General Video Game Playing Escapes the No Free Lunch Theorem

Daniel Ashlock
Department of Mathematics
and Statistics
University of Guelph
Guelph, Ontario, Canada,
dashlock@uoguelph.ca

Diego Perez-Liebana
School of Computer Science
and Electronic Engineering,
Colchester, United Kingdom
University of Essex
dperez@essex.ac.uk

Amanda Saunders
Department of Mathematics
and Statistics
University of Guelph
Guelph, Ontario, Canada,
asaunder@uoguelph.ca

Abstract—**Popular topics in current research within the games community are general game playing and general video game playing. Both of these efforts seek to find relatively general purpose AI to play games. Within the optimization community we are approaching the 20th anniversary of the no free lunch theorem. In this paper we suggest reasons why a games version of a no free lunch result is probably not problematic. This is accomplished by noting that none of the "general" efforts are – or should be – actually general. Technology is proposed to exploit the lack of generality to permit more effective game playing AIs to be designed. A program for classifying games is outlined that consists of gathering performance data on many games for many algorithms and then using the resulting matrix of performance data to create a tree structured classification of the games. This classification is proposed as the basis for assigning games to appropriate algorithms within a more general framework. A novel algorithm that yields more stable tree-based classification is also proposed.**

I. Introduction

General game playing AIs seek to mimic the human ability to learn and then play, at least competently, any of a broad variety of games. The human analogy highlights the well-known problem that different people and different algorithms have games they are good at or bad at. This paper suggest a method that is the equivalent, in human terms, of assembling a team of specialty players that are entered – as a single participant – in a games tournament. This human team might have a chess grand master, an excellent go player, a high-earning professional poker player, a specialist in the analysis of mathematical games, and a video-game obsessed teenager with mad reflexes. When a game was presented to the team, the most likely team member would be put forward to represent the team in playing that game.

This vision hides a difficulty. Suppose the team was presented with the game *Mancala*: who plays? The team forgot to include a board-games nut and so the chess grand master is put forward as the best available choice (or is he?) and the team has an enhanced danger of losing the round of the tournament because of incomplete staffing. Implementing a general game playing AI with sub-experts in the form of more specialized algorithms needs a decent classification of all the types of games it might encounter, algorithms able to deal with each of these games, and an effective procedure for assigning games to algorithms.

In this paper we outline a taxonomic scheme that can simultaneously co-classify games and algorithms, based on performance data. This is the core technology for assembling a team of algorithms that is *complete* in the sense it can deal with the games it might expect to encounter as well as providing the information needed to assign games to algorithms. The paper also explores the notion that this assignment of games to algorithms may be hybrid; one algorithm might be natural for the opening portion of a game while another might be superior in the mid- or end-game.

When attempting to implement a general problem solver, the experience of the optimization community with the no free lunch theorems also suggests that overreach is not only possible but potentially inevitable. The potential for a no free lunch theorem for games is discussed and it is argued that current general game playing efforts fall well below the danger threshold for no free lunch effects; a case is made that the games community is not currently overreaching. A useful implication of the no free lunch theorem is that algorithms should be specialized to their problem domains and the proposal in this paper, in essence, runs with that idea. Placing well-specialized algorithms in a decision framework is a strategy for avoiding overreach and creates infrastructure for incremental improvement of game playing algorithms to a high level of generality.

II. Background

The *no free lunch* (NFL) theorem [1] states that, over the space of all optimization problems, the average performance of a given optimizing algorithm that does not re-sample points is equal to the average performance of all other such algorithms. The key idea behind the proof is that, in the space of all optimization problems, a correct decision for one problem is the wrong decision for another and, over the entire space, this all balances out. This theorem did the research world a service by stomping on claims of universal superiority for one or another algorithm.

There is a clear and helpful corollary to the NFL theorem: the effectiveness of an optimizer on a problem or restricted class of problems increases as the algorithm is specialized to the problem. This specialization of the algorithm often takes the form of incorporating special knowledge about the class of problems into the algorithm. Gradient search [2], for example, only works to optimize functions that *have* a gradient. It is specialized for differentiable functions.

There is a natural question that arises from the furor that occurred after the first publication of the NFL theorem: how could the research community have contained so many people that thought the claims that evolutionary algorithms were universal (”Swiss army algorithms”) were well supported? This belief in the extraordinary powers of a new approach is a repeatable historical phenomenon. Catastrophe theory [3] is another example of a ”savior theory” that would hand us the keys to the world – in the late 1970s. With evolutionary algorithms, the key was that they can solve almost any toy problem and they also cracked some very hard problems like VLSI layout [4]. They set the stage for assuming they could do everything.

Within the games research community something analogous to a universal optimizer has been proposed in the form of general game playing [5] and general video game playing (GVGP; [6]). The games community is following a much better path that the optimization community did in the early days of evolutionary algorithms: they are insisting on the demonstration of at least substantial generality by using multi-game contests to evaluate their general purpose algorithms. It is possible to prove a NFL theorem for some classes of mathematical games – a highly technical effort that will appear elsewhere – and it is easy to embed mathematical games in the space of all games. This means that a sort of NFL result applies to games (or will once the requisite effort to write the proof out and tighten it up has taken place). In this paper we want to make the case that this result will detract in only the most modest fashion from general AI research in games.

The starting point for this is to note that the NFL theorem didn’t shut down optimization research – not even optimization research that developed fairly general purpose algorithms. With its strong exhortation to specialize to your problem, the NFL theorem in fact *helped* optimization research quite a bit. It strongly motivated the valuable research into the impact of representation on search [7], [8]. One of the earliest examples of the impact of representation was in games research [9]. This means that as long as the games research community is building general purpose algorithms for a restricted set of games, they are not likely to run into NFL problems.

A. General is too General

Something that most computer science or math majors learn, or are at least exposed to, is a few facts about the nature of universal spaces that form the underpinning for not fearing no free lunch complications. The simplest instance of the useful viewpoint is the fact that almost all real numbers cannot be described. The number of descriptions, even algorithmic ones, are countably infinite while the number of real numbers is a higher order of infinity, an uncountably infinity [10]. The space of *all* real numbers is appallingly large and considering all of it, other than by aggregational mechanisms like those in calculus, is neither possible nor beneficial. In addition to being indescribable, these numbers are also inapplicable – they are the unconsidered packing form of Euclidean space.

A given instance of the NFL theorem for optimization averages over a space of all optimization problems of some sort. Almost all of these problems are random, in the sense that adjacent points contain little or no information about one another, unless the notion of adjacency is cooked to match the problem. Adjacency in the search space used by evolutionary computation is created by variation operators, like mutation and crossover. If these operators are generic then the mutual information of almost all pairs of nearby points is near zero in almost all optimization problems. That means even very general purpose algorithms – like generic evolutionary algorithms – are already designed incorporating the special knowledge that the problem they are operating on is one that someone has a reason to be interested in. This alone creates a filter for problems in which nearby points often have high mutual information. In this case “mutual information” means that the objective function value for nearby points to a point p is somewhat predictive of the objective function value of p.

What does this mean for general game playing or general video game playing? An AI that could do well on most games a human might enjoy is still operating in an incredibly small subspace of any abstract game space. Suppose we have a game with n possible moves. Scoring of any move considers the complete game history to that point – in other words every string of moves has its own score. Suppose we automatically generate instances of this game by filling in the scores, for all possible play histories up to some maximum length, with a normally distributed random variable. This gives us an uncountable infinite space or games: all but an insignificant subset of these games are of no interest to a human player. There is no pattern, no basis for learning more sophisticated than random sampling, and no reason to bother with such games. These horrible games, however, fill almost the entirety of the game space that contain chess, checkers, and go. The presence of an infinitude of games we will never care about protects us from NFL entanglements in general game AI research.

B. What can be done with this?

While we have only a sketch of an NFL theorem for games, it seems likely that one exists. If it does then the corollary that algorithms should be specialized to their problem also holds for games – but in addition to being a corollary of the NFL theorem this is also just good sense. Why then do we even want general game AI? One natural answer sounds a lot like the rational for climbing Everest – it is there. There are other, more pragmatic reasons. Procedural content generation (PCG) [11] is the algorithmic generation of game content. One type of PCG is the automatic generation of whole games [12]. If one were generating games via evolution, an objective function would be required. A general game playing AI – general at least for the space of games encoded by the representation in question – would be such an objective function. This general AI would also be useful for game level generation [13].

Anyone that plays games knows there are a lot of different flavors of games. The mental machinery needed to play chess is very different from the reflexes needed to play Galaga. As different as those two games are, both lack the social and political dimensions of Risk, the braggadocio needed for a game like Munchkin, or the ability to bluff and read tells that is at the core of skilled play in poker. During a discussion of general video game playing at a recent Dagstuhl conference, it was noted that the games in that year's General Video Game AI (GVGAI) Competition had a strong binary feature: either Monte-carlo tree search [14] was an excellent approach or it was hopeless. This observation suggests a visionary idea: use automatic taxonomy based on algorithm performance to classify games and then develop algorithms for the resulting classes of games. A prototype of this program for game classification has already been tested [15]. A collection of games and different variations of Monte-carlo tree search were juxtaposed and used to build classification trees of both the games and the algorithms. The proposal to use classification trees to segment the general game AI problem space is developed additionally in Section IV.

III. General Video Game AI

The GVGAI Framework is a benchmark that allows conducting research in Artificial General Intelligence via games. It has been used as a framework for the GVGAI [6] Competition since 2014, which is at the fourth edition at the time of this writing. The GVGAI framework would be a natural place to test game classification to improve general AI performance. This section briefly describes these benchmark and the competitions run, followed by an analysis of some of the most relevant controllers submitted to them[1].

A. Framework and Competitions

The GVGAI Framework is a Java port of the original py-vgdl engine developed by Tom Schaul [16], which defined a Video Game Description Language (VGDL) for 2-D classic and arcade real-time games (moves must be supplied within 40ms in the competition setting). GVGAI offers an interface for the implementation of planning and learning algorithms, as well as a collection of more than 140 single and two-player games. Agents have access to a forward model, which allows rolling the game forward to a possible next state by supplying an action.

Implemented controllers can also have access to the game state via a Java object, which accepts queries about the game status (winner, current time step, score), the player's state (position, orientation, health points, resources), the available actions and positions of the other sprites of the game. These sprites are provided by means of *observations*, which camouflage the sprite's type by using arbitrary integer IDs. Information is given about the nature of the sprites, categorized into classes: non-player characters (NPC), static, moving, resources and sprites created by the avatar. Game rules, sprite dynamics and victory requirements are not given to the agent.

Competition rankings are computed on the results obtained by all entries in a set of 10 unknown games. Each game has an independent ranking, sorted by victory rate, score and time steps needed to complete it. Points are provided to each entry according to the current F1 rules: 25 points for the first, 18 for the second, then 15, 12, 10, 8, 6, 4, 2 and 1 for the following positions, with 0 points awarded to the 11th position onwards. The winner of the competition is the controller with the higher sum of points across all games in the final set.

The GVGAI runs two tracks for planning algorithms: single and two-player track [17]. The latter was run for the first time in 2016, featuring 8 submissions (plus 5 sample controllers provided with the framework) in two different legs: IEEE World Congress on Computational Intelligence (WCCI-16) and Computational Intelligence and Games (CIG-16) in 2016. The former and original track has featured in 6 different editions, reaching more than a hundred total submissions: CIG-14, Genetic and Evolutionary Computation Conference (GECCO-15), CIG-15, Computer Science and Electronic Engineering Conference (CEEC-15), GECCO-16 and CIG-16.

B. Competition methods

Table I shows the results of the first edition of the single-player planning track, offering interesting insights about the type of controllers received. Adrien Couëtoux implemented OLETS (Open Loop Expectimax Tree Search; [6]), winner of the 1st edition of this track. OLETS is an Open-Loop tree approach inspired by Hierarchical Open-Loop Optimistic Planning (HOLOP [18]), using an exploration term for the value function and no Monte Carlo simulations.

As can be observed in Table I, the first half of the table is dominated by tree-based methods, mostly Monte Carlo Tree Search (MCTS; [14]) and variants thereof. On a more broader view, these entries belong to the category of *single* agents (i.e. they only use one algorithm), which is the most common type submitted to this edition of the competition, with only 2 entries using a combination of algorithms.

These techniques did not achieve excellent results this round, maybe with the exception of the 4th entry, *Shmokin*. This agent simply starts executing A* to navigate through the level, trying to find a goal to win the game, switching to MCTS if this approach fails. The other mixed approach, *T2Thompson*, provides 6 heuristics that try to achieve game play objectives (shooting enemies, collecting resources, moving towards doors) that use either A* or a steepest ascent hill-climber. The success of mixed approaches supports the notion that partitioning the game space and handing off to appropriate algorithms might work well. Finally, it's also worth highlighting the presence of Evolutionary Algorithms (EA) in the form of Rolling Horizon EA (RHEA; [19]), ranking in

[1]Note that GVGAI is only one of the possible benchmarks for General Video Game Playing. The discussions and insights presented in this paper are applicable to all of them, but GVGAI is shown here as an example due to its popularity and accessibility to the submitted controllers.

Rank	Username	G-1	G-2	G-3	G-4	G-5	G-6	G-7	G-8	G-9	G-10	Points	Victories	Approach
1	OLETS	**25**	0	**25**	0	**25**	10	15	**25**	**25**	8	158	256/500	Tree Search
2	JinJerry	18	6	18	**25**	15	6	18	18	12	12	148	216/500	Tree Search
3	SampleMCTS†	10	18	6	4	18	**25**	6	12	0	0	99	158/500	Tree Search
4	Shmokin	6	**25**	0	12	10	8	0	10	6	0	77	127/500	Tree Search & A*
5	Normal_MCTS	12	0	4	15	4	15	10	4	4	0	68	102/500	Tree Search
6	culim	2	12	8	1	8	4	8	6	10	2	61	124/500	Q-Learning
7	MMbot	15	0	1	2	12	12	2	15	0	0	59	130/500	Tree Search
8	TESTGAG	0	8	15	0	0	1	1	0	2	**25**	52	68/500	Evolutionary Algorithm
9	Yraid	0	6	10	0	0	0	12	0	15	6	49	93/500	Evolutionary Algorithm
10	T2Thompson	0	0	0	10	0	0	0	1	18	18	47	87/500	Hill Climber & A*
11	MnMCTS	8	8	0	0	1	18	4	8	0	0	47	109/500	Tree Search
12	SampleGA†	4	10	12	0	0	2	0	0	8	4	40	76/500	Evolutionary Algorithm
13	IdealStandard	1	6	0	0	6	0	**25**	0	0	1	39	134/500	A*
14	Random†	0	15	0	18	2	0	0	0	0	0	35	78/500	Random
15	Tichau	0	6	0	8	0	0	0	0	1	15	30	55/500	Only action *USE*
16	SampleOSLA†	0	6	0	0	0	0	0	0	1	10	17	51/500	Tree Search
17	levis501	0	0	2	6	0	0	0	2	1	0	11	50/500	Tree Search
18	LCU_14	0	4	0	0	0	0	0	0	0	0	4	54/500	Rule Based System

TABLE I
RESULTS OF THE 1st SINGLE-PLAYER GVGAI COMPETITION. THE GAMES ARE: G-1: ROGUELIKE; G-2: SURROUND; G-3: CATAPULTS; G-4: PLANTS; G-7: PLAQUE-ATTACK; G-6: JAWS; G-7: LABYRINTH; G-8: BOULDERCHASE; G-9: ESCAPE AND G-10: LEMMINGS. †DENOTES A SAMPLE AGENT.

Contest Leg	Entry	Type	Method
CIG-14	OLETS	Single	Tree Search
GECCO-15	YOLOBOT	Meta-heuristic	A*, MCTS, BFS
CIG-15	Return42	Meta-heuristic	A*, Random walks
CEEC-15	YBCriber	Hybrid	Tree Search
GECCO-16	YOLOBOT	Meta-heuristic	A*, MCTS, BFS
CIG-16	MaastCTS2	Single	Tree Search
WCCI-16 (2P)	ToVo2	Hybrid	Sarsa, UCT(λ)
CIG-16 (2P)	Number27	Hybrid	RHEA, MixMax

TABLE II
WINNERS OF ALL EDITIONS OF THE GVGAI PLANNING COMPETITION. 2P INDICATES 2-PLAYER TRACK. HYBRID DENOTES 2 OR MORE TECHNIQUES COMBINED IN A SINGLE ALGORITHM. META-HEURISTIC HAS A HIGH LEVEL DECISION MAKER TO DECIDES WHICH SUB-AGENT MUST PLAY.

the mid-table. RHEA is a technique that evolves sequence of actions in a short time budget in order to choose the next move, as the first action of the best plan found.

Of special relevance is the result obtained by *IdealStandard* (A*) in the game Labyrinth, where the objective is to find the exit. In this game, this agent plays optimally, better than any other technique in this set. This is a clear example of how an over specialized controller can do well in one game and perform poorly on the rest of this set. The other agents that include A* as one of their techniques don't score points in this game, which can be attributed to an incorrect partitioning of the game space, highlighting the importance of an accurate general classification algorithm to decide the technique to use.

Table II summarizes all winners of the planning track for the single and two-player versions of the contest [2]. The 2015 edition of the competition received multiple submissions (the highest so far, about 70 entries), and a proliferation of agents that combine multiple algorithms could be observed. The GECCO-15 leg, as well as the overall championship (determined by summing the points from the three legs), was won by the entry *YOLOBOT* [20], and it is a clear example of this type of entries. *YOLOBOT* starts playing by using a path finding algorithm to populate a list to the closest sprites of each type, while using the forward model to classify the game as stochastic or deterministic. In the former case, the game is played using MCTS; in the latter, Best First Search (BFS) is the algorithm of choice. The other two legs of this edition were won by two different entries: *Return42*, winner of CIG-15, starts determining the stochasticity of the game. A* is used as the main driving algorithm in case the game is stochastic, and random walks are used otherwise. Finally, *YBCriber* [21], winner of CEEC-15, combines reactive avoidance of hazards with Iterative Width (IW; [21]) in their tree search.

YOLOBOT repeated as a winner in the GECCO-16 leg of the single player planing competition, although a new entry, *MaastCTS2* [22], was able to rank first on the CIG-16 leg and become the overall champion. The authors of this controller proposed several enhancements to MCTS, combining it with other techniques. Firstly, they use a Breadth-First initialization with safety pre-pruning (based on IW) to reduce the number of nodes in the tree that counted with more loses. Additionally, the authors complemented MCTS with Progressive History [23] and N-Gram Selection Technique [24], in order to introduce a positive bias towards actions that performed well in earlier simulations.

The 2-Player GVGAI track ran for the first time in 2016, and it featured two different legs. The WCCI-16 one was won by the entry *ToVo2*, a combination of MCTS and Sarsa-UCT(λ). The CIG-16 leg was won by *Number27*, which employed a RHEA technique in combination with MixMax back-ups (as in [25]). Interestingly, the champion of the 2016 edition was an adaptation of *OLETS*, mentioned above, to this track.

[2]For the sake of space, only competition winners are described for editions after 2014. All rankings, framework versions and controllers are available in the competition website: www.gvgai.net

A common pattern for all tracks of the 2015 and 2016 competitions is that none of the submitted controllers is able to lead the rankings in more than 4 out of the 10 games of each final secret test. Between 5 and 7 controllers are able to lead in at least one of the games, and some of them are able to be the top agent in a single game even if they ranked after the 10^{th} position. This suggests that the efforts of combining multiple techniques into a single controller have not still reached a level of performance that dominates in a subset of games. Again, this may be an indication that cleverer ways of partitioning the game state, together with a better or more diverse selection of algorithms, can bring a significant boost in performance.

C. Other work on GVGAI agents

Researchers have also used the GVGAI framework during the last years as a testbed for general artificial intelligence without submitting to the competition. This section revises part of this work, as it is important to understand how other approaches try to tackle this problem. Most efforts are directed towards improving the most used single algorithms in the literature of GVGAI: MCTS and RHEA.

In the case of MCTS, an early attempt by Perez et al. [26] showed that a combination of this tree search technique with evolution and knowledge gathering was able to improve performance in most games of the first set of GVGAI games. However, this approach was still not able to perform better than the vanilla version in some of the games, and subsequent experimental results in other game sets did not provide extraordinary results. More recently, F. Frydenberg et al. [25] proposed several modifications to MCTS (such as MixMax backups, macro-actions, partial expansion and reverse penalties). M. de Waard et al. [27] introduced an enhancement entitled *Option MCTS*, which analyzes the effects of using macro-actions for achieving subgoals. Interestingly, both approaches improve the performance of the vanilla MCTS algorithm. However, this improvement can't be observed across the totality of games used in their experimental study: the performance in some games actually drops down, possibly because over specialization of the improvement introduced.

An algorithm receiving an increasing attention lately is RHEA, with multiple enhancements being proposed. Tuning look-ahead and population size parameters [28], initial seeding of the population [29] and hybridizations with tree search [30], [31] have been shown to again improve the performance of the vanilla RHEA, but fail at producing an improvement over the totality of games used on each study.

These results suggest that it seems to be relatively easy to find enhancements that improve performance in specific, single algorithm-based, techniques. The efforts of these researchers succeed on designing agents that provide a higher average of victory rates than the vanilla versions of the algorithms they intend to improve, but fail to provide a stronger single algorithm that is able to achieve a decent rate of victories in the games tested. Some games of these sets have not been solved once yet by any general agent!

It is reasonable to think that clustering games and applying different techniques in a common agent should provide a step up in performance across the games of a set. Some very recent studies have started to go in this direction. For instance, P. Bontrager et al. [32] analyze the strengths and weaknesses of current GVGAI algorithms, clustering the games by using Principal Component Analysis and Agglomerate Hierarchical Clustering (in fact, these clustering has been used later [28]–[30] to select games for experimental setups). The authors showed that it is possible to build a decision tree to select the algorithm to play with, although they claim that there's also a need for new algorithms in order to improve performance further. Similarly, A. Mendes et al. [33] use J48 and Support Vector Machines to classify 41 known games of the GVGAI corpus and select the most appropriate algorithm to play. The authors are able to provide a meta-heuristic general agent that improves the performance of the algorithms is composed of in isolation. However, they also state that a better selection of features would be required in order to increase the gap with the best single agents, as the most appropriate controller was not always selected by the J48 decision tree.

IV. Stable Classification of Games

The goal of creating a stable family tree of games serves the end of picking diverse sets of games for future competitions. This permits the contest designers to avoid picking games that disproportionately favor one game or another and which provide the broadest possible test of the AIs submitted to the contest.

The prototype study on classifying games [15] used the UPGMA (Unweighted Pair Group Method with Arithmetic mean) hierarchical clustering method to construct family trees of both games and MCTS variants. UPGMA is a clustering method commonly used to transform distance data into a tree. It received attention in [34], and a good description may be found in [35]. It is especially reliable if the distances have a uniform meaning. The classification effort proposed in this paper would provide win/loss or goal achieved/not achieved data and so maintain the desirable uniform meaning.

Given a collection of taxa and distance d_{ij} between taxa i and j, the method first links the two taxa x and y that are least distant. The taxa x and y are merged into a new unit z. For all taxa i other than x and y, a new distance d_{iz} is computed as the average of d_{ix} and d_{iy}, and it is noted that the new taxon z really represents the average of two original taxa. Henceforth, x and y are ignored, and the procedure is repeated to find the next pair of taxa that are least distant. When two taxa u and v are combined into a new taxon w, the new distance d_{iw} is the average of d_{iu} and d_{iv}, weighted according to the number of original taxa in u and v respectively; w contains all the original taxa in both u and v. The procedure ends when the last two taxa are merged.

In order to apply the UPGMA method to games and MCTS variants, we must somehow establish distances between pairs of games and pairs of MCTS variants. In this case the number of victories either against a fixed opponent algorithm (for two player games) or scores above a standard (for one player games) was used to score each of a variety of MCTS variants on a collection of games. The resulting data object is a matrix indexed by the games and the MCTS variants. Treating the rows (scores an MCTS variant got on all the games) or columns (scores the different MCTS variants obtained on a game) as points in Euclidean space permitted the computation of Euclidean distances between pairs of games or points.

This technique has already been used to create family trees of optimization problems [36] and has been seen to be a viable approach to classifying games. There are some problems. Recall that one of the motivations for this approach was that some of the problems in GVGAI were not MCTS-friendly. This means that the performance of a broader variety of algorithms on games is probably necessary. A natural source of algorithms is to mine the GVGAI competitions and adopt algorithms that did well there. The algorithms mentioned in Sections III-B and III-C form a good starting point for the selection of algorithms to drive the classification effort.

A. *Stability of the classification trees*

A second problem is that the UPGMA method, used directly on simple distance data, has been shown to be unstable [37]. The instability demonstrated has the following form. If one point is removed from the data set, and the algorithm is re-run, then the resulting tree can be very different from what would result if the leaf of the tree corresponding to that data point were simply trimmed from the tree. This sort of instability is not acceptable in a tree that is used to classify games and then select the algorithm used to play them. The UPGMA algorithm is widely used in biology and, when the data arise from a common descent process, like biological evolution, the stability of the resulting trees is greater. Performance of different algorithms on a collection of games is unlikely to have this stabilizing property and so a better method of building trees for game classification is needed.

The stability measure uses a simple metric on trees to compute the distance between the tree obtained by trimming a leaf from the one obtained by removing a data point and rebuilding the tree with whatever algorithm is in use. The average distance between such trees, with the average being over all single points that could be removed, is an *instability* measure, in that larger numbers indicate higher degrees of instability. A new clustering algorithm called *bubble clustering* has been found that transforms distance data into trees in more stable fashion. Preliminary results from a bioinformatics-motivated experiment are given in Figure 1. Bubble clustering is compared to the *hclust* package in the statistical platform R [38]. Over a variety of data sets, all forms of bubble clustering tested exhibit higher stability than all forms of hclust tested. The different forms of hclust vary the method of determining the distance between groups of already clustered points.

Bubble clustering operates as follows. The algorithm initializes a matrix of connection strengths between all pairs of point with zero. It repeatedly generates spheres with a radius selected uniformly at random within the diameter of

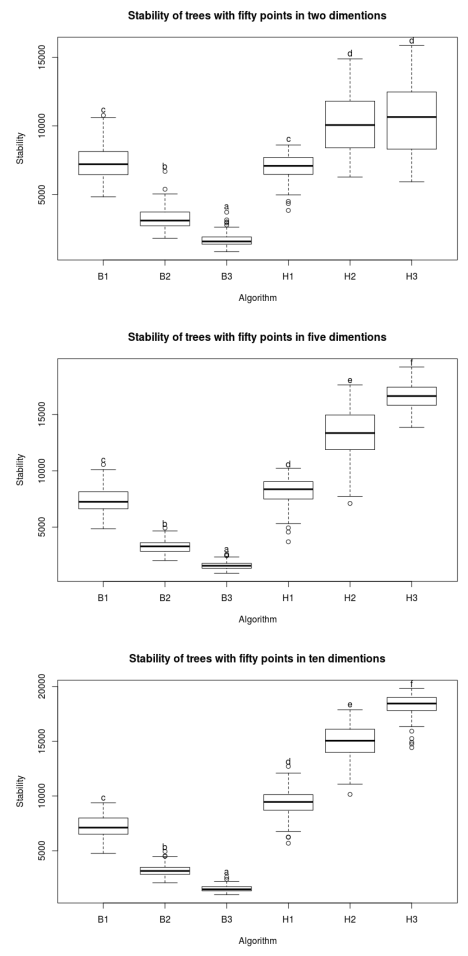

Fig. 1. Shown are relative stability for trees on 100 randomly generated data sets with n=50 data points each with 2, 5, or 10 coordinates distributed uniformly at random in the interval [-5,5]. Six collections of trees are compared produced with different methods; bubble clustering with 10n bubbles (B1), bubble clustering with 100n bubbles (B2), bubble clustering with 1000n bubbles (B3), hclust() using the complete linkage method (H1), hclust() using the average method (H2), hclust() using the centroid method (H3). Boxplots that share a letter are not statistically significantly different while those that have different letters are statistically significantly different

the data space and centered on data points (bubbles). Each time two points are both in such a sphere, their connection strength is incremented by the reciprocal of the number of points in the bubble. A bubble with a small number of points in it indicates points that a more closely coupled, which is why the reciprocal weight for co-membership in a bubble is used. Point density may vary irregularly: the bubble sampling process automatically creates a linkage that compensates for such irregularity. In addition, the number of bubbles used represents a control in a cost/accuracy trade-off. The UPGMA algorithm is modified to deal with linkage data only by taking the largest linkages instead of smallest distances.

The vision presented in this paper for enabling improved AI performance in general game playing is to use bubble clustering on a matrix of algorithm:game success data to generate family trees of games used to assign different sub-algorithms to play those games within a GVGP framework. The algorithms participating in the generation of the success data are still an open area for further work. Bubble clustering with the reciprocal measure can provide the game classification trees at the heart of the effort, but there is also room to tinker with the weighting scheme to enhance the tree stability.

V. Conclusions and Future Work

This paper argues that the implications of NFL theorems are not a problem for general game AIs because the most general domain of interest, within the games arena, is still well below the threshold of completeness where it would suffer from attempting to contradict NFL theorems. Given this, the paper goes on to propose a classification scheme that would permit the partitioning of the set of games of potential interest for assignment to appropriate algorithms. The GVGAI framework is reviewed and a preliminary list of algorithms is supplied based on that review. The GVGAI game set proposed as a test set of games for the effort. Details and possible improvements to the game classification scheme are presented along with preliminary results touching on their stability – and extant problem with these classification techniques.

A. Generalizing bubble clustering

Bubble clustering was proposed for the game classification effort because it solve a potentially problematic stability problem. Of particular value is its ability to operate smoothly on data that is distributed in a complex or irregular fashion. Since the classification of games into types is an off-line activity there is also the ability to perform classification with a very large number of bubbles and so achieve extremely stable classification: in the preliminary work increasing the number of bubbles sampled does improve stability. Given this, bubble clustering is currently a somewhat arbitrary choice that may benefit from additional examination.

Bubble clustering is an example of a more general technique called *associator clustering* (AC). Each sampled bubble is an *associator*. This name arises from the fact that being in a bubble together associates two points. The first example of AC is *k-means multi-clustering* [39], [40]. This method was similar to bubble clustering but used the clusters arising from multiple different executions of the k-means clustering algorithm, with different initial conditions and numbers of clusters, to associate points. Multi k-means clustering showed an exceptional robustness to irregular distribution of data. In general, any reasonable method of telling two points are similar could be used as an associator and AC could potentially use multiple types of associators in the same classification.

In general, an associator is just a way of choosing a collection of points to be associated, and any associator must also have a quality measure that says *how much* the association of points that appear together in an associator should be strengthened. This strengthening factor is the *quality* of the associator. Being together in a randomly sampled bubble and appearing together in one of the clusters of a k-means clustering are the tested examples of associators. If one were clustering game players, it might be possible to create associators derived from their strategic choices. Clustering documents could use common rare words or phrases as associators. No matter what the choice of associator, the modified UPGMA algorithm that joins strongest linkages first can be applied to the resulting matrix of connection strengths. The choice of effective associators for clustering games with algorithm performance data is a central part of future work on useful automatic game classification.

B. Agents, Heuristics and Objectives

The ability to correctly classify games in a stable way opens interesting lines of research, aimed at obtaining good performance in multiple games by means of combining different techniques and methods. Some initial steps have already been taken in GVGAI in this spirit. They are either simple human and ad-hoc classifications [6], [41] (such as differentiating between deterministic versus stochastic games, or the presence or absence of certain type of elements in them), or they are based on more involved clustering techniques which still need further development to perform satisfactorily in a competition setting with unknown games [32], [33].

Deeper research in meta-heuristic agents is one of the next logical steps in General Video Game Playing. These systems would count on a high level decision system that determines which algorithm, heuristic and/or objective must be pursued next within a range of possible choices. This selection mechanism would naturally be strongly influenced by an accurate and real-time classification of games.

It is worth highlighting that this classification does not necessarily need to be only addressed from a game versus game point of view. Alternatively or in parallel, such approach could also consist of analyzing game states rather than complete games. If different algorithms perform better in distinct games, it is not too alien to think that different algorithms can also be used at specific moments during the same game. Especially in complex games, the dynamics, objectives and even rules can change at several points during play. A clear example is Pac-Man, which can be seen as (at least) two sub-games in one: either the player is escaping the ghosts while eating pills, or it is actively chasing them after consuming a power pellet. More complex games (such as real-time or turn-based strategy games, like Civilization) naturally evolve through certain phases where exploration, resource gathering and combat tactics take turns as the primary game objective.

Therefore, an interesting line of future work is the investigation of how to combine multiple general heuristics, where each one of them tries to tackle a different need during game play. For example, C. Guerrero et al. [42] provide an initial study on different heuristics in some games of the GVGAI framework, which try to maximize exploration of the level, discover rules and dynamics or simply maximize the score. How to combine and pick the appropriate heuristic for the given game (or moment within the game) is a problem to be explored in the near future.

One possibility is to build a Multi-Objective approach [43], where each goal is represented by an heuristic and a high level decision mechanism determines their weights dynamically. Another possible alternative could be ensemble systems, where several algorithms (or heuristics) determine the next move to make at each game tick. Each one of these *sub-agents* has a voice, listened by a central decision mechanism. The vote of each agent can be provided in different ways (favorite action, ranking of moves, with or without confidence intervals) and the decision maker can determine how to weight each voice attending to the type of game (or game state) provided by a classification system like the one suggested in this paper.

References

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Trans. Evol. Comp*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[2] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[3] E. C. Zeeman, *Catastrophe theory: Selected papers, 19721977*. Addison-Wesley, 1977.

[4] J. Lienig, "A parallel genetic algorithm for performance-driven vlsi routing," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 29–39, Apr 1997.

[5] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the aaai competition," *AI Magazine*, pp. 62–72, 2005.

[6] D. Perez Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C.-U. Lim, and T. Thompson, "The 2014 General Video Game Playing Competition," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 229–243, 2016.

[7] D. Ashlock and J. Gilbert, "A discrete representation for real optimization with unique search properties," in *Proc. of the IEEE Symposium on the Foundations of Computational Intelligence*, 2014, pp. 54–61.

[8] D. Ashlock, J. Schonfeld, L. Barlow, and C. Lee, "Test problems and representations for graph evolution," in *Proc. of the IEEE Symposium on the Foundations of Computational Intelligence*, 2014, pp. 38–45.

[9] N. L. D. Ashlock, E.Y. Kim, "Understanding representational sensitivity in the iterated prisoner's dilemma with fingerprints," *Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, vol. 4, no. 36, pp. 464–475, 2006.

[10] W. Rudin, *Principles of Mathematical Analysis*. McGraw Hill, 1976.

[11] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6024, pp. 141–150.

[12] T. Mahlmann, J. Togelius, and G. N. Yannakakis, *Towards Procedural Strategy Game Generation: Evolving Complementary Unit Types*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 93–102.

[13] X. Neufeld, S. Mostaghim, and D. Perez Liebana, "Procedural level generation with answer set programming for general video game playing," in *Computer Science and Electronic Engineering Conference (CEEC), 2015 7th*. IEEE, 2015, pp. 207–212.

[14] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4:1, pp. 1–43, 2012.

[15] C. McGuinness, "Classification of monte carlo tree search variants," in *Proc. of the 2016 IEEE Congress on Evolutionary Computation*. Piscataway, N.J.: IEEE Press, 2016, pp. 357–363.

[16] T. Schaul, "An extensible description language for video games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 4, pp. 325–331, 2014.

[17] R. D. Gaina, D. P. Liebana, and S. M. Lucas, "General Video Game for 2 Players: Framework and Competition," in *Proc. of the IEEE Computer Science and Electronic Engineering Conference*, 2016.

[18] A. Weinstein and M. L. Littman, "Bandit-based planning and learning in continuous-action markov decision processes." in *ICAPS*, 2012.

[19] D. P. Liebana, S. Samothrakis, S. M. Lucas, and P. Rolfshagen, "Rolling Horizon Evolution versus Tree Search for Navigation in Single-Player Real-Time Games," in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, 2013, pp. 351–358.

[20] T. Joppen, M. Moneke, N. Schröder, C. Wirth, and J. Fürnkranz, "Informed Hybrid Game Tree Search," Knowledge Engineering Group, Technische Universität Darmstadt, Tech. Rep., 2016.

[21] T. Geffner and H. Geffner, "Width-based planning for general video-game playing," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.

[22] D. J. N. J. Soemers, C. F. Sironi, T. Schuster, and M. H. M. Winands, "Enhancements for Real-Time Monte-Carlo Tree Search in General Video Game Playing," in *IEEE Conference on Computational Intelligence and Games*, 2016.

[23] J. A. M. Nijssen and M. H. M. Winands, "Enhancements for Multi-player Monte-Carlo Tree Search," in *International Conference on Computers and Games*. Springer, 2010, pp. 238–249.

[24] M. J. W. Tak, M. H. M. Winands, and Y. Bjornsson, "N-grams and the Last-good-reply Policy Applied in General Game Playing," *IEEE Trans. on Computational Intelligence and AI in games*, vol. 4:2, pp. 73–83, 2012.

[25] F. Frydenberg, K. R. Andersen, S. Risi, and J. Togelius, "Investigating mcts modifications in general video game playing," in *IEEE Conference on Computational Intelligence and Games*, 2015, pp. 107–113.

[26] D. Perez, S. Samothrakis, and S. Lucas, "Knowledge-based fast evolutionary mcts for general video game playing," in *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 2014, pp. 1–8.

[27] M. d. Waard, D. M. Roijers, and S. C. Bakkes, "Monte carlo tree search with options for general video game playing," in *IEEE Conference on Computational Intelligence and Games*, 2016, pp. 47–54.

[28] R. D. Gaina, J. Liu, S. M. Lucas, and D. Perez Liebana, *Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing*. Cham: Springer International Publishing, 2017, pp. 418–434.

[29] R. D. Gaina, S. M. Lucas, and D. P. Liebana, "Population Seeding Techniques for Rolling Horizon Evolution in General Video Game Playing," in *Proc. of the Congress on Evolutionary Computation*, 2017.

[30] ——, "Rolling Horizon Evolution Enhancements in General Video Game Playing," in *Proc. Computational Intelligence and Games*, 2017.

[31] H. Horn, V. Volz, D. Perez Liebana, and M. Preuss, "Mcts/ea hybrid gvgai players and game difficulty estimation," in *IEEE Conference on Computational Intelligence in Games*, 2013, pp. 1–8.

[32] P. Bontrager, A. Khalifa, A. Mendes, and J. Togelius, "Matching games and algorithms for general video game playing," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[33] A. Mendes, A. Nealen, and J. Togelius, "Hyperheuristic general video game playing," *Proc. of Computational Intelligence and Games*, 2016.

[34] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy; the Principles and Practice of Numerical Classification*. W.H. Freeman, 1973.

[35] D. Swofford, G. Olsen, P. Waddell, and D.M.Hillis, "Phylogenetic inference," in *Molecular Systematics, second edition*, D. Hillis, C. Moritz, and B. Mable, Eds. Sunderland, MA.: Sinauer, 1996, pp. 407–514.

[36] K. M. Bryden, D. A. Ashlock, S. Corns, and S. J. Willson, "Graph based evolutionary algorithms," *IEEE Transaction on Evolutionary Computation*, vol. 10, pp. 550–567, 2006.

[37] D. Ashlock, T. von Konigslow, and J. Schonfeld, "Breaking a hierarchical clustering algorithm with an evolutionary algorithm," in *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 19, 2009, pp. 197–204.

[38] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014.

[39] D. A. Ashlock, E. Kim, and L. Guo, "Multi-clustering: avoiding the natural shape of underlying metrics," in *Smart Engineering System Design: Neural Networks, Evolutionary Programming, and Artificial Life*, C. H. D. et al., Ed., vol. 15. ASME Press, 2005, pp. 453–461.

[40] E. Kim, S. Kim, D. Ashlock, and D. Nam, "Multi-k: accurate classification of microarray subtypes using ensemble k-means clustering," *BMC Bioinformatics*, vol. 10, no. 260, pp. 1–12, 2009.

[41] D. Perez Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul, "General video game ai: Competition, challenges and opportunities," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[42] C. Guerrero-Romero, A. P. Louis, and D. Perez Liebana, "Beyond playing to win: Diversifying heuriscits for gvgai," in *Computational Intelligence in Games , IEEE Conference on*. IEEE, 2017.

[43] D. Perez Liebana, S. Mostaghim, and S. M. Lucas, "Multi-objective tree search approaches for general video game playing," in *IEEE Congress on Evolutionary Computation*, 2016, pp. 624–631.