GENERALIZED ASSET INTEGRITY GAMES

by

© Karl A. Lambert

A Thesis submitted to the

Graduate Studies Office of the Faculty of Engineering and Applied Science

in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

May 2016

St. John's Newfoundland

ABSTRACT

Generalized assets represent a class of multi-scale adaptive state-transition systems with domain-oblivious performance criteria. The governance of such assets must proceed without exact specifications, objectives, or constraints. Decision making must rapidly scale in the presence of uncertainty, complexity, and intelligent adversaries.

This thesis formulates an architecture for generalized asset planning. Assets are modelled as dynamical graph structures which admit topological performance indicators, such as dependability, resilience, and efficiency. These metrics are used to construct robust model configurations. A normalized compression distance (NCD) is computed between a given active/live asset model and a reference configuration to produce an integrity score. The utility derived from the asset is monotonically proportional to this integrity score, which represents the proximity to ideal conditions. The present work considers the situation between an asset manager and an intelligent adversary, who act within a stochastic environment to control the integrity state of the asset. A generalized asset integrity game engine (GAIGE) is developed, which implements anytime algorithms to solve a stochastically perturbed two-player zero-sum game. The resulting planning strategies seek to stabilize deviations from minimax trajectories of the integrity score.

Results demonstrate the performance and scalability of the GAIGE. This approach represents a first-step towards domain-oblivious architectures for complex asset governance and anytime planning.

ACKNOWLEDGEMENTS

I would like to thank my family and friends for their continued support throughout my life. Of utmost importance has been my wife and personal saint, Inga, without whose love and encouragement I would not have completed this great endeavour.

I would also like to acknowledge the efforts of my supervisor, Dr. Leonard Lye, as well as the academic support staff at the Memorial University Engineering Graduate Studies Office. I thank these individuals for their continued patience, sponsorship, and belief in my completion of this programme.

TABLE OF CONTENTS

List of Tables

List of Figures

1 INTRODUCTION

1.1 Background

Modern engineering systems represent complex, high-utility interconnects of people, software, and hardware. Prototypical examples include cyber-physical networks, critical infrastructures, and socio-technical ensembles. These assets are considered highly integrated systems-of-systems with multiple, time-varying objectives and potentially conflicting constraints.

Life-cycle planning is often accomplished through hierarchical management frameworks which combine centralized, aggregated decision making with distributed, autonomous control policies. These frameworks are typically developed in conjunction with compliance standards, safety regulations, and design/operation guidelines. Fundamentally, they represent planning activities supported by expert knowledge, decision-support systems, and procedural consensus. While these management frameworks are proven, they possess significant decision overhead and latency. In real-time online (RTO) scenarios, complex planning actions must be completed with near-optimal performance guarantees in sub-second time intervals. To address these challenges, computational modelling and simulation have become increasingly integrated into the planning process.

Across several industries, probabilistic risk analysis (PRA) and its variants are used to assess the asset condition. Several frameworks exist, such as the Risk-Based Asset

Integrity Management (RBAIM) proposed by Khan et al. (2010). The current state-of-the-art involves dynamically updating a set of probabilistic beliefs regarding the condition of an asset. This condition is typically based on the risks of unwanted component operations, process deviations, or subsystem failures. Architecturally, frameworks such as the RBAIM proceed in a logical manner similar to controllers with closed feedback-loops. This planning can be broken down into three well-defined steps: *monitoring*, *evaluation*, and *prescription*.

During the *monitoring* phase, sensory data are cleaned, streamed, and aggregated into a presentation frame for input into the evaluation module. An *evaluation* module takes as input the pre-conditioned data and decides, almost exclusively through computational processing, a set of numerical values which describe the asset state. This typically involves some reduction mapping, filtration, or classification of the data into a labelled configuration, rating, or score. The evaluation module estimates the expected utility and/or reward derived from being in, or potentially reaching, a set of states. The state-transition likelihoods, costs, risk profiles, long-run gains/losses, and other quantities may also be evaluated. The final component of an asset management framework centers around (typically sequential) decision making. Prescription modules may provide interfaces for reporting and recommendation, but their primary task is action-selection. In the context of artificial intelligence, this phase represents a subset of automated reasoning. In control theory, this process can be viewed as solving for and implementing an optimal control policy. In operations research, it is often referred to simply as planning. Numerically, prescription is effectively a dynamic performance optimization. The objective is to take as

input an asset state or condition and specify a policy, strategy, trajectory, or sequence of state-transitions which satisfice, or optimize, some performance criteria. When the prescription phase completes, the chosen actions are presented as output for effectuation.

At a high level of abstraction, management frameworks for life-cycle asset planning almost exclusively follow this "three-phase" approach. The process of *monitoring-evaluation-prescription* is akin to *observe-decide-act* and other reasoning cycles [Boyd 1976, Stone 2007]. When the asset-environment system becomes more complex and uncertain, one often implements decision making behaviour through *process architectures*. Adaptive control systems, intelligent agents, and cognitive architectures are among the more modern examples. In many cases, stochastic reinforcement learning is applied to recognize patterns, and identify features which lead to incremental performance gains.

In planning problems, one is often faced with an uncertain and indirect knowledge of the asset-environment state; this constitutes *partial observability*. Partial observability may arise from the statistical estimation of properties. For large-scale complex assets, a summary description of the system may be available but suffer from a reduction in representational power and information loss. Partial observability may also arise from raw measurement limitations, as well as through instrumental limits of error. This type of planning commonly adopts the Partially Observable Markov Decision Process (POMDP) model. The literature on POMDPs is vast and well-developed. By themselves, these models are often computationally challenging, and much research has addressed dimensionality reductions and approximate solutions. Typical solutions implement value or policy iteration through dynamic programming [Smallwood and Sondik, 1973]. A further

complication arises when noise, errors, and stochastic effects cause actions to be "imperfect". This constitutes a *tremble*, or deviation from the desired response. For many assets, these imperfections represent mistakes in restorative actions or probabilistic outcomes of maintenance operations. This type of planning commonly adopts a *perturbation analysis* (PA) model, which may be coupled with the aforementioned POMDP approach. PA is essentially a sensitivity-based gradient-descent which reasons about the effects of unwanted and/or unplanned behaviours. A final challenge in planning is dealing with asset-environment systems given poorly specified and/or unknown rewards; this is a model identification problem. In this setting, reinforcement learning (RL) techniques are used. Common RL models include: Q-learning (QL), temporal difference learning (TDL) and probably-approximately-correct (PAC) learning. These methods receive feedback regarding the cause-effect associations between actions (state-transitions) and rewards (or costs). RL models are able to "learn" solutions to POMDPs without explicit specification of the transition probabilities. POMDP, PA, and RL models have been applied to a variety of problems with varying degrees of success [Cao, 2007].

For our purposes, asset management frameworks reduce to sequential decision making through monitoring, evaluation, and prescription. Planning activities are supported by advances in architectures (e.g. autonomous agents, control devices), models (e.g. POMDP, PA, RL), and algorithmic implementations (e.g. dynamic programming, Monte-Carlo sampling). This process has traditionally harnessed domain-specific knowledge, exploiting problem structure and yielding specialized solutions.

The most brittle, least transferrable aspects of governance and planning occur with respect to modelling constructs and performance evaluations. Modelling activities capture the relevant information and logical features of an asset. These must be massaged into desirable, functional descriptions. High-fidelity models require adequate knowledge representations. Working attributes may be mined from characteristic data, and integrated within some information management system. In many ways, this process is time-consuming and partially duplicated across designs [Curran, 2014]. An extension of this process involves the evaluation of asset performance. Models for state-transition systems exist and methods for their analysis are typically well-known. However, the "correct" performance metrics, operational constraints, and degrees-of-freedom are in general not well-known. In a POMDP, one seeks to maximize some sequence of "states" to achieve some "reward" through "actions". This model can be applied off-the-shelf if and only if acceptable, well-defined notions of states, rewards, and actions are known. For the purposes of system identification, evaluation, and optimization, very few "universal criteria" exist. The functional mappings from model attributes to states, rewards, actions, and goals/objectives are again difficult and expensive to construct. These mappings are typically developed on a project-specific basis, rendering them difficult to migrate beyond very narrow conditions.

As engineered systems become increasingly complex and adaptive, the decision making process becomes increasingly convoluted. Objectives, constraints, performance criteria, and control actions become non-stationary and outright obscure. The goals of asset

management become uncertain. In a general sense, the planning problem is ill-posed, and new techniques must be sought.

1.2 Motivation

The governance of complex assets requires systematic procedures. Planning activities consolidate decision-making regarding the fate, utilization, and performance of assets. This form of governance is often accomplished through a spectrum of high-level management frameworks and low-level optimal control policies. For complex assets, these processes require significant automation and intelligent decision support. Despite broad industry acceptance, classical architectures for asset planning are relatively brittle. Solutions are often ad hoc, non-interoperable, task-driven, and project-specific. Under certain conditions, models of the asset and its environment may be over-calibrated. Under others, the analysis may be rendered intractable or invalid. Classical architectures require major rework for new asset classes, models, and mission scopes. There is an emerging need for domain-oblivious, platform-agnostic solutions.

This thesis establishes a planning architecture which is extremely general, yet requires only a basic level of mathematical sophistication. Our methodology centers on the desirability to preserve what foundationally constitutes a *form*. Almost all components, processes, systems, and assets admit symbolic descriptions from which *integrity* is often sought. Integrity in this sense represents "correctness" – a proximity to homeostasis in the form of stable equilibria and ideal conditions. Our approach grounds these abstract ideas within the context of finite discrete state-transition systems. The definition of an asset is generalized to the limits of dependability engineering. Ideas from network science and

graph theory are used to construct metrics for robustness, resilience, and efficiency. Enforcing these model-centric signatures is germane for all assets. Through this interpretation we are able to construct “universal” performance evaluations and planning objectives. This allows a single architecture to automate this process for all assets.

This research is motivated by two major challenges. The first challenge is representational. Solutions must provision for inputs over a massive semantic range. This challenge is tackled through information theory, utilizing similarity metrics such as the graph edit distance (GED) and normalized compression distance (NCD). These metrics are parameter-free, feature-free, alignment-free comparisons of finite objects. These similarity metrics are used in conjunction with fitness indicators for model robustness and equilibrium to yield an integrity score. The second challenge is algorithmic. One must devise a fast procedure for identifying strategies. The prescribed actions must securely defend against adversarial attacks on the asset, while behaving safely albeit opportunistically in the face of naive stochastic environments. This situation presents itself as a combinatorial game which can be efficiently searched using backwards induction and variations of minimax.

The impetus therefore corresponds to generalizing the evaluation and prescription modules of the aforementioned “three-phase” planning architecture. Throughout this work, it is assumed that adequate monitoring is available. High-performance, real-time online (RTO) algorithmic solutions are sought. These solutions must be robust, and make few assumptions regarding the asset or its domain. An architecture which is adaptive yet non-brittle is developed. Performance objectives, evaluation criteria, and optimization procedures are kept domain-oblivious and platform-agnostic. The remainder of this thesis

illustrates the concepts of generalized assets, universal integrity metrics, and game-theoretic optimizations. It develops these ideas from theoretical frameworks to working implementations. The result is a Generalized Asset Integrity Game Engine (GAIGE), which is shown to be versatile and scalable.

1.3 Contributions

This thesis brings to light several areas of research and unites them under a common theme. The contributions to asset integrity planning are summarized here.

1.3.1 Generalized Asset Performability Criteria

Advancements in general modelling are used to formulate an abstract definition for generalized assets. Several fit-for-purpose concepts are explored using the language of graph theory and network science. Dependability metrics, such as reliability, availability, and importance, are used as performance indicators for a variety of reference graphs. The best-known results from information theory are used to define similarity measures. The edit distance and normalized compression distance are used to construct a payoff function for asset fitness. This is defined by an integrity score, which represents the proximity to an ideal (dependable) topological configuration.

1.3.2 Game-Theoretic Planning

Asset planning is formulated as a noisy sequential game. The base game is a two-player zero-sum stochastic game with incomplete information and imperfect actions. The base game is nonetheless mean-field symmetric in payoffs (zero-sum), actions, and information. This form admits a minimax solution. Perturbations to the base game structure

induce deviations away from minimax trajectories. This thesis examines strategies which are resilient to such noise.

1.3.3 Generalized Asset Integrity Game Engine.

This thesis augments brittle and non-competitive planning architectures. These are often based on partially-observable Markov decision processes (POMDP), with a state-space calibrated for a particular asset or domain. Classical, decision-theoretic planning is based on policy or value iteration that is optimized for *harsh environments*. In harsh environments, risk sources are stochastic, albeit naive. Loads and effects such as wind, waves, storms, earthquakes, freeze-thaw cycles, and solar damage - are by themselves applied passively to reduce the asset condition. These risk sources have no direct knowledge of inspection and maintenance practices. The harsh environment might act in an extreme manner, but possesses neither the intent, nor the intelligent look-ahead to disrupt asset persistence. Background aging processes, themselves mixtures of stochastic and deterministic mechanisms, include such things as corrosion, crack-propagation, fatigue, or other incidental damage. These make up the standard antagonists in the so-called harsh environmental regime.

Game-theoretic planning extends integrity reasoning to *hostile environments*. In hostile environments, intelligent adversaries work in conjunction with natural risks to actively deny asset performance. Game-theoretic asset integrity planning seeks to find action sequences which are robust against *all possible outcomes*. Strategies must securely defend against worst-case attacks while ensuring safe, opportunistic utilization. This thesis combines several algorithms to deliver a fast, anytime-optimal response.

2 GENERALIZED ASSET PERFORMANCE

2.1 The Generalized Asset

Abstractly, an *as-set* is a basic set equipped with a performance measure. Basic sets are finite, discrete collections of distinct, countable objects. A performance measure is a way of assigning value to any subset of elements in the set. A complex asset is enriched with additional structure, such as a partially-ordered power-set. Complex assets may feature recursively-nested subsets and interacting measures. Being composed of sets, measures, and their operations, these "abstract assets" are studied more formally in pure mathematics.

More concretely, the term asset often refers to a real-world system. Most engineered systems can be categorized as assets. These systems incorporate many elements whose continued structural existence and correct operation generates some reward. Complex assets extend this concept further. They include large-scale structural ensembles of interacting components and subsystems. Complex assets are typically identified by massively modular interconnectivity. Their utilization produces emergent, uncertain risks and rewards at multiple scales. Payoffs are typically measured in terms of socio-economic utility. The ownership of complex assets is distributed across many stakeholders, who share the costs, benefits, and risks associated with the asset.

Examples of modern (complex) assets include cyber-physical networks, socio-technical systems, critical infrastructures, civionics platforms, and high-utility

interconnects. While these objects have become increasingly complex, their modelling and simulation has become increasingly fit-for-purpose. All assets nonetheless admit a sequential evolution in time. It is possible to witness the asset at discrete time-steps, and infer temporal difference relations between the set of previous and current observations. Of course, highly-variable conditions can significantly affect the belief in system configuration. This belief is based on partially observable states, incomplete information feedback, and variable utilization profiles based on moving performance constraints.

Several modelling frameworks are capable of capturing the workings of complex assets. A modern approach involves the use of a modelling language. Modelling languages express information, semantics, and systems knowledge in a structure that is defined by a consistent set of rules [Jezequel et al., 2002]. The relevant system characteristics, including components, events, relations, and process behaviours, can be described in a modelling language. Several model description languages (MDL) also incorporate the ability to specify performance requirements and constraints which must be satisfied. In some cases, these boundary-like conditions are left out, or the MDL lacks a direct syntax for their specification. These "incomplete" descriptions are effectively domain-oblivious. Any information they convey regarding the performance state(s) of a real-world system must be extracted from the appropriateness of the model representation itself. This is in direct contrast to a domain-specific MDL, which possesses enough expressive power to also describe the fitness of the system at hand. Put another way, a domain-specific MDL encodes not only the structure and dynamic behaviour of the actual system, but also some implicit impression of its overall performance. This performance is gauged through pre-

constructed indicators, measurable states, and/or a sense of fitness condition. The domain-oblivious or general-purpose MDL encodes a dynamical system through a model, but does not embed any proper assessment of its own configuration.

The line between both types of model description (and their language) is not necessarily crisp. For example, some descriptions embed meta-data, error-checking and control, aggregated observations, and system-level scoring. In this work, the more general case is assumed; i.e. efforts to describe and embed the state of a system into its own description are *agnostic*. The setup for *generalized assets* is that they function like any MDL which coherently realizes, interprets, and encapsulates a complex real-world system.

In summary, a *generalized asset* is an umbrella term used to reference a finite discrete collection of information emitted and presented by a *source*. In this work, the source refers specifically to an abstract model of an actual, real-world system. The generalized asset captures the relevant workings through some MDL. We mainly consider domain-oblivious MDLs, where asset descriptions do not implicitly encode assertions regarding an overall fitness level or global score. This definition of *generalized asset* is broad and all encompassing. In motivating the development of a useful architecture, we follow with a review of the most common asset modelling frameworks.

2.2 Modelling Institutions

The vast majority of generalized assets are given in terms of a model description language (MDL). An MDL expresses the relevant objects, states, relations, and transitions

of the system at hand. Paradigms for abstraction, reasoning and modelling assets vary greatly; Table 2.1 lists several common modelling institutions.

Table 2.1. Modelling Institutions.

Institution	Diagrammatic Mechanism	Abstraction for
Formal System	Axiomatic	Reasoning Rewriting
Association Scheme	Set Indexing / Design	Algebraic Structures Combinatorial Designs
Finite State Machine	Language / Automata	Models of Computation
Process Calculus Process Algebra	Message Passing	Actors Simulations
Block Diagram	Process Flow	Systems Modelling
Vector Addition System (and variants)	Traversal Space	Distributed Systems
Petri-Net (and variants)	Stochastic Queue	Queuing Networks Concurrent Processes
Boolean Circuitry	Logic Tableau	AC0, NC0 Analogies
Belief Networks (Including Neural, Boltzmann, Markov, Bayes)	Revision / Propagation	Machine Learning Optimization Pattern Recognition Generation Classification

Each institution is a mathematically well-defined methodology for representing information, and ultimately modelling the behaviour of a logical system. Their ontological makeup can be categorized by their primary *diagrammatic mechanisms* and *modelling abstractions*. Diagrammatic mechanisms can be vaguely understood as the methods through which information is logically traversed, in the sense of being tagged, parsed, updated, and/or reasoned about. Each institution provides a modelling abstraction, which services a theoretical scope and range of practical applications.

Many of these institutions are weakly interchangeable, as morphisms between them exist under appropriate conditions [Goguen and Burstall, 1992]. A common, unifying theme is that all of these systems involve the propagation of causality in the form of state-transitions or other transactions. Many logical operations in one institution are also available in another, or some equivalence of operations exist [Diaconescu, 2008].

These institutions often admit a single, common ontological interpretation in *graph-theoretic form* For example, *graphs* (and their *labelling*) generalize the diagrammatic mechanisms of finite state machines, block diagrams, vector addition systems, petri-nets, circuits, networks, and trees. Graphs can be used as abstractions for conceptual and semantic models. They provide structures for sub-symbolic, connectionist reasoning. Graphs are essentially a very general, frequently used institution for representing and processing information. They are encountered in computer science, systems engineering, and well-adopted by formal language theory. Graphs are at the heart of many programming paradigms. In engineering design, they serve as the basis for the Unified Modelling Language (UML). They also provide a standard way of formatting information in the Process Specification Language (PSL). The PSL is foundational to ISO 18629, which provides standards for industrial automation systems and integration [ISO 18629, 2006].

2.3 Graphs and Their Relatives

2.3.1 Preliminaries

A graph consists of two finite sets, *V* and *E*. Each element of *V* is called a *vertex*. The elements of *E* are called *edges*. The edges in *E* are pairs of vertices. Together, *V* and *E* form a graph, *G*. Graphs model pairwise relationships (edges) between objects (vertices). The basic notion of a graph can be extended in several ways:

1. When the set *E* contains ordered pairs of vertices, we obtain a directed graph, or a digraph. Each edge in a digraph has a specific orientation.
2. When the set *E* contains repeated elements, it becomes a *multiset*. The resulting graph is then a *multigraph*.
3. When an edge can be formed from a vertex to itself, we obtain a "loop". Graphs containing loops or self-edges are known as *pseudographs*.
4. Allowing edges to be arbitrary subsets of vertices gives rise to hypergraphs.
5. Allowing *V* or *E* to be an infinite set, one obtains an *infinite graph*.
6. By allowing vertices to reference or signify groups of vertices and edges together (subgraphs), one obtains a *metagraph*. Metagraphs are "graphs of graphs".

For notational convenience, an edge directed from vertex *u* to vertex *v* may be represented as *{u,v}*, or more concisely as *uv* when context allows. The *order* of a graph *G* is the cardinality of its vertex set. The *size* of a graph *G* is the cardinality of its edge set. Given two vertices, *u* and *v*, if $uv \in E$, then *u* and *v* are said to be *adjacent*. If $uv \notin E$, then

u and *v* are non-adjacent. Furthermore, if an edge *e* has vertex *v* as an endpoint, we say that *v* and *e* are *incident*. The *neighborhood* of a vertex *v*, denoted *N(v)*, is the set of vertices adjacent to *v*. For a set of vertices *S*, the neighborhood is the union of neighborhoods of the vertices in *S*. The *degree* of *v*, denoted by *deg(v)*, is the number of edges incident with *v*, with self-loops counted twice. In simple graphs, this is the same as the cardinality of the vertex neighborhood, *N(v)*. The *maximum degree* of a graph *G*, is defined as $\Delta(G) = max(deg(v) \mid v \in V(G))$. The *minimum degree* of a graph *G*, is defined as $\delta(G) = min(deg(v) \mid v \in V(G))$. In *normal* graphs, the *handshaking lemma* applies, yielding a result that says the sum of the degrees of the vertices is equal to twice the number of edges. This result is also known as the *first theorem of graph theory*.

A *path* in a graph is a sequence of distinct vertices, $v_{1,}v_{2,}\ldots,v_k$, such that $v_i v_{(i+1)}$ is an element of *E* for $i = 1,2,\ldots,k-1$. The *length* of a path is the number of edges on the path. A *cycle* in a graph is a sequence of vertices $w_{1,}w_{2,}\ldots,w_{(r-1)},w_r$, such that $w_{1,}w_{2,}\ldots,w_{(r-1)}$is a path with $w_1 = w_r$, and $w_{(r-1)}w_r \in E$. Essentially, a cycle is a closed path. Self-loops can also be considered cycles in the degenerate case. The length of a cycle is defined as the number of edges on the cycle. An odd cycle has even length, and vice-versa. A graph of order *n* is considered a tree graph, or simply a *tree*, if and only if it is acyclic and contains *n-1* edges.

A *degree distribution* is a probability distribution of the in- and out- degrees of all the vertices in a graph. A *path-length distribution* is a probability distribution of the lengths of non-cycle paths between all vertex pairs in the graph.

A graph is considered *connected* if every pair of vertices can be joined by a path. Each maximal connected piece of a graph is called a *connected component*. A graph is *strongly connected* if every vertex is reachable from every other vertex through some path. If the removal of a vertex *v* from *G* causes the number of components to increase, then *v* is called a *cut vertex*. If the removal of an edge *e* from *G* causes the number of components to increase, then e is called a *bridge*. The smallest connected graph contains two vertices sharing a single edge with unit degree and path distributions. A graph is isomorphic to another graph if there is an edge (and label) preserving bijection between all vertices in one graph and all vertices in the other graph. A graph is homomorphic to another graph if there is an edge (and label) preserving surjection between all vertices in one graph and all vertices in the other graph.

2.3.2 Special Graphs

There exist several types of graphs with special attributes. The most crucial to our discussion are the null, empty, and complete graphs. The null graph is simply the null set, and contains no vertices or edges. The empty graph on n vertices, denoted by E_n, is the graph of order *n* where *E* forms an empty set. The complete graph on *n* vertices, denoted K_n, is defined as the graph of order *n* where ∀u ∈ V, ∀v ∈ V, uv ∈ E.

A graph is called $\langle K_v, K_e \rangle$*-complete* if the number of edges is related to the number of vertices by the following equation:

$$|E| = K_e\left(\frac{|V|(|V|-1)}{2}\right) + K_v|V| \qquad 2.1$$

Where:

- $|E|$ denotes the cardinality of the edge set, or graph size.
- $|V|$ denotes the cardinality of the vertex set, or graph order.
- K_e is the number of edges connecting each pair of non-identical vertices.
- K_v is the number of self-adjoint edges, or allowable self-loops per vertex.

This definition of $\langle K_v,K_e\rangle$-*completeness* is unique and not found in the standard literature. By convention, with $K_v > 1$ we have the *pseudograph* property, and for $K_e > 1$ we have the *multigraph* property. In a graph-theoretic sense, this construction would be termed a pseudo-multi-graph. For simplicity, we refer to it as the $\langle K_v,K_e\rangle$-*complete* graph of order n. This graph can be denoted by the triple $\langle K_v,K_e,K_n\rangle$ indexing a multiplicity over the complete graph K_n. It can be concisely read off as $K_{(v,e,n)}$. Figure 2.1 illustrates a $K_{(v,e,n)}$ for $<1,2,5>$ and $<3,3,3>$.

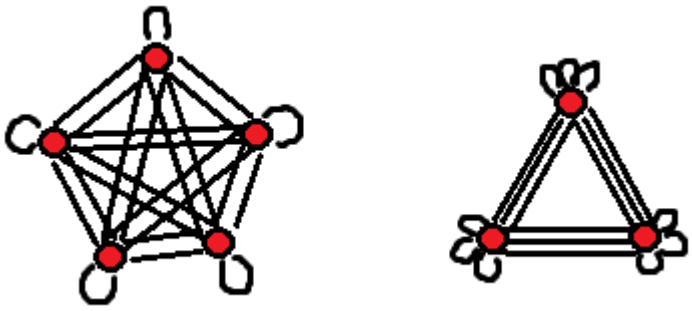


Figure 2.1. $K_{(1,2,5)}$ **and** $K_{(3,3,3)}$**.**

A *weighted graph* is a graph in which each edge has an associated weight, cost, or distance. The weights are typically *metric*, and can be made to represent functional

evaluations. In a weighted graph, the weight of an edge *e* is denoted by *w(e)*. If the edge *e* directs vertex *u to v*, we can write *w(u,v)*. If no explicit weight is given, the edge is assumed to have weight 1 if it exists. Non-edges are usually given the weight 0 or ∞ , depending on the context. A weighted graph *G* can be represented by a *weighted adjacency matrix* $A = \{a_{ij}\}$, where $a_{ij} = w(v_i, v_j)$.

In general, a weighted $K_{(v,e,n)}$ forms a Cartesian product space that is represented by a multi-dimensional adjacency array. This array can be indexed by (3+1)-tuples which take a source vertex, a destination vertex, a valid edge between them, and point to a corresponding weight or traversal cost. For example, $K_{(3,4,5)}$could be represented by an adjacency list.

The length, or number of items in this list, is found by substitution into equation 2.1 giving:

$$|E|=4(\frac{|5|(|5|-1)}{2})+3|5|=55 \qquad 2.2$$

Each of the $|E|=55$ edge weights of the $K_{(3,4,5)}$ complete graph can be queried from the adjacency list. If one assigns integer labels, this can be accomplished via some production of the form $\langle a_{i\in n}, a_{j\in n}, a_{ij}\rangle$, where $ij \in \{e\}$ if $i \neq j$, and $ij \in \{v\}$ if $i = j$.

2.3.3 The $K_{(1,2,N)}$ Asset Representation

Many generalized assets epitomize complex, dependable systems. These systems can often be given in the form of $K_{(v,e,n)} = \langle 1,2,\mathrm{N}\rangle$ weighted complete graphs. The $K_{(1,2,N)}$ graph forms an $N \times N$ weighted adjacency matrix with all N^2 elements filled. Recall that

if adjacency does not exist between two model constituents, the edge element can take on a vanishingly small or large value, depending on the context.

This case has special significance, because many real-world assets are described, handled and processed using the equivalent of a $K_{(1,2,N)}$representation. The NxN weighted adjacency matrix is commonly yet unknowingly encountered across several disciplines. It is an efficient data structure for describing relations between dense, highly inter-dependent objects. Furthermore, transformations are often imposed so as to convert to and from this NxN format. The resulting adjacency matrices are nonetheless capable of capturing all the relevant state-transitions and process interactions between elements.

Finally, because the $K_{(1,2,N)}$ graph admits an $N\times N$ matrix, one can often extract and operate over certain matrix attributes, such as the characteristic polynomial or eigenvalue decomposition. This makes certain assets more amenable to spectral analysis [Gertsbakh and Shpungin, 2011].

2.3.4 Graph Dynamical Systems

Many generalized assets evolve as a graph dynamical system (GDS). Typically, the generalized asset is presented to a decision agent (or its architecture) chronologically, over a finite number of discrete time-steps. In the most common setup, a generalized asset is an encoding of what is essentially an active or live $K_{(1,2,N)}$graph. At each time-step, there is a noisy realization of an $N \times N$ weighted adjacency matrix which represents the current belief in the real-world asset configuration. The generalized asset performance can then be assessed by analyzing the sequence of graphs (or matrices), as they evolve over time. Formally, a graph dynamical system (GDS) consists of:

- A *graph* G, with vertex set $v(G) = \{1,2,\ldots,n\}$.
- For each vertex i, a state $x_i \in X$, where X is some finite set of states. The *system state* is given by the n-tuple, $x = (x_{1,}x_{2,}\ldots,x_n)$.
- A *vertex function*, f_v, for each vertex v, which maps the state of vertex v at time $t$to the vertex state at time $t+1$ based on the states associated with x. Particular emphasis may be placed on the states of the vertices adjacent to v.
- An *update scheme* that governs how the vertex functions are applied, so as to induce a discrete dynamical system with map $F : X^n \to X^n$.

Characterizing the performance of a GDS is computationally difficult [Zelazo and Mesbahi, 2010]. Research in this area seeks local-to-global relationships, where from local graph properties (and update rules) one seeks to infer the emergence of global behaviour.

The GDS formalism is therefore applicable to a wide variety of complex systems, and is perhaps the most dominant subsumption within our notion of generalized assets.

2.4 Performance Evaluations

Generalized assets must have their configurations analyzed and assessed. This process constitutes a performance evaluation, which determines the overall asset behaviour, and to what extent the asset is functioning. Whether qualitative or quantitative, highly-detailed or of low resolution, a performance evaluation essentially maps local and global states to a more condensed encoding. This reduction may output a numerical score, tuple, or alpha-numeric string of relevant information. Mappings from states and configurations to their resulting scores can be constructed from two distinct evaluation paradigms. We categorize these as being either fitness-based or similarity-based. These evaluation types can also be relative or absolute. In most planning architectures, the purpose of a performance evaluation is to ultimately make informed decisions about the fate of the asset.

2.4.1 Fitness-Based Evaluations

In fitness-based evaluations, several well-known and desirable properties are composed from the ground-up into an overall score. This technique utilizes both indicators as well as metrics. Indicators identify the existence potential for factors which cannot be well-defined or well-measured. Indicators are typically more qualitative, and are often

derived from heuristics. Their development is predominantly driven by experience and a sense of best-practice. For our purposes, metrics are a more concrete, quantitatively stronger form of indicator. They may be applied to uncertain or partially-observable properties through estimation and/or approximation.

Both indicators and metrics are essentially measures of expectation based on experience, observations and information-gains. Combining these measures into a score identifies the fitness-level of the asset. This evaluation can be either absolute, relative or sometimes both, depending on how the scoring functions are used. An example of both types of fitness is captured by the *Elo* ratings system used in Chess, or the *TrueSkill* system used by other competitive ladders [Herbrich and Graepel, 2006]. These ratings systems specify strength of play which can be interpreted as a score or fitness level. The higher the rating, the more potent the player, and the more statistically likely to defeat any random opponent selected from the set of all players. This is an example of an absolute measure based on the ground state of the ladder. It corresponds to a difference between the probability distributions of a player's strength, and the distribution of strengths for all players. In addition, the higher the ratings difference between two randomly selected players, the more likely the higher rated player will defeat the lower rated player. This is again an assessment of the difference between two probability distributions. This scaling is often non-linear, with small differences in rating being more pronounced at high levels of play. In this way, the ratings system can also be used as a relative measure of performance.

Fitness evaluations often scale monotonically. An increase (or decreases) in key-performance measures will directly correspond to an increase (or decrease) in fitness level.

The main advantages and disadvantages of fitness-based evaluations are contained in Table 2.2.

2.4.2 Similarity-Based Evaluations

In *similarity-based* evaluations, asset configurations are compared and contrasted. The differences between asset configurations are then associated with a score. The similarity (or difference) may be relative or absolute depending on the chosen reference datum. Similarity-based evaluations typically assign more or less *importance* to patterns of discrepancy based on their regularity, magnitude, frequency, and location.

An example of this type of evaluation is the *edit distance* between two strings. The edit distance typically counts the number of operations required to transform one string into the other. In most contexts, one string represents the achievement of some goal, ideal, or reference configuration. The other string represents the current sample for comparison. In this way, the similarity between strings represents the performance of one with respect to the other. Similarity-based evaluations are able to capture the notions of "performance" and "integrity" in a general sense, as these are both manifestations of "deviations from correctness". This has the advantage of being nearly domain-oblivious. Nonetheless, some notion of proximity to "correctness", i.e. the reference design object, must be known or estimated in advance. Table 2.2 summarizes the key benefits and drawbacks of fitness-based and similarity-based evaluations.

Table 2.2. Fitness vs. Similarity-Based Performance Evaluations

Performance Evaluation	Advantages	Disadvantages	Examples and Applications
Fitness-Based	– Specialized, customizable. – Ascertains detailed, actionable knowledge about the asset. – Low computational complexity in the best case.	– Requires domain-specific knowledge. – Difficult to construct and evaluate. – High computational complexity in the worst case.	– Ratings systems. – Voting/Auctions. – Analytical Hierarchy/Network Process. – Evolutionary and/or Genetic fitness. – Classifiers.
Similarity-Based	– Applicable almost everywhere. – Requires little domain knowledge. – Simple to construct and evaluate. – Computational costs are fixed/known.	– Requires baseline reference object(s). – Provides vague, hard to interpret knowledge about the asset. – Operates at a higher level of abstraction.	– Edit distances (Hamming, Levenshtein, etc.) – Sorensen-Dice Index. – Jaccard Coefficient. – Information-theoretic distances. – Kolmogorov Complexity. – Entropy estimation. – Discrepancy Analysis. – Anomaly detection.

2.5 Binary Monotone Fitness

The performance of real-world assets underscores dependable operation and utility production. These outcomes should be low-risk, safe, secure, reliable, available, and of high-quality. Many of these systems are modelled as networks and graphs. The relevant information regarding components and state-transitions is captured by some model description language, which ultimately expresses a generalized asset. Classical (non-quantum) components and systems exist in only one discrete state at a time. In dependable systems engineering, the state function is almost always a fitness-based performance evaluation. This implies a many-to-one composition of metrics with domain-specific parameters, optimizations, and tunings.

Because several fitness-based evaluations share similar mathematical properties, it is sometimes possible to abstract away from a particular asset, its domain, and specific analysis parameters. This is particularly true when systems are composed of near-identical, tightly-coupled elements [Cox, 2009]. These elements are frequently queried together using similar access patterns and return similar states. Under these conditions, it becomes efficient to replace the individual, potentially real-valued state evolution functions with a simple logical map. This mapping is typically a truth table which determines whether or not components or systems exist above or below some threshold value. This can be viewed kind of pass/fail test criteria, which flags the value 1 denoting existence above some threshold (e.g. activation potential), and 0 denoting the existence below said threshold. A

structure function, Φ, can then be defined for the specified threshold which distinguishes between two states: a functioning or active state and a failed or inactive state.

Fitness-based performance evaluations, when based on threshold exceedance or binary compliance criteria, are often minimally sufficient for actionable decision-making and planning at a large-scale [Cox, 2009]. In many cases, this does not alleviate the need for a more intricate fitness-based analysis [Bier, 2005]. Nonetheless, this practice can be applied to many components and systems, as it rapidly imparts the most fundamental and crucial information. A discussion is therefore necessary to appreciate the induced scope with respect to generalized assets.

2.5.1 Binary Monotone and Coherent Systems

A system is considered *monotone* and *coherent* if and only if it satisfies both the *monotonicity* and *coherence* requirements for its structure function. Monotonicity requires (i) that a structure function Φ be non-decreasing in each argument, and (ii) that the function maps to zero when all components are failed, $\Phi(\vec{0}) = 0$, and maps to one when all components are functioning, $\Phi(\vec{1}) = 1$. Condition (i) implies that the system can not deteriorate (that is, change from the functioning state to the failed state) by improving the performance of a component. Condition (ii) implies that if all the components are in the failure state, the system necessarily has to be in the failure state (although this is not necessarily sufficient). Similarly, if all the components are in the functioning state, the

system is in the functioning state. Coherence requires the system be (i) monotone, and (ii) each component is relevant and actually contributes to the overall structure function.

The combination of Boolean (or binary) threshold states, monotonicity and coherence yields a *binary monotone system,* which is a useful description for rapidly assessing the condition, state, or fitness of a sub-region within a model. Many of the terms and definitions found in this section have been adapted from the work of Aven (1991), [Aven and Jensen, 1991].

2.5.2 Structure Functions

2.5.2.1 Series-Parallel Systems

A system that is functioning if and only if each component is functioning is called a *series* system. The structure function for a series system is given by:

$$\Phi(\vec{x}) = x_1 x_2 \ldots x_n = \prod_{i=1}^{n}(x_i) \qquad 2.3$$

A system that is functioning if at least one component is functioning is called a *parallel* system. The structure function for a parallel system is given by:

$$\Phi(\vec{x}) = 1 - (1 - x_1)(1 - x_2)\ldots(1 - x_n) = 1 - \prod_{i=1}^{n}(1 - x_i) \qquad 2.4$$

The expression on the right-hand side can also be given in coproduct form:

$$\Phi(\vec{x}) = 1 - (1 - x_1)(1 - x_2)\ldots(1 - x_n) = \coprod_{i=1}^{n}(x_i) \qquad 2.5$$

2.5.2.2 k-out-of-n Systems

A system that is functioning if and only if at least *k* out of *n* components are functioning is called a *k-out-of-n: good* system. Series and parallel systems represent the boundary cases of *k = 1*, and *k = N*, respectively. A series system is an n-out-of-n system, and a parallel system is a 1-out-of-n system. The structure function for a k-out-of-n system is given by:

$$\Phi(\vec{x}) = \begin{cases} 1 \;\; if \;\; \sum_{i=1}^{n} x_i \geq k \\ 0 \;\; if \;\; \sum_{i=1}^{n} x_i < k \end{cases} \qquad 2.6$$

2.5.2.3 Minimal Cut and Path Sets

A cut set *K* is a set of components that by failing causes the system to fail, i.e., $\Phi(\overrightarrow{0_K}, \vec{1}) = 0$. A cut set is minimal if it can not be reduced without losing its status as a cut set. A path set *S* is a set of components that by functioning ensures that the system is functioning, i.e. $\Phi(\rightarrow 1_S, \rightarrow 0) = 1$. A path set is minimal if it can not be reduced without losing its status as a path set.

2.5.3 Dependability Metrics

2.5.3.1 Reliability

The *reliability* of a system represents the probability that it has not failed. For binary-monotone-coherent systems, the series-parallel and k-out-of-n structure functions help define the reliability. The reliability of a k-out-of-n structure of independent components, all of which share an identical probability of non-failure (reliability) *p,* is given by:

$$R = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{(n-i)} \qquad 2.7$$

Where:

- *R* is the reliability of the k-out-of-n system.
- *p* is the probability of non-failure.
- *n* is the total number of components.
- *k* is the minimum number of functioning components for non-failure.
- *i* is the index.
- *"n choose i"*, or $\binom{n}{i}$ is the *binomial coefficient,* given by $\frac{n!}{i!(n-i)!}$ with $i \leq n$, and $i, n \in \mathbb{N}$.

This system is sometimes referred to as "*i.i.d.* k-out-of-n:G", where *i.i.d.* denotes independent and identically distributed, and G or F denote whether the combinatorial *threshold k* leads to a "*good*" or "*failed*" system.

There exist a number of methods for reliability computation of a general structure such as a network or a graph. Many of these methods are based on the minimal cut (*path*) sets. For smaller systems which are either very reliable (or unreliable), the so-called inclusion-exclusion principle may be applied. There also exist state enumeration methods, factoring (pivot-decomposition) methods, and many others. For a complete treatise of these methods, the reader is referred to [Aven and Jensen, 1991]. More generally, reliability analysis may concern itself with multi-state and non-monotone systems. These systems may feature components with non-identical failure probabilities, time and age-dependent failure models, inter-related failures, and other effects. A presentation of these more advanced reliability models is beyond the scope of this work.

2.5.3.2 Availability

The *steady-state availability (A)*, and *failure frequency (w)* are perhaps the two most important measures of *repairable and self-healing systems* [Mishra, 2008]. Several other steady-state availability measures can derived from these parameters. For example, there is the mean failure-repair cycle time (MCT), the mean up-time (MUT) and mean-downtime (MDT) during a failure-repair cycle, and the expected number of system failures/repairs during a specified time interval (*T*). Classically, the MCT, MUT, and MDT measures have been derived from the mean-time-to-failure (MTTF), mean-time-between-failures (MTBF), and mean-time-to/between-repairs (MTTR, MTBR).

These are given by:

$$MTTF = \frac{1}{w} \tag{2.8}$$

$$MTBF = \frac{A}{w} \tag{2.9}$$

$$MTTR = \frac{(1 - A)}{w} \tag{2.10}$$

Where *w* is given by failure expectations, according to the failure distributions of the components, or system. For *i.i.d. k-out-of-n:G systems*, the *point availability A* (at time *t)* is given by the ratio of expectation in uptime over the total time:

$$A = \frac{E\{\text{uptime}\}}{E\{\text{uptime}\} + E\{\text{downtime}\}} = \frac{A(k, n)}{A(k - 1, n)} \tag{2.11}$$

A powerful multi-dimensional Markov model developed by [Khatab et al., 2009], has been developed for analyzing state-transition systems subject to stochastic deteriorations and renewals. They model the a*vailability of non-i.i.d., k-out-of-n:G systems subject to repair via priority queues.* Their formulation utilizes the formalism of a stochastic automata network (SAN), which through *Kronecker algebra* is able to represent very large scale finite capacity queueing networks. The *stationary availability* of each component and the system are evaluated. With several assumptions made, the corresponding numerical problem is solved using algorithms from Monte Carlo simulation.

This method is highly sophisticated, and the technical details can be found in [Khatab et al., 2009].

2.5.3.3 Improvement and Importance

The main dependability concern is to certify that real-world assets operate effectively in the presence of process deviations and component failures. This analysis leads to network improvement and importance measures, where priority is given to elements (for graph-defined assets: vertices, edges) that contribute the most to the process or system.

Concepts from *reliability importance* can be generalized to include the contributions of a component towards other fitness-based performance metrics. These might include *maintainability*, *serviceability*, *availability*, *risk*, etc. Two importance measures in the literature are *Improvement Potential and Birnhaum's measure*. These are again treated in detail via [Aven and Jensen, 1991]. There are also techniques based on *risk achievement* or *risk reduction*, as well as the *criticality importance*, and *Fusell-Vesly's measure* [Cox, 2009]. Several of these measures also operate on graphs where the *quality* of edges and vertices can undergo variations. Each measure of importance depends on slightly different interpretations. However, they are all based on the contribution of a component's criticality to the overall system performance. This often implies a parametric sensitivity analysis for the system [Mishra, 2008]. Importance measures can be useful tools in the system optimization process.

Qualitatively, this procedure can be described as follows:

1. Identify the most important units by means of the chosen importance measure.
2. Identify possible improvement actions for these units.
3. Estimate the effect on reliability, availability, and performance by implementing the improvement.
4. Perform cost evaluations.
5. Make an overall evaluation and take a decision.

This procedure can also be accomplished through simulation and importance sampling [Zio et al., 2006]. As an example, consider the reliability of an *i.i.d., k-out-of-n:G* system. The value of this metric can be improved by increasing the number of components in the system. An increase in components from $n-1$ to n gives the *reliability improvement*:

$$\Delta R = R(k,n) - R(k,n-1) = \binom{n-1}{k-1} p^k (1-p)^{(n-k)} \qquad 2.12$$

for $n \geq k$

As *n* increases, this improvement will become progressively smaller. The problem of determining the optimal system size *n*, the reliability level *R*, and the threshold level *k* is an issue which arises in *system design*. The *sensitivity* of a change in this system reliability, with respect to changes in component reliability, can be found using:

$$\frac{dR(k,n)}{dp} = k\binom{n}{k} p^{(k-1)} (1-p)^{(n-k)} \qquad 2.13$$

Generalized assets may include systems which have the potential for *load-sharing and redistribution.* The lifetime distribution for component failures is arbitrary and not

necessarily exponential. Repair or replacement of failed components is delayed until some local criterion is met. Until restoration, the surviving components share (perhaps disproportionately) the load offered to the system. Components under these loading cycles are assumed to undergo *accelerated aging*. This results in a more contracted life model and representative changes in the failure distributions.

This more general situation represents a *stand-in* utilization model, as opposed to a *stand-by* model. For stand-in systems, switching and maintainability concerns become far more complex. These models have been investigated numerous times and are difficult to analyze directly. Evaluating the importance and improvement potential for stand-in systems is an area of ongoing research. The works of [Mishra, 2008], [Zio et al., 2006] can be referenced for details.

2.6 Graph Fitness

The properties of graphs serve as extremely useful performance metrics for generalized assets. In a graph-defined asset, discrete elements undergo state-transitions which interact to produce distinct patterns. Many of these patterns will be discernable at small scales, while manifesting behaviours which are statistically, topologically, or dynamically congruent at larger scales. It therefore becomes necessary to monitor, evaluate, and act-upon the most relevant properties of a graph. These properties are numerous, and their relevance varies by application.

Graph fitness is a performance concept based on well-defined measures. These measures can be based on graph performance indicators (GPI) or well-defined metrics such as resilience, efficiency, robustness, and decomposability.

2.6.1 Graph Performance Indicators (GPI)

The fitness of graph-defined assets is sometimes measured using graph performance indicators (GPI). These are categorized based on the nature of the underlying graph topology, and overlying coverage processes as summarized in Table 2.3.

Table 2.3. Categorization of GPI Types.

GPI Type	Topology		
Coverage	Activity	Static	Dynamic
	Static	– No add/remove of vertices or edge. – Fixed vertex and edge states. – e.g. bitmap image, non-mutable array.	– Vertices and edges can be added or removed. – Vertex and edge states do not change. – e.g. reconfigurable Boolean circuits, fabric/lattice switching.
	Dynamic	– Existence of vertices and edges is fixed. – Vertex/edge states can vary by process. – e.g. Network flow, traffic/capacity, petri nets, timed automata.	– Creation/destruction of vertices, edges. – Variable states based on coverage processes and rates/locales of element introduction/removal. – e.g. disease spreading, annealing, social games.

Topological dynamics are changes in graph structure over time. GPIs for this situation center on the *robustness* of a graph to changes in its topology. These metrics track the ability of a graph to maintain structural properties while undergoing permutations to its vertex or edge set. *Decomposability* characteristics focus on the changes to graph structure in response to both random and targeted vertex/edge removals. These removals may occur through discrete disconnection patterns or sweeping partition operations. The surviving graph(s) may potentially have new *strength* and *robustness* characteristics. For example, if the surviving components are more (or less) centralized and tightly clustered, the graph

is said to have *hardened* (or *softened)*. Decomposition operations can therefore lead to non-monotonic changes in graph properties. The opposite is also true. The *recomposition*, or addition of new vertices and edge patterns may increase topological performance while reducing coverage performance, through a phenomenon known as *Braess' Paradox* [Braess, 1968, 2005]. It is therefore important to understand the induced, coupled dynamics between topological and coverage processes.

Coverage dynamics represent changes in the quantities of interest which affect vertex or edge states over time. When vertex/edge states are merely Boolean existence values we degenerate to a form of topological dynamics. It can be shown that any coupled (coverage + topological) dynamics can be transformed into an equivalent static graph topology [Rozenberg, 1997]. Every possible realization of the graph then becomes a coordinate lookup in some large configuration-space. However, there is some inherent difficulty in finding and applying this transformation [Fan and Mostafa, 2006]. For reasons of algorithmic tractability, this method is rarely considered.

GPIs for *coverage* may also involve gauging the *utilization* of the graph through the *percolation* and *articulation* of its *activation*. *Activation* represents a distribution of volume or energy of coverage. It is directly associated with the discrete packets of exchange such as network flow, traffic, utility, etc. *Percolation* interfaces with topology to measure the potential for coverage. It measures the capability of activation to hold or spread. Percolation GPIs deal with reachability and connectivity. They may be based on *routability* and *conductivity*, which concern the quality of routes, lengths, durations, etc. Percolation is affected by the congestion levels of paths, circuits, and random walks between sources

and sinks. *Articulation* expresses the *switching resolution* of a coverage process. Its performance relates to percolation through the saturation of pathways and the (in) ability to route and/or spread activation. However, articulation is less concerned with re-routing, latency, and throughput capability. It is more concerned with specificity and responsiveness to sensitive coverage adjustments. Articulation GPIs seek to characterize the ability to manipulate and redirect activation at small scales.

Generalized assets can possess both *topological dynamics* as well as *coverage dynamics.* This conglomeration leads to graph dynamical systems (GDS) which are extremely difficult to analyze and possessing GPI which are hard to compute in general.

2.6.2 Graph Resilience

For *G* with fixed *P*, the "resilience of *G* with respect to *P*", is the minimum number r, such that by removing r edges from *G*, one almost surely obtains a graph H not having the property P. The most commonly accepted definition of graph resilience was put forth by [Sudakov, 2008]. The local resilience of a graph *G* with respect to property *P* measures how much one has to change *G* locally so that *P* no longer holds. The global resilience of a graph is defined analogously, where the changes and properties are global. The resilience of many graph properties can be used to construct valid GPIs for complex assets.

2.6.3 Graph Efficiency

Efficiency is another commonly used metric to assess graph models and graph-defined assets. It is predominantly a measure of coverage potential, and expresses a resistance to failure from the perspective of coverage inadequacy. It can be computed at both local and global scales [Latora and Massimo, 2001].

The *average efficiency* of a graph G is given by:

$$\eta_{avg}(G) = \frac{2}{N(N-1)} \sum_{i<j \in G}^{n} \frac{1}{d(i,j)} \qquad 2.14$$

The *average local efficiency* of a graph G is given by:

$$\eta_{local}(G) = \frac{1}{N} \sum_{i \in G}^{n} \eta_{avg}(G_i) \qquad 2.15$$

The *average global efficiency* of a graph G is given by:

$$\eta_{global}(G) = \frac{\eta_{avg}(G)}{\eta_{avg}(G^{ideal})} \qquad 2.16$$

Where:

- N is the total number of nodes in the graph G.
- $d(i,j) \simeq d_{min}(i,j)$ approximates the shortest path between node i and j.
- G_i is the local subgraph consisting only of a node i's immediate neighbours, but not node i itself, in a graph G with N total nodes.
- G^{ideal} is a reference graph with idealized properties, such as the complete graph or random graph, depending on the context.

2.6.4 Decomposability Metrics

Measures of decomposability study the inter-connectedness of a graph as it survives partitions through vertex and/or edge removals. The decomposition process can be systematic or essentially random. A graph-defined asset suffering random component failures will exhibit fitness degradation. Resistance to this process is effectively a form of graph *reliability*. Similarly, an adversary seeking to inflict maximal damage to an asset may have his actions viewed as a dismantling of key structural features in a model. Resistance to this process can be measured by the *Isoperimetric Number,* which is also known as the *Cheeger Constant.*

Robustness to these and other effects can be captured by several topological and coverage-based GPIs. Examples include the *Hosoya* or *Wiener Index*, the *Estrada Index*, or *Tutte Polynomial*. These measures are outlined in detail in the work of [Bunke et al., 2008]. As an example in graph robustness, let $\Omega(v, e)$be the set of all connected graphs *G* with *v* vertices and *e* edges. Assume that the components fail independently of each other with probability $1 - p$. The familiar reliability equation gives the probability that a graph *G* is connected:

$$R(G) = \sum_{r=1}^{v} S_r(G) p^r (1-p)^{(v-r)} \qquad 2.17$$

Where $S_r(G)$ is the number of connected induced subgraphs of *G* that contain exactly *r* vertices. An *r-cutset* of *G* is defined to be a set of *r* vertices in *G* that when removed from *G*, leave it disconnected. The number of *r-cutsets* of *G* is denoted $C_r(G)$.

Since any set of *r* vertices in *G* must be either connected or disconnected, the following relation holds:

$$S_r(G) + C_{(v-r)}(G) = \binom{v}{r} \qquad 2.18$$

Given *p*, there is always at least one locally best graph in $\Omega(v, e)$, i.e., a graph that is the "most reliable" in the sense of connection probability. This interpretation of graph reliability can be seen as a probability of surviving successive cuts and removals while remaining connected.

The *isoperimetric number*, or *Cheeger constant* is a numerical measure of a graph's disposition towards *bottlenecks*. It is yet another measure used to assess the inter-connectedness of a graph. Let *G* be a finite undirected graph with vertex set *V(G)* and edge set *E(G)*.

For an allocation of vertices $B \subseteq V(G)$, let $\partial B = \{(x, y) \in E(G): x \in B, y \in V(G)/B\}$ denote the collection of all edges from a vertex x in *B* to a vertex y outside of *B*. Then the *isoperimetric number*, or *Cheeger constant* of *G*, denoted *h(G)*, is given by:

$$h(G) = min\{\frac{|\partial B|}{|B|}: B \subseteq V(G), 0 < |A| \leq \frac{1}{2}|V(G)|\} \qquad 2.19$$

Note that $h(G)$is positive if, and only if, *G* is connected. The value of $h(G)$is large if partitions of the vertex set *B* lead to subsets with many edges between them.

2.7 Normalized Compression Distance

A reasonable notion of similarity is how difficult it would be to transform one object into another, using the most efficient transformation possible. The normalized information distance (NID) is universal in this regard [Li et al., 2004]. Unfortunately, the NID is also incomputable in general [Vitanyi et al., 2008]. Admissible information-theoretic distances have been successfully applied to a variety of objects. The most prolific of these is the normalized compression distance (NCD).

2.7.1 Kolmogorov Complexity

Kolmogorov complexity is a notion of information content. It is based on two principles: (a) all data can be represented as a bit string; (b) the shorter this string can be described, the less information is contained in it. These principles are detailed in [Li et al., 2004] and [Vitanyi et al., 2008]. The main idea is that with respect to some universal Turing machine U, there must be some minimal description for any given data given by:

$$K_U(x)=min\{|y|:U(y)=x\} \qquad 2.20$$

The Kolmogorov complexity $K_U(x)$of a string x is uncomputable in general. There can be no algorithm which computes the Komolgorov complexity of *x* for all *x*, so that $\forall x, K_U(x) \notin T(U)$. This result can be bounded from above, and for every algorithm which bounds it, there is another algorithm which provides a better bound. Fortunately, all computable *compressors* approximate $K_U(x)$.

The Kolmogorov complexity of a file is a lower bound on the length of the ultimate compressed version of that file. If one assumes that the natural data contain only effective regularities that a good compressor finds, then $K_U(x)$is only slightly smaller (up to a constant factor) than the length of the compressed version $C(x)$; that is $C(x) + O(|x|) \approx K_U(x)$.

2.7.2 Normalized Information Distance

The *normalized (symmetric) information distance* (NID) is given by:

$$NID(x,y)=\frac{max[K(x \mid y), K(y \mid x)]}{max[K(x), K(y)]} \qquad 2.21$$

When the incomputable functions $K(*)$ are approximated by a good choice of compressor $C(*)$, one obtains the normalized compression distance.

2.7.3 Normalized Compression Distance

The normalized compression distance (NCD) expresses the similarity between any pairs of finite objects. The NCD is an information-theoretic measure of how difficult it is to convert one object into the other through computational means given by:

$$NCD(x,y)=\frac{C(x\|y)\text{-}min[C(x), C(y)]}{max[C(x), C(y)]} \qquad 2.22$$

Where || is the familiar *concatenation* operator, and $C(x\|y)$denotes a compression of the concatenated representation of objects x and y. This compression is assumed to be roughly symmetric, so that $C(x\|y) \simeq C(y\|x)$.

When the data emanates from generalized assets, which can be represented by graphs or as models described in some language, the notion of *concatenation* is extended to these descriptions. That is, two models (e.g. graphs) *X* and *Y* are combined (joined, concatenated) into a single equivalent model (or graph) *XY*.

The idea behind the NCD is that if *X* and *Y* share common information, they will compress together better than separately, as the compressor will be able to reuse the recurring patterns found in one of them to more efficiently compress the other. In practice, the NCD is a non-negative number, $0 - \epsilon \leq r \leq 1 + \epsilon$, representing how different two objects are. Similar numbers represent more similar objects. The ϵ in these bounds is due to imperfections in compression techniques. For most standard compression algorithms, one is unlikely to see an epsilon above 0.1 [Vitanyi et al., 2008].

The NCD is intended to be universally applicable. As a similarity metric, the NCD has been put through numerous stress tests, for instance in [Nykter et al., 2008]. Its main advantages include being parameter-free, feature-free, alignment-free, and resistant to noise [Cebrian et al., 2007]. It can be explicitly computed, and is useful in clustering, classification, and anomaly detection tasks. This makes the NCD both theoretically and practically appealing, as access to a simple lossless compressor (such as GZIP or LZW) allows one to evaluate the similarity of generalized assets. If a reference configuration of known fitness is available, then the NCD allows one to complete a "universal" performance evaluation.

3 INTEGRITY GAMES

3.1 Background

3.1.1 Context

The time-evolution of generalized assets can be examined through many lenses. The theory of (general) dynamical systems provides the broadest scope, investigating all matters of dynamical behaviour. Dynamical systems come in many "flavours", with specific sub-fields dedicated to symbolic or arithmetic sequences, topologies, graphs, and other structures. Emphasis is placed on answering important structure and existence questions. A formal study may determine the particularities of reachability, stability and approximation. The potential for various events, behaviours, boundary effects, asymptotic limits, and attractor (or repellor) regions is often sought. The applications of dynamical systems theory have spawned entire fields, including coding theory, (optimal) control theory, and game theory, to name but a few [Picci and Gilliam, 1999].

A survey of governance, management, planning, and control activities reveals that they are reducible to complex decision making. These processes may involve concurrent, distributed action-selection, which can nonetheless be transformed into a time-sequential dynamical system. Using information feedback, update-rules, and resource constraints, the decision process manifests itself as strings of symbols in some alphabet, walks over a tree, traversals over a graph, transitions between points in a space, or coverages of a set.

3.1.2 Terminology

When certain conditions are met, one has a particular type of dynamical system called an integrity game. This definition of integrity game shares some of the notions used by other fields. The most general terminology can be found in the literature on algebraic and symbolic dynamical systems theory. Its subset, optimal control theory, is used predominantly in the applied sciences. This terminology is relatively ubiquitous in process control and systems engineering, although many different formulations, modelling approaches, and solution techniques are in use [Picci and Gilliam, 1999], [Smolensky, 1986]. Game theory matured in conjunction with these and other fields, but has since adopted its own terminology. The game formulation is well-suited to problems in the social sciences, operations research, economics, business analytics, and computer science. Game theory is a mature platform for a highly-studied variation of the common theme: optimizing performance.

Table 3.1 outlines several of the prevalent terminologies encountered in game theory, control theory, and their parent, dynamical systems theory.

Table 3.1. Outline of terminology encountered by field of study.

Terminology	Game Theory	Control Theory	Dynamical Systems
Systems Context	– Game – Auction – Tournament	– Plant – Process – Signal	– Generalized Systems
Representation of States	– Fitness Landscape – Payoff Matrix – Decision Tree	– State Space – Configuration Plot – Time/Frequency Domain	– Phase Space – Phase Portrait – Geometric Manifold
State Variables	– Payoffs/Rewards – Costs/Risks – Utility	– State-Variables – Costs/Errors – Gains/Losses	– State-Variables – Functions/Measures – Penalty Gradients
Source of Governing Dynamics	– Players – Actors – Agents	– Controls – Controllers – Forcing Functions	– Evolution Functions – Potential Forms – Drifts/Exchanges
Uncertainty & Feedback Mechanisms	– Imperfect Information – Incomplete Information	– Error Terms – Reference Signals – Partial Observability – Noise/Filters	– Flows – Diffeomorphisms – End/Boundary Effects
Traversals	– Moves – Decisions – Actions	– State-transitions – Controls	– Paths – Evolutions – Propagations
Sequence of State Visits	– Strategies – Action Profiles	– Policies	– Trajectories – Orbits (ergodic)
Criterion for Optimality	– Minimax Strategy – Nash Equilibrium – Solution Refinements – Learning Rate	– Optimal Policy – Adaptability – Stability – Transient Response – Asymptotic Behaviour	– Lyapunov Stability – Basins of Attraction – Bounds/Limits – Conjugacy – Invariants
Performance Objectives	– Find optimal strategies. – Provide conditions for a win/loss	– Find optimal policies – Design optimal controllers.	– Describe potential behaviour and confine transfer functionals.
Canonical Types	– Combinatorial – Differential – Evolutionary – Stochastic – Quantum	– Robust – Adaptive – Intelligent – Statistical – H-infinite	– Measure-Preserving – Topological – Graphical – Symbolic – Sequential

3.1.3 Ontogenesis

Game theory extends aspects of decision theory and overlaps much of control theory. In terms of decision-making, game-theory replaces the single-decider situation with multiple-players or rational agents. In terms of control, game theory urges multi-controller design in the presence of intelligent adversaries. This is sometimes the equivalent of developing a *robust* controller which defends against the expectation of *worst-case* behaviour. Classical game theory goes further, by requiring the potential for dynamical processes to be equipped with intelligent behaviour derived from inductive inference and/or analytical look-ahead. If the equivalent behaviour were formulated using the language of control theory, the underlying processes and filters might be classified as *acausal* and/or *anti-causal*. Thus, in game-theory one is concerned with dynamics which can affect the state of the system through their conscious motivation. In control theory, the opposing dynamics are implicitly naive.

Models based on game theory allow one to address *risks* (and risk sources) which are not only *naive* and *stochastic* (so-called *harsh environments*), but also superimposed with *adversarial dynamics* (so-called *hostile environments*). Thus, where control theory often seeks to *optimize asset performance in the face of a harsh environment*, game theory seeks to *satisfice asset performance in the face of hostile environments*. In terms of planning, game theory deals with purposeful, active performance denials, which subsume any accidental performance degradations.

The theory of games has become central to artificial intelligence, search algorithms, and architectures for autonomous reasoning. Modern game theory developed primarily out of research into operations research and mathematical economics. It has origins with Zermelo (1913), Borel (1921), Von Neumann and Morgenstern (1928, 1944) and of course with Nash (1950). From there, it expanded through Kuhn and Tucker (1951), Shapley (1953), Selten (1965, 1975), Conway (1970) and Smith (1972). Major contributions followed in Harsanyi (1973, 1992), Rubenstein (1982), Mertens (1985), and van Damme (1993) [Dimand and Dimand, 2002]. More recently, investigations into the computational complexity of games (and their solutions) have been put forward by [Condon, 1992], [Daskalakis, Goldberg, and Papadimitriou, 2008], and [Chen et al., 2010]. Interest into the design of game-playing agents, as well as the combinatorial and algorithmic aspects of game solution concepts, has given rise to the field of algorithmic game theory. Significant progress in this area has been made by authors such as Koller, Nisan, Roughgarden, and Tardos, among others [Nisan et al., 2007].

3.1.4 Classification

An integrity game constitutes a deterministic base game which is perturbed in a nondeterministic manner. In the present work, we consider only a restricted class of base games and perturbative effects.

The base game is taken to be a standard combinatorial game. This represents a sequential game where players alternate turns. On their respective turns, players choose to

transition the game into a new state, and receive an immediate fixed reward. This is done by selecting amongst a static, finite set of discrete actions. This process is repeated until termination criteria are met. Such games can be solved using the minimax theorem and by adversarial search algorithms [Hauk, Buro, and Schaeffer, 2006].

The perturbed game results when stochastic effects, such as noise, errors, trembles of hand (imperfect actions), and partial observability (incomplete information) come into play. Insufficient assumptions regarding opponent beliefs, rules, or number of players can also play a role. This alters the structure of the base game by essentially corrupting inputs and state evaluations, partially randomizing state-transitions, and modifying payoffs. Such games can be approximately solved using generative model discovery, reinforcement learning, and sampling based algorithms [Bowling and Veloso, 2000].

3.2 The Base Game

3.2.1 Specifications

A base game can be specified in one of several forms. An extensive-form game organizes the sequence of player decisions or moves into a decision tree. The extensive-form efficiently captures the choices at every decision point, including chance events from nature. A normal-form or strategic-form game describes strategy spaces and rewards by way of a payoff matrix. This organization is useful for games where decisions are simultaneous or premeditated before play occurs. The normal-form can also represent a degenerate case of extensive-form. When information feedback between sequential choices is minimal, the situation is effectively simultaneous. Furthermore, every extensive-form game can be transformed into a unique normal-form representation. However, the converse is not necessarily true. A normal-form game may admit multiple extensive forms. Converting an extensive-form game into a normal-form may require an exponential blow up in the size of the payoffs [Bowling and Veloso, 2000].

Finally, there is the *succinct-form*, which is the specification we adopt for the current version of our planning architecture. Succinct-form games lend themselves well to computational exploitation via symmetry and induction (e.g. transposition and refutation tables) [Schoenebeck and Vadhan, 2006], and [Fortnow et al., 2005].

Definitions found in this sub-section have been adapted from the works of [Brown and Shoham, 2008]. These authors follow a standard notation which has been used elsewhere in the literature on games.

3.2.2 Extensive Form

Much of the game-theoretic terminology is defined for extensive-form games. Specifying a game in extensive form has many useful applications. A brief discussion of their framework is therefore necessary. Formally, a finite extensive-form game consists of:

- A finite set of players $I = \{1,2,\ldots, i\}$;
- A finite set of nodes X that form a *rooted tree*, with a labelled root node $x_0 \in X$, and a set of terminal nodes $Z \subset X$;
- A set of *transition functions* that describe for each non-terminal node:
    - The player $i(x)$who moves at x;
    - The set *A(x)* of possible actions at *x*.
    - The successor node *n(x,a)* resulting from action *a*.
- Payoff functions for each player which assign payoffs to players as a function of the terminal node reached $u_i: Z \to \mathbb{R}$;
- An *information partition* $h(x)$ which defines for each node x, the set of nodes that are possible given what player $i(x)$ knows. Thus, if the node $x'$is known to be reachable from the current node x, or $x' \in h(x)$, then the player $i$moving at node x, or$i(x)$, can be the same player moving at node x'. It follows that if $x' \in h(x)$, then $i(x')=i(x), A(x')=A(x)$, and $h(x')=h(x)$;
- The set of information sets available when player i moves from position *x*: $H_i=\{S \subset X: S=h(x) \mid \exists x \in X, i(x)=i\}$;

- The set A_i of actions available to i at any of his information sets $h \in H_i$.

3.2.3 Normal-Form

A game in normal-form is a structure $\Gamma=\langle N, S, F\rangle$, where:

- $N = \{1,2,...,n\}$ is a finite set of players;
- $S = \{S_1, S_2, ..., S_n\}$ is an *n-tuple* of *pure strategy sets*, one for each player;
- $F = \{F_1, F_2, ..., F_n\}$ is a tuple of payoff functions, one for each player which maps strategies to rewards.

This definition has been included for completion. Normal-form games are conceptually useful for identifying certain equilibria, but require additional computational overhead for more refined solutions. For efficiently solving integrity games, they rank below their more preferred *succinct-form* (most ideal) as well as their *extensive-form counterparts*. Hence, further discussion regarding this type of game specification is omitted.

3.2.4 Succinct-Form

Games in succinct-form often allow for smaller representations than normal-form. Describing a game of n players, each facing s strategies, requires a listing of ns^n utility values. In games where symmetry (of information, actions, strategies, payoffs, etc.) is exploited, a combinatorial reduction in the number of utility values is possible [Fortnow, 2005].

For 2-player integrity games, the *base game* can be succinctly modelled by n^2 utility values, where n in this case is the maximum number of moves, actions, choices, or state-transitions possible at any round. The succinct-form directly coincides with the $K_{(v,e,n)}$asset representation discussed in Sections

Special Graphs and The $\boldsymbol{K}_{(\mathbf{1},\mathbf{2},\boldsymbol{N})}$Asset Representation. For 2-players, $K_{(1,2,N)}$ forms a base integrity game in succinct-form. This specification is defined by N^2 utility values (one for each potential state-transition), usually given by some $N \times N$ weighted adjacency matrix.

3.2.5 Strategies

3.2.5.1 Pure Strategies

A *pure strategy* for player i in an extensive-form game is a function $s_i : H_i \rightarrow A_i$ such that $s_i(h) \in A(h)$ for each $h \in H_i$. A *strategy* is a complete plan explaining what a player will do in every situation. It represents a sequence of action selections. Let S_i denote the set of pure strategies available to player i, and let S=$\{S_1 \times S_2 \times \ldots \times S_I\}$ denote the set of pure strategy *profiles* $\forall i \in I$. Similarly, let the set s=$(s_{1,\ldots}, s_I)$ denote a particular strategy profile, and let s_{-i}denote the strategies of $i's$opponents.

3.2.5.2 Mixed Strategies

In an extensive-form game, a *mixed strategy* σ_i for player i is a probability distribution Δ over the set of $i's$ pure strategies S_i, or $\sigma_i \in \Delta(S_i)$.

3.2.5.3 Behavioural Strategies

A *behavioural strategy* for player *i* in an extensive-form game is a function $\sigma_i{:}H_i \rightarrow \Delta(A_i)$ such that $support(\sigma_i(h)) \subset A(h), \forall h \in H_i$.

3.2.5.4 Kuhn's Theorem and Perfect Recall

A classic result in game theory, known as *Kuhn's Theorem*, states that in a game of *perfect recall*, i.e. where players may remember all their previous moves/states as well as their previously encountered information sets, then for any mixed strategy there is an equivalent behavioural strategy. Thus, these terms are often used interchangeably.

3.2.6 Solution Concepts

In game theory, a *solution concept* is a formal rule for predicting how a game will be played [Leyton-Brown and Shoham, 2008]. These predictions describe which strategies will be adopted by rational agents, and therefore constitute a "solution" for the result of the game.

3.2.6.1 Nash Equilibrium (NE)

Recall that S=$\{S_1 \times S_2 \times ... \times S_I\}$ is the set of pure strategy profiles, with $S_{i \in I} \in S$ the set of all profiles for player i. The resultant payoff function for some strategy profile$s \in S$ is given by f=$(f_1(s),..., f_I(s))$. Also recall that s_i and s_{-i} denote the strategy profiles for player $i$and all of his opponents, respectively. The payoff function f depends entirely on the strategy profile$s$, which represents the strategy chosen by player i as well as all the other players $-i$.

A strategy profile $s^* \in S$ is said to be a *Nash Equilibrium (NE)* if no unilateral deviation in strategy by any single player is profitable for that player. Formally, $NE \rightarrow \forall i, s_i \in S_i : f_i(s_i^*, s_{-i}^*) \geqslant f_i(s_i, s_{-i}^*)$. When this inequality holds strict for all players, the Nash Equilibrium is said to be strict. When $\exists s_i^* \in S_i : f_i(s_i^*, s_{-i}^*) = f_i(s_i, s_{-i}^*)$, the Nash Equilibrium is said to be *weak*. *NE* can exist for either pure or *mixed* strategies. In 1951, Nash showed that for every game with a finite number of players, in which every player can choose from finitely many pure strategies, there must exist at least one (possibly mixed) Nash Equilibrium [Nash, 1950]. As a solution concept, NE is more commonly found within the context of normal-form games.

3.2.6.2 Subgame Perfect Equilibrium (SPE)

This solution concept is a *refinement* (or subset) of the classical Nash Equilibrium. Refinements enforce stricter conditions on the optimality of behaviour, and impose greater requirements on the rationality of players. *Subgame perfection* posits that players will always seek a Nash Equilibrium going forward even if some off-equilibrium play was observed. By definition, a *subgame* Γ' of some extensive-form *game* Γ, consists of:

- A subset Y of the set of nodes X, where Y is rooted by a single non-terminal node x, and contains all of $x's$ successors, and;
- Y has the property that if $y \in Y$ and $y' \in h(y)$ then $y' \in Y$;
- Γ' shares the same information sets, feasible moves, and payoffs at terminal nodes as Γ.

Subgame perfect equilibrium therefore prescribes a NE to be played at each subgame. As a solution concept, SPE is more commonly associated within the context of extensive-form games.

3.2.6.3 Minimax Theorems

In their most general form, *minimax theorems* are *fixed-point* theorems from *variational* analysis [Ricceri and Simons, 1998]. Under broad conditions, a dynamical system, and hence most game structures, will admit an approximately stable saddle region. The *saddle* is effectively a *fixed-point solution* to the game (or dynamical system) which can be found using various techniques [Ricceri and Simons, 1998].

A particular example is known as the *max-min inequality.* Given a real-valued function over some cross-product of compact vector spaces, $f: X \times Y \to R$:

$$\sup_x \inf_y f(x,y) \leq \inf_y \sup_x f(x,y) \qquad 3.1$$

Which holds $\forall x, y, f(x,y) \in \mathbb{R}$. In 1928, Von Neumann studied the problem of solving "games of strategy" (*gesellschaftsspiel),* and produced the following result, known as the *Von Neumann Minimax Theorem (VNMM)*:

$$v = \min_X \max_Y X^T AY = \max_Y \min_X X^T AY \qquad 3.2$$

Where *v* is called the *value of the game*, X, Y are the *mixed strategy solutions* for players *Min and Max*, and *A* is the *payoff matrix* of the game in normal-form.

Along with NE, the VNMM is one of the most fundamental theorems of game theory. It states that every finite, two-player, zero-sum game (2PZSG) must possess

optimal mixed strategies. This optimality is achieved under minimax conditions and is known as the value of the game. Proofs can be found in the original papers [Von Neumann, 1928], [Von Neumann and Morgenstern, 1944].

3.2.7 Transformed Minimax Potential

Any finite N-player general-sum game (NPGSG) can be transformed into an equivalent (N+1)-player zero-sum game, or (N+1)ZSG. This can be achieved by invoking an arbitrator or referee player [Cai and Daskalakis, 2012]. This process creates a *global potential function* out of the rewards or payoffs, summing them to zero. Additionally, strictly competitive situations can be *partitioned*, where a single player is isolated from the remaining contingent of players. This contingent forms an aggregate non-cooperative force [Cai and Daskalakis, 2012]. The combination of these transformations can reduce many situations into an equivalent set of problems which are more readily solved. We summarize this process by the following implication diagram(s); NPGSG → (N+1)ZSG → 2PZSG*.

The resulting game, 2PZSG*, is also called the *transformed minimax potential* of the game (or dynamical system). Structurally, this is a finite, two-player zero-sum game where the payoffs follow a potential reward function that attenuates (or discounts) in long-run expectation towards zero. The two players are the net resultants of a binary partition, condensation, or mean-field approximation of the actors. They represent a duality of forces, the *Minimizer* and the *Maximizer*, or MIN and MAX. These players seek to strictly oppose one another in perfect competition. Each player has the fundamental objective of minimizing (or maximizing) their respective payoffs.

In 2PZSG*, payoffs are typically captured by a *Von Neumann-Morgenstern Utility functio*n (VNM-Utility). These payoffs are complete, transitive, monotonic, and necessarily risk-averse [Bergstrom, 2014]. It is presumed that agents will seek to maximize (or minimize) their long run expected VNM-utility. This is both an *admissible* criterion for decision making, as well as a notion of *rationality* between agents [Bergstrom, 2014]. Both *VNM-utility* and *VNM-rationality* are considered somewhat artificial, as humans are rarely capable of this behaviour [Kreps, 1988]. Furthermore, this so-called VNM-behaviour is seldom observed in practice [Kreps, 1988]. By default, we will assume that in 2PZSG and 2PZSG* players *attempt* to adopt VNM-like behaviour.

When the transformed minimax potential respects VNM-behaviour, several solution concepts become equivalent. The problem of finding a minimax solution is called MINIMAX, and the problem of finding a maximin solution is called MAXIMIN. These concepts also have dual representations in linear programming, where finding a solution is called LP. The problem of finding a pure strategy NE is called NASH, although sometimes the mixed strategy NE are used interchangeably [Nihan, Roughgarden, et al., 2007]. It can be shown that for any symmetric 2PZSG, and by proxy any 2PZSG*, the following implication is true; MINIMAX = MAXIMIN = NASH = LP.

Additionally, if more than one optimal mixed strategy exists, then there are infinitely many optimal mixed strategies [Leyton-Brown and Shoham, 2008]. At least one pure strategy solution (such as a NE) is guaranteed to exist, and there may exist several pure strategy solutions. For a thorough discussion of these results equipped with proofs,

the reader may again be diverted to the works of [Leyton-Brown and Shoham, 2008] and [Nihan, Roughgarden, et al., 2007].

While not always valid or appropriate, the 2PZSG*, or transformed minimax potential, is a reduction which allows for simplified analysis. It consequently assumes VNM-behaviour in utility and rationality. As a zero-sum *potential game*, it maintains the notion that one player's gains are balanced by losses to the other players as a whole. In the two-player sense, this yields values for the game which can be seen as saddle-point solutions. These transcend the game specification; as Nash Equilibria in normal-form games, principal variations in extensive-form games, or minimax trajectories for succinct-games.

3.3 The Perturbed Game

3.3.1 Specifications

As with its *base game* counterpart, a *perturbed game* can be specified in one of several formats. However, there is less interchangeability between specifications and solution concepts for *perturbed games* [Jackson et al., 2011]. Since we are dealing with sequential planning of generalized assets, only the extensive-form variation of perturbed games will be covered. We will later adapt extensive-form solutions to a succinct-form as part of an overall model for solving integrity games.

3.3.2 Perturbations of the Base Game

Let Γ be a standard extensive-form *base game* as defined in Section Extensive Form. A *perturbed game* ~Γ is a copy of the base game where *every pure strategy* is played with non-zero probability. Thus $\forall i \in I, \forall s_i \in S_i, Pr(s_i) \geq 0$, and ~Γ can be interpreted as a restriction on playing *only totally mixed strategies* $\sigma_i \in \Delta(S_i)$.

In their most basic form, perturbations to the base game structure result in all strategies (i.e. sequences of actions) being "on the table" without regard to how sub-optimal they may be. A *dominated* strategy in the base game is technically feasible throughout the perturbed game. There are several reasons for advocating that strategies be *totally mixed* [Leyton-Brown and Shoham, 2008]. The examination of perturbed gameplay (over some base game) is a common technique for assessing the *stability* and *robustness* of equilibrium

solutions to noise. Perturbations also capture modelling errors, misjudgment of states and beliefs, as well as behavioural imperfections. A summary of these motivating factors is presented in Table 3.2.

Table 3.2. Sources of Perturbation to the Base Game

Nomenclature	Source Class	Effects(s)
Incomplete Information	– Partially-Observable States. – Unknown/Uncertain Player Motivations.	– Misinterpretation of game state. – Errors in utility and payoff functions. – Uncertainty in opponent beliefs, motives, available actions. – Emergence of unexpected patterns.
Imperfect Actions	– Non-deterministic Actions and/or Selection Mechanisms. – Control Costs.	– Incorrect moves and/or improper action-selection, despite perception of correctness. – Deviations from expected state-transitions and/or payoff results.
Misc. Modelling Errors	– Unknown Actors, Rules, Environments. – Approximating Assumptions. – Infeasible or unable to capture real-world complexity.	– Misrepresentation of the situation through incorrect (imprecise, inaccurate) game structures. – Incorrect, contracted, or insufficient analysis. – Sub-optimal or even massively detrimental decision making.

The study of perturbed games leads to several *refinements* in the interpretation and prediction of rationally "correct" play. These refinements are often motivated by arguments from *admissibility*. Admissibility criteria require decision rules which are not dominated by alternatives in the sense of some estimator such as Bayesian expectation or some coherent risk/loss function [Kreps, 1988], [Leyton-Brown and Shoham, 2008]. For two-player games, *admissibility* implies that no strategy s_A that is (weakly) dominated by another strategy s_B is legally allowed to be played.

That is, for any player i, a strategy $s^* \in S_i$:

- *Weakly dominates* another strategy $s' \in S_i$, if $\forall s_{-i} \in S_{-i}\{u_i(s^*, s_{-i}) \geq u_i(s', s_{-i})\} \wedge \{\exists s_{-i}{:}u_i(s^*, s_{-i}) = u_i(s', s_{-i})\}$;
- *Strictly dominates* another strategy $s' \in S_i$, if $\forall s_{-i} \in S_{-i}\{u_i(s^*, s_{-i}) > u_i(s', s_{-i})\}$.

In perturbed games, the standard solution concepts seek to implement strategies which tolerate deviations from base game equilibrium behaviour (such as NASH or MINIMAX). The limits of this *tolerance* typically involve some notion of admissibility, or remaining undominated in the face of perturbations. Refinements may also be defined from other desirable properties, such as the *preservation of inherited inference*, or by way of *forward/backward induction*. This has led to increasingly stronger refinements over the *subgame perfect equilibrium*. Examples include the *sequential equilibrium* proposed by [Kreps and Wilson, 1982], as well as the *proper equilibrium* of Myerson, the *Markov perfect equilibrium*, and the concept of *Mertens Stability* [Nisan, Roughgarden, et al., 2007].

3.3.3 Extensive-Form Trembling Hand Perfection

One of the most significant refinements is the *trembling hand perfect equilibrium*, a solution concept first proposed by Selten (1975). Trembling hand perfect equilibrium takes into account the possibility for deviations from equilibrium as a result of "trembling hands". Under this regime, players have *perfect recall* of their previous actions, but fumble certain individual moves, and may (with small probability) choose unintended strategies

for the remainder of the game. There are differing, incomparable notions of normal-form and extensive-form trembling hand equilibria [Jackson et al., 2011].

Formally, a (mixed or behavioural) strategy profile σ is an $\epsilon - perfect$ *equilibrium iff* it assigns strictly positive probability to all pure strategies, and only pure strategies that are best replies get probability greater thanϵ. A (mixed or behavioural) strategy profile$\sigma$is then an *extensive-form trembling-hand perfect equilibrium iff* it is the limit point of a sequence of $\epsilon - perfect$equilibria with $\epsilon \to 0+$.

The notion of $\epsilon - perfect$ equilibrium maintains that for mild perturbations of the information sets away from the complete information of the base game, one can expect correspondingly mild perturbations in best reply behaviour. Through trembling hand perfect equilibrium, one may recover any subgame perfect equilibria which vanished as a result of perturbations less than ϵ. To do so, it merely assumes "perfection" (or optimality) in the sequential responses to successive perturbations of the game structure.

3.3.4 Robustness, Stability, and Adaptability Concepts

It is worth noting that in perturbed games, players *can* be better off by ignoring some of the information potentially available to them [Jackson et al., 2007]. For real-world assets, players typically incur additional control costs to gain knowledge or refine their beliefs of the asset-environment system. For example, paying (e.g. trading energy) to reduce the uncertainty and/or noise associated with an observation, shaping an inspection to be of higher fidelity, or forcing probabilistic guarantees on a maintenance action. In

such cases, the relationship between the value of ϵ and the expected equilibrium payoffs can have profound effects. There can exist correspondences between ϵ, σ, and u, where all players benefit from relaxing their knowledge of the game structure. This can be understood as having players face the costs of discovering information vs. the costs of fully tailoring strategies to imperfect information, such as $\epsilon - approximate$ uncertainty tresholds. This is in addition to the other justifications (c.f. Table 3.2) that are based on players (or the modeler) incorrectly perceiving states, actions or payoffs.

Approaches based on this model vary, but are broadly referred to as $\epsilon - robust$ or $\epsilon - stable$ methods. These methods essentially seek large basins of attraction, which are considered to be of lower risk and more preferable then narrow yet higher-performing corridors of play. They emphasize state-transition trajectories which remain *stable* (minimal variance) as the risk landscape is perturbed. They also anticipate *robust* and *securable* payoffs in the face of uncertain information. This is in contrast to the principles behind the so-called *adaptive methods*, which seek to continuously exploit discernable changes in the game structure. Adaptive methods expropriate feedback, and the belief that other players possess fewer computational resources and/or will exhibit fewer information gains as the game evolves. Roughly speaking, adaptive methods are appropriate when opponents will make many more "mistakes", and these mistakes will be noticed and can be exploited. A detailed, formal treatment of these ideas can be found in [Bowling and Veloso, 2000].

To recapitulate, solution concepts which are *adaptive* work well in *asymmetric games*, where (i.) some players have an inherent structural advantage from which to launch adaptive strategies, and (ii.) perturbations preserve this asymmetry overall (and hence the advantage). In these games, several opponents may be considered to have significantly greater (or fewer) resources. There may also be more (or less) information available at each round, or some subset of players must pay more (or less) for it. The disadvantaged players may also possess greater uncertainty in their knowledge of states and actions. They may be restricted in their action sets and possibly subject to move penalties. Disadvantaged players may also exhibit irrational behaviour through access to fewer computational resources or some natural disposition towards sub-optimality.

This asymmetry is sometimes indirectly captured parametrically by $\gamma \in [0,1]$, which expresses the level of non-cooperation present in the game structures Γor $\sim\Gamma$. When γ is small, the level of competition is small. The parameter γ can also be interpreted as a *hostility index* for the environment.

Limiting the ruleset (i.e. the resources and actions available to players) may handicap even the most diligent and computationally rational opponents. When the game structures Γ or $\sim\Gamma$ are sufficiently asymmetric, the opportunity for *strategic potency* is likely to be reduced. This corresponds to a significant shift in the location of saddle-point equilibria or even their complete degeneration (c.f. *games with no value* [Sion and Wolfe, 1957]). As $\gamma \to 0$, the asymmetry and resulting discrepancy between player abilities is

maximized (in expectation). At this point it may make sense to relax the assumption of adversarial dynamics completely, reducing the game to that of a one-player optimal stochastic control problem [Filar and Vrieze, 1997]. In these circumstances, small or even time-dynamic values of γ appreciably close to zero will emphasize solutions which anticipate stochastic deviations, and attempt to classify environment types by their generating distribution(s). These conditions are ripe for adaptive techniques, which will inevitably exploit discovered asymmetries and increase the overall game performance, as it represents planning under naive environments.

As $\gamma \to 1$ we achieve symmetry and opponents are considered strictly competitive. This represents the situation of intelligent adversaries with full-scale resources, equivalent rulesets, and equal-and-opposite objectives. In these circumstances, solution concepts based on stability and robustness are preferred. Thus for large values of γ, we seek trajectories which are designed to survive against, or in an evolutionary sense *avoid disappointment, regret, or invasion* by, any reasonable perturbations of the landscape. For extensive-form games, this leads to the acceptance of trembling-hand perfect (or $\epsilon - approximate$ equivalent) equilibrium solutions over their adaptive counterparts. For sequentially perturbed base games, this approach remains valid insofar as inductive inference is preserved. It can be shown that backwards induction is preserved for games of *perfect recall*, as well as for games with the *Markov property* [Filar and Vrieze, 1997].

In perfect recall games, a complete history of state-transitions is available and in the worst-case *ex ante* responses to perturbations are admissible. In Markov games, future

state-transitions are history independent and depend on the current state (i.e. memoryless). In both cases, induction can be applied, yielding a set of appropriately stable and/or robust equilibrium solutions. This is often accomplished via a variant of minimax search over the fitness of outcomes; e.g. taking into account the probabilistic "quality" of expected outcomes by evaluating moments such as the mean and standard deviation [Cai and Daskalakis, 2012].

3.4 A Succinct Integrity Game for Generalized Assets

3.4.1 Problem Description

Consider the situation of an *asset manager* responsible for making decisions regarding the performance and utilization of a generalized asset. The generalized asset can be of any type, but consider for the moment a graph-defined state-transition system. Several graph performance indicators have been used to construct a series of reference configurations for the asset. These configurations have been ranked according to their overall fitness level and partitioned by equivalence class. The asset manager has determined the utility generated by maintaining the asset near a particular configuration for an entire time-step. Using the normalized compression distance as a similarity metric, the costs of transforming configurations are deducted from the utility. This forms an *integrity score*, which represents the net payoffs or rewards from transitioning between the asset reference configurations, *labelled as integrity states*.

The problem facing the asset manager is how to optimize the performance of the asset in the absence of any additional information. No domain knowledge is available, and the coupled asset-environment system is considered sufficiently novel and complex so as to rule out the feasibility of using expert assessments. Where possible, procedures which "learn" to filter unwanted noise have already been applied. Any process of reducing uncertainty is bounded and converges long after the asset is expected to be operational. The design lifetime of the asset is unspecified but known to be finite. The asset is expected to persist in an unknown but hostile environment throughout lifecycle. Environmental

effects include bounded stochastic integrity disruptions as well adversarial dynamics. The asset is deemed sufficiently important to warrant the attentions of intelligent opponents, about which very little is known in terms of resources and/or capabilities.

The asset manager is tasked with formulating state-transition plans which maximize the integrity score over some long-run sequence of time-steps. This ultimately corresponds to navigating a dynamic integrity landscape with the objectives of maximizing utility while minimizing risks. Translated, the asset manager must prescribe state-transition trajectories which pursue desirable configurations while avoiding unwanted ones. Furthermore, these trajectories must be optimal in that over-time, they accumulate the greatest possible integrity score in the presence of perturbations owing to noise, modelling errors, and unforeseen deviations.

Before the asset is made operational, a series of expected long-run average integrity scores are given as payoffs to a base game in succinct-form. The succinct-form integrity scores correspond to an array of length quadratic in the number of distinct integrity states. For simplicity, these scores are positive integers. At each time-step, the asset is monitored and noisy data regarding its configuration are extracted. The expected integrity scores for the remainder of the game are evaluated, with one score assigned to each potential one-time-step state-transition. The array is then updated and given to the asset manager for analysis, planning, and governance.

3.4.2 Formulation

This problem is correctly modelled as a sequentially perturbed two-player zero-sum game (~2PZSG). In the worst-case, all actors (including the environment) conspire against the asset manager. Since this scenario is not explicitly ruled out, we can generate a transformed minimax potential resulting in a 2PZSG* base game with perturbations arriving at each time-step. This requires a certain amount of symmetry in the game structure and posits Von-Neumann behaviour (VNM-rationality, VNM-utility). We must naively assume the capabilities of all agents are approximately equal (at least in long-run expectation), thereby massaging the game into ~2PZSG* form.

Because the asset evolves forward in time and partial information feedback is present, we presume an ordering of moves. Therefore, any simultaneous play is "accidental" and a normal-form specification would be inappropriate. While extensive-form is applicable, the problem suggests the use of succinct representations which are updated at each time-step. With no explicit move ordering given, we may assume two scenarios: (i) the asset manager has at his disposal the "first-move" in which a starting state can be specified and decided, and (ii.) the adversary decides the initial state of the problem. Since the initial design of the asset is very likely to be stipulated in advance, the base game is assumed to commence with the asset manager having already made the "zeroth move". Ideally, we would seek methods for a rapid analysis from any/all start state(s) to any/all end state(s).

In the next chapter, we detail a working model and provide algorithms for this entire problem. In the Appendices, we provide several engineering-related examples. These examples illustrate the process of problem formulation and analysis using the aforementioned methods.

3.4.3 Preliminary Analysis

The ~2PZSG* class efficiently models the generalized asset integrity problem. These "succinct integrity games" can be analyzed by taking the limits of the game structure. This can be done with respect to several properties including the robustness, stability, and adaptability concepts outlined in section 3.3.4.

For this purpose, one may consider the *adversarial* index γ, which represents the degree of non-cooperation in the actions of the opponent and/or environment. This parameter expresses the qualitative risk(s) of deviating from equilibrium play in lieu of incomplete beliefs, irrational behaviour, model deterioration, mistake probabilities, and/or errors incurred by computational and/or analytical limitations. By *deviation risk* we mean the product of the *frequency* and *intensity* of deviations from nondeterministic best-response behaviour (optimality conditions). By itself, γ captures the environmental severity, look-ahead, and overall competitive intelligence. For small values of γ, the environment is *naive* and *harsh*, while for large values it is *adversarial* or *hostile*. Alternatively, $1/\gamma$ can be thought of as a measure of random move generation on behalf of the environment.

The *payoff perturbation threshold* $\epsilon_{payoffs,max}$, expresses an asymptotic upper bound in the magnitude of one-stage deviations to the payoff functions. For simplicity we will write $\epsilon = \epsilon_{payoffs,max}$ and let u_k be some payoff function which maps state-transitions to utility values. Let (v_s, v_t)denote a transition as a source-destination pair of states. Then the resulting map $u_k{:}(v_s, v_t)$ gives the integrity score awarded immediately after transitioning from integrity level v_s to v_t at the end of time step k. The perturbation threshold ϵ therefore sets the absolute minimum and maximum one-stage payoff deviation limits. So for $u_{k+1}:(v_s, v_t)$, we have $u_k - \epsilon \leq u_{k+1} \leq u_k + \epsilon$.

The *average signal-to-noise ratio* $D_{sn} = \mu_u \mu_\epsilon$, expresses the degree of pronunciation and discernibility in the *average payoffs* μ_u with respect to the *average perturbations in payoff*s μ_ϵ. When perturbation thresholds exceed payoffs, there is very little controllability over game outcomes and the effects of the adversarial index are reduced. When perturbations vanish, the payoffs remain effectively fixed and the adversarial index dominates any decision making. Thus D_{sn} affects the long-run sensitivity of the base game $\Gamma$to changes in γ and ϵ. We note that D_{sn} drives the interval limits of dynamic range for perturbed games: $\sim\Gamma_{min}(D_{sn}) \leq \sim\Gamma(u, \gamma, \epsilon) \leq \sim\Gamma_{max}(D_{sn})$.

Variability in these game parameters entices an examination of several limit cases. For a brief analysis, we consider the succinct integrity game structure $\sim\Gamma(u, \gamma, \epsilon)$, and supply it with four boundary scenarios.

3.4.3.1 Scenario 1: $\boldsymbol{\gamma \to 0, \frac{\epsilon}{u} \to 0}$

In this scenario there is a total lack of competition. Additionally, the effects of payoff changes are vanishingly small. The only integrity antagonists are the naive effects of a stochastic environment. This situation allows for a direct stochastic optimization. Depending on the context, the objective is to find the minimum or maximum of some function of the payoffs u, subject to interval constraints. Without adversarial resistance *or* payoff perturbations, VNM-behaviour on the part of the asset manager is trivial to maintain. This VNM-behaviour as a criterion, is akin to recognizing that many versions of this problem are frequently treated (or posed) in a manner that are *convex* [Schoenebeck and Vadhan, 2006].

At each time interval, the asset manager selects the action which offers the highest long-run (possibly discounted) expected payoff in the base game. This process continues indefinitely or until termination criteria are met.

3.4.3.2 Scenario 2: $\boldsymbol{\gamma \to 0, \frac{\epsilon}{u} \to \infty}$

This scenario arises when the process of integrity degradations emanates from non-competitive sources. However, in this case the effects of noisy payoffs are made appreciably large. The problem statement emphasizes that for large perturbations, any filtering or learning of the payoffs cannot be accomplished within a finite horizon setting. Under these conditions, the signal-to-noise ratio approaches zero, and there is no control over the payoff structure. Any strategic consistency is based on random play (the ability

to conjunction form of true random play). In this scenario, the net costs of integrity restorations will be essentially random. The asset will generate a long-run utility as a function of the stochastically realized state-transition sequence.

3.4.3.3 Scenario 3: $\boldsymbol{\gamma \to 1, \frac{\epsilon}{u} \to 0}$

In this limit case, the adversarial index tends towards one while simultaneously maintaining a maximal signal-to-noise ratio. The asset integrity score is actively denied under perfect competition from an intelligent opponent. With vanishingly small noise effects, the situation becomes one of deterministic 2PZSG dynamics. A suitable Nash equilibrium, minimax trajectory, or principal variation will always exist. If each player is guaranteed to: (i.) play perfectly (in a deterministic sense), (ii.) possesses equal and opposite beliefs, actions, and utility payoffs (i.e. complete game symmetry), then this solution concept forms what is essentially a "nemesis contract" between players.

3.4.3.4 Scenario 4: $\boldsymbol{\gamma \to 1, \frac{\epsilon}{u} \to \infty}$

Here, the adversarial index approaches one while the magnitude of the perturbations greatly exceed the payoffs. The environment again consists of intelligent adversaries. These antagonists compete with the asset manager to deny any long run expected utility generated by the asset. This process is manifested by an opposition to any accumulation of the integrity score. However, the presence of uncontrollably large noise effects ensure the state-transition sequences will be realized non-deterministically. Since learning the generating distribution for the noise effects is strictly unreliable, both players are again

forced to play a kind of the sequential best-response equilibrium. The resulting game dynamics involve piecewise strategy reformulations through ongoing replanning and adaptation at each round, or when applicable (e.g. viz change-detection). This situation results in sequential re-evaluations of the expected minimax trajectory or principal variation. When adaptive techniques and/or learning routines are available, convergence to some initially unknown generating function of the perturbations may be possible. In this case, the asset manager may form a set of beliefs which affect trajectory assessments. These are based on estimates of the historical, current, and future integrity states. Based on the proximity to termination criteria (such as failure risks or resource expenditures), the asset manager will prescribe an immediate action with concern for short-term survival and long-run security. In this scenario, the net costs of integrity restorations will be essentially random. The asset will generate a long-run utility as a function of the stochastically realized state-transition sequence.

3.5 Conspectus

This chapter presented a self-contained review of several game-theoretic concepts. A brief summary of the background, context, and terminology was provided in order to relate game-theory to the theory of control and general dynamical systems. An ontogenesis and classification of various game studies was undertaken in order to demonstrate a thorough literature review of the subject matter. This chapter emphasized two major game structures, namely a base game and a perturbed game. For each game type, the overall structure and its representation were discussed. An outline for the normal-form, extensive-form, and succinct-form game specifications was coupled with definitions of pure, mixed, and behavioural strategies.

Several solution concepts and major theorems were also reviewed. These were limited to the Nash Equilibrium, Subgame Perfect Equilibrium, Kuhn's Theorem, as well as the minimax theorem. A methodology for approximating general game types through a conservative transformation to a two player game was also provided. Emphasis was placed on Von Neumann rational behaviour and utility. For the perturbed game, the extensive-form trembling-hand equilibrium was defined. The importance of robustness, stability, and adaptability concepts were also discussed.

Finally, the chapter concluded with a scenario formulated as a succinct integrity game. In this more appropriate game description, an asset manager seeks to optimize the integrity score of a (generalized) asset. Several modelling parameters were introduced, and a qualitative analysis of the limit cases was provided. This chapter abridges much of the

theoretical foundations for the remaining work, which concentrates almost exclusively on implementation details and practical applications.

4 ARCHITECTURE AND IMPLEMENTATION

Generalized asset governance can be developed into working architectures for autonomous planning. Successful implementations should prescribe a sequence of actions which optimize the utility generated by the asset. For generalized assets, domain-oblivious reward concepts advocate the need for payoff structures evaluated from graph fitness indicators and information similarity metrics. This is a situation of dynamic performance optimization, which can be reformulated as a *~2PZSG** type game known as a *succinct integrity game*.

This chapter compiles the results from previous chapters into a Generalized Asset Integrity Game Engine (GAIGE). At its core, the GAIGE executes a modified minimax search algorithm which exploits a generic problem structure. Arguments from combinatorial symmetry, dynamic programming, sequential optimality, and backwards induction are used to establish a model transposition equivalence which returns minimax trajectories in linear time, $O(N)$. This result constitutes an online, reactive planner which can be augmented in $O(\sqrt{T})$ time through the use of nearline methods such as test drivers, sampling, and bandit algorithms. The combination of online and nearline algorithms deliver a hybrid (heterogeneous), anytime evaluation procedure. The GAIGE is shown to be capable of prescribing epsilon-approximate trembling-hand-perfect strategies. A discussion on how to solve perturbed integrity games using the GAIGE is also presented.

The GAIGE targets several objectives and requirements at both the architectural and algorithmic levels. The remainder of this chapter addresses the development, operation and performance of a GAIGE implementation. Prototypical use-cases are also tested and benchmarked.

4.1 Objectives and Requirements

Autonomous planning can be engineered to meet design goals, objectives, requirements, specifications, and constraints. These stipulations manifest themselves at multiple scales. A separation of concerns establishes two primary layers of abstraction, (i.) the architecture, and (ii.) the algorithmic implementation. The former specifies the overall framework and organization of the process, while the later details a particular realization or construction.

The literature on artificial intelligence, search, and planning systems provides a basis for the development of guidelines and expectations. However, the vast majority of technological engagements draw from the theme of (optimal) policy and stochastic control [Kaelbling et al., 1998]. At the time of this writing, the notions of generalized assets, their (integrity) governance, and general game playing agents are still emerging [Genesereth et al., 2005], [Kiekintveld 2008], [Finnsson and Bjornsson, 2010]. Unfortunately, adoption levels within the physical and industrial asset integrity communities have so far remained low. Nonetheless, many of these emerging research areas have active annual workshops and related journals. For example, the *General Intelligence in Game-Playing Agents (GIGA)* proceedings, or the *Ontology Modeling in Physical Asset Integrity Management* publications of [Ebrahimipour and Yacout, 2015]. This section anticipates their eventual usage in the field of (physical or industrial) asset integrity management, and elucidates a set of desirable characteristics. The proposed requirements emphasize solutions which offer the greatest range of application for the lowest overall complexity.

4.1.1 Architectural Requirements

Performing autonomous planning typically mandates high-performance architecture. The minimum capabilities vary according to the objectives. With respect to generalized asset integrity games, the primary objectives are to prescribe strategies and support efficient gameplay. Architecturally, this requires at least some degree of modularity, along with the integration of numerous components (c.f. massively modular and parallel distributed processor architectures, which are both instances of reasoning over graphs). For succinct integrity games, it is enough to require a three-phase architecture consisting of monitoring, evaluation, and prescription stages.

In keeping with the impetus of this thesis, we tread through several architectural requirements and summarize their practical benefits. Table 4.1 presents a simple three-phase planning architecture. Table 4.2 offers a high-level summary of the semantic and non-functional requirements for such architecture.

Table 4.1. A three-phase architecture with sub-components for autonomous planning

Phase	Modular Components, Tasks and Responsibilities	Stage
Monitoring	Sensory Activation and Acquisition.	1
	Signal Pre-processing, filtering, conditioning.	2
	Production of raw input stream.	3
Evaluation	Generate Live/Active Asset Configuration	4
	Query Reference Configurations	5
	Evaluate Integrity Scores	6
	Perform Minimax Transposition Search	7
	Produce Minimax Tableau	8
	Yield trajectories/strategies	9
Prescription	Query offline knowledge oracle (deliberation base)	10
	Estimate Deviation Risks and Stability Beliefs	11
	Formulate and Select Actions	12
	Output Prescription Results to Effectors/Actuators	13
Repeat	Return/Retrieve Monitoring Data	14 → 1

Table 4.2. Semantic and Non-Functional Architectural Requirements

Requirement	Type	Description	Benefits
Domain-Oblivious	Semantic	Does not discriminate based on asset class, prior knowledge, or information context.	Remains valid across multiple scopes and domains.
Platform-Agnostic	Semantic	Does not depend on any specific computational setting or hardware environment.	Improves ease of deployment. Portable across machinery types.
Model-Driven	Semantic	Driven by direct model contact, embedded descriptions, and structural configurations.	Reduces dependence on big data aggregations. Avoids data-centric processes and large throughput operations. Simplifies analytics, reduces overhead and latency.
Non-Brittle	Non-Functional	Does not require retrofitting across projects. Minimal parametric tuning.	Reduces engineering rework and redevelopment.
Graceful Degradation	Non-Functional	Retains limited functionality, self-stabilizes, and is fault-tolerant.	Reduces external dependencies. Avoids or contains catastrophic failures.
Scalable	Non-Functional	Acceptable scaling laws. Latency, throughput, and robustness capabilities will not scale disproportionately with additional resources.	Reduces the risks of scope drift. Predictable long-run management and life-cycle costs. Expands/contracts to suit future needs.
Accessible	Non-Functional	Interoperable and versatile use cases. Supports diverse user types and experience levels.	Accommodates and empowers a range of potential users. Lowers the barrier to entry. Hastens learning curve.
Transparent	Non-Functional	Open to study, diagnose, test, modify, and customize.	Accommodates variable skill levels, reverse engineering, and more advanced usage.

Additionally, several functional requirements are prevalent for each of the major architectural components. Assuming a three-phase approach to planning, these components can be monitoring, evaluation or prescription based. Each architectural component is subject to several well-defined, technical capabilities. The demands placed on automated planning architectures vary greatly [Ghallab et al., 2004]. We follow a set of functional demands from a predominantly non-parametric, non-Bayesian paradigm. We term this approach *hypermodern*. The hypermodern requirements somewhat juxtapose the more classical demands found in the probabilistic setting, from which are associated the Markovian and Bayesian decision agents.

The inherent prevalence of Bayesian approaches (with respect to multi-agent decision making) have led to the consideration of cognitive, behavioural, and non-Bayesian revision or rule-update schemes. In these non-Bayesian settings, agents may use simple rules such as linear or convex combinations of information. Results derived directly from Bayesian and other probabilistic approaches are considered robust when the number of possible outcomes is finite, and the number of marginals of the data-generating distribution(s) are unknown [Owhadi et al., 2015a]. However, Bayesian approaches are also known to be generically brittle. In particular, any given prior and model can be slightly perturbed to achieve any desired posterior conclusions [Owhadi et al., 2015b]. The mechanisms causing *shattering*, *brittleness* and *robustness* suggest that Bayesian learning and robustness are antagonistic requirements, with a missing definitive notion of stability [Owhadi et al., 2013].

These issues raise concerns about the general applicability of Bayesian inference in a continuous world under finite transformations of an information structure. A close inspection of many practical probabilistic decision agents (who implement the theorems of Bayes and Cox), suggests that these are in fact held together by non-Bayesian feedback loops. These non-Bayesian feedback loops are typically associated with a performance evaluation of what is essentially Bayesian Inference. In lieu of these interpretations, we avoid further complications by seeking non-Bayesian architectures. One such alternative is the aforementioned hypermodern approach.

A concise overview of a *hypermodern* decision agent is that it is predominantly non-parametric. In the probabilistic setting, the higher an agent's expectation of utility from an action, the higher the probability of choosing that action [Cao 2007]. Probabilistic agents typically form parametric beliefs about the world through progressive information gains. These agents presuppose a coherent set of rules for the asset-environment behaviour. In the hypermodern setting, this preference structure either does not hold, or is not required. Emphasis is therefore placed on more ad hoc measurements, high-frequency updates, unordered beliefs, and action-selection rules which in general cannot be composed into "smooth" probability distributions. The hypermodern agent scales well with access to high-frequency, low-latency (HFLL) monitoring and control. This is in contrast with the typical probabilistic agent, which tends to prefer low-frequency, high-throughput (LFHT) transactions. Hypermodern agents also perform better in the face of context-switching environments [Cao 2007].

Table 4.3 establishes the functional requirements for a typical monitoring component within a three-phase planning architecture. For our purposes, this monitoring is required to be pre-conditioned. We also require a real-time or near-real-time reporting protocol, but do not impose a particular timing constraint, deadline, or penalty scheme. The monitoring component must also support interactive online input streams from multiple sources. The combination of these functional requirements, as well as the overall semantic and non-functional requirements of Table 4.2, lend themselves to a high-frequency low-latency (HFLL) approach.

Table 4.4 proposes several requirements for the evaluation modules or components. For hypermodern decision agents, the predominant stipulation is that an evaluation of monitoring inputs possesses the *anytime* property. An evaluation component is said to be *anytime scalable* if it is guaranteed to improve its solution quality (i.e. monotonic-increasing in expectation) with additional (temporal) resources. This entails the use so-called *anytime algorithms*, which are reviewed in Section 4.1.3.6. Architecturally, the evaluation component is tasked with offering a trade-off between solution quality and computational resources. Algorithmically, this can be accomplished in several ways, including roll-out style algorithms, or sampling-based improvement techniques [Silver and Veness, 2010], [Kocsis and Szepesvari, 2013]. Our choice of implementation is in line with meeting the set of hypermodern demands, which prefer to avoid sampling and parametrization during runtime. The proposed evaluation component hybridizes both online and nearline evaluations to remain stable and robust. Other methods for delivering anytime guarantees may come at the cost of being numerically unstable or weakly

approximate [Browne et al., 2012]. In a hybrid setup, the online evaluation returns secure results to the prescription component for the time-critical deployment of strategies. Meanwhile, an ongoing nearline (or offline) evaluation can be queried for less time/safety-critical and more opportunistic results. This combination of online and nearline (or offline) analytics augments the action-selection process by offloading a minimal amount of work to the prescription phase.

Table 4.5 outlines the functional requirements of a hypermodern prescription phase. The prescription component collects the outputs from the evaluation and produces a shortlist of the best available strategies along with the chosen alternative(s). Potential trajectories are generated and may be recorded so as to inform and improve the offline evaluation component. The prescription component is primarily responsible for accepting the evaluation results and implementing the corresponding control actions. The currently prescribed action-plans, as well as summary data, may be presented through an external interface. This requirement supports a human-in-the-loop (e.g. expert user) or meta-level AI. The presentation of strategies and prescribed actions affords cognitive-level pattern recognition, process supervention, quality auditing, and governance oversight if necessary. This may be accomplished by way of visual displays, data visualizations, info-metrics, or standardized reports. The architectural requirements emphasize that prescription occurs in near-real-time when external interrupts are not present.

Table 4.3. Functional requirements for a hypermodern *monitoring* component within three-phase planning (action-selection) architecture.

Monitoring Requirement	Functional Description	Technical Considerations
Conditioned Inputs	Accept a pre-processed, pre-filtered input stream from sensory data or model description feeds. Example: Clean and convert input signals. Only inputs expressed as strings of positive-semi definite integers on the interval $[0, 10^{10}]$ will be accepted.	Offers a separation of architectural concerns; upstream filtering vs. downstream processing, etc. Complex planning and decision making processes are isolated from the nuances of data gathering and synthesis. Input stream need not be noise-free but should be reasonably well-defined and bounded in some acceptable language/type.
Real-Time Reporting	Inputs must be delivered at a rate considered to be *real-time* or *near-real-time;* with hard or soft deadlines/penalties for violation of these timing and reporting requirements. Example: Sensory information is queried/polled, cleaned, and made available as input data every second while the asset is active.	Supports live/active asset monitoring. Improves change-detection-rate. Quality of service decays with respect to deadline exceedance. Can create "information overload" scenarios. Does not necessarily increase the resolution of actionable knowledge.
Online Pass-through	As monitoring data is streamed in, it must be immediately passed through to the evaluation module to be operated on in an online (or nearline) manner. Example: Inputs are revealed and processed sequentially without pooling them for batch processing.	Captures dynamic activity and anomalous events near the source. Relays this information to the evaluation module with minimal overhead. Allows operations to be conducted in an online, nearline, or streamline manner. Avoids an offline analysis which requires an aggregate, batch, or larger queue/store for datasets.
Interactive Asset-Environment	Monitoring must support interactive inputs. Example: An input sequence is periodically injected with data owing to some potentially interactive transactions, or commands as a result of some output process.	Allows for slipstream activity. Captures out-of-order causality. An *interactive* input stream may be related to any number of previous input streams, as well as to the relationship between them. Interactive sequences are difficult to predict and often require *speculation-free* monitoring, evaluation, and prescription.

Table 4.4. Functional requirements consistent with a hypermodern *evaluation* component within a three-phase planning (action-selection) architecture.

Evaluation Requirement	Functional Description	Technical Considerations
Anytime Performance Scaling	The evaluation component can be queried at anytime, returning the best-known solutions or results.	Evaluation module(s) progressively enhance solutions. Algorithms do not require specifying the resources for completion (non-contract).
RTO Analysis	Performs evaluation in a real-time online sense.	Produces a real-time online (RTO) analysis, and supplies the prescription component with immediately available online-calibre results. These results are reactive but safe/secure.
DD Analysis	Augments RTO evaluations through a deep nearline or offline deliberation.	Conducts a deep-deliberation (DD), in a nearline or offline sense, and supplies the prescription component with delayed results. These results are more adaptive, predictive and opportunistic.
Competitive Evaluations	The ratio of online to offline analysis (competitive ratio) is bounded.	The evaluation procedure does not "know" the entire set of inputs (including future inputs), but can nonetheless implement an effective evaluation.

Table 4.5. Functional requirements for a hypermodern *prescription* component within a three-phase planning (action-selection) architecture.

Prescription Requirement	Functional Description	Technical Considerations
Strategy Profiling and Decision Ranking	Takes as input the results/solutions from the evaluation components and compiles a set of strategies (and strategy profiles).	Yields a set of top-ranked candidate state-transition trajectories. Isolates the pre-requisite resources and action sequences to achieve optimal asset behaviour/operation.
Plan Generation and Action-Selection	Formulates an optimal* plan. Prescribes the sequence of actions leading to desired goals/state(s).	Selects the most appropriate state-transition trajectory. * = optimal w.r.t criteria/definitions. Outputs the single best-response move/actions as well as the expected trajectory for the asset.
Control and Effectuation	Sends a control signal or policy revision protocol to actuators and/or affectors.	Cyber-physically implements a realization of the move, action, or desired state-transition.
Ongoing/Active Reporting	Produces as output a summary of the active decision-making process.	Outputs should display an active summary of relevant information in a human-interpretable form. E.g. Summary Reports, Tabular, Charts, Visual displays, etc.
I/O Interfacing	Provides users with access to input/output options and configurable parameters (if applicable).	Allows expert supervision, advanced tuning, auditing and diagnostics.

4.1.2 Algorithmic Requirements

The implementation of an architecture for generalized asset integrity planning necessitates the use of computational resources. In a three-phase architecture, each component may require the deployment of its own computational procedures. The suitability of an algorithm will vary by task; whether it is for monitoring and pre-conditioning the input streams, evaluating strategies and solving succinct integrity games, selecting and implementing control actions, or visually presenting the results to the user. Each of the monitoring, evaluation, and prescription components can be computationally intensive, challenging to implement and sensitive to design choices [Sleight and Durfee, 2013].

In keeping with the impetus, we narrow our discussion to the key evaluation components within a three-phase planning architecture. The conditioning of inputs/outputs, pre/post processing, and dataflow to/from the non-evaluation components are considered negligible. These are relatively straightforward tasks representing the interface between the evaluation component(s) and the monitoring or prescription phases. For our purposes, we define an *evaluation component* to consist of three (3) major algorithmic tasks:

1.) Estimate the state-transition costs.

2.) Directly solve the base game.

3.) Attempt to improve solutions to the perturbed integrity game.

These tasks are listed by order of precedence. They are *ongoing* in the sense that they must be performed at each iteration or when signaled that a change in the asset is detected.

In (1), we seek algorithms which evaluate the integrity score of the asset relative to a pre-computed set of reference configurations (integrity states). For each integrity state or idealized reference configuration, a complex function may need to be evaluated. This function scores the fitness-level of the asset based on model dependability metrics and uses the normalized compression distance to ascertain the relative similarity to each of the states. Repeating this process for each integrity state will populate an array which expresses the estimated integrity score for each potential state-transition. This array contains the integrity scores or state-transition costs, and constitutes the payoff structure for the base game.

In (2), we seek algorithms which are calibrated towards efficiently solving a class of succinct integrity games. Hypermodern decision agents demand high-frequency, low-latency information updates. As such, this portion of the "game solver" should be compatible with real-time online constraints. Algorithms which directly solve the base game are required to be deterministic. They must report the principal variation as a pure or mixed strategy Nash Equilibria up to some fixed evaluation depth. For a two-player succinct integrity game, this process corresponds to returning the set of all minimax state-transition trajectories.

In (3), we invoke the *anytime* requirements, and seek algorithms which iteratively improve solutions to the perturbed integrity game. No additional algorithmic requirements are imposed, although it would seem that a deep, nearline or offline search of the state space is the most appropriate. In the deliberation setting, it becomes more feasible to attempt a

proper learning of the generator functions for the stochastic noise and other perturbative effects. Table 4.6 postulates a set of desirable characteristics and acceptable computational specifications. Taken together, these circumstances influence the choice of deployed algorithms.

Table 4.6. Algorithmic Modules for Integrity Game Solvers under Hypermodern Evaluation Demands.

Algorithmic Configuration	Working Description	Functional Requirements	Complexity Requirements
Integrity Score	Evaluation of a cost function.	Estimates the integrity scores between state-transitions.	Yields a payoff matrix (input array) within $O(N\ logN)$ using standard compression (NCD) techniques.
Online Evaluation	RTO solver for Baseline Integrity Games.	Computes all minimax trajectories.	Solves the base integrity game (2PZSG) in $O(N)$ time.
Nearline or Offline Evaluation	DDO solver for the Perturbed Integrity Games.	Computes long-run stable strategies and/or learns complex noise generating functions.	Yields an epsilon-approximate globally optimal principal variation in up to $O(N^2)$time, with $O(N\ logN)$ being ideal.

4.1.3 Supplemental Notions

4.1.3.1 Automated Planning

Automated planning is a branch of artificial intelligence which conducts the algorithmic search for strategies and actions. The solutions to automated planning problems are often more complex than classical control and classification problems. This is due to the multi-dimensional and highly dynamic problem structure. Planning problems contend with multiple intelligent agents, each interacting competitively or cooperatively, reporting fuzzy beliefs, non-transitive preferences, and chaotic knowledge about the environment. The automated search for plans, strategies, or action sequences is performed through algorithms which explore the state and/or configuration space of the system. This is accomplished through either a direct traversal towards a goal, and/or indirect sampling to synthesize admissible and thereafter optimal solutions.

4.1.3.2 Real-Time Online Algorithms

Real-Time Online (RTO) algorithms are subject to both real-time and online requirements. Real-time conditions must guarantee a response within strict time constraints, often referred to as deadlines. For “hard” real-time constraints, violation of deadlines leads to system failure. For “soft” real-time constraints, violations are tolerated as progressively-critical faults which degrade the quality of service but may nonetheless allow for recovery. For complex asset governance, deadline requirements may vary. Acceptable response times are typically in the order of several seconds. In this context,

real-time planning algorithms will at any given time receive data, process them, and output results so as to affect the asset at near this time. Online algorithms are restricted to a piece-wise, subset handling of inputs. This often implies a sequential realization of the problem. An online algorithm receives a sequence of requests and performs an immediate action in response to each request in kind. Since they do not know all the given information at once, online algorithms are forced to make decisions that may turn out to be sub-optimal or even detrimental. The study of real-time online (RTO) algorithms has focused on the quality of decision making that is possible under both real-time and online conditions. In this setting, RTO planning is designed to work fast and abruptly, performing decision making after each input request, and producing well-formed responses within seconds.

4.1.3.3 Offline Algorithms

Offline algorithms are given access to the entire range of inputs in advance. An offline algorithm formulates a plan which by itself still represents taking an action in response to each request sequentially. Unlike online algorithms, the choice of each action can be based on the entire sequence of requests. An offline algorithm essentially knows the future and implements a zero-regret policy trajectory, navigating the state-space optimally.

4.1.3.4 Performability of Online and Offline Algorithms

The most common approach to assessing the *performability* of RTO planning is to assume a specific stochastic model of the source of inputs, requests, event arrivals, state

realizations, etc. Within such a model, an online algorithm may be considered optimal if it chooses its actions so as to minimize some cost functional. Here, the cost depends on the sequence of requests generated by the stochastic source, and on the sequence of actions chosen by the online algorithm in response to those requests. The choice of stochastic model hinges on data being readily available about the observed sequence history, and also requires faith that the future will resemble the past [Zilberstein, 1996]. For these reasons, stochastic input models may be rather limited for some forms of asset integrity planning. Rather inappropriately, much of the theory of stochastic control, risk-based scheduling, and performance analysis is based on this approach [Borodin and El-Yaniv, 2005]. One alternative to stochastic models is a worst-case approach inspired by minimax regret and stochastic game theory. Here, the optimality of an online algorithm is evaluated by contrasting its cost with that of an optimal offline algorithm processing the same sequence of requests. In literature, this is known as the competitive ratio.

4.1.3.5 The Competitive Ratio

The competitive ratio is defined as the maximum, over all possible input sequences, of the ratio between the cost incurred by the online algorithm and the cost incurred by the optimal offline algorithm [Borodin and El-Yaniv, 2005]. In this model, an optimal online algorithm is one whose competitive ratio is a minimum. The main virtue of the competitive ratio approach is that it avoids commitment to a particular stochastic input model. However, the approach is pessimistic and essentially assumes the request sequence is chosen by an all-knowing (offline) adversary. In practice, one seeks RTO algorithms which

perform well on typical request sequences while maintaining a small competitive ratio. This balance can often be achieved through anytime algorithms, which hybridize both the online and offline settings.

4.1.3.6 Anytime Algorithms

Anytime algorithms constitute a class of search techniques which provide automated planning suitable for stochastic learning and the performance optimization of complex assets. These algorithms hybridize aspects of real-time online fast-response with offline deep-calculation and future-proof deliberation. An anytime algorithm can return a valid, admissible solution to a problem even if it is interrupted prematurely. Anytime planning algorithms find and report incrementally better solutions as additional computational resources are provided. In general, one assumes monotonically increasing results [Zilberstein, 1996]. As the time to perform the algorithmic search increases, the quality of the plan is expected to approach optimality. This is in contrast to *contract algorithms*, which take as input some fixed amount of computation, run to completion, and provide a single best answer. Contract algorithms guarantee a correct output only after proper termination, with no guarantees on intermediate results. When interrupted before finding a global optimum, anytime algorithms will return partial, best-known approximate solutions. Anytime algorithms therefore scale with any allocation of computational resource.

4.1.3.7 Anytime Performance

Anytime performance is typically evaluated through the use of competitive ratios and performance profiles [Borodin and El-Yaniv, 2005]. A performance profile estimates the quality of results based on the input and the amount of time that is allotted to the algorithm. It is a mapping of time to the quality of expected results. The probability of a result being correct (certainty), the error bounds (accuracy), and the amount of discrimination between other results (specificity), each contribute to the quality. The competitive ratio, which measures the degree to which an online algorithm approaches the performance of an offline algorithm, is also used. When both these techniques are combined, the performance of an anytime algorithm can be monitored and potentially controlled by solving some meta-level computational resource allocation problem. For additional details on managing the stopping times of multiple anytime algorithms, see for example the works of [Zilberstein, 1996], and [Borodin and El-Yaniv, 2005].

4.1.3.8 Desirable Anytime Characteristics

In terms of asset integrity, the techniques for automated planning, performance optimization, and algorithmic search are inter-related. Anytime conditions have been applied to several families of algorithms [Thayer and Wheeler, 2010]. These include: numerical optimization (e.g. gradient descent, hill-climbing), heuristic search (e.g. tabu search, particle swarm), nondeterministic algorithms (e.g. Monte Carlo and genetic perturbation methods), probabilistic inference (e.g. Deep Belief Networks inspired from Bayes, Markov, Boltzmann, ANN), combinatorial search (graph coloring), and discrete

symbolic processing (string matching). Applying anytime constraints may alter the algorithmic implementation of the numerical method, particularly if it is inherently sequential, embarrassingly parallel, or fundamentally offline. This can potentially lead to program code which is more ad-hoc, difficult to deploy, and/or harder to maintain.

Irrespective of the implementation, there exist several characteristics which are considered desirable, if not required, for a legitimate anytime planning algorithm [Zilberstein, 1996]. A summary of these characteristics includes:

1. *Interruptible*: The algorithm can be stopped at any-time and provide some answer.
2. *Pre-emptive*: The algorithm can be suspended and resumed with minimal overhead.
3. *Monotonic*: The quality of the best-known and returnable result is a non-decreasing function of the computation time.
4. *Measurable quality*: The quality of an approximate result can be determined approximately, often at runtime.
5. *Nondeterministic Consistency*: For a given input, the quality of the result with respect to computation time is approximately the same each time.
6. *Diminishing Returns*: The improvement in solution quality is the largest at the early stages of computation, and the improvement diminishes over time.
7. *Completeness*: Given infinite resources, for any inputs, the globally optimal solutions should be found, or reported that they do not exist.

In addition, requirements usually specify a strong competitive ratio, hard or soft real-time constraints, tailored performance profiles, and bounds on the overall computational complexity.

4.2 The Game Engine

The Generalized Asset Integrity Game Engine (GAIGE) is a game-playing agent for the rapid generation of decision sequences. In its prototypical form, it aims to implement a basic anytime search over the state-space of possible move orderings, returning the most promising strategies and formulating integrity plans guided by game-theoretic optimality criteria (i.e. equilibrium corridors).

4.2.1 Scope

The GAIGE uses the architectural and algorithmic guidelines of Section 4.1 for autonomous planning and governance. These are conservative by design, emphasizing several of the robustness and stability concepts of Chapter 3. High performance decision-making is sought in the face of adversarial dynamics, hostile environments, and anytime preemptive feedback. This entails a time-sequential, dynamic optimization of a complex evaluation function (the integrity score) based on the principles of Chapter 2. Hyper-modern demands are imposed, signifying a requirement that solutions avoid over-fitting and scale well (low-latency, high-throughput) across domain types, sizes, and uncertainty regimes.

These requirements emphasize numerical solutions which are “easy to implement”, and in general, tolerant of forced modelling errors (i.e. when strong uncertainty reduction is infeasible). This leads to the anticipation of low complexity anytime algorithms with (competitive) asymptotic performance guarantees. The prototypical GAIGE utilizes an

anytime search procedure for plan generation. This yields a prescription of reactive strategies which secure the immediate survival of the asset (in a deterministic and myopic sense). Given additional resources, the GAIGE explores strategies which are increasingly adapted to the perturbations.

4.2.2 Minimax Transposition Search (MTS)

In their basic form, minimax algorithms return *principal variations* and constitute a type of adversarial search procedure. For many games, (principal) variations can be thought of as (the best and/or most likely) lines of play. Principal variations represent the expected path(s) of traversal, such as move-orderings for decision-trees or vertex-cycles in graphs.

Seeking principal variations rapidly and robustly is crucial if one is trying to elaborate plans within adversarial environments. The search routines for many high-performance game engines typically involve some form of *minimax search* (e.g. Chess and Go). Minimax is typically employed in 2-player finite discrete sequential games (i.e. combinatorial games). Variants of the basic minimax process have been developed to exploit problem-specific features, taking advantage of special structure and providing accelerated results. Standard enhancements include alpha-beta pruning (e.g. *negascout*), moving-window searches with test-drivers (e.g. *MTD-f*), iterative deepening and quiescence (i.e. *intelligent fan-out*), as well as number-theoretic exploits (e.g. *magic and conspiracy numbers*), [ChessProgramming.com Wiki pages, 2015]. In more complex

games, search procedures may include contributions from fictitious and/or randomized play, pattern matching, opening books, advanced rulesets, and endgame databases.

The primary goal of minimax enhancements is to reduce the amount of computational effort required to return principal variations. Several of the aforementioned approaches can reduce the worst-case running time of $O(k^t)$, where k is the average branching factor, and t is the search depth. In the GAIGE, the principal variations are equivalent to the minimax trajectories through the game dynamical system. These paths also represent elements of a set of "stable" oscillations for an ergodic family of systems under *2PZSG* dynamics. Therefore, enumerating the elements of such a set is equivalent to returning all the minimax principal variations.

Minimax Transposition Search (MTS) is the search procedure used to solve the repeated base game component of the GAIGE. MTS is a search procedure which proceeds via backward induction from a finite stopping horizon (fixed search depth). It is inspired by dynamic programming and the principal of optimality first proposed by Bellman (c.f. optimal substructure, [Bellman, 1957]). The MTS algorithm acknowledges the inherent symmetry and substructure present in the succinct-form base integrity game. This manifests itself in several ways. First, since all k actions are potentially available to either player at any time-step, the search admits a constant branching factor. This branching is bounded by some fixed branching rate k, supporting a simplified decomposition and traversal. Second, we note that every state-transition reward/cost is fixed throughout all time-steps. Informally, this leads to a realization that after a finite number of state-

transitions, one will be revisiting a particular reward state (as the same player to move and minimax). This repetition of previously visited states through different permutations of a move ordering is known as a transposition. The MTS algorithm works by exploiting the symmetry and invariance of information, actions, payoffs, and state-transitions. By making use of a special substructure whereby "everything is eventually a transposition", the MTS is able to return all principal variations in both linear time and space.

Figure 4.1 presents the MTS algorithm in its entirety, with comments and pseudocode adapted from a prototypical GAIGE implemented in the Python (version 2.79) programming language. The program code was developed solely by the author as part of the thesis work, with references to the work of Bubek et Al. for the Bandit algorithms component.

Figure 4.1. The Minimax Transposition Search (MTS) algorithm as implemented in the GAIGE.

	Algorithm MTS (page 1).	
1	**Initialization with inputs N, T.** *aPayoffs* ← $\{a_{11,}a_{12,}\ldots,a_{ij}\}$; *aCS* ← $\{0\}^N$; *aMinCS* ← $\{0\}^N$; *aMaxCS* ← $\{0\}^N$;	Initialize the set of N^2 actions (state transition payoffs). Also initialize arrays for cumulative sums, and their min and max.
2	**for** *t = 0, ..., T* **do**	For each of the *T* rounds.
3	*x ← 0;* *aMulti ← 0;* *aLineOut ← {"Depth %s Min – Path [" % (t+1)};*	Initialize location index, transposition count, and set line outputs for the *min player*.
4	**for** *j = 0, ..., N* **do**	For each of the *N* sources (j^{th} row).
5	*aMinX ← -1; aMin ← ∞;*	Initialize location/bounds.
6	**for** *i = 0, ..., N* **do**	For each of the *N* destinations (i^{th} col).
7	**if** *aMin >= aCS[i] + aPayoffs[x]:*	Check for a new minimum.
8	**if** *aMin == aCS[i] + aPayoffs[x]:* *aMulti ← aMulti + 1;*	Check for multiple paths leading to same cumulative payoffs (transpositions).
9	**else** *aMulti ← 1;* *aMin ← aCS[i] + aPayoffs[x];* *aMinX ← {i};*	Update the location of the minimum cumulative sum and its value.
10	*x ← x + 1;*	Increment the location index.
11	**end for**	Exit the *i* loop.
12	*aLineOut ← aLineOut + {"%s" % aMinX};* **if** aMulti > 1: *aLineOut ← aLineOut + {"*%s" % aMulti};* *aLineOut ← aLineOut + {", "};*	Format and store the line outputs.
13	*aMinCS[j] ← aMin;*	Update the minimum cumulative sum.
14	**end for**	Exit the *j* loop.

	Algorithm MTS (page 2).	
15	*aLineOut ← aLineOut + {"] Sum of Payoffs ["};* **for** *q = 0, ..., size(aMinCS)* **do** *aLineOut ← aLineOut + {"%s, " % aMinCS[q]};* **end for**	Format the line outputs to show the accumulated payoffs up to the current depth.
16	**output** *flush(aLineOut, "print()")*	Flush line outputs to the screen and/or store in logs.
17	*x ← 0;* *aMulti ← 0;* *aLineOut ← {"Depth %s Max – Path [" % (t+1)};*	Reset location index, transposition count, and set line outputs for the *max player*.
18	**for** *j = 0, ..., N* **do**	For each of the *N* sources (j^{th} row).
19	*aMaxX ← N; aMax ← 0;*	Initialize location/bounds.
20	**for** *i = 0,...,N* **do**	For each of the *N* destinations (i^{th} col).
21	**if** *aMax <= aMinCS[i] + aPayoffs[x]:*	Check for a new maximum.
22	**if** *aMax == aMinCS[i] + aPayoffs[x]:* *aMulti ← aMulti + 1;*	Check for multiple paths leading to same cumulative payoffs (transpositions).
23	**else** *aMulti ← 1;* *aMax ← aMinCS[i] + aPayoffs[x];* *aMaxX ← {i};*	Update the location of the maximum cumulative sum and its value.
24	*x ← x + 1;*	Increment the location index.
25	**end for**	Exit the *i* loop.
26	*aLineOut ← aLineOut + {"%s" % aMaxX};* **if** aMulti > 1: *aLineOut ← aLineOut + {"*%s" % aMulti};* *aLineOut ← aLineOut + {", "};*	Format and store the line outputs.
27	*aCS[j] ← aMax;*	Update the cumulative sum with the new maximum.
28	**end for**	Exit the *j* loop.

Algorithm MTS (page 3).		
29	*aLineOut ← aLineOut + {"] Sum of Payoffs ["};* **for** *q = 0, ..., size(aCS)* **do** *aLineOut ← aLineOut + {"%s, " % aCS[q]};* **end for**	Format the line outputs to show the accumulated payoffs up to the current depth.
30	**output** *flush(aLineOut, "print()")*	Flush line outputs to the screen and/or store in logs.
31	**end for**	Exit the main loop.
32	**return** 0	End of algorithm MTS.

4.2.3 Stochastic and Adversarial Optimal (SAO) Bandits

Multi-armed bandits (MAB) are well-known in the literature on sequential decision analysis and statistical process control. In the bandit setting, an agent must sequentially choose actions so as to maximize the cumulative, expected, or discounted long-run reward. Through sequential (possibly noisy) feedback, it becomes possible to build a model of the relationship between actions and rewards. At each time-step, the agent may choose actions in order to improve its model (*exploration*), or select actions believed to yield high rewards according to the model (*exploitation*). This results in an *exploration-exploitation dilemma* which is hard to solve in general. The basic MAB problem has been extensively studied in [Agrawal and Goyal, 2012] with provisions for safety vs. risk dilemmas studied in [Galichet et al., 2013].

Variations on the MAB theme have been cross-pollinated by research into planning algorithms and experimental design [Kocsis and Szepesvari, 2013]. The basic MAB formulation is similar to an MDP, with extensions showing correspondence to POMDP models [Silver and Veness, 2010]. These problems can be solved using dynamic programming techniques such as value or policy iteration [Filar and Vrieze, 1997]. Unfortunately, these techniques typically require algorithms of exponential complexity in the number of independent arms. More recent approaches utilize stochastic reinforcement learning (c.f. Q-learning, H-infinite control) with adaptive sampling. Such methods typically recycle the ideas of Thompson (c.f. Thompson Sampling), Gittins (c.f. Gittins

Index), or Whittle (c.f. Whittle Index). The complexity of these approaches typically ranges from linear to cubic in the number of actions [Agrawal and Goyal, 2012].

MAB algorithms have also been developed under the *probably-approximately-correct* (PAC) learning framework of Valiant [Valiant, 1984]. In the PAC framework, one seeks approximate solutions which are optimal in the sense of two-sided error (epsilon, delta) bounds on the notion of *regret*, which is typically a convex loss function. Fast, scalable classes of bandit algorithms utilizing these ideas include the *epsilon-greedy*, *UCB*, *Softmax* (*Boltzmann*), and *EXP* bandits. These algorithms work online and deliver fast, near-optimal solutions in potentially sub-linear time [Audibert and Bubeck, 2009]. These algorithms represent the current state-of-the-art, solving the exploration-exploitation dilemma across many problem classes, including several variations of MAB, MDP, and POMDP.

The GAIGE utilizes an algorithm known as the Stochastic and Adversarial Optimal (SAO) bandit. The SAO was developed for the MAB framework by Bubeck and Slivkins [Bubeck and Slivkins, 2012]. This algorithm has been modified and implemented by the author as part of a near-line extension of the MTS component of the GAIGE. The SAO works by minimizing the *competitive regret*, or alternatively maximizing the *competitive ratio* as outlined in Section 4.1.3. Thus, one typically seeks to compare the *online* rewards received by a MAB algorithm up to some stopping-time with respect to some *offline* benchmark. A standard offline benchmark is the best-possible action-sequence in hindsight [Audibert and Bubeck, 2009]. What makes the SAO algorithm exceptional is that it achieves the "*best of both worlds*" after a reasonable number of rounds, *T*.

In SAO, the worst-case regret is bounded by $O(log(T))$ when the reward environment is stochastic, and $O(\sqrt{(T)})$ when it is adversarial. This represents an *attack-defense balance* which gracefully enhances the agnostic behaviour of the MTS. A complete discussion of the SAO is available through the work of Bubeck and Slivkins (2012), [Bubeck and Slivkins, 2012]. These authors also provide the relevant theorems and proofs of convergence. We have recruited their ideas and present them in Figures 4.2 and 4.3. As mentioned, the SAO algorithm is anytime scalable and competitive. It has been revised and implemented as part of the near-line component of the GAIGE in order to solve the perturbed integrity game.

Figure 4.2. The Multi-Armed-Bandit (MAB) framework with both adversarial and stochastic reward feedback.

	Known parameters: *N* actions; *T* rounds; non-degenerate choice: ($T \geq N \geq 2$).
	Unknown parameters:
	(i.) Adversarial setting: non-parametric → none;
	(ii.) Stochastic setting: N *i.i.d.* probability distributions $v_{1,\dots}, v_N$ on $[0,1]$. Each v_i parameterized by mixture of moments, θ_i: e.g. $\theta_i = f(location\mu_i, scale\sigma_i, etc.)$.
	For each round $t = 1,2,\dots,T$;
	(1) A MAB algorithm chooses an action $A_t \in \{1,\dots,N\}$ (possibly randomly).
	(2) The environment selects rewards according to the exposed model: – An adversary simultaneously selects rewards $g_t = (g_{1,t},\dots,g_{N,t}) \in [0,1]^N$. – Each reward $g_t \sim v_i$ is drawn stochastically and independently.
	(3) The forecaster observes (and receives) the reward $g_{A_t,t}$. – In the MAB framework, the forecaster does not observe the rewards from other arms.
	Goal: Minimize the regret, defined for each respective environment:
	– Adversarial model: $$\widehat{R_n} = \max_{i\in\{1,\dots,N\}} \sum_{t=1}^{T} g_{i,t} - \sum_{t=1}^{T} g_{A_t,t}$$ – Stochastic model: $$\widetilde{R_N} = \sum_{t=1}^{T} \max_{i\in\{1,\dots,N\}} (\theta_i) - \theta_A$$

Figure 4.3. The Stochastic and Adversarial Optimal (SAO) algorithm of Bubeck and Slivkins (2012).

	Algorithm SAO (page 1).	
1	**Initialize the SAO Bandit with inputs N, T, $\boldsymbol{\beta > 1}$.** *S* ← *{1, …, N};* *K = N;*	Initialize the set *S* of active *strategy profiles* (perceived as *arms or actions* in the Bandit framework).
2	**for** i = *1, …, K* **do**	For each of the *K* actions (where each action denotes an MTS minimax strategy to follow).
3	$\tau_i \leftarrow \{T\}$; $p_{i,t} \leftarrow \{1/K\}$;	The time τ_i when action *i* is deactivated, and its probability $p_{i,t}$ of selection at time *t*.
4	**end for**	End of initialization.
5	**for** *t = 1, …, T* **do**	Begin the main loop.
6	**play** $A_t \in S$ with $p_{i,t}$	Select an active action with the appropriate probability.
7	**for** *i = 1, …, K* **do**	For each action, test several properties (4 total under SAO).
8	**if** $(i \in S) \wedge (\max_{j \in S} \widetilde{H}_{j,t} - \widetilde{H}_{i,t}) > \alpha_1(i,t)$	Test function 1. Deactivation threshold.
9	**then** $S \leftarrow S \setminus \{i\}; \tau_i \leftarrow t;\ q_i \leftarrow p_{i,t};$	Deactivation: Remove action *I* from the active set. Update the deactivation time τ_i and probability q_i at time of deactivation.
10	**end if**	
11	**if** $\lvert\widetilde{H}_{i,t} - \widehat{H}_{i,t}\rvert > \alpha_2(i,t)$	Test function 2. Consistency condition.
12	**or** $(i \notin S) \wedge (\max_{j \in S} \widetilde{H}_{j,t} - \widetilde{H}_{i,t}) > \alpha_3(i,t)$	Test function 3. Sub-optimality threshold.
13	**or** $(i \notin S) \wedge (\max_{j \in S} \widetilde{H}_{j,t} - \widetilde{H}_{i,t}) > \alpha_4(i,t)$	Test function 4. Significance threshold.
14	**then** *call* algorithm **EXP3.P**: $p_{i,t+1} \leftarrow$ EXP3.P($\delta \in (0,1)$)	If one of tests 2-4 is *true*, then the environment satisfies the properties of being adversarial, and we update with *EXP3.P*.
15	**end if**	

Algorithm SAO (page 2).		
16	**end for**	End of testing phase (*exploration*). Exit the $i \in K$ loop.
17	**for** *i = 1, …, K* **do**	Update the action selection probabilities $\forall i \in S$.
18	$p_{i,t+1} \leftarrow \left(\frac{q_i \tau_i}{t+1}\right)\{1\}_{i \notin S} + \frac{1}{\lvert S \rvert}\left(1 - \sum_{j \notin S} \frac{q_i \tau_i}{t+1}\right)\{1\}_{i \in S}$	
19	**end for**	End of updating phase (*exploitation*). Exit the $i \in K$ loop.
20	**end for**	Exit of main loop.
21	**return** 0	End of algorithm SAO.

The SAO algorithm of Figure 4.3 makes use of following notation, terminology, and statistical test functions:

- The *cumulative reward of a fixed strategy i* up to time *t*, and its *average:*

$$G_{i,t} = \sum_{m=1}^{m=t} g_{i,m}, \quad H_{i,t} = \frac{1}{t} G_{i,t} \qquad 4.1, 4.2$$

- The *estimated cumulative reward* from strategy *i* up to time *t*, and its *average*:

$$\tilde{G}_{i,t} = \sum_{m=1}^{m=t} (g_{i,m} A_{i,m} \,/\, p_{i.m}), \quad \tilde{H}_{i,t} = \frac{1}{t} \tilde{G}_{i,t} \qquad 4.3, 4.4$$

- The *algorithm's cumulative reward* from strategy *i* up to time *t*, and its *average*:

$$\hat{G}_{i,t} = \sum_{m=1}^{m=t} (g_{i,m} A_{i,m}), \quad \hat{H}_{i,t} = \hat{G}_{i,t} \,/\, \sum_{m=1}^{m=t} A_{i,m} \qquad 4.4, 4.5$$

Table 4.7. Formulae used by the SAO bandit algorithm (ref. Figure 4.3).

	Function	Derived Formula
1	Deactivation threshold	$\alpha_1(i,t) = 6\sqrt{\frac{4K\log(\beta)}{t} + 5\left(\frac{K\log(\beta)}{t}\right)^2}$
2	Consistency condition with $t_i^* = min(\tau_i, t)$	$\alpha_2(i,t) = \sqrt{\frac{2\log(\beta)}{\sum_{m=1}^{m=t} A_{i,m}}} + \sqrt{4\left(\frac{Kt_i^*}{t^2} + \frac{t - t_i^*}{q_i \tau_i t}\right)\log(\beta) + 5\left(\frac{K\log(\beta)}{t_i^*}\right)^2}$
3	Differential sub-optimality at deactivation time.	$\alpha_3(i,t) = 10\sqrt{\frac{4K\log(\beta)}{\tau_i - 1} + 5\left(\frac{K\log(\beta)}{\tau_i - 1}\right)^2}$
4	Significance of rewards at deactivation time.	$\alpha_4(i,t) = 2\sqrt{\frac{4K\log(\beta)}{\tau_i} + 5\left(\frac{K\log(\beta)}{\tau_i}\right)^2}$

The functions in Table 4.7 represent bounds which provably hold *with high probability (w.h.p.).* These are implemented as inequalities against some reward, loss, or regret criterion.

For example, line (11) of Figure 4.3 performs a statistical test, $|\widetilde{H}_{i,t} - \widehat{H}_{i,t}| > \alpha_2(i,t)$, which evaluates to *true* if a consistency condition for adversarial environments is met. In this case, line (11) implies that for any strategy or action, the absolute difference in $\widetilde{H}_{i,t}$, the average of the cumulative reward estimates, and $\widehat{H}_{i,t}$, the average cummulative rewards experienced thus far, should *not* exceed a limit threshold in order to be *consistent* with the random play of a stochastic environment.

The lines (11, 12, and 13) in Figure 4.3, and the corresponding bounds $\alpha_{2,}\alpha_{3,}\alpha_4$ of Table 4.7, ensure that if any one of the three statistical tests evaluate to *true*, the disjunction will be *true*, and the SAO algorithm then utilizes the EXP3.P revision protocol to guard against adversarial dynamics. This is in contrast to the UCB/INF-inspired bandit updating of lines (17, 18), which are calibrated for stochastic environments.

Ultimately, the SAO component of a prototypical GAIGE implementation will compute the strategy selection probabilities using the following schemes:

- If exploration/testing indicates an *adversarial* payoff structure, utilize a variant of *EXP3.P*-bandits for revision:

EXP3.P:	$p_{i,t+1} \Leftarrow \hat{p}_{i,t+1} \leftarrow (1-\gamma)\frac{exp(\eta\tilde{G}_{i,t})}{\sum_{k=1}^{k=K} exp(\eta\tilde{G}_{k,t})} + \frac{\gamma}{K}$			4.6
with	$\beta = \frac{\sqrt{ln(K\delta^{-1})}}{nK}$	$\gamma = 1.05\frac{\sqrt{K\ ln(K)}}{n}$	$\eta = 0.95\frac{\sqrt{ln(K)}}{nK}$	4.7, 4.8, 4.9

- If exploration/testing indicates a *stochastic* payoff structure, utilize a variant of *UCB/INF*-bandits for revision:

UCB/INF:	$p_{i,t+1} \Leftarrow \tilde{p}_{i,t+1} \leftarrow \frac{q_i\tau_i}{t+1}\{1\}_{i\notin S} + \frac{1}{\lvert S\rvert}\left(1 - \sum_{j\notin S}\frac{q_i\tau_i}{t+1}\right)\{1\}_{i\in S}$	4.10
with	$\{1\}_C$ denoting an indicator function returning 1 if the clause *C* is *true* and 0 *otherwise*.	4.11

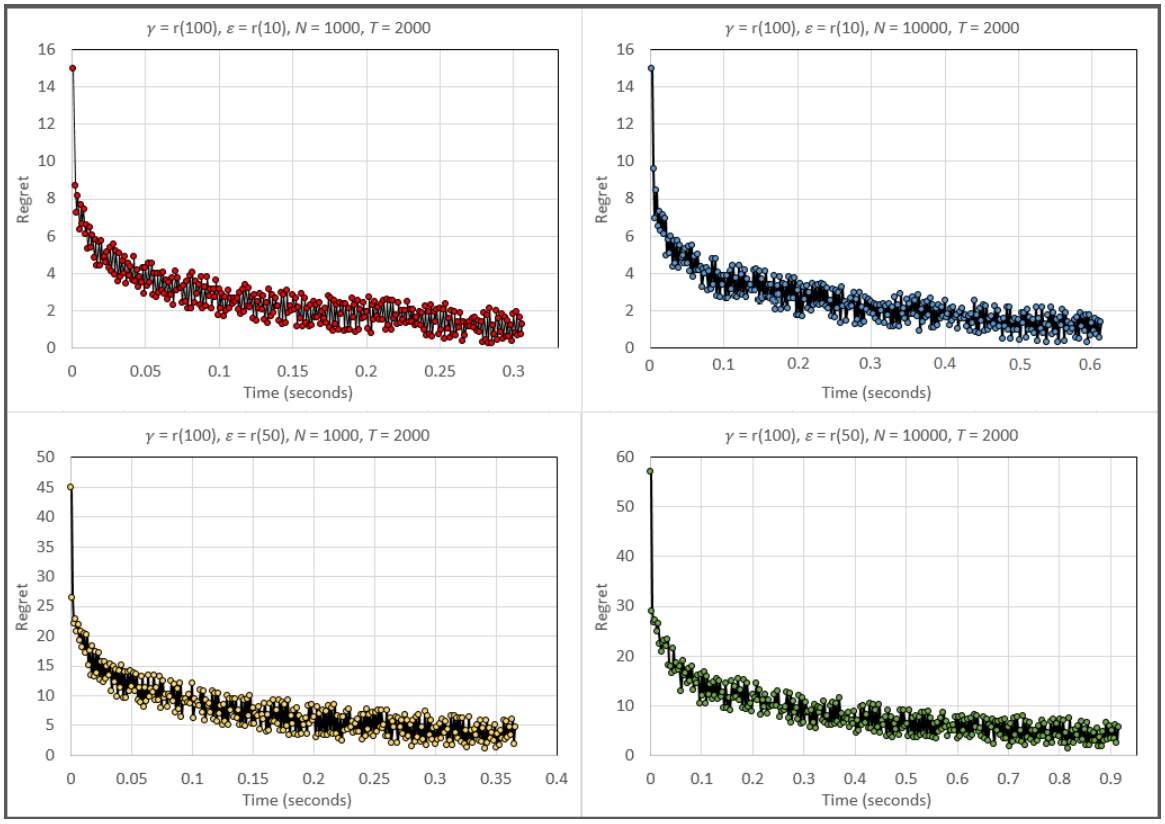
The SAO bandit balances between attacking a weak adversary (stochastic reward environment), and defending itself from a more devious adversary that targets the algorithm's weaknesses. This an example of *defensive forecasting*, which avoids being overly aggressive if the reward sequence is seemingly stochastic. It can nonetheless find optimism in the face of uncertainty [Vovk et al, 2008]. The SAO bandit also has the advantage of avoiding the *Bayesian pathology* of greedy conditioning and myopic variance reduction [Owhadi et al., 2015]. This makes the algorithm indirectly useful for change-point detection, as well as being less prone to overfitting, less brittle, and less readily shattered (c.f. *VC-dimension*) than many alternatives in the literature [Owhadi et al., 2013].

4.3 Benchmarks

Preliminary benchmarks for the GAIGE were conducted using an AMD Phenom II 6-core processor operating at 4.017 GHz with 8 GB of DDR3 RAM at 1866 MHz. The operating system was a 64-bit Lubuntu with Linux Kernel 4.2 and Python 2.77. All tests with the GAIGE were run with multiprocessing modules enabled. The test suite consisted of a varying a set of inputs randomly at each time-step so as to simulate a sequence adaptations to small but ever-present perturbations to the base game. For each round, the array elements a_{ij} of the base game are perturbed by at most $|\varepsilon| = \pm\, integer(\gamma a_{ij})$ according to a uniform normal distribution with parameters $\varepsilon_\mu = 0$ and $\varepsilon_\sigma = 1$, and fixed values γ, N, T by test run. For each iteration through the GAIGE, the MTS algorithm takes $O(N)$ steps to produce pure strategies. These strategies can be mixed by a nearline bloom out of the SAO bandit algorithm, requiring up to an additional T = N steps to achieve regret-minimization within the bounds $L_{stochastic} = O(NlogN)$ to $L_{adversarial} = O\left(N\sqrt{N}\right)$ total time. For these tests, the MTS + SAO anytime responses are forced after a random number of iterations between the bounds N and $N\sqrt{N}$, which tends towards $O(NlogN)$ in expectation. The level of regret achieved by this amount of computation is not necessarily *minimal* but is *competitive*. These tests are therefore setup to provide a crude analysis of a regret ratio (e.g. competitive or minimal) for a prototypical GAIGE implementing the MTS + SAO algorithms on hard inputs. The benchmarks also offer a glimpse into the expected compliance of the GAIGE as a hypermodern solver with soft RTO deadline constraints.

Figure 4.4 displays a small set of benchmarks which illustrate the convergence rates of GAIGE for a variety of conditions.

Figure 4.4. Benchmarked regret convergence for the GAIGE.



Results show that for large $\varepsilon = f(\gamma), N$ and large T, the perturbations do not dominate the base payoffs and sufficient runtime is present for a near-complete regret minimization. As T becomes larger than approximately $\sqrt{N}$, the GAIGE converges on the ε generating functions and fixates towards a single mixture of pure strategies which provide a regret in the vicinity of ε.

The converged strategies chosen by the GAIGE are robust to perturbations of magnitudes less than or equal to $|\varepsilon_{max}|$. The selected strategies offer competitive (and in some cases minimal) regret, and correspond to an ε-approximate trembling-hand perfect equilibrium. This is a highly sought after solution concept for general game-playing agents. Results show the GAIGE achieves these results on modest hardware and for relatively large datasets within sub-second execution times.

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Recapitulation

This thesis examines asset integrity governance under extremely general conditions. The problem is classically formulated as an instance of risk-based planning. The integrity of a physical asset is then governed using the solutions prescribed by an MDP or POMDP model. The first chapter of this thesis highlights several gaps in the existing approaches. It also serves to contrast the difference between *specialized* and *generalized* techniques. Many of the existing asset integrity frameworks are most suitable for/when:

- *Well-defined, project-based integrity assessments*; where sufficient domain context, expert knowledge, and a priori information produce asset-specific performance measures.
- *Assets operating in harsh environments*; where risk-sources are naïve (non-adaptive), and risks represent stochastic background processes and/or accidental event arrivals.
- *Offline analysis*; where integrity modelling is performed using large batches of real or simulated data, and action planning is accomplished over longer, predetermined time-scales.
- *Non-autonomous planning and governance*; where human experts are required to be in the loop on a continuous basis, and direct supervision may be critical to the evaluation and assessment of integrity plans.

Extending these frameworks constitutes the bulk of this work. Asset integrity governance is fundamentally revisited using a more abstract and general interpretation. In Chapter 2, several performance measures are identified from the literature on dynamical models, graphs, complex networks, and dependable systems. This material is crucial to the understanding of the generalized asset as an information structure. Through this paradigm, generalized assets are typically expressed in one of several well-defined modelling forms:

- *Description Languages*; such as the Model Description Language (MDL), the Process Specification Language (PSL), Architectural Component Language (ACL), Unified Modelling Language (UML), and their corresponding file-types: .uml, .xml, etc.
- *Schematic-Defined*; Organizational flowcharts, process flow and logic control diagrams, with file-types: .cad, .fea, etc.
- *Graph-Defined*; Block diagrams, state-transition diagrams, binary-monotone systems, complex networks, etc.

Generalized asset performance is assessed using a combination of fitness-based and similarity-based measures. Chapter 2 also advocates a two-part compilation of model information. This reasoning is valid for all assets, and performed in the following manner:

- *First*, construct several reference configurations using the appropriate description type. For example, binary monotone fitness for MDL, GPI for graph-defined assets, reference files for schematics, etc.

- *Second*, compute the NCD or edit distance between all pairs of reference configurations.

This procedure can be coupled with the expected utility of having the asset maintain a particular configuration. This results in a set of integrity scores which represent universal performance criteria. The integrity scores can be interpreted as state-transition costs, payoffs, derived utility, benefits or rewards, depending on the context.

These values form an array which is representative of a succinct-form game. Chapter 3 elaborates these ideas further, and serves the dual purpose of emphasizing and recruiting game-theory as a basis for asset integrity planning. Several key points can be concluded from this portion of the work:

- *Risks and their sources* - are in general not simply naïve, but rather adaptive and in a broad sense optimal under symmetric information and actionable resources.
- *Hostile environments* - expose an asset to harsh environments as well as the actions of intelligent adversaries.
- *Game-theoretic planning* - extends decision-theoretic planning, and is essential for assets tasked with persisting in hostile environments.

Games can be reasoned about using several different formats. This work contrasts the normal-form, extensive-form, and succinct-form game representations. The most important solution concepts are also examined. These include but are not limited to; Nash Equilibrium, Trembling Hand Equilibrium, Von Neumann Minimax Potential.

Several trends in high-performance planning are outlined in Chapter 4. This work proposed a set of guidelines for solving integrity games via autonomous game-playing agents. Requirements were specified at both the architectural and algorithmic levels. The overall theme was based on a set of demands and constraints called *hypermodern*. Hypermodern planning agents predominantly advocate:

- *Domain-oblivious*; Planning in the face of extreme uncertainty, including few, if any, assumptions regarding the nature of the problem.
- *Agnostic learning*; Unsupervised improvement of activities and results, in the presence of little or no reinforcement feedback, context, or dependencies.
- *Anytime scalable*; Solution quality improves with additional computational resources and exhibits a strong competitive ratio.
- *Defensive Forecasting*; The agent responds in a robust, non-brittle, interruptible real-time online manner (pessimistic reflex). Whilst through anytime scaling, gracefully increases its optimism through a near-line expansion or offline component (proactive deliberation).
- *Non-standard analysis;* The agent is essentially non-bayesian, non-parametric, and mitigates the need for sampling.

Chapter 4 also presented the algorithms implemented by a prototypical GAIGE. Inspired by optimal substructure and dynamic programming, the MTS algorithm was developed to take advantage of the repeated symmetry of the base game. The minimax trajectories produced by MTS are used to generate a preliminary action plan, while also being fed into the SAO bandit algorithm for additional deliberation. The SAO bandit tests the performance of each pure minimax strategy against the game history to determine the nature of the payoff perturbations. The probabilities of selecting a particular strategy are then updated according to the amount of regret experienced by the algorithm, and whether it is consistent with stochastic or adversarial lines of play. This solves the perturbed game by devising mixed strategy equilibria which are approximately trembling hand perfect. The best-known strategies are always available for output, to be leveraged for any appropriate policy control, management platform, or governance oversight.

Finally, a prototypical GAIGE implementation is benchmarked on synthetic data sets. This serves as a first-step towards additional validation, which will stress-test the expected behaviour across several asset classes and operational domains. Results from these rudimentary benchmarks show a GAIGE capable of delivering competitive regret minimization on arbitrary problem types and scaling into large problem sizes.

5.2 Recommendations for Future Work

The themes surrounding this research are relatively broad and shallow, albeit their combination is novel. Generalized asset integrity games are a theoretical, multi-disciplinary construct with many potential applications. Chapter 1 draws on risk-based asset integrity management, existing models and their functionality. Chapter 2 draws its inspiration from results in model theory, information theory, graph theory, complex networks, and dependable systems engineering. Chapter 3 is devoted to game theory, in particular the algorithmic expectations and complexity issues of game structures and their solutions. Meanwhile, Chapter 4 sought lessons from requirements design, systems architecture, autonomous planning, and anytime algorithms. The culmination point is the GAIGE which, as with many frameworks in their infancy, is currently more of a conceptual placeholder than a field-proven technology. It is therefore recommended that future work expand these ideas in four major directions:

(I.) ***Enhanced Representational Power.*** Support the ever more general asset representations and universal evaluation measures as they become available. Plans in the GAIGE represents a sequence of actions, which themselves represent prescribed state-transitions between model reference configurations. What exactly such state-transitions may entail physically is of upstream or downstream concern. The current work examined several modelling institutions and settled on evaluation functions based on graph topology and description complexity. Future work along this line would recruit additional ideas from model and information theory to

produce representations of ever more broader and general classes of assets, and their (integrity) performance.

(II.) ***Architectural and Algorithmic Improvements*.** Autonomous reasoning, planning, and decision-making techniques are continuously improving. Future work along this line would see the GAIGE utilize a wider and more appropriate palette of solution concepts from game theory, optimal control, and dynamical systems theory. It is recommended to recruit from the literature on general game-playing agents, sampling algorithms, agnostic machine learning, and anytime optimization. These endeavours would seek to devise even better regret and complexity bounds.

(III.) ***Implementation Scale-Up.*** The current GAIGE implementation is only prototypical. It was developed as a simple software agent in the Python language. The program code runs as a script/daemon and makes use of multiprocessing capabilities, but is far from optimized. A full-scale software implementation would likely be refactored into C to work in conjunction with other tools and interface with several management information systems. In the limit, a full-scale solver based on the GAIGE could potentially recruit the power of a GPU cluster or custom hardware.

(IV.) **Additional validation.** General game-playing agents are somewhat novel, and there is no strongly agreed upon method of benchmarking. As such, a more encompassing, standardized test suite would have to be developed. Future work should assess the game-playing strengths and weaknesses across highly distinct asset types.

5.3 Closing Remarks

This research was undertaken with respect to a single maxim: abstraction and generalization over focus and specialization. In a world of increasingly fit-for-purpose engineering, application-specific frameworks, and tailor-made solutions, is it at all possible to do more with less? At the very least, this thesis explores this question. In and of itself, the pursuit yields value-adding research and development which is often overlooked – particularly within the sphere of engineering. More pragmatically, these efforts contribute by addressing a real-world, industrial impetus. This work surveyed a large, multi-disciplinary body of literature, and produced a working prototype which holds its own against the state-of-the-art. In essence, the answer to our question appears to be affirmative.

Bibliography

Agrawal, S., & Goyal, N. (2012). Analysis of Thompson Sampling for the Multi-Armed Bandit Problem. *Workshop and Conference Proceedings, Issue 23, Journal of Machine Learning Research.* Microsoft Research, India. pp. 2-26.

Audibert, J.-Y., & Bubeck, S. (2009). Minimax Policies for Adversarial and Stochastic Bandits. *Institute for Research in Computer Science and Automation.* INRIA, Paris, France. pp. 1-10.

Aven, T., & Jensen, U. (1999). Stochastic Models in Reliability. *Applications of Mathematics: Stochastic Modelling and Applied Probability. Springer-Verlag.* New York, NY, U.S.A. pp. 1-284.

Bellman, R. (1957). Dynamic Programming. *Princeton University Press.* Princeton, NJ, U.S.A. pp. 3-18.

Bergstrom, T. (2014). Notes on Uncertainty and Expected Utility. *UCSB Lecture Notes on Economics, University of California, Santa Barbara.* Santa Barbara, CA, U.S.A. pp. 5-7.

Bier, V. (2005). Game-Theoretic and Reliability Methods in Counterterrorism and Security. *University of Wisconsin-Madison.* Madison, WI, U.S.A. pp. 1-13.

Borodin, A., & El-Yaniv, R. (2005). Online Computation and Competitive Analysis. *Cambridge University Press.* Cambridge, United Kingdom. pp. 1-414.

Bowling, M., & Veloso, M. (2000). An Analysis of Stochastic Game Theory for Multi-agent Reinforcement Learning. *School of Computer Science, Carnegie Mellon University.* Pittsburgh, PA, U.S.A. pp. 3-10.

Boyd, J.R. (1976). Destruction and Creation. *Technical Paper, Doctrines on Military Strategy.* Washington D.C., U.S.A., pp. 1-13.

Braess, D. et al. (1968, 2005). On a Paradox of Traffic Planning. *Journal on Transportation Science, translated from the original German version, 1968.* INFORMS, Vol.39, No.4. pp. 1-5.

Browne, C.B., et al. (2012). A Survey of Monte Carlo Tree Search Methods. *Computational Intelligence and AI in Games, IEEE Transactions.* pp. 1-43.

Bubeck, S., & Slivkins, A. (2012). The Best of Both Worlds: Stochastic and Adversarial Optimal Bandits. *Workshop and Conference Proceedings, Issue 23, Journal of Machine Learning Research. Princeton University.* Princeton, NJ, U.S.A. pp. 1-23.

Bunke, H., et al. (2007). A Graph-Theoretic Approach to Enterprise Network Dynamics. *Defense Science and Technology, Birkhauser Publishing.* Bern, Switzerland. pp. 1-237.

Cai, Y., & Daskalakis, C. (2012). On Minimax Theorems for Multiplayer Games. *Lectures on Advanced Game Theory, Massachusetts Institute of Technology, MIT.* Cambridge, MA, U.S.A. pp. 1-16.

Cao, X. (2007). Stochastic Learning and Optimization: A Sensitivity Based Approach. *Springer Science Publications,* Hong Kong University, Hong Kong. pp. 7-42, 53-142.

Cebrian, M., et al. (2007). The Normalized Compression Distance is Resistant to Noise. *IEEE Transactions on Information Theory, Volume 53, Issue 5.* pp. 1895-1900.

Chen, X., et al. (2010). Settling the Complexity of Two-Player Nash Equilibria. *Journal of the ACM 56 (3).* pp. 1-6, 16.

Cox, L.A. (2009). Risk Analysis of Complex and Uncertain Systems. *International Series in Operations Research and Management Science. Springer.* Denver, CO, U.S.A. pp. 1-97.

Condon, A. (1992). The Complexity of Stochastic Games. *Journal of Information and Computation. CiteSeer Publishing.* pp. 1-17.

Curran, R., et al. (2014). Transdisciplinary Lifecycle Analysis of Systems. *Proceedings of the 22nd ISPE Inc. International Conference on Concurrent Engineering, July 2014.* pp. 3-40.

Daskalakis, C., Goldberg, P., & Papadimitiou, C. (2008). The Complexity of Computing a Nash Equilibrium. *Computer Sciences Division, University of California at Berkeley.* Berkeley, CA, U.S.A. pp. 1-8, 54, 62-65.

Diaconescu, R. (2008). Institution-Independent Model Theory. *Institutions, Madhyamaka, and Universal Model Theory, Birkhauser Publications.* pp. 1-4.

Dimand, M.A., & Dimand, R.W. (2002). The History Of Game Theory, Vol 1-2-3: From the Beginnings to 1945, and from 1945 to the Modern Era. *Business and Economics Series, Routledge Publishing.* pp. 142-148, pp. 531-587.

Ebrahimipour, V., and Yacout, S. (2015). Ontology Modeling in Physical Asset Integrity Management. *Springer Publications – Production & Process Engineering.* pp. 33-88.

Eppinger, S., & Browning, R. (2012). Design Structure Matrix Methods and Applications. *The MIT Press, Cambridge, Massachusetts, U.S.A., and London, England.* pp. 317-324.

Fan, J., & Mostafa A. (2006). Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. *College of Computing, Georgia Institute of Technology,* Atlanta, GA, U.S.A. pp. 1-11.

Filar, J., & Vrieze, K. (1997). Competitive Markov Decision Processes. *Springer Series on Operations Research & Decision Theory.* pp. 1-73, 153-156, 161.

Finnsson, H., & Bjornsson, Y. (2010). Learning Simulation Control in General Game-Playing Agents. *AAAI Publications, 24th Conference on Artificial Intelligence*. pp. 1-2.

Fortnow, L., et al. (2005). On the Complexity of Succinct Zero-Sum Games. *Proceedings of the 20th Annual IEEE Conference on Computational Complexity.* pp. 301-333.

Galichet, N., Sebag, M., & Teytaud, O. (2013). Exploration vs. Exploitation vs. Safety: Risk-Aware Multi-Armed Bandits. *Workshop and Conference Proceedings, Issue 29, Journal of Machine Learning Research.* INRIA, Paris, France. pp. 245-260.

Genesereth, M., Love, N., & Pell, B. (2005). General Game Playing: Overview of the AAAI Competition. *AI Magazine Volume 26, Number 2.* AAAI Society. pp. 63-72.

Gertsbakh, I., & Shpungin, Y. (2011). Network Reliability and Resilience. Section 1.3.5: Using D-Spectrum and Signatures. *Springer Briefs in Electrical and Computer Engineering.* pp. 21-24.

Ghallab, M., Nau, D., & Traverso, P. (2004). Automated Planning: Theory and Practice. *Morgan Kaufmann Publishers.* pp. 23-33.

Goguen, J.A., & Burstall, R.M. (1992). Institutions: Abstract Model Theory for Specification and Programming. *Journal of the Association for Computing Machinery 39,* pp. 95–146.

Hauk, T., Buro M., & Schaeffer, J. (2006). Rediscovering *-Minimax Search. *In Computers & Games, Springer Verlag. Department of Computing Science, University of Alberta.* Edmonton, Alberta, Canada. pp. 1-5.

Herbrich, R., & Graepel, T. (2006). TrueSkill (*TM*): A Bayesian Skill Rating System. *Technical Report, MSR-TR-2006-80. Microsoft Research Laboratories.* Cambridge, United Kingdom. pp. 1-5.

ISO 18629-14:2006 (2006). Industrial Automation Systems and Integration: Process Specification Languages and Ontologies. *Parts 11-14. Logic Flow and Resource Theories. International Standards Organization.*

Jackson, M.O., et al. (2011). Epsilon-Equilibria of Perturbed Games. *Department of Mathematical Economics, Stanford University.* Stanford, CA, U.S.A. pp. 1-35.

Jezequel, J.M., et al. (2002). The Unified Modeling Language. Model Engineering, Concepts, and Tools. *5th International Conference. Springer Publishing.* pp. 120-188.

Kaelbling, L.P., et al. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Journal of Artificial Intelligence, 101, 99-134.* U.S.A. pp. 1-36.

Kiekintveld, C. D. (2008). Empirical Game-Theoretic Methods for Strategy Design and Analysis in Complex Games. *PhD Dissertation, Computer Science and Engineering. University of Michigan, 2008.* pp. 1-3.

Khan, F. (2012). Development of a Tool for Risk Based Integrity Assessment of Process Components. *Final Report, Process Engineering, Memorial University of Newfoundland*, St. John's, NL, Canada. pp. 1-89.

Khatab, A., et al. (2009). Availability of K-out-of-N:G Systems with Non-Identical Components Subject to Repair Priorities. *Journal of Reliability Engineering and System Safety, Volume 94, Issue 2*. pp. 1-8.

Kocsis, L., & Szepesvari, C. (2013). Bandit Based Monte-Carlo Planning. *Computer and Automation Research Institute of the Hungarian Academy of Sciences, Kende.* Budapest, Hungary. pp.1-13.

Kreps, D.M. (1988). Notes on the Theory of Choice. *Underground Classics In Theoretical Economics, Westview Press.* pp. 9-56.

Kreps, D.M., & Wilson, R. (1982). Sequential Equilibria. *Econometrica, Vol. 50, No.4, Published by the Econometric Society.* U.S.A. pp. 866-876.

Latora, V., & Massimo, M. (2001). Efficient Behaviour of Small World Networks. *Physical Review Letters, Volume 87, Edition Number 19-8701*. pp. 1-2.

Leyton-Brown, K., & Shoham, Y. (2008). Essentials of Game Theory: A Concise, Multidisciplinary Introduction. *Thesis Lectures on Artificial Intelligence and Learning, Second Edition, Morgan & Claypool Publishers.* pp. 1-88.

Li, M., et al. (2004). The Similarity Metric. *IEEE Transactions on Information Theory, Volume 50.* pp. 1-14.

Misra, B.M. (2008). Handbook of Performability Engineering. *Springer Publishing.* London, UK. pp. 1-211, 308-314.

Nash, J.F. (1950). Equilibrium Points in N-Player Games. *Proceedings of the National Academy of Sciences.* U.S.A. pp. 48-49.

National Aeronautics and Space Administration (NASA). (2015). *Proceedings from the UTM 2015 conference on Unmanned Traffic Management Systems*. Weblink: http://utm.arc.nasa.gov/index.shtml.

Nisan, N., Roughgarden T., et al. (2007). Algorithmic Game Theory. *First Edition. Cambridge University Press.* New York, NY, U.S.A. pp. 1-726.

Nykter, M., et al. (2008). Critical Networks Exhibit Maximal Information Structure-Dynamics Relationships. *Physical Review Letters, 100, 058709.* pp. 1-10.

Owhadi, H., Scovel, C., & Sullivan, T. (2013). When Bayesian Inference Shatters. *Mathematics and Statistics Theory, Cornell University Library.* ArXiv.org. pp. 1-7.

Owhadi, H., Scovel, C., & Sullivan, T. (2015)a. On the Brittleness of Bayesian Inference. *Department of Computing and Mathematical Sciences, California Institute of Technology.* Pasadena, CA, U.S.A. pp. 1-16.

Owhadi, H., Scovel, C., & Sullivan, T. (2015)b. Brittleness of Bayesian Inference under Finite Model Information in a Continuous World. *Electronic Journal of Statistics, Volume 9.* pp. 1-7.

Picci, G., & Gilliam, D.S. (1999). Dynamical Systems, Control, Coding, Computer Vision: New Trends, Interfaces, and Gameplay. *Progress in Systems and Control Theory, Vol.25, Birkhauser Publishing.* pp. 324-333.

Ricceri, B., & Simons, S. (1998). Minimax Theory and Applications. *Nonconvex Optimization and Its Applications, Volume 26, C.I.P. Issue.* pp. 1-271.

Rozenberg, G. (1997). Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. *Scientific Publications.* pp. 401-442.

Schoenebeck, G., & Vadhan, S. (2006). The Computational Complexity of Nash Equilibria in Concisely Represented Games. *Proceedings of the 7th IEEE Conference on Electronic Commerce.* pp. 270-279.

Silver, D., & Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. *University College of London.* London, United Kingdom. pp. 4-9.

Sion, M., & Wolfe, P. (1957). On a Game Without a Value. *Contributions to the Theory of Games III, Annals of Mathematics Studies 39, Princeton University Press.* U.S.A. pp. 229-306.

Sleight, J., & Durfee, E.H. (2013). Organizational Design Principles and Techniques for Decision-Theoretic Agents. *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems.* International Foundation for Autonomous Agents and Multi-agent Systems. pp. 463-470.

Smallwood, R., & Sondik, J. (1974). The Optimal Control of Partially Observable Markov Processes Over a Finite Horizon. *Journal of Operations Research, Vol.21, INFORMS.* pp. 1071-1088.

Smolensky, P. (1986). Information Processing in Dynamical Systems: Foundations of Harmony Theory. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1.* MIT Press, Cambridge, MA, U.S.A. pp. 194-281.

Stone, P. (2007). Learning and Multi-agent Reasoning for Autonomous Agents. *Technical Slides, Department of Computer Sciences, University of Texas at Austin*, Austin, TX, USA. pp. 1-45.

Sudakov, B., & Vu, V.H. (2008). The Local Resilience of Graphs. *Wiley Inter-Science Series. Department of Mathematics, UCLA.* Los Angeles, CA, U.S.A. pp. 1-23.

Thayer, J., & Wheeler, R. (2010). Anytime Heuristic Search: Frameworks and Algorithms. *Department of Computer Science, University of New Hampshire.* Durham, NH, U.S.A. pp. 1-8.

Valiant, L. (1984). A Theory of the Learnable: Probably Approximately Correct. *Research Contributions in Artificial Intelligence and Language Processing.* ACM. pp. 1-9.

Vitanyi, P.M. Et al. (2008). Normalized Information Distance. *Lecture Notes, CWI.* Amsterdam, Netherlands. pp. 1-33.

Von Neumann, J. (1928). On the Theory of Games. *Proceedings Rendered to the Academie of Sciences, 18th of June, 1928.* pp. 1689-1691.

Von Neumann, J., & Morgenstern, O. (1944). Theory of Games and Economic Behavior. *Princeton University Press.* Princeton, NJ, U.S.A. pp. 1-2.

Vovk, V., Takemura, A., & Shafer, G. (2008). Defensive Forecasting. *International Statistical Review.* pp. 1-15.

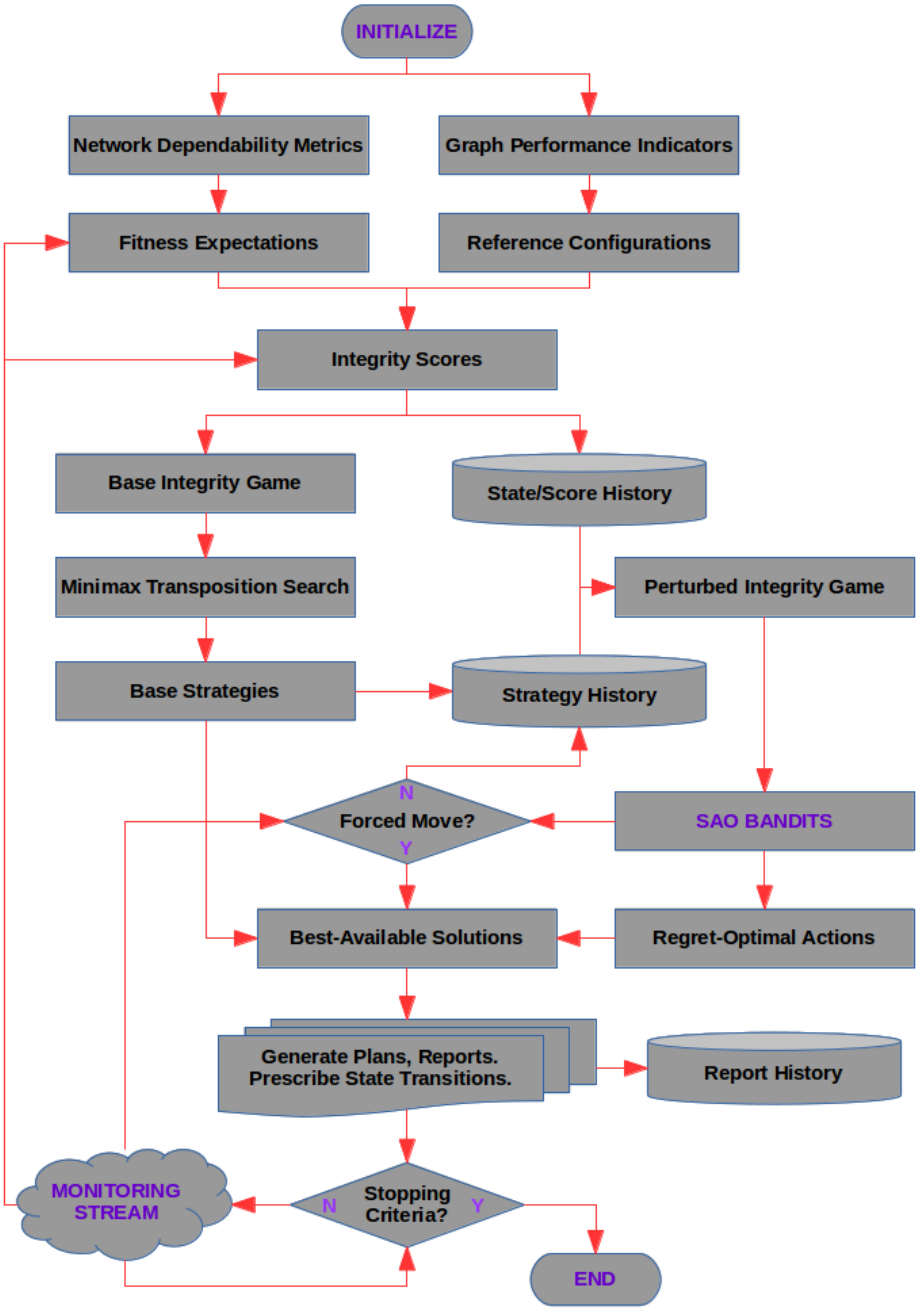
Wikispaces on Chess Programming. (2015). *Proceedings retrieved from the wikispaces on chess-programming, chessprogramming.wikispaces.com, Search in Chess and Chess Engines.* pp. n.

Zelazo, D., & Mesbahi, M. (2010). Graph-Theoretic Methods for Networked Dynamic Systems: Heterogeneity and H-norm Performance. *Department of Aeronautics and Astronautics, University of Washington,* Washington, OR, U.S.A. pp. 1-33.

Zilberstein, S. (1996). Using Anytime Algorithms in Intelligent Systems. *AI Magazine, Volume 17, Number 3.* AAAI, U.S.A. pp. 73-83.

Zio, E., et al. (2006). Determining the Minimal Cutsets and Fussell-Vesely Importance Measures in Network Systems by Simulation. *Department of Nuclear Engineering, Polytechnic of Milan.* Milan, Italy. pp. 1-7.

Appendix I – Architectural Overview of the GAIGE



Appendix II –Potential Applications and Worked Examples

This appendix revisits the concepts of generalized assets through examples. Each scenario is representative of a broad class of assets, processes, systems, as well as their typical modelling paradigms. The selected case studies illustrate how a distinct, real-world asset may be approached and refactored as a generalized asset. This epitomizes much of the pre-processing required before ongoing planning and analysis can be effectuated using the GAIGE. It is hoped that these worked examples will further demonstrate the range of potential applications, shedding light on the inherent versatility of this formulation, architecture, and implementation.

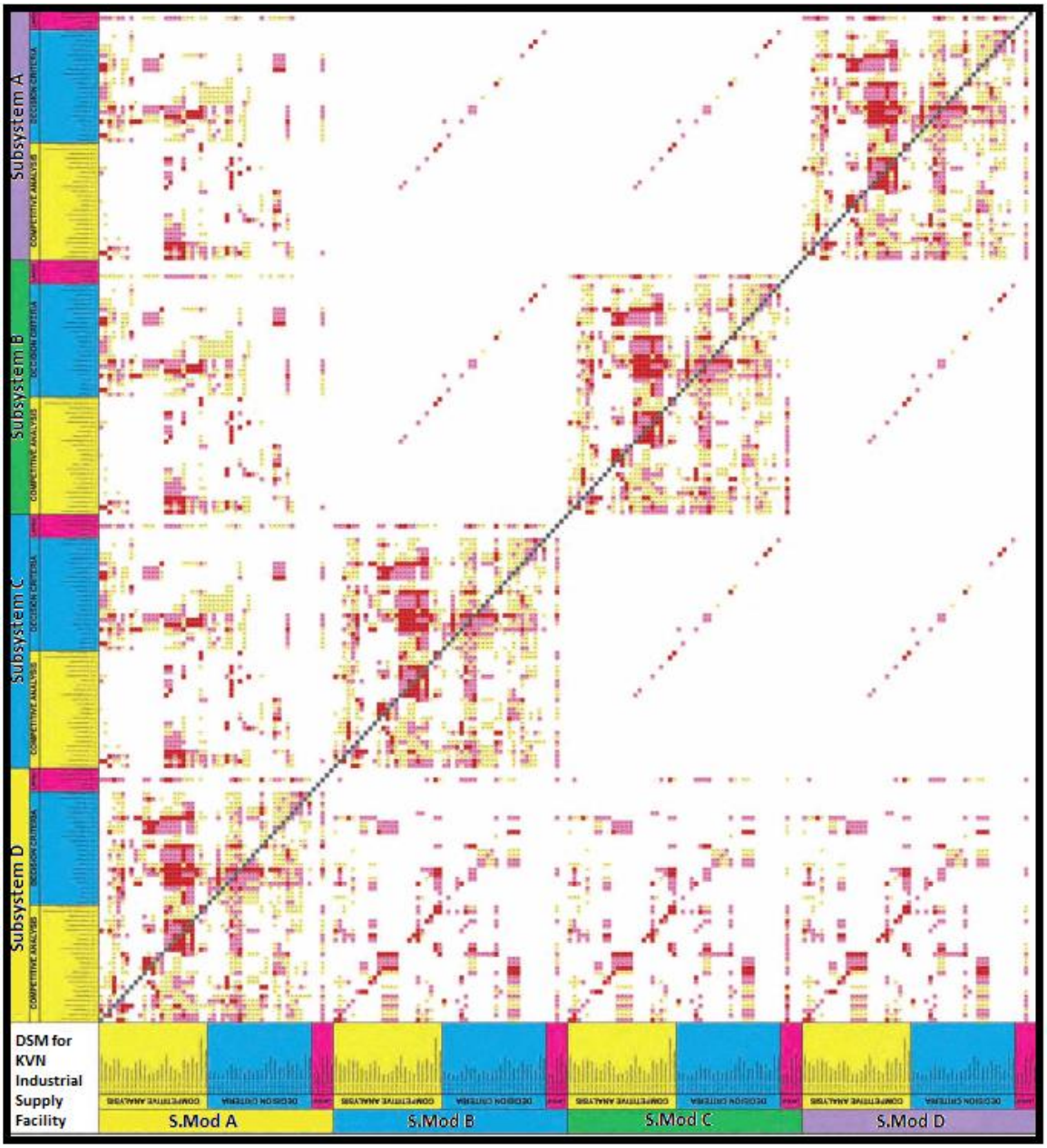
Example 1. Cyber-Physical Asset given by its Design-Structure-Matrix (DSM). Case study of a networked industrial supply facility for the KVN Company.

Systems engineering of products, processes, and organizations require tools and techniques for decomposition and integration. A design structure matrix (DSM) provides a simple, compact, and visual representation of a complex system that supports innovative solutions to decomposition and integration problems [Eppinger and Browning, 2012]. Generalized Assets may be specified using a DSM, which offers an alternative representation for the analysis of critical system components and their interactions. In this example, a physical asset is a localized industrial network within some warehouse, port, or

manufacturing and supply facility. The physical asset components consist of heavy-duty materials handling equipment, transportation machinery, and storage modules. This asset also has a number of cyber components including a wireless sensor network for surveillance, monitoring and tracking services, autonomous robots and devices for scanning and packaging, and an integrated information management system. The objective is to operate these multi-domain subsystems and their critical components together in a parsimonious manner. Here, the generalized asset is the active information structure which captures the relevant features of the entire cyber-physical system and unifies their indicators for evaluation. The generalized asset integrity represents a global state of idealness or correctness between the many functional dependencies, interactions, and process influences. This also includes the fitness levels of the various subsystems and components.

Figure A2.1 depicts a DSM model which has already been developed for this particular cyber-physical asset. The DSM maps naturally into a generalized asset representation. The DSM model implicitly includes many of the graph performance indicators and fitness-based assessments of Section 2.1, such as vertex centrality and binary monotone importance. Therefore, less work is required to compose integrity scores "from the ground up". As an added bonus, the nature of this particular asset is such that many of its components are locally intelligent and implement self-diagnostic protocols. This allows a treatment of certain decision-making criteria, such as repair, maintenance, and servicing forecasts, to act as elements within the DSM itself.

Figure A2.1. DSM for the KVN industrial supply facility as a generalized asset.



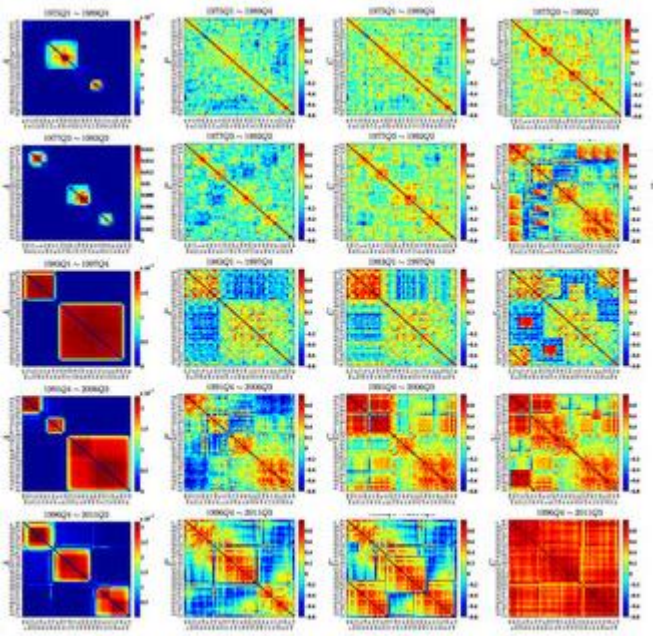
The “raw” generalized asset is therefore a 216 x 216 matrix formed directly from the DSM itself. By amalgamating knowledge from KVN engineers, systems integrators, suppliers, and floor managers, the major dependencies were reduced to the following criticality levels: 3 (red) – very strong, 2 (pink) – medium, and 1 (yellow) – low. Although these interactions revealed many dynamic relationships, a static and domain-oblivious analysis was performed to derive the baseline reference configurations for the generalized asset. Many scenarios were considered, with most falling into one of the following categories:

- The class of complete shutdown conditions, owing to multiple failures, unplanned downtime, catastrophic exposures, etc.
- A class of planned shutdown or (near)-offline conditions.
- Various “lights out” operating conditions, owing to major operations performed autonomously by cybernetic systems, and/or with humans-out-of-the-loop during off-hours.
- A set of high-output and maximum performance conditions in terms of energy use, risk exposure, and utility return.
- The class of low-energy, low-hazard, or low-risk conditions, again owing to times where business continuity can be maintained without heavy production and equipment aging. This includes scenarios with no presence of hazardous materials, minimal storage, occupancy, personnel, as well as other dependencies which mitigate workplace entropy.

- Various cyclical and transient conditions, owing to various on-demand and just-in-time production and off-equilibrium output to optimize value, chiefly resulting from the cycling of logistical processes such as resupply, rerouting, and housekeeping tasks.
- A set of randomly generated “anomalous” conditions owing to the cyber-physical nature of the asset, as well as the potential exposure to intelligent threats, software risks, and other unforeseen operating regimes.

These conditions were used to construct a set of representative reference configurations from the (unwieldly large) set of possible configurations. Figure A2.2 illustrates how a DSM-defined generalized asset would appear configured under a number of different scenarios.

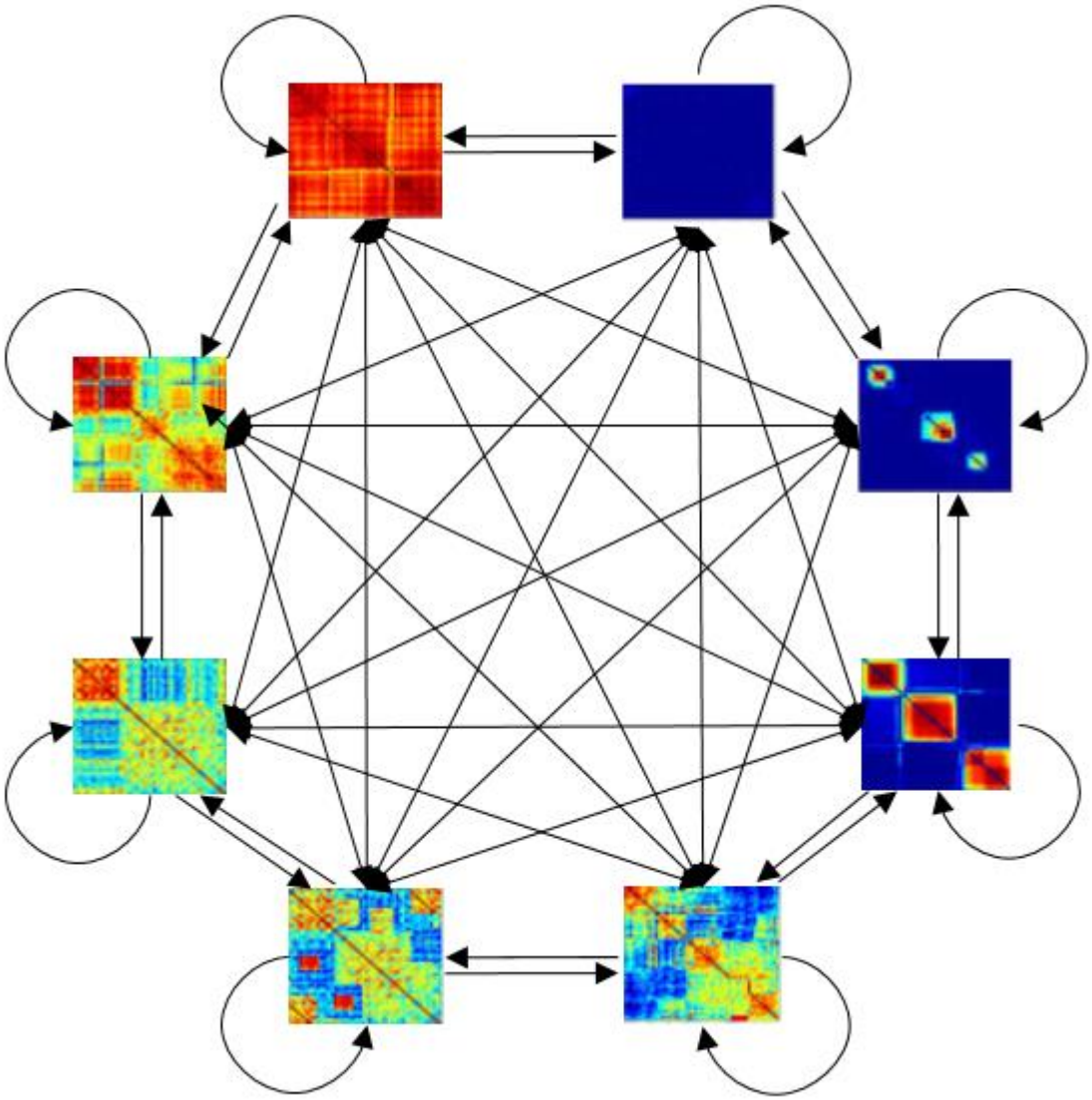
Figure A2.2. DSM for KVN industrial supply facility under different scenarios.



Constructing the set of reference configurations characterizes an offline pre-processing step through fitness-based metrics which require significant computational resources. Further evaluations involve the hashing of integrity scores based on similarity-based measures such as the normalized compression distance (NCD). This alleviates the need for additional fitness-based assessments, and is conducive towards hypermodern analysis. In this example, the reference DSM are sufficiently compact, suppressing the need for sophisticated decomposition and encoding schemes, such as the joint spectral distance or algebraic matrix characterization methods. The standard $< K_v, K_e, K_N > -asset$ representation is applicable, with $K_v = 1$, $K_e = 2$, $K_N = 20$ offering a complete state-transition-graph with 20 vertices, and 400 edges. The state-transition-system formed by this asset is therefore a 20 x 20 weighted adjacency matrix, with each of the 400 elements

requiring an integrity score evaluation. For simplicity of depiction, Figure A2.3 illustrates the state-transition-graph formed by a more simplified $< 1,2,8 >$ asset.

Figure A2.3. A <*1, 2, 8*>-*complete* graph for the KVN facility.



The similarity between each pair of DSM configurations is computed using the NCD. To perform the NCD computations, the DSM are first converted into bit arrays and saved as .dsm files without header metadata in a manner similar to the raw (uncompressed) .gif file format. For this application the GZIP program was used as the compressor to compute NCD values. The GZIP program implements a lossless compression algorithm of Lempel and Ziv (LZW), and is suitable for NCD comparisons between generic file objects.

Each of the reference configurations assumes an inverse monotonic relationship between the NCD and long-run expected Von-Neumann utility. If the generalized asset is modelled coherently and consistently, then large differences between descriptions (e.g. two particularly different DSM configurations) correspond to large differences in derived utilities. This is useful in developing the final integrity scores, which map and scale the NCD values into integer numbers for computational speedups. The integrity scores are the assigned edge weights of the final state-transition graph which is used as input into the GAIGE and corresponds to a succinct-form integrity game. The integrity scores always represent compound "*move and hold*" operations. The magnitude of a score is derived from two contributions: (i.) the NCD value of the "*move*", and (ii.) the VNM-utility value of the "*hold*". The *move* is captured by the inverse NCD score, and signifies the degree of rework costs. A large NCD parallels numerous component change-orders for the asset manager, such as maintenance and repair activities at the KVN facility. A large NCD could also represent numerous disruptions and more focused destructive efforts on the part of an intelligent adversary or natural antagonist. The *hold* simply provides for the vector sum of the relative utility derived from all productions, operations, costs and losses, revenues,

liabilities, and other socio-economic benefits for maintaining the asset in a particular global state (therein normalized for one time-step). In the current work, the combined *move and hold* integrity score is collected and rewarded immediately and is not future or past discounted.

Because of real-world limitations, some state transitions may be "nigh impossible" to achieve. This might represent the cyber-physical constraints of delivering extreme repairs or damages in relatively short time steps. In these extreme cases, a particular edge of the graph would be blocked and the respective integrity score(s) would simply be replaced with a null symbol, effectively reducing the number of state-transitions available to a player. This particular example uses reference configurations which were chosen in such a way as to always be available within one time step, albeit with potentially strategically important magnitudes, reflected by very large or very small integrity scores.

Figure A2.4 tabulates the NCD, VNM, and overall integrity scores for one potential reconfiguration of the asset. These scores represent the potential reward derived from a global state transition from the offline condition (source vertex labelled as the 0^{th} vertex, *vertex_i*) to any of the $N = 20$ other reference conditions (*vertex_j* destination vertices). Values in the VNM ($/minute) column are present-value forecasts derived from an expected annual operating income of $63M for the asset. This corresponds to approximately $120/minute of uptime in the maximal state and $70/minute uptime in the "sweet spot" of ideal working state (in a static sense).

Figure A2.5 plots the performance scaling of these reconfiguration metrics. In these figures, the source vertex is *vertex_0*, i.e. the asset existing in an offline condition. In kind,

vertex_20 denotes a fully working configuration at maximum levels of production. This is evidenced by the higher VNM utility. This transition however, does not correspond to the highest integrity score. This is because a reconfiguration from *vertex_0* to *vertex_20* is dominated by the *move* component rather than the *hold* component in the compound *move-and-hold* operation registered by the integrity scores. In other words, minute for minute, it is far more expensive to restore the KVN facility from a totally offline to a completely working state then it is to capitalize on profits incurred immediately thereafter. If the integrity game were played over a single time-step, the optimal transition would be vertex_0 to vertex_14, resulting in the best compromise between the NCD and VNM for a maximum integrity score yield. Viz. generalized assets, the payoffs owing to a reconfiguration are realized by a combination of transition effects and their immediate returns, and these do not always correlate with higher performance levels using classical performance metrics such as network fitness or system availability.

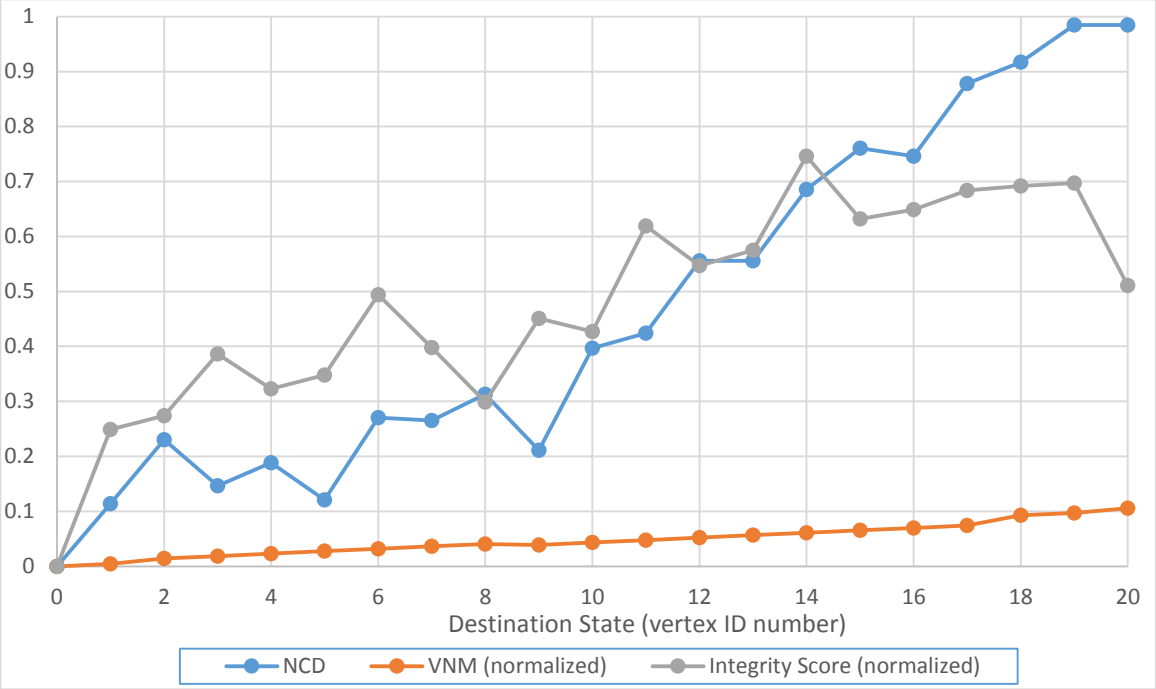
Figure A2.4. Asset reconfiguration potentials from the offline state to all other reference conditions.

Source	Destina-tion	NCD	VNM ($/mi-nute)	VNM (nor-malized)	Integrity Score	Integrity Score (normalized)
0	0	0	0	0	0	0
0	1	0.1141	5	0.0044	1000	0.249
0	2	0.2303	16	0.0141	1100	0.274
0	3	0.1467	21	0.0185	1550	0.386
0	4	0.1883	26	0.0229	1300	0.323
0	5	0.1209	31	0.0274	1400	0.348
0	6	0.2706	36	0.0318	1200	0.299
0	7	0.2652	41	0.0362	1600	0.398
0	8	0.3132	46	0.0406	1984	0.494
0	9	0.2111	44	0.0388	1812	0.451
0	10	0.3967	49	0.0432	1717	0.427
0	11	0.4242	54	0.0477	2490	0.619
0	12	0.5555	59	0.0521	2200	0.547
0	13	0.5555	64	0.0565	2310	0.575
0	14	0.6856	69	0.0609	2054	0.511
0	15	0.7604	74	0.0653	2540	0.632
0	16	0.7458	79	0.0697	2608	0.649
0	17	0.8782	84	0.0741	2750	0.684
0	18	0.9174	105	0.0927	2780	0.692
0	19	0.9845	110	0.0971	2800	0.697
0	20	0.9847	120	0.1059	3000	0.746

- Vertices ordered by k-out-of-N:G criteria.

- EdgeID = <vertex_source, vertex_sink>

Figure A2.5. Plot of the performance scaling for a reconfiguration of the asset (offline to all other configurations).



These figures provide some insight into the expected integrity levels and performance characteristics of the generalized asset. Enumerated to exhaustion (every source to every destination), they also express a succinct form payoff matrix for the base integrity game. The full 400 x 400 input array is used to initialize governance of the asset via the GAIGE. As the asset evolves in real-time, the integrity scores are updated as component conditions are reported through an aggregation of various monitoring and sensory channels which integrate a supervisory control and data acquisition (SCADA) stream. The GAIGE periodically performs a simple check to see if the array has sufficiently changed to warrant a re-computation of the optimal strategies. If the integrity scores have

changed significantly, the GAIGE performs an MTS of the new array and attempts additional deliberation through the use of SAO-bandits. The best-available strategies are always available for generation as an output to the user or to the information management system. The planning horizon for the asset accounts both stochastic and adversarial disruptions. Asset behaviour was simulated over 360,000 one-minute time-steps using historical data obtained from standard operations at the KVN facility. This is equivalent in duration to 6000 hours of operation or a period of 24/7 uptime across 250 days. At each time-step, a new payoff matrix was generated based on a perturbation of the previous integrity scores, with each value drawn from an unknown distribution amongst some family of distributions (e.g. Bernoulli and exponential). For stopping-time simplicity, a random amount of additional deliberation (up to 10 seconds) was accorded every 100^{th} time-step to visualize the effect of SAO-bandit "corrections". The strategies reported by the GAIGE were smoothed up to larger time-scales for illustration. Figure A2.6 reports a sample sequence of prescribed state-transitions relative to five initial conditions (given by DSM configurations at vertices 0, 5, 10, 15, and 20) over 1 hour intervals. For these time steps no SAO-bandit deliberation was performed. A perturbation event was introduced at time-step 20, causing many of the strategies to resettle under different minimax trajectories. The event can be interpreted as an arbitrary disruption, tremble, or noisy realization of state which causes the integrity scores to change, inducing a correction to the optimal lines of play.

Figure A2.6. An example set of state-transition trajectories provided by the GAIGE.

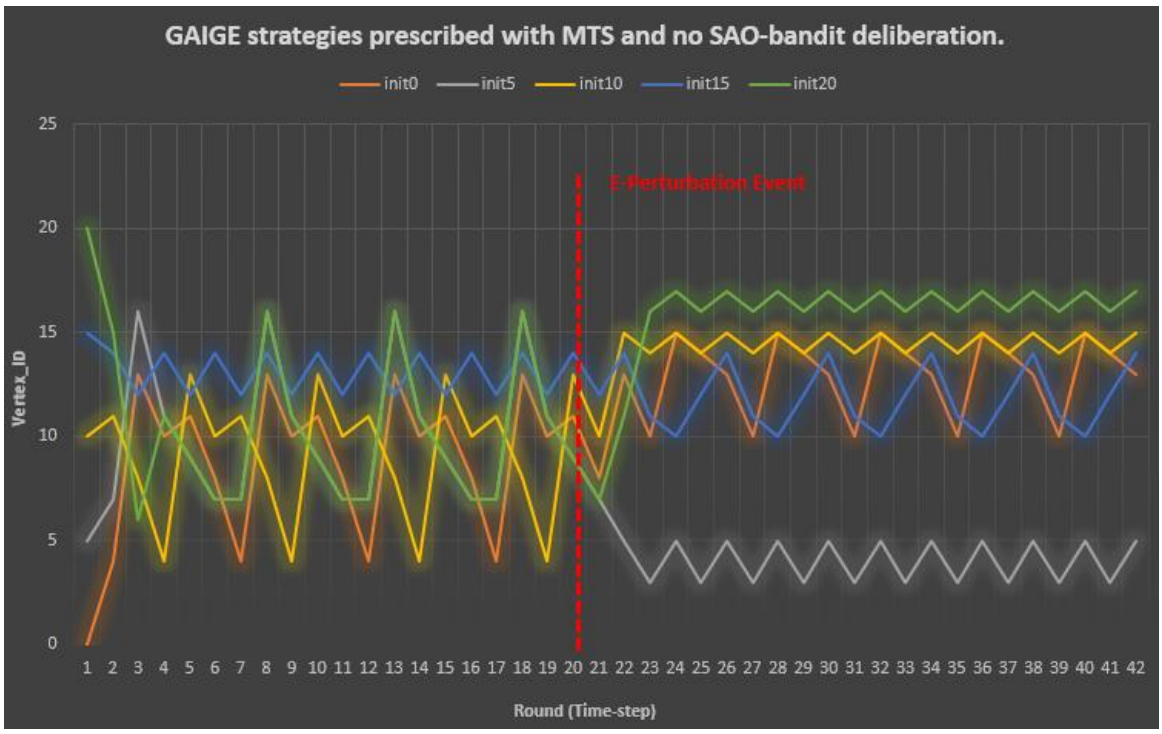
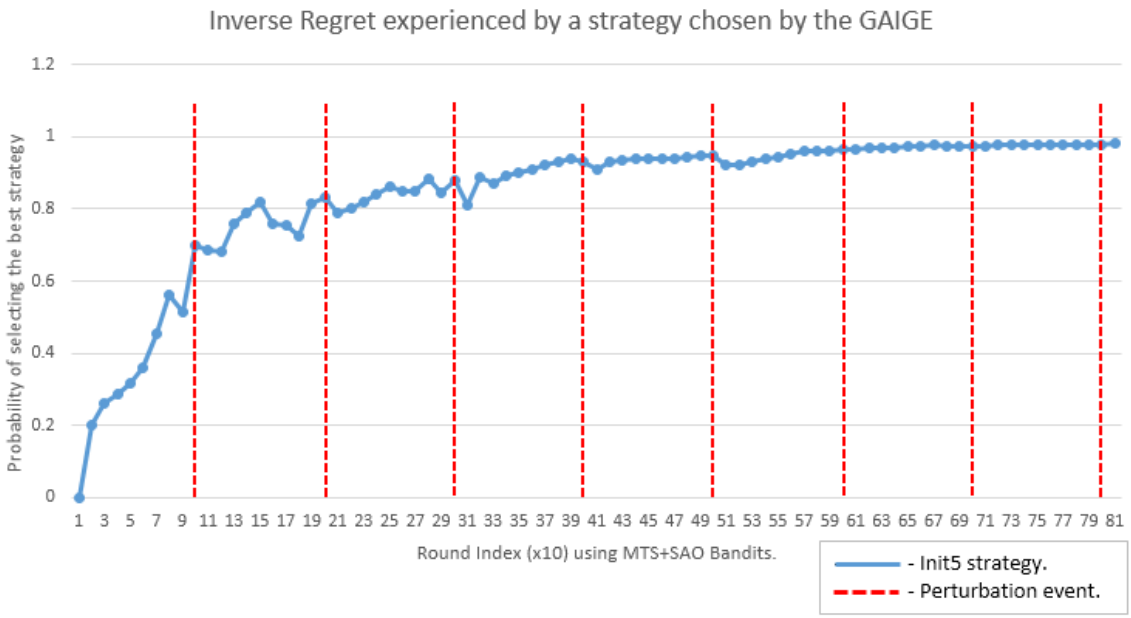


Figure A2.7 showcases the level of regret experienced by the GAIGE relative to a correct line of play in hindsight. The figure is developed for a single strategy profile based on the asset being initialized in configuration 12. The GAIGE chooses actions using the MTS algorithm only, except every 10^{th} round where it is allowed access to up to 10 seconds of additional deliberation using the SAO-bandit. These anytime intervals are counted as a single update step, during which a history of regret is used to examine the most promising strategies and adjust future actions.

In these tests, *regret* was revealed as the difference between the total integrity score so far accumulated by the GAIGE (using MTS and every so often MTS+SAO), and the integrity score of a perfectly "omniscient" player using a fixed strategy that also accounts for all future perturbation events. This corresponds to the probability of selecting the correct strategy or sequence of actions. Figure A2.7 shows a convergence which is progressively more resilient to adversarial and stochastic perturbations. This is a defensive yet opportunistic integrity plan which also corresponds to an approximate trembling-hand equilibrium.

Figure A2.7. Example of the (inverse, cumulative) regret from the actions prescribed by GAIGE using a combination of MTS and SAO algorithms.



The GAIGE would be regularly updated in a live setting, where it would interface and support other information systems and decision support tools. A more robust, scaled-up version could theoretically act as a force-multiplier for the asset integrity authorities at KVN. Additional context, expert/domain knowledge, and specialized optimizations would likely be required for an actual production environment. Nonetheless, by way of this generalized, agnostic methodology, one can effectively govern the high-level operational configuration of any DSM-defined asset. The KVN facility is a networked industrial centre for supply and distribution operations. It represents a cyber-physical asset. Passing the DSM representation into generalized asset form and then planning via the GAIGE enables advanced budgeting for supply chain disruptions, equipment downtime, and machinery throttling.

Example 2. Critical infrastructure assets given by a complex evolving network of interdependencies. Case study: Integrity governance for an Unmanned Aerial Traffic Management System (UTM).

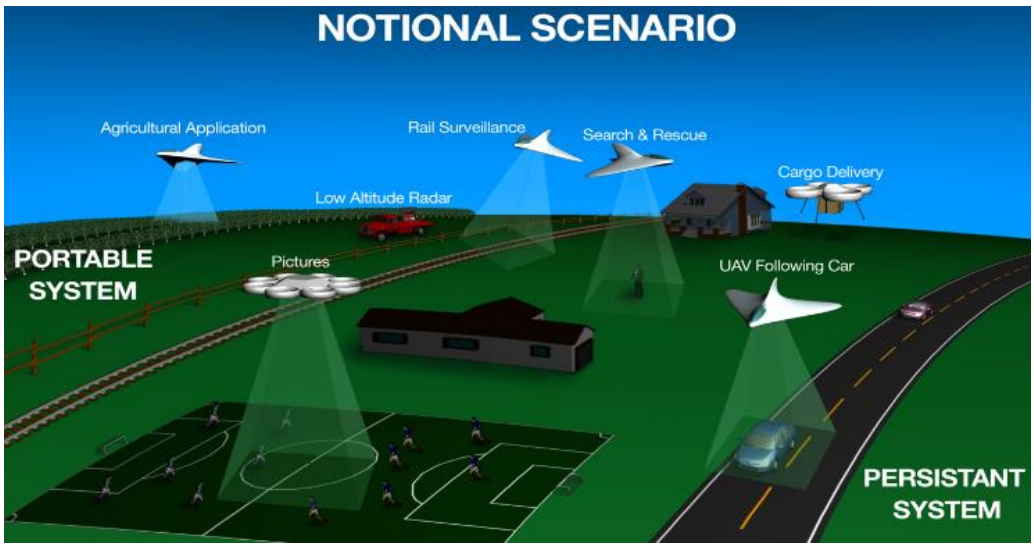
Unmanned Aerial Vehicles (UAVs) and Systems (UASs) are witnessing widespread deployment as force-multipliers for several industries and public-serving sectors. A small subset of potential applications include agricultural, forestry, and natural resources data collection and condition monitoring, rail and road network surveillance, low-altitude radar and speed enforcement, aerial telemetry and streaming, search and rescue, and cargo delivery.

It is anticipated that the increasing ubiquity of UAVs and UASs will drive the development of Unmanned Aerial Traffic Management Systems (UTMs). A notional UTM may be small or large scale, while publically or privately owned and operated. Various UTM architectures have been put forth, with no particular consensus on the applicable topology (e.g. distributed or centralized), legal frameworks, regulations, or governance practices. Nonetheless, the potential for low-altitude, short and long term air-space leasing has driven the need for integrated management systems which autonomously guide and direct fleets of UAS assets. The near-term goals of UTM architectures are to enable low-altitude UAV/UAS operations with demonstrated safety and security. The long-term goals for a UTM typically seek to tighten requirements and strengthen capabilities. These may include addressing emerging threats and vulnerabilities, improving efficiency, and increasing asset/fleet capacity, autonomy and endurance.

UTMs are typically classified as *portable* or *persistent*. A *persistent* UTM is optimized for high-availability missions and implements centralized governance of a large territory, such as a provincial district. In this sense it is very much akin to existing air-traffic management (ATM) infrastructure. The *portable* UTM is more embedded. It is favored for ad hoc, high-utility missions and private projects. It manages a smaller fleet of assets over a localized region, such as a city or county. The portable UTM can often be field-deployed alongside several of the UAV/UAS whereas the persistent UTM operates out of a fixed facility. Both classes of UTM support differing business models owing to different advocacy groups and use cases.

Figure A2.8 illustrates a UTM based on novel proposals from NASA and commercial partners who are enabling the development of a Low Altitude Traffic and Air Safety (LATAS) Platform.

Figure A2.8. Notional UTM for UAV/UAS operations, courtesy of NASA.

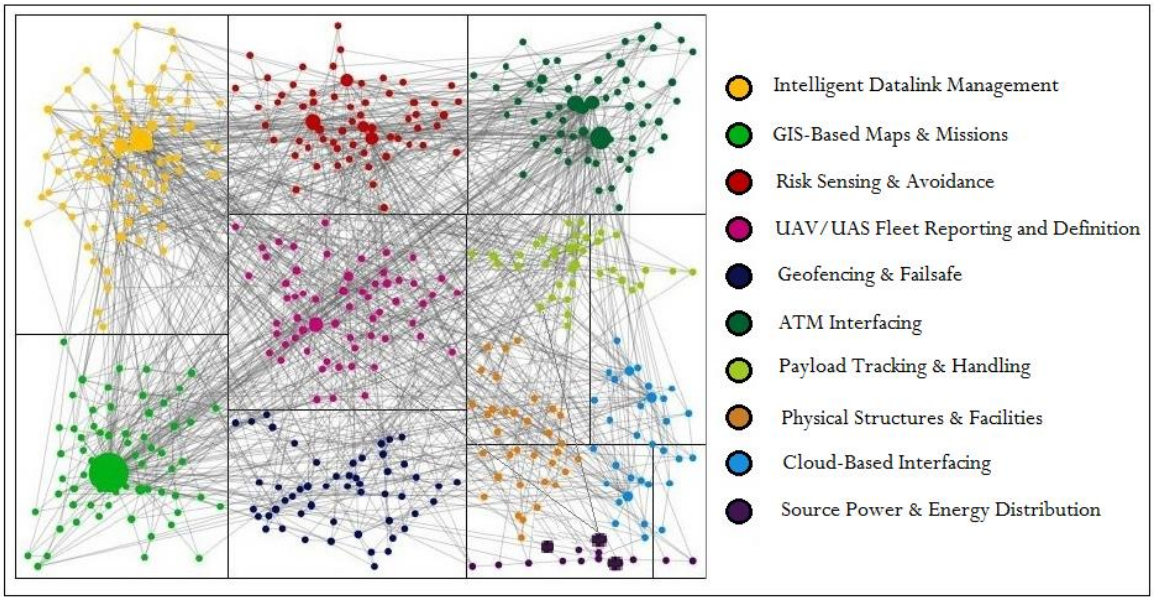


Ongoing UTM and LATAS development efforts seek to connect and integrate leading airspace management technologies into an infrastructure-as-a-service (IaaS) package for commercial and recreational drone operators, as well as regulators and air traffic controllers.

In this example, we consider a private IaaS provider certified by a national regulating authority to operate a fixed entity (persistent) county-level UTM. The UTM is modelled holistically as a system-of-systems (SoS). This choice of modelling approach allows a rapid integration of critical asset features. During preliminary design, a static dependability analysis of the components and subsystems was conducted. This was a bottom-up offline analysis, beginning with reliability block diagrams (RBD) and fault-tree analyzes (FTA). Results provided insight into the availability and disposition of several asset dependencies and their logical relationships. This knowledge was used to perform a series of fitness evaluations of the systems within the overall SoS. The SoS itself is a large collection of hierarchically clustered interacting manifests. The SoS is also a graph, and when properly defined embodies yet another example of a generalized asset.

The expected (dynamic) operating risks, importance levels, and criticality of various elements are found using several of the graph-performance indicators (GPI) discussed in Chapter 2. Figure A2.9 presents a summary of the SoS which is used to form a reference configuration for the UTM as a generalized asset.

Figure A2.9. The generalized asset information structure for an Unmanned Aerial Traffic Management System (UTM).



The proposed UTM is modelled as a generalized asset given by a SoS with critical modelling ensembles:

- ***Intelligent Datalink Management***. Manages the communications, protocols, link and access control events, error correction and control, etc.
- ***GIS-Based Maps and Missions***. Provides the collaboration between geomatics and GIS systems to enable GPS/satellite guided mapping and dynamic mission planning services.
- ***Risk Sensing & Avoidance***. Encompasses the cognitive protocols, SCADA, and swarm optimization algorithms for the primary sensing, detection, and avoidance tasks.

- ***UAV/UAS Fleet Reporting and Definitions***. Primary SCADA for fleet operations including individual and coordinated mission logistics, airspace status, physical health, and condition monitoring for deployed and reserve UAV/UAS assets.
- ***Geofencing and Failsafe***. Manages the failsafe behaviour for various flight modes, including operating range constraints, signal strength limitations, airspace coordinate safety limits, emergency procedures and lost-link return/evacuation policies.
- ***Air Traffic Management (ATM) Interfacing***. Encapsulates the correspondence with proper ATM systems from other authorities for weather information, auxiliary traffic reports, or other commands and notifications.
- ***Payload Tracking and Handling***. Manages the end-to-end processing of physical payloads for delivery or digital contingency services such as reconnaissance and remote data acquisition. Handles the induction queuing, shipping confirmation, and de-queueing of parcels and packets through the UTM facilities and airspace infrastructure.
- ***Physical (Ground) Structures and Facilities***. Management of the UTM supporting physical infrastructure, including operating grounds and facilities, warehousing, ground vehicles, and related monitoring, security and surveillance.
- ***Cloud-Based Interfacing.*** Includes components which regulate the cloud-based platforms and services such as physical hardware, remote storage and access, internet-driven protocols, processes and devices, and user-enabled functionality.

- ***Source Power and Energy Distribution.*** The subsystems responsible for interfacing with power/utility infrastructure and supplying energy. Includes regulation and distribution of electrical power and backups within the UTM.

The UTM is expected to persist in a hostile environment. The UTM is exposed to a variety of risks but is predominantly sensitive to adverse weather conditions, extended power grid outages, communications jamming and disruption, as well as unforeseen technical failures. Adversarial threats include deliberate airspace congestion, landing zone obfuscation or harassment, malicious actions against drones, false orders/requests, and many others. Multiple safety and security measures are in place to minimize the attack surface by design. Despite these efforts, the UTM remains exposed to the continuous threat of cyber-attacks. Several potential vectors exist, including actions which glitch the SCADA or GIS subsystems to provide incorrect mission commands, as well as disabling or hijacking aerial assets. Attacks may also attempt to compromise the privacy and safety of sensitive data or payload information. Other vectors include the distributed denial-of-service (DDoS) attacks on the various external interfaces. A weaker adversary might utilize these tactics in an attempt to disrupt the quality of service (QoS). Finally, there are vulnerabilities to physical intrusion and damage to the facilities, owing to perimeter surveillance and security failures (or attacks).

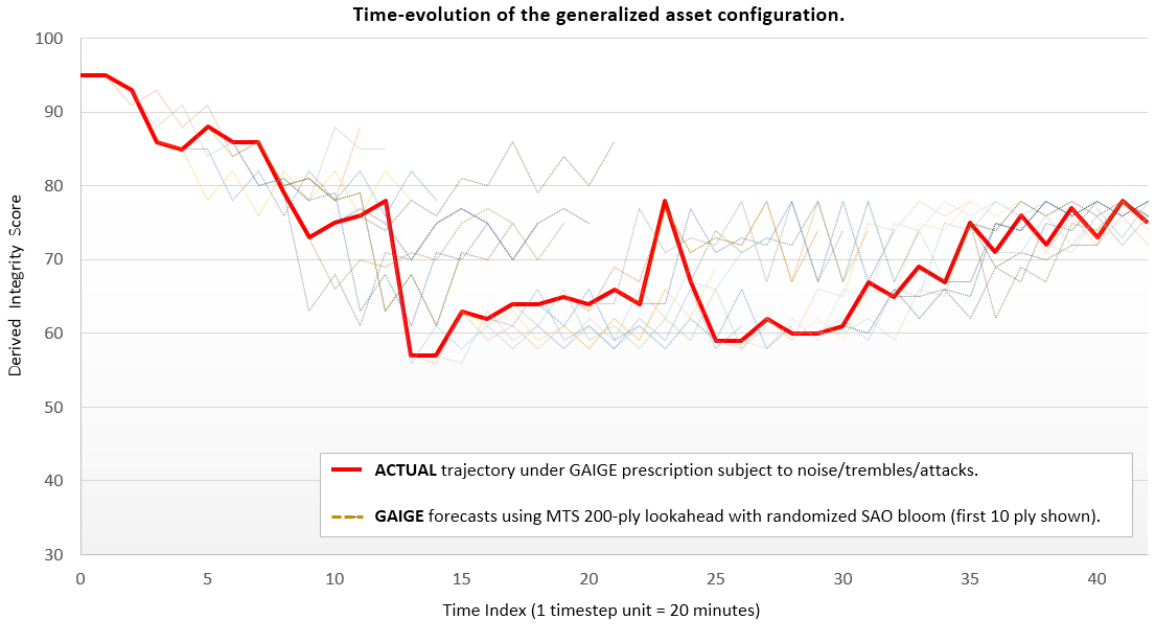
The performance of this generalized asset is calculated for several scenarios which manifest themselves as $N = 100$ significant reference configurations in a manner similar to the previous example. A set of $N^2 = 10{,}000$ baseline integrity scores are derived from long-run VNM-utility expectations and reconfiguration efforts computed using the NCD. Results are formulated into a succinct payoff array for input into the GAIGE. The UTM gathers and processes data asynchronously. Update frequencies range from sub one-second time intervals for dynamic positioning and sensing subsystems, to upwards of 7 days based on a lack of change-detection in certain physical components and ground facilities. As such, the update priorities for elements within the generalized asset may be adjusted based on change criticality, plausibility, or detection. This scheduling reduces the size and number of active array elements considered by the GAIGE at any one time-step. State-transition actions which recover the asset from immediate issues are evaluated before deliberating over less impending concerns. A prudent allocation of anytime resources also serves the defensive forecasting requirements, where the initial best-available plans are also the most secure (MTS), and deprecated opportunistically as optimization times permit (SAO-bandits).

Generalized asset operations are simulated for a horizon of 10 hours, corresponding to a session of normal UTM missions/activities subject to naïve discrete event arrivals and their ideal response/recovery. Naïve events are sampled using a variety of distributions, and model the ever-changing weather conditions, airspace traffic reports, sense and avoid trajectory deviations, signal strengths, ATM and cloud-based notifications.

To impress the need for game-playing agents in the face of adversarial dynamics, a series of deliberately planned remotely-executed minimax trajectory disruptions are introduced. These represent intelligent attacks which can be recovered from relatively quickly, yet are useful for examining fleet and airspace resilience. In this example, a GPS navigation glitch is considered, which triggers the failsafe and geofencing protocols. An electrical power disruption is considered, engaging the backup supplies and initiating a return-to-home procedure on all UAV/UAS. Finally, two cloud-based DDoS attacks are considered, resulting in temporary QoS adaptation. For simplicity, the class of 'catastrophic' and/or large-scale physical denial of the fleet or ground facilities is not considered. Such events would inevitably result in a near one-shot transition to the completely failed endgame state. In such a configuration the UTM activities are suspended indefinitely and no further analysis is required from an integrity planning perspective.

The planning horizon for the GAIGE is therefore $T = 36{,}000$ one-second time-steps simulating 10 hours of hostile environment persistence. During this time, the asset undergoes adaptive oscillations as changes to the UTM configuration are observed and innovated upon in the manner prescribed by the GAIGE. At each time-step, the GAIGE re-plans a sequence of actions and transitions the asset accordingly. This results in a series of planned vs. actual trajectories. Figure A2.10 illustrates the time-evolution of the asset under GAIGE recommended governance from a particularly safe starting configuration until the end of simulation.

Figure A2.10. Evolution of the UTM under governance from the GAIGE.



In the above plot, the horizontal x-axis indexes a time-step which averages 20 minutes of condition monitoring and integrity evaluations. The vertical y-axis indexes a derived integer integrity score scaled between 0 and 100. This represents the payoff level awarded at the end of a round of simulated gameplay. In this regard, the reference configurations have been ranked and scaled in order to illustrate how the integrity of a UTM would be governed. Correct UTM management seeks to continuously maximize the area under the curve(s) with respect to utility yields; the adversary attempting to do the opposite (minimize the area); and nature acting as a random noise signal (which can aid either party). This generalized asset evolves in accordance with the discrete events registered within the UTM. At each round, the GAIGE updates its forecast of what it believes to be an optimal sequence of state-transitions. This planning is conducted using a 200-ply lookahead for the

backwards induction of the MTS algorithm. Random amounts of SAO bandit iterations (nearline bloom) are also included. The faint dotted lines in Figure A2.10 illustrate the GAIGE-prescribed plans at the end of each round. For clarity only the first 10 state transitions are plotted. The bold red line indicates the time-history of noisy realizations of state-transitions, as prescribed by the GAIGE (updated at each round), but owing to incomplete state observations (noise), imperfect actions (trembles), and intelligent disruptions.

The operational time frame simulated in this example was subject to several stochastic disruptions and adversarial attacks. An interpretation of the results demonstrates that current versions of the GAIGE are overly conservative. The GAIGE often hedges against hostilities at every time step. This results in grim prognostics towards low-yield asset operations, a byproduct of minimax aspirations from continuously competing against the world. In practice this extreme prudence would (hopefully) become unnecessary through a combination of protective measures and the influential deterrents of policing and honest citizenry (i.e. human integrity). Realistically an adversary will only strike with bounded resources, placing an upper bound on the frequency, amplitude and phase of superimposed attack patterns. In game theory even the slightest resource disparity between attacker and defender can introduce unwinnable conditions for the poorer player and autopilot strategies for the richer. In this example, the offensive prowess of adversaries (in terms of their observational and degradation capabilities) were assumed to be in balance with the restorative capabilities of the UTM. The development of better adversarial threat models would allow the integrity governance of cyber-physical infrastructure to be less

pessimistically challenged. Unfortunately this process would be require expert input and result in a more domain-specific approach. As adversarial due diligence becomes more common place, these additional modelling details will be required before production-scaled assets are authorized to operate within real-world hostile environments.

The (actual) time-evolution of the asset given by its changing integrity scores can be seen as a series of reconfiguration expenditures. Figure A2.10 shows the GAIGE buffering against the long-run losses from an impending set of adversarial and stochastic disruptions. These are the expected trajectories in terms of reconfigurations, which are minimax worst case unless the bandits risk otherwise.

Figure A2.10 can also be interpreted in purely UTM game-playing terms. At the end of time-step 5, weather conditions force the asset into a lower integrity score. The GAIGE then reassesses its integrity plan from this new (somewhat unforeseen and unavoidable) condition. In time-step 6, the GAIGE actually considers a potential vulnerability to attack in round 11, and updates its integrity plan in defense of this credible future. The adversary senses this innovation and strikes with an unannounced DDoS attack during time-steps 8 and 9 (instead of 11). This forces the asset to drop into a hardened configuration with safer fleet posturing and higher alert levels. This caution results in a lower integrity score for several rounds. The DDoS releases after time-steps 12-13, at which point the GAIGE advises a cautionary relaxation out towards higher integrity scores. At time-step 23, the GAIGE finally decides it is time for a series of potent restoration and recalibration events, only to be immediately denied by the adversary who initiates GPS-jamming activities. The detection of these events forces a purposeful transition to a lower

integrity score – the UTM hardens in an attempt to pre-emptively buffer against further risk of airspace and IaaS integrity denial. At the low points around time-step 25, the GAIGE anticipates an opportunity by around time-step 30 to recover a higher-orbit minimax trajectory. This is in line with the long-run prescriptions made before the GPS-jamming. The asset eventually recovers to a reasonably maintainable integrity level by time-step 35 onwards. At this point the GAIGE is learning perturbation-robust strategies, and proactively fixing around an integrity level of 75 by the end of simulation.

The problem of sustaining safe and secure UTM operations within a low-altitude high-traffic airspace is a hypermodern challenge. The combination of UAV/UAS assets, payloads and requests, fleet readiness, mission planning, physical facilities, and supporting infrastructures must collectively synergize to deliver a networked infrastructure as a service (IaaS). The UTM paradigm is still in its infancy, but expected to mature within the next decade. Whatever the particular architecture or design, a UTM can be modelled as a time-evolving SoS, which can in turn be formulated as a generalized asset. The usual domain-oblivious fitness-based performance indicators from the theory of dependable networks, graph dynamics and topology continue to apply. The agnostic, similarity-based evaluations from the theory of information and computability continue to serve a purpose, indicating how much an asset must reconfigure itself in terms of model description to achieve a desired outcome.

Potential applications of generalized asset integrity games to engineering problems.

Formulating the problems of KVM facility management or UTM governance within the framework of generalized asset integrity games allows a compact representation of the information critical to imparting changes of global state. Both DSM and SoS defined assets have been shown to be readily integrated into this framework. This process could be demonstrated for other modelling institutions, such as .XML or .UML drawings, DFA, state-transition tables, and extended to arbitrary connectionist diagrams. This generality is extremely useful for rapidly (approximately) validating engineering proposals with minimal modelling and simulation overhead. In this way, several reference configurations for a generalized asset can be pre-processed to yield a succinct-form payoff structure which can be passed into the GAIGE for minimal-regret planning and analysis. As demonstrated by examples, the GAIGE is particularly adept at scoring the performance of complex cyber-physical assets, critical infrastructures, and future service platforms such as the KVN facility or UTM concept. This work establishes a framework for generalized asset performance, integrity games, and the GAIGE itself. The overall process is expected to remain flexible and nearly identical across domains. Many potential applications show promise in benefitting from this approach. Among those actively being investigated: geospatial watershed integrity, water-distribution integrity, the integrity of sensitive ecosystems, *aqua-* and *agri-* culture logistics management, and high-speed rail transport systems.