Automated Extraction of Problem Structure

Anthony Bucci[1], Jordan B. Pollack[1], and Edwin de Jong[2]

[1] DEMO Lab, Brandeis University, Waltham MA 02454, USA
{abucci*,pollack}@cs.brandeis.edu
http://demo.cs.brandeis.edu/
[2] Decision Support Systems Group, Universiteit Utrecht
dejong@cs.uu.nl

Abstract. Most problems studied in artificial intelligence possess some form of structure, but a precise way to define such structure is so far lacking. We investigate how the notion of problem structure can be made precise, and propose a formal definition of problem structure. The definition is applicable to problems in which the quality of candidate solutions is evaluated by means of a series of tests. This specifies a wide range of problems: tests can be examples in classification, test sequences for a sorting network, or opponents for board games. Based on our definition of problem structure, we provide an automatic procedure for problem structure extraction, and results of proof-of-concept experiments. The definition of problem structure assigns a precise meaning to the notion of the underlying objectives of a problem, a concept which has been used to explain how one can evaluate individuals in a coevolutionary setting. The ability to analyze and represent problem structure may yield new insight into existing problems, and benefit the design of algorithms for learning and search.

1 Introduction

Most problems studied in artificial intelligence possess some form of *structure*. Taking chess as an example, different players can be compared with regard to strategy, tactics, and other aspects of their play. Thus, there are several *dimensions* along which the behavior of players can be compared. Precise knowledge of such problem structure would benefit both our insight into problems and the design of algorithms. It has so far been unclear however how such dimensions might be defined precisely, and how informative dimensions might be determined. We investigate how these notions can be made precise, and propose a formal definition of problem structure. Based on this, we describe an automatic mechanism to explore and represent problem structure.

We consider problems where the performance of a candidate solution, or *candidate* for short, is determined by the outcomes of *tests*. We consider problems where the performance of a candidate is determined by the outcomes of *tests*. For example, a classifier may be evaluated on the errors it makes in classifying test examples; an evolved checkers player may be evaluated on its scores against some set of opponents; and a sorting network can be evaluated on its ability to sort test sequences. This class of *test-based problems* defines a broad range of problems.

* Corresponding author

The structure of a problem consists of a space and a mapping of the candidates and tests into this space. The structure space is such that the outcome of a candidate on any test can be uniquely determined given only the coordinates of the candidate. Furthermore, a structure space is of minimum dimension given this constraint. The structure space captures essential information about a problem in an efficient manner.

Since the quality of a candidate is determined by its outcomes on tests, tests may be viewed as objectives in the sense of Multi-Objective Optimization (MOO; see [1], e.g.). In this view, the structure space may be seen as a projection of the tests onto a smaller set of dimensions or objectives, such that a one-to-one mapping exists between the candidate objective vectors for the two spaces. This resulting set of objectives will typically be unknown at first, but is fundamental in the sense that it represents all relevant relations between candidates and tests in an optimally compact way. The axes spanning the structure space may therefore be called the *underlying objectives* of a problem. The term *underlying objectives* was first introduced in work on coevolution [2], where it was observed that the tests in a coevolutionary algorithm tended to identify the objectives that governed the evaluation of learners. A simpler version of the same idea was presented in the form of the *ideal test set* and *test dimension* of [3].

In the realm of machine learning of game strategies, Arthur Samuel notes that terms for the evaluation polynomial of his checkers player should ideally be generated by the learning program itself. Samuel mentions the idea of an orthogonal set of terms to be used in this evaluation polynomial [4]. Along similar lines, Susan Epstein has argued that to be optimized, a game player should experience a variety of opponents with varying skill levels [5]. We feel the present work offers a precise way to discuss concepts like "orthogonal set of terms," and to clarify which variety of opponents a game player requires to be optimized.

The structure of this paper is as follows. In section 2, we present a mathematical definition of a *coordinate system* for a test-based problem. We give an example from geometry to motivate our choice of definitions, and explore some of the properties and implications of the definitions. In section 3, we present a polynomial-time dimension-extraction algorithm which, given a problem, constructs a coordinate system for it. The coordinate system need not be minimal, but it is guaranteed to *span* the problem in a certain sense and satisfy an independence criterion. Finally, in section 4, we present some experimental validation of the formal and algorithmic ideas. We run the dimension-extraction algorithm on the population in a coevolutionary simulation run on a game with known dimension; we see that the algorithm correctly deduces the dimension or overestimates it, depending on the game.

2 Geometrical Problem Structure

Let $p : S \times T \to 2$ be any function, where S and T are *finite* sets[3] and 2 is the partially ordered set $0 < 1$. Here the set S is interpreted as the set of candidate solutions; T is the set of tests or test cases, and 2 is the outcome of applying a test to a candidate. The function p encodes the interaction between a test and a candidate; intuitively, we

[3] The finiteness assumption is not strictly necessary, but it greatly eases the exposition.

can think of it as a payoff function. Such functions appear often in optimization and learning problems. For example:

Example 1 (Function approximation). Let $f : T \to \mathbb{R}$ be a target function defined over a set T, and let S be a set of model functions $T \to \mathbb{R}$. The problem is to find a function in S that matches f as closely as possible. Notice that if $h \in S$ is some candidate function, then a point $t \in T$ can serve as a test of h. For example, we can define $p : S \times T \to 2$ by $p(h,t) = \delta(f(t), h(t))$, δ the Kronecker delta function.

Example 2 (Chess). Let $S = T = \{\text{deterministic chess-playing strategies}\}$. For any two strategies $s_1, s_2 \in S$, define $p(s_1, s_2) = 1$ if s_1 beats s_2, 0 otherwise. Then p is of form $S \times T \to 2$.

Example 3 (Multi-objective Optimization). Let S be a set of candidate solutions, and for each $0 \leq i \leq n-1$, let $f_i : S \to 2$ be an objective. The optimization task is to find (an approximation of) the non-dominated front of these n objectives. Let $T = \{f_0, \ldots, f_{n-1}\}$, and define $p : S \times T \to 2$ by $p(s, f_i) = f_i(s)$ for any $s \in S$, $f_i \in T$.

In this section, we will use such a function p to define an abstract coordinate system on the set S. This coordinate system will give a precise meaning to the notion of *underlying objectives*. In all of our examples, S will be finite. At first glance it is not obvious what a coordinate system on a finite set might look like. One of the major contributions in this paper is forwarding an idea about how we might do that.

2.1 Motivation

As a motivating example for the definitions to follow, let us consider the 2-dimensional Euclidean space E_2, namely the set $\mathbb{R} \times \mathbb{R}$ with its canonical coordinate system and pointwise order. Write $x : E_2 \to \mathbb{R}$ and $y : E_2 \to \mathbb{R}$ for the two coordinate axes; for any point in E_2, the function x returns the point's x coordinate and the function y returns its y coordinate. $p \leq q$ holds for two points $p, q \in E_2$ exactly when $x(p) \leq x(q)$ and $y(p) \leq y(q)$ both hold. Now consider these two families of subsets of E_2. For each $r, s \in \mathbb{R}$:

$$X_r = \{p \in E_2 | x(p) \geq r\} \tag{1}$$

$$Y_s = \{p \in E_2 | y(p) \geq s\} \tag{2}$$

Geometrically, X_r is the half plane consisting of the vertical line $x = r$ and all points to the right of it. Y_s is the half plane consisting of the horizontal line $y = s$ and all points above it. Figure 1 illustrates these two families.

For brevity, let us write $\mathcal{X}$ for the family $(X_r)_{r \in \mathbb{R}}$ and $\mathcal{Y}$ for $(Y_s)_{s \in \mathbb{R}}$. In other words, an element of the family $\mathcal{X}$ is one of the sets X_r, and an element of $\mathcal{Y}$ is one of the sets Y_s. We would like to show that $\mathcal{X}$ and $\mathcal{Y}$ can act as stand-ins for the coordinate functions x and y. In particular, $\mathcal{X}$ and $\mathcal{Y}$ satisfy the following three properties:

1. **Linearity**: For all $r, s \in \mathbb{R}$, $X_r \subset X_s$ or $X_s \subset X_r$. Furthermore, $X_r = X_s$ implies $r = s$. Similarly, $Y_r \subset Y_s$ or $Y_s \subset Y_r$ and $Y_r = Y_s$ implies $r = s$.

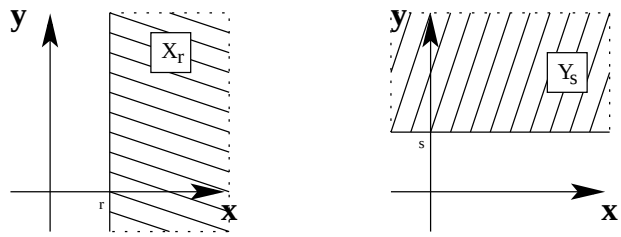

Fig. 1. Typical members of the families $\mathcal{X}$ and $\mathcal{Y}$; see text for details.

2. **Independence**: There exist $r, s \in \mathbb{R}$ such that X_r and Y_s are incomparable; that is, neither is a subset of the other.
3. **Spanning**: For all $p \in E_2$ define $f(p) = \inf_r\{p \in X_r\}$ and $g(p) = \inf_r\{p \in Y_r\}$. Then f and g are well-defined functions from E_2 to $\mathbb{R}$, and $p \leq q$ in E_2 exactly when $f(p) \leq f(q)$ and $g(p) \leq g(q)$ both hold.[4]

Property 1 states that the family $\mathcal{X}$ is linearly ordered by $\subset$; $\mathcal{Y}$ is as well. Property 2 states that the two families $\mathcal{X}$ and $\mathcal{Y}$ give independent information about E_2. Finally, property 3 states that $\mathcal{X}$ and $\mathcal{Y}$ can together be used to recover the order on E_2; this is the sense in which they span the space.

Properties 1-3 make no reference to the special qualities of E_2. In fact, they require only the family $\mathcal{X} \cup \mathcal{Y}$ of subsets of E_2. Since we can define families of subsets in any set, particularly finite ones, these three properties are a suitable abstract notion of coordinate system which can be fruitfully extended to finite sets.

2.2 Terminology

We will require some terminology from discrete math, which we review next.

Recall that a *preorder* on a set S is a reflexive, transitive, binary relation on S. Unless we state otherwise, the symbol $\leq$ will be used for preorders; we will also write $s_1 \geq s_2$ to mean $s_2 \leq s_1$. The *reflexive* property means that for any $s \in S$, $s \leq s$ holds. The *transitivity* property means that for any three $s_1, s_2, s_3 \in S$, $s_1 \leq s_2$ and $s_2 \leq s_3$ together imply $s_1 \leq s_3$. A preorder is similar to a partial order. *Partial orders* are also *antisymmetric*, meaning: whenever $s_1 \leq s_2$ and $s_2 \leq s_1$ both hold, it must be that $s_1 = s_2$. In a preorder, antisymmetry may fail: both these relations may hold, but it may still be that $s_1 \neq s_2$. Preorders commonly arise from functions into sets that are already ordered. For instance, if $f : S \to \mathbb{R}$, then we can compare two $s_1, s_2 \in S$ using f. Namely, there is a preorder $\leq_f$ on S defined: $s_1 \leq_f s_2$ exactly when $f(s_1) \leq f(s_2)$. Antisymmetry of $\leq_f$ is then equivalent to f being injective.

A *linear order* is a partial order which satisfies the *trichotomy law*: for any two s_1, s_2, either $s_1 \leq s_2$, $s_2 \leq s_1$, or $s_1 = s_2$ must hold. A partial order need not satisfy this property. In other words, a partial order can have *incomparable* elements, meaning two $s_1, s_2 \in S$ such that neither is $\leq$ the other. The canonical example of a partial order

[4] In this example $f = x$ and $g = y$. This property is the definition of the order on E_2 in disguise.

is the power set of a set. The power set is partially ordered by inclusion: given any two subsets of a set, it need not be true that one is a subset of the other. We refer the reader to a discrete mathematics text such as [6] for more details and discussion of these concepts.

2.3 Coordinate Systems

Before defining a coordinate system on S, we will need some preliminary definitions to simplify notation.

Let $p : S \times T \to 2$ be any function on the finite sets S and T. For each $t \in T$, define the set $V_t = \{s \in S | p(s,t) = 0\}$. The set V_t is therefore the subset of all candidates which do poorly against the test t. We can use these sets to define a preordering on T. Namely, define $t_1 \leq t_2$ if $V_{t_1} \subset V_{t_2}$. Observe that in general this will be a preorder: there is no guarantee that $V_{t_1} = V_{t_2}$ implies $t_1 = t_2$. However, reflexivity and transitivity hold. It will be convenient to define two formal elements $t_{-\infty}$ and t_∞ and extend the order $\leq$ from T to $\overline{T} = T \cup \{t_{-\infty}, t_\infty\}$ by defining $t_{-\infty} < t < t_\infty$ for all $t \in T$. That is, $t_{-\infty}$ and t_∞ are respectively the minimum and maximum of $\leq$ extended to $\overline{T}$. For any subset $U \subset T$, we will write $\overline{U}$ for $U \cup \{t_{-\infty}, t_\infty\}$. Under the mapping $t \mapsto V_t$, $t_{-\infty}$ corresponds to $\emptyset$ and t_∞ corresponds to S. This formal device will make certain arguments easier. In particular, for any $s \in S$ and any $U \subset T$, there will always be $t_1, t_2 \in \overline{U}$ such that $p(s,t_1) = 0$ and $p(s,t_2) = 1$. $\overline{U}$ will always have a minimum and a maximum.

[3] argues that a function like p induces a natural ordering on the set S which is related to the idea of Pareto dominance in multi-objective optimization. We argue that this ordering captures important information about how two candidate solutions in S compare to one another in an optimization problem defined by p. Let us write $\preceq$ for this ordering; then for any $s_1, s_2 \in S$, $s_1 \preceq s_2$ holds if $p(s_1,t) \leq p(s_2,t)$ for all $t \in T$. For instance, in the multi-objective optimization example, $s_1 \preceq s_2$ exactly when $f_i(s_1) \leq f_i(s_2)$ for all objectives $f_i \in T$. In the multi-objective optimization literature the latter condition means s_2 *covers* s_1.

With these preliminaries, we can define a coordinate system on S. The sets V_t will play a role analogous to the X_r and Y_r above. The ordering $\preceq$ on S is the one we wish to span.

Definition 4 (Coordinate System). *A family $\mathcal{T} = (T_i)_{i \in I}$ of subsets of T is a* coordinate system for S *(with* axes T_i*) if it satisfies the following two properties:*

1. ***Linearity****: Each T_i is linearly ordered by $\leq$; in other words, for $t_1, t_2 \in T_i$, either $V_{t_1} \subset V_{t_2}$ or $V_{t_2} \subset V_{t_1}$.*
2. ***Spanning****: For each $i \in I$, define $x_i : S \to \overline{T_i}$ by: $x_i(s) = \min_{t \in \overline{T_i}}\{s \in V_t\} = \min_{t \in \overline{T_i}}\{p(s,t) = 0\}$, where the minimum is taken with respect to the linear ordering on $\overline{T_i}$. Then, for all $s_1, s_2 \in S$, $s_1 \preceq s_2$ if and only if $\forall i \in I, x_i(s_1) \leq x_i(s_2)$.*

The definition of $x_i(s)$ as the minimal $t \in \overline{T_i}$ such that $p(s,t) = 0$ implies that $p(s,t) = 1$ for all $t < x_i(s)$. The requirement that T_i be linearly ordered guarantees that if $s \in V_{t_1}$ and $t_1 < t_2$, then $s \in V_{t_2}$ as well. It follows that if $t > x_i(s)$, then $s \in V_t$; i.e.,

$p(s,t) = 0$. Consequently, if $T_i = \{t_0 < t_1 < \cdots < t_{k_i}\}$ is an axis and $x_i(s) = t_j$, we can picture s's placement on the axis like this:

$$\begin{array}{lccccccccccccc} p(s,t) & 1 & & 1 & & \dots & & 1 & & 0 & & \dots & & 0 \\ T_i & t_0 & \longrightarrow & t_1 & \longrightarrow & \dots & \longrightarrow & t_{j-1} & \longrightarrow & t_j & \longrightarrow & \dots & \longrightarrow & t_{k_i} \end{array}$$

This picture is the crux of what we mean by "axis." For any candidate s, the above picture holds. s's coordinate on a particular axis is exactly that place where it begins to fail against the tests of the axis. Intuitively, we can think of an axis as representing a dimension of skill at the task, while s's coordinate represents how advanced it is in that skill.

We have not assumed independence because we would like to consider coordinate systems that might have dependent axes. Much as in the theory of vector spaces, we can show that a coordinate system of minimal size must be independent. However, as we will see shortly, in this discrete case there is more than one notion of independence which we must consider.

Definition 5 (Dimension). *The* dimension *of S, written* $\dim(S)$*, is the minimum of* $|\mathcal{T}|$ *taken over all coordinate systems $\mathcal{T}$ for S.*

Remark 6. Because S and T are finite, $\dim(S)$ will be well-defined if we can show at least one coordinate system for S exists. We will do so in section 2.4.

In the meantime, let us assume coordinate systems exist and explore some of their properties.

Definition 7 (Weak Independence). *A coordinate system $\mathcal{T}$ for S is* weakly independent *if, for all $T_i, T_j \in \mathcal{T}$, there exist $t \in T_i$, $u \in T_j$ such that V_t and V_u are incomparable, meaning neither is a subset of the other.*

Then we have a theorem reminiscent of linear algebra:

Theorem 8. *Let $\mathcal{T}$ be a coordinate system for S such that* $|\mathcal{T}| = \dim(S)$*. Then $\mathcal{T}$ is weakly independent.*

Sketch of Proof. Suppose $\mathcal{T}$ is not weakly independent. Then there are two axes, call them T_i and T_j, such that all tests in T_i are comparable to all tests in T_j. Consequently, we can create a new coordinate system $\mathcal{T}'$ as follows. First, $\mathcal{T}'$ has all the axes as $\mathcal{T}$ except T_i and T_j. Create a new axis T_k by forming $T_i \cup T_j$ and then arbitrarily removing duplicates (which are t, u such that $V_t = V_u$). The resulting T_k is then linearly ordered, and so can be an axis. Put T_k in $\mathcal{T}'$. Then, $\mathcal{T}'$ is also a coordinate system for S, but $|\mathcal{T}'|$ is one less than $|\mathcal{T}|$, contradicting the fact that $\mathcal{T}$ was minimal. Thus, $\mathcal{T}$ must be independent. □

2.4 Existence of a Coordinate System

In this section we prove that any function $p : S \times T \to 2$ with S and T finite gives rise to a coordinate system on S. Simply put, the set of all chains in T satisfies definition 4. Once we can show one such coordinate system exists, we know that a minimal one exists and there is a reasonable notion of the dimension of S.

Definition 9. *A* chain *in T is a subset $C \subset T$ such that, for all $t_1, t_2 \in C$, either $V_{t_1} \subset V_{t_2}$ or $V_{t_2} \subset V_{t_1}$; further, $V_{t_1} = V_{t_2}$ implies $t_1 = t_2$.*

Let $\mathcal{C}$ be the set of all chains in T. Then:

Theorem 10. *$\mathcal{C}$ is a coordinate system for S.*

Proof. Write $\mathcal{C} = (C_i)_{i \in I}$. By definition, each C_i is linear. Thus we need only check that this family spans $\preceq$.

($\Rightarrow$) Assume $s_1 \preceq s_2$. We want to show $\forall i, x_i(s_1) \leq x_i(s_2)$. Consider a $C_i \in \mathcal{C}$ and imagine $C_i = \{t_0 < t_1 < \cdots < t_{k_i}\}$. If $x_i(s_1) \not\leq x_i(s_2)$, i.e. $x_i(s_1) > x_i(s_2)$, we must have the following situation:

$$\begin{array}{lcccccccccc} p(s_1,t) & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 0 & \dots & 0 \\ C_i & t_0 \to & t_1 \to & \dots \to & t_{j_2-1} \to & t_{j_2} \to & \dots \to & t_{j_1-1} \to & t_{j_1} \to & \dots \to & t_{k_i} \end{array}$$

$$\begin{array}{lcccccccccc} p(s_2,t) & 1 & 1 & \dots & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ C_i & t_0 \to & t_1 \to & \dots \to & t_{j_2-1} \to & t_{j_2} \to & \dots \to & t_{j_1-1} \to & t_{j_1} \to & \dots \to & t_{k_i} \end{array}$$

where $x_i(s_1) = t_{j_1}$ and $x_i(s_2) = t_{j_2}$. However, then $p(s_1, t_{i_2}) > p(s_2, t_{i_2})$, which contradicts the assumption that $s_1 \preceq s_2$. Thus, $x_i(s_1) \leq x_i(s_2)$. This argument holds for any C_i and any $s_1, s_2 \in S$; therefore we have our result.

($\Leftarrow$) Assume $\forall i \in I, x_i(s_1) \leq x_i(s_2)$. We have the following for each $C_i \in \mathcal{C}$:

$$\begin{array}{lcccccccccc} p(s_1,t) & 1 & 1 & \dots & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ C_i & t_0 \to & t_1 \to & \dots \to & t_{j_1-1} \to & t_{j_1} \to & \dots \to & t_{j_2-1} \to & t_{j_2} \to & \dots \to & t_{k_i} \end{array}$$

$$\begin{array}{lcccccccccc} p(s_2,t) & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 0 & \dots & 0 \\ C_i & t_0 \to & t_1 \to & \dots \to & t_{j_1-1} \to & t_{j_1} \to & \dots \to & t_{j_2-1} \to & t_{j_2} \to & \dots \to & t_{k_i} \end{array}$$

where $x_i(s_1) = t_{j_1}$ and $x_i(s_2) = t_{j_2}$. It is clear from the diagram that for all $t \in C_i, p(s_1,t) \leq p(s_2,t)$. This fact holds for any C_i. That is, we have for all $t \in \bigcup_{i \in I} C_i, p(s_1,t) \leq p(s_2,t)$. However, $\bigcup_{i \in I} C_i = T$, meaning we have $s_1 \preceq s_2$.

Combining the above two implications, we have shown that $s_1 \preceq s_2$ if and only if $\forall i \in I, x_i(s_1) \leq x_i(s_2)$, for any $s_1, s_2 \in S$. Hence, $\mathcal{C}$ is a coordinate system for S, as we set out to show. $\square$

3 Dimension-Extraction Algorithm

In this section we give a polynomial-time algorithm that finds a weakly-independent coordinate system for a set of candidates. The algorithm accepts as input a set of candidates, a set of tests, and the outcome of each candidate for each test. Given this input, the goal is to construct a coordinate system such that (i) the position of a candidate

in the constructed space uniquely identifies which tests it passes and fails, and (ii) the dimension of this coordinate system is minimal. Since an efficient optimal algorithm is not available, an algorithm will be presented that satisfies (i) but uses heuristics to minimize the dimension, and is therefore not guaranteed to satisfy (ii).

The main idea of the algorithm is as follows. We start out with an empty coordinate system, containing no axes. Next, tests are placed in the coordinate system one by one, constructing new axes where necessary. A new axis is required when no axis is present yet, or when a test is *inconsistent* with tests on all existing axes. Two tests t, u are inconsistent if V_t and V_u are incomparable. We now discuss two aspects of coordinate systems that inform our algorithm.

In a valid coordinate system, the tests on each axis are ordered by strictness; any test must at least fail the candidates failed by its predecessors on the axis. This knowledge informs our heuristic for choosing the order in which to consider tests: the first step of the algorithm is to sort the tests based on the number of candidates they fail.

A second aspect of coordinate systems is that a test whose set of failed candidates is the union of the sets of candidates failed by two other tests can be viewed as the combination of those tests. For example, if a test A on the first axis fails candidates 1 and 3 and a test B on the second axis fails candidates 2 and 5, then a test located at position (A,B) in the coordinate system must fail the union of the candidate sets: candidates 1,2,3, and 5. Since such a composite test provides no additional information about which tests a candidate will pass of fail, it can be safely discarded. Therefore, the second step of the algorithm is to remove any tests that can be written as the combination of two other tests.

Once the tests have been sorted and superfluous tests removed, the procedure is straightforward; tests are processed in order and are either placed on an existing axis if possible, or on a new axis if necessary. The pseudocode of the algorithm is as follows:

4 Experiments

As a validation of the ideas presented in the previous sections, we applied our dimension-extraction algorithm to the populations of a coevolutionary simulation. Here we report the procedure we used and the results of the experiments.

Naturally, the question arises whether this algorithm will really extract useful coordinate systems from a problem. This question clearly bears much further empirical study. Here we are content to address the simpler question of whether the dimension extraction algorithm will give meaningful answers for particular problems in which we know what the underlying objectives are.

4.1 Method

The algorithm of fig. 2 was applied to the populations in a variant of the Population Pareto Hill Climber (P-PHC) algorithm presented in [7]. Briefly, a population of candidates and a separate population of tests is maintained by the algorithm. At each time step, the tests are treated as objectives that the candidates are trying to maximize. Each

```
Input:
List candidates, tests
boolean play(cand, test)
boolean consistentWith(test1, test2)
Test and(test1, test2)

Output:
Tree dimensions

Algorithm:
sort tests by number of fails
for each test1, test2, test3 ∈ tests (with test1 ≠ test2 ≠ test3)
  if test3 = and(test1, test2)
    remove test3 from tests
  end
end

for each test ∈ tests
  for each leaf ∈ dimensions
    if consistentWith(test, leaf)
      add test as child to leaf
    end
    if test was not added to a leaf
      add test as new leaf to root of dimensions
    end
  end
end
```

Fig. 2. Algorithm for coordinate system construction. The algorithm accepts sets of candidates and tests and their outcomes, and constructs a coordinate system that reflects the structure of the problem. Axes in this coordinate system consist of tests, and the location of a candidate in this induced space uniquely identifies which tests it will fail or pass.

candidate is given a single offspring, and the parent is replaced if the offspring does at least as well as the parent on each test.

Tests are incented to find distinctions between candidates. If a and b are two candidates, a test t makes a distinction between them $t(a) \neq t(b)$. Each test is given one offspring; an offspring replaces its parent if it makes a distinction the parent does not make. It is possible for an offspring to lose distinctions which the parent also makes; we are not concerned with this possibility in this algorithm. Except for this variation in test selection, all other algorithm details are the same as those reported in [7].

Two numbers games were used as test problems [8]. The first domain was the COMPARE-ON-ONE game presented in [2]. In this game, candidates and tests are both n-tuples of numbers. c and t are compared on the single coordinate where t is maximal. c "wins" the game if it is larger than t on that coordinate. This game has been shown to induce a pathology known as "focusing" or "overspecialization;" in conventional coevolutionary algorithms; see [7] or [2] for details.

The second domain was the TRANSITIVE game. Again, candidates and tests are n-tuples of numbers. This time, when a candidate c interacts with a test t, c wins if it is at least as large as t on all dimensions.

Observe that the coevolutionary algorithm does not have access to the fact that individuals are tuples of numbers. The games are given as black boxes to the P-PHC algorithm and it must make best use of this win/loss information. Consequently, when we run our dimension-extraction algorithm on the P-PHC populations, we are hoping to see the algorithm discover the number n which is the true dimension of the game.

We used the following procedure to estimate the number of dimensions. 10 independent copies of P-PHC were run for 2,000 time steps. At each time step, the estimated number of dimensions in the current population was output according to the dimension-extraction algorithm. This value was averaged across the 10 runs to obtain a single "average run." Then the following statistics were calculated across all 2,000 time steps: the 10th and 90th percentiles; the upper and lower quantiles; and the median. The number of true dimensions of the underlying problem was varied from 1 to 16 and statistics were gathered for each number of dimensions.

4.2 Results

Our results are presented in figure 3. These figures are box plots of the estimated number of dimensions versus the true number of dimensions. The boxes span the lower and upper quartiles of the dimension estimates; the whiskers give the 10th and 90th percentiles. The plus marks the median of the dimension estimates. The dotted line gives the expected answer.

The figure on the left gives the results for COMPARE-ON-ONE. There is good agreement between the estimated value of the number of dimensions and theoretical value for dimension ranging from 1 to 16. Further, the variance in the estimates is generally quite small.

The figure on the right gives the results for TRANSITIVE. In this case the algorithm consistently overestimates the number of dimensions of the problem. There is a larger amount of variance in the estimate as well when compared with COMPARE-ON-ONE. We only display up to 10 dimensions in the figure, enough to see the trend.

5 Conclusions

A notion of *problem structure* with application to a broad class of problems in artificial intelligence, including learning and search, has been proposed. Problem structure here takes the form of a *coordinate system* whose axes consist of tests, and knowledge of the position of a test uniquely specifies the behavior of that test.

The structure of a problem is an intrinsic property. Thus, any existing problems for which candidates are evaluated using tests must have an associated coordinate system of the kind defined in this paper. For most problems, the question of what the underlying objectives are is new, and it is given a precise meaning by the definition of problem structure presented here. The definition, and the preliminary algorithm for extracting

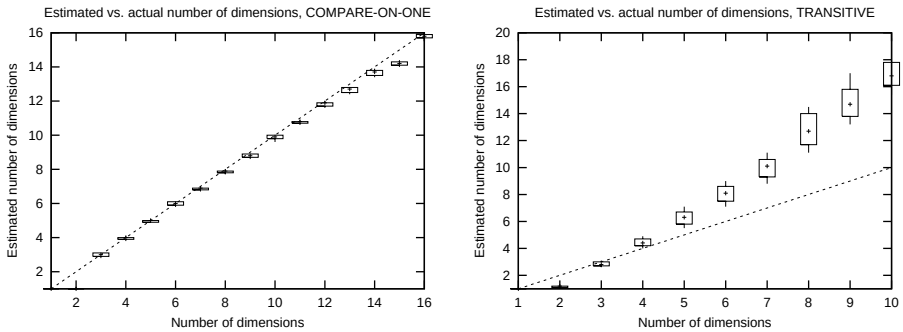

Fig. 3. Estimated number of dimensions in two numbers games, applying the algorithm in fig. 2 to the populations of a coevolutionary algorithm; see text for details. The left figure is the estimate for the COMPARE-ON-ONE game; note the tight correspondence with the theoretical number of dimensions. On the right is the estimate for the TRANSITIVE game; here the algorithm consistently overestimates.

problem structure, may therefore yield new insight into existing problems. While computationally challenging, this permits asking intriguing questions such as: what is the dimension of chess, and what are the underlying dimensions of chess?

The formal definition of problem structure that has been presented directly suggests ways of extracting problem structure automatically. A preliminary algorithm for coordinate system construction has been provided, and demonstrated on example problems. It is our hope that the notion of problem structure that has been proposed may incite the study of problem structure as a general property of problems; if efficient algorithms for problem structure extraction can be identified, it may become possible to better understand existing problems of interest by the algorithmic analysis of their structure, thereby providing new insight into existing problems in an automatic manner.

References

1. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Computation **3** (1995) 1–16
2. De Jong, E.D., Pollack, J.B.: Ideal evaluation from coevolution. Evolutionary Computation **12** (2004)
3. Bucci, A., Pollack, J.B.: A mathematical framework for the study of coevolution. In De Jong, K., Poli, R., Rowe, J., eds.: FOGA 7: Proceedings of the Foundations of Genetic Algorithms Workshop, San Francisco, CA, Morgan Kaufmann Publishers (2003) 221–235
4. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM Journal of Research and Development **3** (1959) 210–229 Reprinted in E. A. Feigenbaum and J. Feldman (Eds.) 1963, *Computers and Thought*, McGraw-Hill, New York.
5. Epstein, S.L.: Toward an ideal trainer. Machine Learning **15** (1994) 251–277
6. Scheinerman, E.R.: Mathematics: A Discrete Introduction. 1st edn. Brooks/Cole, Pacific Grove, CA (2000)

7. Bucci, A., Pollack, J.B.: Focusing versus intransitivity: Geometrical aspects of coevolution. In Erick Cantú-Paz et al., ed.: Genetic and Evolutionary Computation - GECCO 2003. Volume 2723 of Lecture Notes in Computer Science., Springer (2003) 250–261
8. Watson, R., Pollack, J.B.: Coevolutionary dynamics in a minimal substrate. In L. Spector et al., ed.: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco, CA, Morgan Kaufmann Publishers (2001)