perception domain. As the number of features in a data science problem gets larger, the dimension of the feature space grows. Higher dimensional tasks are more computationally difficult for common model-fitting techniques, such as clustering and regression. Also, as a given set of data is spread over a larger feature space, it generally grows more sparse, and the distribution can become harder to characterize accurately.

Additionally, if the data is distributed more heterogeneously across this space, then it can be difficult to characterize accurately; more diverse, outlying, or smaller subgroups of observations may not be classified correctly. Similar to the inherent domain complexity due to the distribution of paths through the state transition graph, these are inherent difficulties with the data domain (feature space and training data distribution) that will have some impact on any classification or prediction task in that domain.

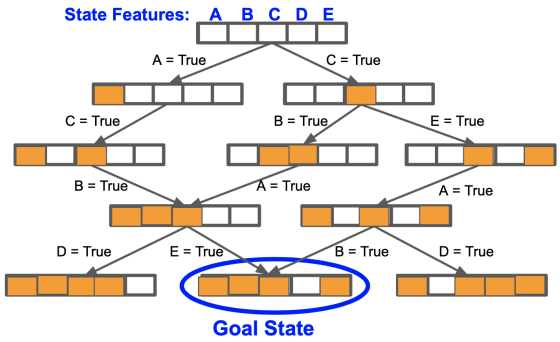

Figure 3: A simple example of a state transition graph. Possible states are depicted as nodes in the graph, connected by edges that represent actions transitioning between states. In this case, the state definition consists of five Boolean features. In the initial state, all features are set to False (white). Actions affect the state by flipping a selected feature to True (orange), and only certain actions are possible in each state. One state is highlighted as the goal state for the current task. There are relatively few possible states, at most two possible actions in each state, and multiple intersecting paths through the graph lead to the goal state. This domain and task have low complexity.

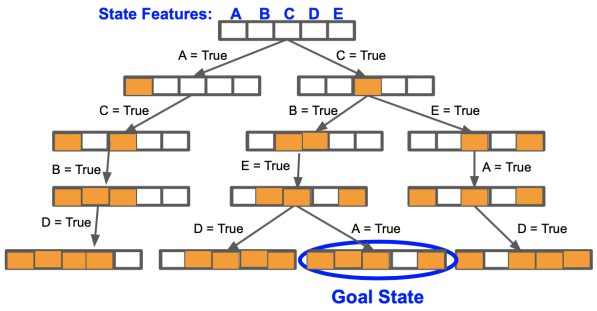

Figure 4: An example of a state transition graph for a more complex domain. Note that there is only a single path to each of the final states, meaning that if an agent makes a single incorrect decision pursuing the current task, then it will be unrecoverable, and the agent will fail the task. The same holds for all states in the domain, and no matter which state is selected as the goal state, it will be challenging to achieve that goal successfully. The set of possible paths leading to a given state is more sparse in this domain than in the first domain; this domain is more complex.

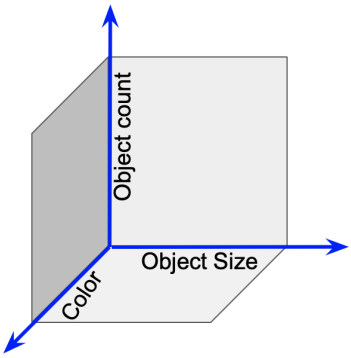

Figure 5: A 3-dimensional feature space for a dataset containing images with three features: Object count, Object Size, and Color. Each observation in the training dataset can be plotted as a point in this 3-dimensional space. ML algorithms can then characterize the distribution of points across this space (using clustering, regression, CNNs, or other techniques) such that they are able to make predictions about how new data points might fit in the existing distribution.

Complexity Measures

Dimensionality In both Classical AI and Data Science applications, dimensionality impacts the difficulty of completing tasks in the domain and is a key measurement tool for estimating domain complexity.

In Classical AI, increasing the number of possible states (including increasing the environment size, environment features, and number of possible interactions with objects or external entities) increases the size of the state transition graph. Similarly, increasing the number of possible actions increases the breadth or tree-width of the state transition graph. Both increase the dimensionality of the environment space and significantly increase the difficulty of tasks that involve navigating the state transition graph to reach a goal state. If a given goal state is only accessible by very long paths (which require the agent to make a long series of correct decisions), then this increases the dimensionality of the task solution and performance spaces, the complexity of the domain, and the difficulty of reaching the goal.

In Data Science, increasing the number of possible features, the number of possible values for each feature (for example, by increasing the set of possible objects or environment conditions) increases the size of the feature space and the domain complexity with respect to the environment. This increases the difficulty of fitting models to characterize the distribution of the training data across that space. As vocabularies increase, sensor resolution increases, or additional variables are added, the dimensionality of the data science problem increases, and task difficulty increases as well. Meanwhile, as the number of possible classes or prediction values increases, the set of possible parameterizations for fitting the model to the feature space can increase exponentially, increasing the complexity with regards to task solution and performance.

Sparsity In addition to dimensionality, sparsity is an important characteristic of the complexity component spaces described in the previous section. It relates to the rarity of successful strategies, or how thinly distributed the information necessary to make successful decisions can be. This is related to how difficult it is to develop successful strategies and perform well.

In Classical AI action and planning domains, the sparsity of paths that lead through the state space and the rarity of successful paths to the goal, both impact the complexity with regards to the task solution space and performance space (see figure 4). When most states can be accessed through only a very small number of paths, the state space is more difficult to navigate, and the domain is less forgiving of suboptimal decisions. It is easy to unintentionally enter a state from which the goal is no longer accessible. Significantly more memory or computation may be required to correctly identify the optimal path throughout the entirety of execution.

In data science and perception domains, a sparsely distributed training dataset, especially when spread across a large feature space, significantly increases complexity with regards to the task solution space and performance. When there are many classes or prediction values, which only have a few observations in the training data (i.e., many classes appear rarely in the training data) and when the training data in general is spread sparsely throughout a large dimensional feature space, there will be many modeling solutions with similar fit (given this minimal information), and it will be challenging to meaningfully model the data distribution in a way that supports accurate predictions.

Heterogeneity Finally, heterogeneity refers to the diversity of important information in a given domain component. Problems that include more diversity generally require more information, and therefore, more complicated solutions to address correctly.

Classical AI action/planning problems that have a very heterogeneous state space (environment space) in which different states often have different possible action sets (and the same action may produce very different transitions depending on the feature values of the current state) have an increased complexity with regards to the task solution and performance space; they require more information to navigate successfully. This often occurs in contexts where a diverse set of skills is required to navigate the state space; domains with high skill set complexity will have high heterogeneity.

In data science and perception domains, heterogeneity can occur in the feature space when different features have very different properties with respect to the data distribution (i.e., combining data from different sensors in cases with high skill set complexity). Additionally, classes or prediction values, which reflect very diverse subgroups in the data, may be more difficult to model efficiently and accurately.

Domain-complexity level in a transdisciplinary light differs in different ways. Domain-complexity level can indicate that the domain is simple, complicated, or complex in different ways. For this reason, we propose using a final, sin-

gle value of a complexity level, rather than complexity level values for the environment, policy, solution, and planning spaces described in Section 5. These four spaces' complexity level can be expressed as a bar chart or spider chart (with its caveats). The skills space expressed in these four spaces would show what skills are needed for an agent to perform a task successfully in that domain.

7. Conclusion

Returning to our example of the AUV from (Wilson, McMahon, and Aha 2014), we can express the difficulty that was encountered when transitioning from the simulator domain to the real-world domain in terms of the components of our complexity levels. The real world had higher dimensionality with regards to the environment space complexity and greater heterogeneity with respect to the perception-task solution space (where noise and environmental factors meant a diverse collection of different sensor readings might represent a single system state). These unexpected increases in domain complexity, as compared to the simulator domain, caused the AI to behave pathologically, constantly replanning in response to the discrepancies between its perception of the real world and its less complex internal state space. Once the complexity of the deployment domain was understood, a bounding box technique was introduced, enabling the robot to operate successfully in the open ocean.

8. Next steps

This paper is an early attempt at synthesizing several measures and aspects of domain complexity that are hypothesized to apply generally across domains and tasks. Many questions still remain. We believe that these questions need to be formulated and addressed within a research agenda that treats domain-*general* complexity (as opposed to domain-*specific* complexity) as a first-class citizen. We are not claiming that each such research question should empirically apply to every domain, but ideally, should be applicable to a diverse set of domains, allowing general claims to be made. As next steps, we state three questions below that could (potentially) be investigated in an experimental setting with the goal of yielding general insights about domain complexity:

1. First, can we derive strong theoretical connections between domain complexity and *difficulty*? For instance, will it always be the case that tasks in a more complex domain will be *necessarily* more difficult, or just more difficult *in expectation*? And can difficulty be studied independently of complexity? These are questions with which multiple communities, especially within AI and open-world learning, are only just beginning to grapple. In addition to stating theoretical claims, it is also important to test these claims empirically. We believe that designing appropriate experimental methodologies that allow us to validate general claims is (in itself) a promising area for future research to tackle.
2. Second, can our measures of domain complexity be used to define and *quantify* complexity in so-called complex systems, such as networks and dynamical systems, as well as other non-linear systems (e.g., chaotic differential equations)? Is one system more complex than another, and if so, then along what dimensions? Moreover, does this have theoretical ramifications, validated by appropriate experiments, for analyzing such systems?
3. Third, in the cases of infinite action and state spaces, what are the appropriate mathematical frameworks for distinguishing between domains of (arguably) differing complexity? In real analysis, for instance, there are different hierarchies of infinity that are well understood. For instance, integers and real numbers both form infinite sets, but the latter has provably greater cardinality than the former. Could similar claims be made for complexity?

These questions are not exhaustive, and some have several other questions associated with them that may require several parallel lines of theoretical and experimental research to fully investigate. We emphasize that the one commonality between all these questions is their lack of dependence on a single domain or model. Rather, all of them aim toward an agenda that is as domain-general as possible. As noted earlier, we believe that this is the central element that distinguishes other field-specific studies of complexity from our proposal.

References

Agarwal, M.; Aggarwal, V.; Quinn, C. J.; and Umrawal, A. K. 2021. Stochastic top-k subset bandits with linear space and non-linear feedback.

Amir, O.; Doshi-Velez, F.; and Sarne, D. 2019. Summarizing agent strategies. *Autonomous Agents and Multi-Agent Systems* 33.

Carroll, T. L., and Byers, J. M. 2017. Dimension from covariance matrices. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27(2):023101.

Chaslot, G.; Bakkes, S.; Szita, I.; and Spronck, P. 2008. Monte-carlo tree search: A new framework for game ai.

Claussmann, L.; Revilloud, M.; Glaser, S.; and Gruyer, D. 2017. A study on al-based approaches for high-level decision making in highway autonomous driving. 3671–3676.

Falniowski, F. 2020. Entropy-based measure of statistical complexity of a game strategy. *Entropy* 22(4).

Hernández-Orallo, J., and Dowe, D. L. 2010. Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence* 174(18):1508–1539.

Holzinger, A.; Biemann, C.; Pattichis, C. S.; and Kell, D. B. 2017. What do we need to build explainable ai systems for the medical domain?

Ingrand, F., and Ghallab, M. 2013. Robotics and artificial intelligence: a perspective on deliberation functions. *AI Communications* 27.

Jackson, M. C. 2020. How we understand "complexity" makes a difference: Lessons from critical systems thinking and the covid-19 pandemic in the uk. *Systems* 8(4).

Jung, S.; Sinha, K.; and Suh, E. S. 2020. Domain mapping matrix-based metric for measuring system design complexity. *IEEE Transactions on Engineering Management* 1–9.

2016. What is complexity theory?

Kim, J. 2014. How Complexity Domain Impacts Software Development Process.

Legg, S., and Hutter, M. 2007. Universal intelligence: A definition of machine intelligence.

Pereyda, C., and Holder, L. 2020. Measuring the complexity of domains used to evaluate ai systems.

Prnjat, O., and Sack, L. 2001. Complexity measurements of the inter-domain management system design. In *Proceedings. Ninth IEEE International Conference on Networks, ICON 2001.*, 2–7.

Remus, R., and Ziegelmayer, D. 2014. Learning from domain complexity. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2021–2028. Reykjavik, Iceland: European Language Resources Association (ELRA).

Senator, T. 2019. Science of artificial intelligence and learning for open-world novelty (sail-on). *SAIL-ON*.

Stephenson, M.; Renz, J.; and Ge, X. 2020. The computational complexity of angry birds. *Artificial Intelligence* 280:103232.

Štuikys, V., and Damaševičius, R. 2009. Measuring Complexity of Domain Models Represented by Feature Diagrams. *Information Technology And Control* 38(3):179–187.

Wilson, M. A.; McMahon, J.; and Aha, D. W. 2014. Bounded expectations for discrepancy detection in goal-driven autonomy. In *AAAI 2014*.