Objective Set Compression

Test-Based Problems and Multi-Objective Optimization

Edwin D. de Jong[1] and Anthony Bucci[2]

[1] Institute of Information and Computing Sciences
Utrecht University
PO Box 80.089
3508 TB Utrecht, The Netherlands
dejong@cs.uu.nl

[2] DEMO Laboratory
Michtom School of Computer Science
Brandeis University
415 South St.
Waltham, MA 02454
abucci@cs.brandeis.edu

Summary. We consider a class of optimization problems wherein the quality of candidate solutions is estimated by their performance on a number of tests. Classifier induction, function regression, and certain types of reinforcement learning, including problems often attacked with coevolutionary algorithms, can all be seen as members of this class. Traditional approaches to such *test-based problems* use a single objective function that aggregates the scores obtained on the tests. Recent work, by contrast, argues that useful finer-grained distinctions between candidate solutions are obtained when each test is treated as a separate objective, and that algorithms employing such multi-objective comparisons show favorable behavior relative to those which do not. Unfortunately, the number of tests can be very large. Since it is well-known that high-dimensional multi-objective optimization problems are difficult to handle in practice, the question arises whether the multi-objective treatment of test-based problems is feasible. To begin addressing this question, we examine a method for reducing the number of dimensions without sacrificing the favorable properties of the multi-objective approach. Our method, which is a form of dimension extraction, finds *underlying objectives* implicit in test-based problems. Essentially, the method proceeds by placing the tests along the minimal number of coordinate axes that still preserve ordering information among the candidate solutions. Application of the method to the strategy set for several instances of the game of Nim suggest the technique has significant practical benefits: a type of compression of the set of objectives is observed in all tested instances. Surprisingly, we also find that the information contained in the arrangement of tests on the coordinate axes reveals important information about the structure of the underlying problem.

1 Introduction

Certain problem domains encountered in machine learning and computational intelligence applications involve an evaluation of candidate solutions that is derived from a set of *tests*. The outcomes a candidate receives on these tests are integrated into a scalar or vector which reflects different aspects of the quality of the individual, and is used to make decisions about keeping, discarding, or modifying that candidate. These domains are called *test-based problems* [De Jong and Pollack, 2004].

Before introducing the ideas that follow it will be useful to carry forward several illustrative examples of test-based problems:

- **Classifier Induction**: We are given a set of labeled data points and asked to produce a model, a neural network for example, that classifies them as well as possible. We fix a network topology and consider the task as a search through M, the space of possible weights for that topology. In other words, we seek a particular set of weights $m^* \in M$ which minimizes some classification error over the given data set. Each data point can be thought of as a test which an $m \in M$ either passes (classifies correctly) or fails (classifies incorrectly).[3] The error measure is an integration of this test information into a final evaluation of m.
- **Function Regression**: We are given a set of coordinates (x_i, y_i) representing the inputs and outputs of an unknown function and are tasked with finding a function that produces those pairs. We fix a space F of functions (for instance, genetic programs) and construe the task as a search through F for an f^* which minimizes a measure like RMS error. As with the classifier induction example, we can view each pair (x_i, y_i) as a test of a candidate function $f \in F$. f's outcome on test x_i is its error $|f(x_i) - y_i|$. These individual errors are then integrated over all pairs to form an evaluation of f, e.g. its RMS error.
- **Learning Games of Strategy**: We aim to learn a competent strategy for an instance of the game of Nim. Let P_1 be the set of first-player strategies and P_2 the set of second-player strategies. In order to see how good a particular strategy $r \in P_1$ is, we play it against a variety of second-player opponents $s \in P_2$. Each opponent s can be thought of as a test of r: r plays s with the outcome being a win or loss for r. These outcomes across many P_2 players can then be integrated into an evaluation of r, for instance its worst-case outcome.
- **Coevolutionary Algorithms**: Roughly speaking, coevolutionary algorithms search through instances of two or more distinct roles, utilizing individuals playing one role to evaluate individuals playing the other. A seminal example is Hillis' coevolution of sorting networks (one role) against sets of unsorted lists (a second role which Hillis called *parasites*)

[3] In practice we could further differentiate between false positive and false negative outcomes, but for the sake of the example we are simplifying matters.

[Hillis, 1990]. Each possible sorting network is a candidate solution whose sorting abilities can be partially tested by a parasite. To put it differently, a sorting network's evaluation is derived from its outcomes against a number of parasites acting as tests.

The connections between coevolution and multi-objective optimization are worth exploring in more detail. Recent work examining these connections has produced a theory which suggests that any test-based problem can be viewed as a multi-objective optimization problem. In this chapter, we aim to explore some of the outcomes of this point of view. In the remainder of this section we will explore the intellectual source of these ideas and then give an overview of the rest of the chapter.

It is worth pointing out that the method described here can be viewed as performing dimension *extraction* in the sense of [Bucci et al., 2004]. The notion of dimension reduction has recently been applied in other work in Evolutionary Multi-Objective Optimization; see, for instance, [Deb and Saxena, 2006] and [Saxena and Deb, 2007]. Also see the discussion of feature selection and feature extraction in the chapter by Brockhoff et al. in this volume.

1.1 Coevolution, Test-Based Problems and MOO

Traditionally, coevolutionary algorithms have integrated outcomes against multiple tests into a single fitness value, often by averaging or maximizing over all values.[4] That is, a candidate solution interacts with several test individuals and is then given a fitness that is an average or the maximum of its results against these tests (for a discussion on treating interactions in Cooperative Coevolution as tests, see [Bucci and Pollack, 2005]).

Both [Ficici and Pollack, 2001] and [Noble and Watson, 2001] argue that a finer-grained comparison can be made by viewing each test as its own objective. Rather than averaging or maximizing over all outcomes, the idea is to treat each outcome as a separate component of a vector of outcomes. Then, to compare two candidate solutions, the same Pareto dominance or Pareto covering relations utilized in multi-objective optimization are employed.

The process of transforming single-objective problems into multi-objective problems by separating the different criteria contributing to the quality of individuals has been named multi-objectivization [Knowles et al., 2001]. The application of this idea within coevolution is called *Pareto coevolution*. In Pareto-coevolution, each test in the population is treated as if it were an objective in a massive multi-objective optimization problem.[5] For instance,

[4] Simple fitness proportional coevolutionary algorithms typically use a (weighted) average of outcomes as fitness. Cooperative Coevolutionary algorithms sometimes use the maximum outcome as fitness value [Potter and Jong, 2000].

[5] A critical difference from evolutionary multi-objective optimization being that in coevolution not all objectives are in hand in advance, but rather are discovered during search.

if there were a population of 100 parasites in the sorting network example, a Pareto coevolutionary approach might evaluate a sorting network with a 100-dimensional vector of numbers, each number an outcome against a different parasite. Initial approaches to Pareto coevolution bought finer-grained comparison information at the cost of large outcome vectors like this; increasing the dimensionality of the objective space generally complicates the search problem. In other words, while Pareto coevolution has advantages, for instance its mitigation of cycling dynamics, large outcome vectors introduce new problems; see also the chapter by Ficici in this volume. As a result, there has been a drive to reduce the size of these vectors without losing too much of what was gained by using them in the first place.

Along these lines, [De Jong and Pollack, 2004] presents empirical results suggesting that a Pareto coevolutionary algorithm could find what were dubbed the *underlying objectives* of a problem. These are hypothetical objectives that determine the performance of candidate solutions without the need to test candidates against all possible tests. [De Jong and Pollack, 2004] applies a two-population Pareto coevolutionary algorithm, DELPHI, to instances of a class of abstract test games. Figs 13 and 15 of that work suggest that evaluator individuals[6] evolve in a way which tracks the underlying objectives of the problem domain. The results suggest that the algorithm is sensitive to the presence of underlying objectives even though it is not given explicit information about those objective. [Bucci and Pollack, 2003] makes a similar observation, also empirical though using a different algorithm; Fig. 5 of that work suggests a similar sensitivity to underlying objectives. In both cases, clusters of individuals, rather than single individuals, move along or collect around the objectives of the problem domain. The problem domains considered, namely numbers games [Watson and Pollack, 2001], were designed to have a known and controllable number of objectives, but the algorithms used in these two studies did not rely on that fact. The work therefore raises the question of whether underlying objectives exist in all problem domains, and whether search learning algorithms can discover them.

A partial answer to this question is found in the notion of *coordinate system* [Bucci et al., 2004]. Coordinate systems, which were defined for a class of test-based problems,[7] can be viewed as a formalization of the empirically-observed underlying objectives of [De Jong and Pollack, 2004]. To elaborate, a coordinate system consists of several *axes*. Each axis is a list of tests ordered in such a way that any candidate can be placed somewhere in the list. A candidate's placement is such that it passes all tests before that spot and fails all tests after it. For this reason, an axis can be viewed as measuring some aspect of a candidate's performance: a candidate that places high on an axis is "better than" one which sits lower in the sense that it passes more tests (more

[6] What we are here called *tests*.

[7] Specifically, problems with a finite number of candidates and a finite number of binary-outcome tests.

of the tests present in the axis, not more tests of the problem as a whole). Formally, an axis corresponds to a numerical function over the candidates, in other words to an objective function. It can be proven [Bucci et al., 2004] that every problem of the considered class possesses at least one coordinate system, meaning it has a decomposition into a set of axes. In short, every such problem has some set of objective functions associated with it, one for each axis in a coordinate system for the problem.

Besides defining coordinate systems formally, [Bucci et al., 2004] gives an algorithm that finds a coordinate system for a problem domain in polynomial time. The algorithm, though fast, is not guaranteed to produce the smallest-possible coordinate system for the problem. Finite domains must have a minimal coordinate system, but in general even finite domains can have distinct coordinate systems of different sizes. The algorithm is not coevolutionary per se, as it examines the outcomes of tests on candidates. It is therefore applicable to the entire class of test-based problems.

1.2 Chapter Overview

To summarize, recent theoretical work on coevolutionary algorithms has elucidated a theory of underlying objectives for test-based problems which establishes a conceptual link between multi-objective optimization and coevolution. We can now view coevolution as a form of multi-objective optimization in which not all objectives are explicitly given *a priori*, but are nevertheless present theoretically and can be extracted. Although this theory originated in coevolutionary algorithms research, it is focused on problem structure and indifferent to which search algorithm is used. Coordinate systems can be extracted from any test-based problem, which includes classifier induction, function regression, or game strategy learning problems.

However, there are two questions left unaddressed by this story. First, that *minimal* coordinate systems exist as theoretical objects does not guarantee they can be extracted algorithmically; previous work gave an algorithm which could find *some* coordinate system, but not necessarily a minimal one. Secondly, even if they could be found, there is no guarantee the extracted coordinate systems are meaningful, i.e. that their structure relates to characteristic features of the problem. With these questions left open, it is possible coordinate systems are simply mathematical curiosities that have limited relevance in practice. The aim of understanding underlying objectives and establishing a bridge between test-based problems and multi-objective optimization would not be met.

This chapter will focus on these two questions by developing and applying an exact coordinate system extraction algorithm to small instances of the game of Nim. The exact algorithm is guaranteed to identify a minimum-dimensional coordinate system, but can only be applied to small problems due to its computational complexity. After giving the necessary background,

we will describe the exact extraction algorithm and prove it produces a minimal coordinate system, giving a positive answer to the question of whether minimal coordinate systems can be discovered algorithmically. We argue that the extracted coordinate system can be interpreted as *compressing* objective information of the problem, in the sense that knowing where a candidate solution lies in a coordinate system is equivalent to knowing how that candidate performs against all possible tests. Since the number of axes in a coordinate system can be no more than the number of tests, but may be significantly smaller, a minimal coordinate system is a maximally-compressed view of candidate solutions' performance.[8]

In the game of Nim, where candidates are players and tests are potential opponents, we observe that a substantial compression does take place. Minimal coordinate systems for all tested instances of Nim have significantly fewer axes than tests. We also observe that the axes can be interpreted: each axis directly tests a candidate's strength at a particular game configuration. It is worth noting that, although each axis tests on a corresponding game configuration, not all game configurations correspond to axes. There are significantly fewer axes than game configurations, too. This correlation between axes and game configurations is noteworthy given that the coordinate system extraction algorithm is insensitive to details of the application domain. There is no reason to expect that an axis, while theoretically meaningful, would correspond in an intuitively-meaningful way with a candidate player's ability at the game. Nevertheless, across the tested instances of Nim, extracted coordinate systems consistently represent ability at specific game configurations. These results validate that the notion of coordinate system is not just a theoretical curiosity; rather, coordinate systems give a view into performance that is both compressed in size and intuitively meaningful.

In short, this set of ideas establishes fruitful connections among multi-objective optimization, coevolution, machine learning, and game strategy learning techniques. The theoretical and algorithmic notion of coordinate systems provides a way to view a test-based problem as a sort of multi-objective problem, allowing conceptual cross-fertilization among these disciplines. The results on Nim suggest that this conceptual link can go both ways: treating Nim strategy learning as a multi-objective optimization problem can yield insights into the nature of the game itself as well as how to learn or evolve strategies to play it.

[8] Maximally-compressed with respect to our set of assumptions, of course – naturally there are many ways to compress this information.

2 Preliminaries

2.1 Definition of Problem Structure

Our notion of problem structure is based on the observation that while the number of possible tests in a problem can be very large, these tests may test on a limited number of *underlying objectives* [De Jong and Pollack, 2004]. The question we aim to address is how these underlying objectives may be identified. If this is possible, it permits accurate evaluation using only a limited number of tests. In the following, we consider how a minimal set of objectives can be identified for which the information provided is equivalent to the information provided by the set of all possible tests.

Let $\mathbb{S}$ be the set of candidate solutions in a problem; these can for example be classifiers, or game playing agents. Let $\mathbb{T}$ denote the set of tests. In the example of classification these would be test points; in game playing they would be opponents. A test can be viewed both as an object $T \in \mathbb{T}$ or as a function $T : \mathbb{S} \longrightarrow \{0, 1\}$ that returns the outcome of the test for a given candidate solution. We will write the outcome of a test $T \in \mathbb{T}$ for a candidate solution $S \in \mathbb{S}$ as $T(S)$.

Definition (**Objective**): An *objective* is a function that assigns a value to candidate solutions which measures an aspect of its performance: $o : \mathbb{S} \longrightarrow \mathbb{R}$.

Without loss of generality, we will assume that higher values are to be preferred. An ordered set of objectives OS can be viewed as a vector function that accepts a candidate solution and returns an outcome vector, where each element i is the outcome of objective OS_i: $OS(S) = OS_1(S), OS_2(S), \ldots, OS_n(S)$.

The set of all tests $\mathbb{T}$ can be viewed as a set of objectives; each test can be viewed as a binary objective whose value for a given candidate solution S is given by the outcome of the test $T(S)$. Since $\mathbb{T}$ is given as part of the problem formulation, the corresponding objectives will be called the *initial objectives*:

Definition (**Initial Objectives**): $O_{\text{init}} = \mathbb{T}$.

The question of identifying a minimal set of objectives is thus reduced to the problem of compressing the initial objectives to a minimal set of objectives that provides equivalent information, as we define next.

2.2 Objective Compression

A given set of objectives $O1$ can be compressed to yield an equivalent but smaller set of objectives $O2$. For this purpose, the notion of equivalence is defined as follows.

Definition (**Equivalence**): We define two objective sets $O1$, $O2$ to be equivalent, written $equiv(O1, O2)$, if the following criteria hold:

- Information preservation. This criterion holds if a mapping f exists such that $\forall S \in \mathbb{S} : f(O2(S)) = O1(S)$. If this is the case, the objective values

assigned by $O1$ can be reconstructed from the objective values assigned by $O2$.

- Order preservation. For the transformation to be meaningful, the second set of objectives should result in the same preference information. Though dependent on the preference function, this will generally be achieved if $\exists i : O1_i(x) > O1_i(y) \iff \exists j : O2_j(x) > O2_j(y)$, where $x, y \in \mathbb{S}$, $1 \leq i \leq |O1|$, and $1 \leq j \leq |O2|$.

2.3 Order Preservation: Pareto-dominance

As an example, we demonstrate that the second condition is sufficient to guarantee order preservation for the preference function of Pareto-dominance. The Pareto-dominance preference function states that a candidate solution x is preferred over another candidate solution y, or *dominates* it, with respect to the objectives in $O1$, written $x \overset{O1}{\succ} y$, if:
$\forall i : O1_i(x) \geq O1_i(y) \wedge \exists i : O1_i(x) > O1_i(y)$ with $1 \leq i \leq |O1|$.

If two objective sets $O1$ and $O2$ are equivalent, then $x \overset{O1}{\succ} y \iff x \overset{O2}{\succ} y$. This can be seen as follows:

Assume $x \overset{O1}{\succ} y$. Then $\forall i : O1_i(x) \geq O1_i(y)$ and $\exists i : O1_i(x) > O1_i(y)$. Therefore, $\nexists i : O1_i(y) > O1_i(x)$. Thus, due to the order preservation condition, $\nexists j : O2_i(y) > O2_i(x)$, and hence $\forall j : O2_j(x) \geq O2_j(y)$.

Furthermore, since $\exists i : O1_i(x) > O1_i(y)$, the condition guarantees that $\exists j : O2_j(x) > O2_j(y)$. Since we have $\forall j : O2_j(x) \geq O2_j(y)$ and $\exists j : O2_j(x) > O2_j(y)$, it follows that $x \overset{O2}{\succ} y$, and thus the forward implication has been shown. The backward implication is analogous.

2.4 Problem Structure

Assume we are given a problem P with initial objectives O_{init}. Then P's problem structure is defined by the smallest set of objectives $O_{\min}$ that is equivalent to O_{init}:

Definition (**Problem Structure**):$O_{\min} = \arg\min_{O_i \in \mathbb{O}} |O_i|$ such that $equiv(O_i, O_{\text{init}})$, where $O_i \in \mathbb{O}$, and $\mathbb{O}$ is the set of all possible objective sets given some representation of objectives.

We define the *evaluation dimension* of a problem as the lowest number of dimensions d for which a correct coordinate system exists, or equivalently as the cardinality of the problem structure:

Definition (**Evaluation Dimension**): $d_{eval}() = |O_{\min}|$

2.5 Coordinate Systems for Test-Based Problems

In the following, we present a representation for objectives in test-based problems. Based on this representation, a search algorithm will be used to identify the smallest objective set that is equivalent to the initial objectives, and which therefore represents the structure of the problem.

The objective sets we will consider take the form of *coordinate systems*. Each axis in a coordinate system represents an objective. The position of a test on an axis is determined by which candidate solutions are defeated by the test. We will write $SF(T, S)$ to indicate that test T defeats candidate solution S, meaning it assigns a zero outcome to the candidate solution. Failing a solution S is written $SF(S)$.

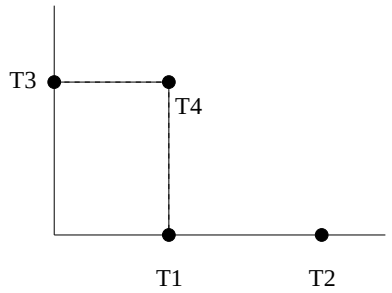

Fig. 1. Example of a coordinate system. Monotonicity: $T2$ defeats strictly more candidate solutions than $T1$. Compositionality: $T4$ defeats all and only those candidate solutions that are defeated by either $T1$ or $T3$.

Definition (**Solution Failure**): $SF(T, S) \iff T(S) = 0$. Solution failures can be grouped together in sets. We write $SF(T)$ to denote the set of all solution failures made by T: $SF(T) = \{SF(S) | S \in \$ \wedge SF(T, S)\}$ The set of all possible solution failures is named SFS: $SFS = \{SF(S) | S \in \$\}$

We now proceed to define the elements of a coordinate system.

Definition (**Axis**): An axis A represents an ordered set of increasing solution failure sets: $A = A_1, A_2, \ldots A_n$ with $A_i \subseteq SFS$ such that $\forall i < j : A_i \subset A_j$. The elements A_i of an axis will be called *coordinates*.

Definition (**Coordinate System**): A coordinate system CS is an ordered set of axes: $CS = A^1, A^2, \ldots A^n$.

Definition (**Position**): A position P in an n-dimensional coordinate system is an ordered set of coordinates, one for each axis: $P = A^1_i, A^2_j, \ldots A^n_k$.

The coordinate systems that will be defined feature the following two properties, which motivate viewing them as coordinate systems, see Figure 1:

Definition (**Monotonicity**): If test $T1$ has a higher position on an axis than $T2$, this implies that $T1$ defeats all candidate solutions defeated by $T2$, in addition to one or more other candidate solutions.

Definition (**Compositionality**): If test $T3$'s position is spanned by two tests $T1$ and $T2$ which reside on different axes, i.e. $T3$'s position is ($T1$,$T2$), then the set of candidate solutions defeated by $T3$ is the union of the sets of candidate solutions defeated by $T1$ and $T2$: $SF(T3) = SF(T1) \cup SF(T2)$.

Both tests and candidate solutions can be embedded into a coordinate system:

Definition (**Test Embedding**): The *embedding* $CS(T)$ of a test T in a coordinate system CS is the position obtained by choosing the highest coordinate on each axis for which T still makes all corresponding solution failures: $CS(T) = \{A_j^i \in CS, 1 \leq i \leq n | SF(T) \supseteq A_j^i \quad \wedge \quad \nexists k > j : SF(T) \supseteq A_k^i\}$.

Definition (**Interpretation of Test Positions**): The set $SF(P)$ of solutions defeated by a test at position P is obtained by taking the union of the solution failure sets represented by the position's coordinates: $SF(P) = \bigcup_{1 \leq i \leq n} P_i$.

Definition (**Solution Embedding**): The embedding $CS(S)$ of a candidate solution S in a coordinate system CS is the position obtained by choosing the highest coordinate on each axis for which S is not included in the corresponding solution failures: $CS(S) = \{A_j^i \in CS, 1 \leq i \leq n | S \notin A_j^i \quad \wedge \quad \nexists k > j : S \notin A_j^i\}$.

Since a coordinate system represents an objective set, $CS(S)$ may be interpreted as the objective vector for candidate solution S; each axis represents one objective, and the coordinate on the axis represents the value of the objective. In order to map coordinates into numerical values, any monotonic assignment of coordinates to values may be used, for example the index of the coordinate on the axis can be employed such that a candidate solution with position (A_8^1, A_5^2) would have objective values $(8, 5)$.

Definition (**Interpretation of Solution Positions**): The set $TS(P)$ of tests solved by a candidate solution at position $P = CS(S)$ is obtained by assembling all tests whose coordinates do not exceed that of the solution: $TS(P) = \{T \in \mathbb{T} | \nexists A^i \in CS(T) | A^i \supset P_i\}$.

Definition (**Correctness**): A coordinate system CS is *correct* for a given set of tests $TS \subseteq \mathbb{T}$, written $correct(CS, TS)$, if for each test the set of solution failures equals the set of solution failures represented by the embedding of the test: $\forall T \in TS : SF(CS(T)) = SF(T)$

Our central theorem states that the objectives represented by a correct coordinate system are equivalent to the initial objectives:

Theorem (**Correctness of coordinate systems**): $correct(CS, O_{\text{init}}) \Longrightarrow equiv(CS, O_{\text{init}})$

Proof: The coordinate systems that have been defined above are a specific way to represent objectives for a test-based problem. We will now demonstrate

that the objectives represented by a correct coordinate system are always equivalent to the initial objectives. Therefore, by restricting the search to correct coordinate systems, it is guaranteed that any coordinate systems found will be equivalent to the initial objectives. We can thus perform objective compression by searching for the smallest correct coordinate system.

Given a correct coordinate system CS, proving equivalence to the initial objectives O_{init} requires establishing the properties of information preservation and order preservation. Regarding information preservation, it is to be shown that a mapping f exists such that $\forall S \in \mathbb{S} : f(CS(S)) = O_{\text{init}}(S)$. In other words, the outcomes $\{T(S)|T \in \mathbb{T}\}$ need to be reconstructed from the coordinates of S. Thus, f can be based on the composition of (1) the embedding function $CS(S)$, which determines the position of a candidate solution in the coordinate system, and (2) the interpretation function $TS(P)$, which determines the tests solved by a candidate at position P: $TS(CS(S))$. This function returns the tests solved by S. Given all tests solved by S, the outcome of any test can be determined by seeing whether the test is part of this set. This yields the desired reconstruction function:

$$f_i(S) = \begin{cases} 1 & \text{if} \quad T_i \in TS(CS(S)) \\ 0 & \text{otherwise} \end{cases}$$

Next, the property of order preservation is to be shown. Assume $\exists i : O_{\text{init},i}(x) > O_{\text{init},i}(y)$. Thus $\exists T \in \mathbb{T} : T(x) > T(y)$; since we are assuming binary tests, this means $T(x) \wedge \neg T(y)$. Given that CS is a correct coordinate system, we know that $TS(CS(y))$ yields the tests solved by y, and must therefore contain a test that is not present in $TS(CS(x))$. Since the axes are monotonic, this implies y must have a higher coordinate for some axis.

Conversely, assume y has a higher coordinate than x for some axis: $TS(CS(y)) > TS(CS(x))$. Then there must exist a test solved by y that is not solved by x. Thus, y has a higher coordinate for the initial objective corresponding to this test. □

3 The Exact Algorithm and Nim

3.1 Exact Algorithm

We present an algorithm that performs exact identification of problem structure, meaning the dimensionality of the extracted coordinate system is guaranteed to be minimal. The algorithm operates by considering all possible coordinate systems in order of increasing dimensionality, and returning when a correct coordinate system has been found. This guarantees that the smallest possible coordinate system, measured in terms of its dimensionality, will be found.

As described, a coordinate system consists of axes whose coordinates represent solution failure sets. Given a set of candidate solutions, the number of

all solution failure sets is the power set of this set, and thus exponential in the size of the solution set. Considering all assignments of all solution failure sets to axes is therefore prohibitive. However, the only requirement for a coordinate system is that it can represent the given set of tests. Thus, the coordinate system must contain positions corresponding to the solution failure sets represented by the tests, but need not contain positions corresponding to other solution failure sets. This greatly reduces the search problem; due to this observation, we can restrict the search by only considering subsets of the solution failure sets represented by tests.

Since the number of all possible coordinate systems grows very quickly, the exact algorithm that will be presented will in general still not be feasible for problems of realistic size. Its purpose however is to permit studying the structure of small example problems, and to provide a starting point for efficient structure approximation algorithms.

The algorithm starts from an empty coordinate system. Each axis is initially defined by two virtual tests: ORG (origin), which passes all solutions, and INF (infinity), which defeats all solutions. To search for a correct coordinate system, all tests are visited in turn. For each test, it is determined whether the test can be placed in the current coordinate system; if not, the coordinate system is adapted to permit representing the test. If this fails given the current dimensionality because one or more additional axes are required, the dimensionality is increased.

The placement of tests is stored in a state vector, representing the current placement of each test. Two search operators are employed: `find\_first\_full\_state(state)` finds the first correct coordinate system from the current state by visiting the tests in turn and placing each test correctly, adapting the coordinate system where necessary. `inc_full_state` increases the current state; this operator is applied when the current state does not permit the construction of a correct coordinate system. By continuing the search for a correct coordinate system while the state so far permits this and increasing the state once it is found that it does not permit this, a correct coordinate system of the given dimensionality will be found if it exists. Since the dimensionality is incremented only if no correct system is found, the first coordinate system found by the algorithm is guaranteed to be minimal. The algorithm therefore returns once a correct coordinate system is found. The pseudocode of the main loop of the algorithm is as follows:

`calculate_options(T)` accepts a test and for each axis i determines the highest coordinate for which the test still makes all corresponding solution failures. These coordinates are written $T^i_{\min}$, and represent the embedding of the test in the partially constructed space. The successor of a coordinate $A^i_{\min}(T)$ is written $A^i_{\max}(T)$. The coordinate of a test on axis i must lie between $A^i_{\min}(T)$ and $A^i_{\max}(T)$. Potentially, a new coordinate must be created. The options for such a new coordinate are constrained by $A^i_{\min}(T)$ and $A^i_{\max}(T)$; the solution failure set it represents must be a superset of the former

```
FIND_STRUCTURE()
 1: for num_dims = 1 to num_tests do
 2:   init_axes()
 3:   state = [0, 0, 0, 0, 0]
 4:   for i = 1 to num_tests do
 5:     calculate_options(T)
 6:     init_test_state(i)
 7:   end for
 8:   while !done do
 9:     if find_first_full_state(state) then
10:       done = true! {success; break outer for loop and return}
11:     else
12:       ok = inc_full_state
13:       if !ok then
14:         done = true {failure; increment num_dims}
15:       end if
16:     end if
17:   end while
18: end for
```

and a subset of the latter. Therefore, the algorithm must search the powerset of $A^i_{\max}(T) \setminus A^i_{\min}(T)$.

`init_test_state(T)` places a test at the highest possible existing coordinates, i.e. $A^1_{\min}(T), A^2_{\min}(T), \ldots, A^n_{\min}(T)$.

`find_first_full_state(state)` looks for the first feasible state from the current state by ensuring a correct placement for each test in turn. This is done by calling `inc_test_state` until a correct placement for a test is found.

`inc_test_state(i)` increments the state element corresponding to test T, and fails if this exceeds the range for this state element. If the increment succeeds, the element represents a new placement for the test. The order of the options considered is as follows: First, the test is placed at the position that its embedding would indicate, i.e. $A^1_{\min}(T), A^2_{\min}(T), \ldots, A^n_{\min}(T)$, using INIT-TEST-STATE(T). The next n options consider placing the test on one of the n axes. If this succeeds, the test defines a new coordinate for the axis concerned. Finally, all combinations of solutions failures in $A^i_{\max}(T) \setminus A^i_{\min}(T)$ are considered.

`inc_full_state` moves to the next state by increasing the state vector.

3.2 The Game of Nim

We will explore the notion of problem structure and its extraction by applying the search algorithm to the Game of Nim.

Nim originates from the Chinese game Tsyanshidzi (picking stones game). The first European reference to a possible Nim-type game dates from 1503 [Singmaster, 1996]. The name is thought to stem from the German imperative

'nimm' (take), and is proposed in [Bouton, 1902]. The game also features in Alain Resnais' 1961 movie *Last Year at Marienbad*.

The game starts by placing a number of rows of small objects such as matches on a table, where each row is called a *counter*. The players take turns, and on each turn must take one or more matches from a single row. The player to take the last match wins the game. In the *misère* version of the game, the outcome is reversed.

Nim is an *impartial game*, meaning each player has the same available moves in every position. An interesting result of combinatorial game theory is the Sprague-Grundy theorem, independently discovered by Roland P. Sprague [Sprague, 1936] and Patrick M. Grundy [Grundy, 1939]. The theorem states that every impartial game is equivalent to a nim position, augmented with the possibility of adding matches.

In 1901, the Harvard mathematician Charles L. Bouton presented an optimal strategy for the game of Nim [Bouton, 1902]. To compute the strategy, the counters of a Nim position are written below one another in binary notation. Next, the columns of the numbers are summed. If all sums are even, the game position is a *safe combination*. To play optimally, the player must merely select a move that results in a safe combination.

3.3 Results

We apply the structure extraction algorithm to Game of Nim versions with the following initial configurations: [1 3], [4], and [2 2].[9] We employ the misère version of the game, which has been noted as being more difficult to analyze. For all of these small game configurations, the structure extraction algorithm took less than a second. However, due to the high computational complexity of the exact algorithm, larger versions of the game quickly lead to high computational requirements.

To apply the structure extraction algorithm, we first generate all strategies for a game. No distinction is made between first and second players; for both players, all combinations of the legal moves in all game configurations are considered. The complete set of strategies is then played against itself in a full squared comparison. Since some game configurations never occur for some strategies, some players will behave identically while having different strategy representations. Therefore, any first players with outcome vector identical to other first players are removed, and likewise for duplicate second player. This yields an outcome matrix representing all unique first and second players, which serves as input to the structure extraction algorithm. The tests in the matrix are first sorted according to the number of solution failures they represent; since a complete search is performed, this does not affect the

[9] The notation $[x_1 x_2]$ means this game instance has two piles of sticks and the starting configuration has x_1 sticks in the first pile and x_2 sticks in the second pile.

dimensionality of the outcome, but it may serve to consider the more likely coordinate systems first.

Results for Nim [1 3]

The first version we apply the algorithm to is the [1 3] configuration, which has a row containing a single match and one containing three matches. There are 7 non-empty game configurations. Different configurations have different numbers of legal moves; the full number of strategies is 144. However, after removing first and second players with duplicate outcome vectors, 6 first player strategies (candidate solutions) and 9 second player strategies (tests) remain. The resulting outcome matrix for unique strategies is shown in table 1.

	T0	T4	T8	T24	T28	T32	T48	T52	T56
S0	1	0	1	1	0	1	1	0	1
S1	0	0	0	0	0	0	0	0	0
S2	1	1	1	1	1	1	1	1	1
S3	1	1	1	0	0	0	1	1	1
S72	1	0	0	1	0	0	1	0	0
S75	0	0	0	0	0	0	1	1	1

Table 1. Outcome matrix of unique strategies for Nim[1 3].

The result of applying the structure extraction algorithm to the outcome matrix for this game is a two-dimensional coordinate system; see Table 2. The first axis consists of tests T48, T0, and T24, representing the solution failure sets $\{S1\}$, $\{S1, S75\}$, and $\{S1, S75, S3\}$, in addition to ORG and INF. Interestingly, all coordinates on the axis thus correspond exactly to actual tests, rather than being arbitrary subsets of solution failures. The same holds for the second axis, containing tests $T56$ and $T52$.

Since a correct coordinate systems embeds all tests onto positions that correspond to their outcomes, a question is where the four tests that do not lie on the axes are located. This is shown in Table 3 (left) and visualized in Figure 2. For any of these tests, the set of solutions defeated is the union of the solution failure sets of its coordinates.

A next question is where the candidate solutions are embedded in the space. This is shown in Table 3 (right) and Figure 3. The embedding of the solutions shows how two objectives are sufficient to evaluate the solutions and express the relations between them. For example, it is immediately seen that solution S2 is the optimal solution, as it has the highest coordinate on both dimensions. Comparing this with the initial situation in which all nine tests are objectives demonstrates the utility of objective compression: by identifying the compositionality implicitly present in the problem, the number of objectives required for evaluation is reduced from nine to two.

dim 1	ORG	T48	T0	T24	INF
S1	1	0	0	0	0
S75	1	1	0	0	0
S3	1	1	1	0	0
S0	1	1	1	1	0
S2	1	1	1	1	0
S72	1	1	1	1	0

dim 2	ORG	T56	T52	INF
S1	1	0	0	0
S72	1	0	0	0
S0	1	1	0	0
S2	1	1	1	0
S3	1	1	1	0
S75	1	1	1	0

Table 2. The two-dimensional coordinate system for Nim[1 3].

test	coordinates
T8	(T0 , T56)
T4	(T0 , T52)
T32	(T24 , T56)
T28	(T24 , T52)

test	coordinates
S0	(T24 , T56)
S1	(ORG, ORG)
S2	(T24 , T52)
S3	(T0 , T52)
S72	(T24 , ORG)
S75	(T48 , T52)

Table 3. Left: Coordinates of the tests not lying on the axes. Each test is precisely a composition of two other tests; see outcome matrix and text. **Right:** Coordinates of the candidate solutions.

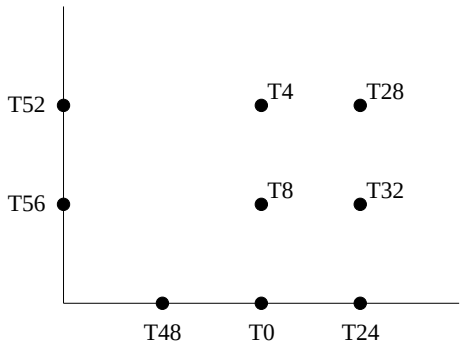

Fig. 2. Embedding of the tests into the coordination system found for Nim[1 3].

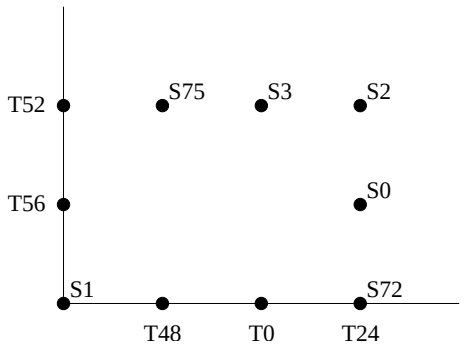

Fig. 3. Embedding of the candidate solutions into the coordination system found for Nim[1 3]. Two dimensions are sufficient to evaluate the solutions and express the relations between them, as compared to the original nine dimensions required when using all tests as objectives. This substantial reduction demonstrates the potential value of objective compression.

Apart from the potential computational advantage offered by objective compression, an interesting theoretical question is whether the automatically extracted coordinate system can tell us something about the structure of a problem. To this end, we now interpret the axes that have been identified.

Table 4 shows the actions selected by the axis tests in all of the seven different game configurations that can occur in Nim[1 3], the resulting game configuration after the action, and the outcome for the test. It is immediately seen that the tests in dimension 1 only differ in a single configuration, viz. configuration 3, which represents the game state [0 3], i.e. a single row with three matches. The first test on the axis, $T48$, removes all three matches and thus leaves no matches, thereby losing the game. $T0$ leaves two matches, so that its outcome depends on the opponent strategy. Finally, the last strategy on the axis, $T24$, leaves a single match, and thereby secures a sure win. Thus, the particular coordinate system that has been extracted has a clear interpretation in this dimension: the objective represents the quality of the move selected by a test in game situation [0 3].

For the second dimension (Table 5), the result is similar. The tests only differ in game situation 6, which represents the configuration [1 2]. The tests select increasingly good moves, leaving [0 2] (result depending on opponent), and [1 0] (sure win) respectively.

Results for Nim [2 2]

Nim[2 2] has 8 configurations, resulting in 288 strategies. The resulting outcome matrix contains 36 unique tests and 6 unique candidate solutions. The extracted minimal coordinate system is 4-dimensional; thus, a high degree of compression is observed. As in Nim[1 3], all dimensions can be interpreted,

	C1	C2	C3	C4	C5	C6	C7	After	Result
T48	1	1	3	1	1	1	1	[0 0]	sure loss
T0	1	1	1	1	1	1	1	[0 2]	depends
T24	1	1	2	1	1	1	1	[0 1]	sure win

Table 4. Actions selected by the tests on the first axis in the seven game configurations. The axis apparently tests on the ability to select a good action in situation 3, which represents configuration [0 3]. Moving down the table, the resulting configurations show the increasing quality of the moves.

	C1	C2	C3	C4	C5	C6	C7	After	Result
T56	1	1	3	1	1	3	1	[0 2]	depends
T52	1	1	3	1	1	2	1	[1 0]	sure win

Table 5. Actions selected by the tests on the second axis. Again, the axis concerns the behavior of tests in a single configuration: [1 2], and the tests represent increasingly good moves for this configuration.

and the different options for a dimension always concern a single game configuration.

Results for Nim [4]

Nim[4] has four configurations, and 24 strategies. There are six unique tests and five unique candidate solutions. The problem is two-dimensional, and as for the previous two problems the dimensions have a clear interpretation. The coordinate system and the embedding of the candidate solutions is visualized in Figure 4.

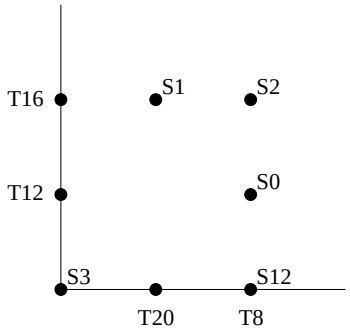

Fig. 4. Embedding of the candidate solutions into the coordination system found for Nim[4].

4 Discussion and Conclusions

We have presented an algorithm that is able to extract the *underlying objectives* of an arbitrary test-based problem. Any test-based problem has underlying objectives. Viewing each test as a separate objective yields an initial objective set. By compressing this initial set into a smaller number of objectives, meaningful underlying objectives can be obtained. The algorithm that has been presented delivers a set of objectives that are guaranteed to be minimal in number, yet which provide evaluation information that is equivalent to the initial objective set.

The optimal dimension extraction algorithm has been applied to small versions of the Game of Nim. A main finding is that by extracting the underlying objectives, the number of required objectives can be drastically reduced. In other words, the extraction algorithm substantially decreases the dimensionality of the objective space, without loss of information. For example, for a version of the game with 36 distinct tests, the initial set of 36 objectives, was reduced to an equivalent objective set containing only 4 objectives. This demonstrates that the notion of objective set compression is not merely a theoretical possibility; in the example domain that has been explored, the number of objectives can be greatly reduced. While there is no guarantee that the rate of compression in other games will be comparable, it has been shown that a substantial reduction of the number of objectives is possible at least for some problems.

Objective compression is a new analytical tool that has several applications. First and foremost, it may serve to increase our insight into existing problems. By identifying the underlying objectives of a problem, intrinsic information about the structure of the evaluation function implicit in the problem is revealed. Our results with the Game of Nim showed that the underlying objectives can be highly interpretable; in all cases, objectives represented orthogonal dimensions of performance in the game where each subsequent position on an axis represented a better strategic choice, and where each axis corresponded to a single state that may arise during play.

A second consequence of theoretical interest is that any test-based problem is characterized by an *evaluation dimension*: the minimal number of objectives for which an objective set exists that is equivalent to the initial objective set. The evaluation dimension forms an intrinsic property of the problem. Thus, an open question for any test-based problem is what its evaluation dimension is; we expect the evaluation dimension has implications for the complexity of evaluation and search. An open question is how the evaluation dimension or the problem structure relate to existing notions of dimensionality or complexity, such as the teaching dimension [Goldman and Kearns, 1995].

In addition to these theoretical implications, problem structure extraction may find applications in learning and search. If the relevant dimensions of performance for a problem can be identified or estimated, this may help to select which tests are to be used in evaluation. An illustration of this principle

has been given in [De Jong and Bucci, 2006], which details a coevolutionary algorithm that guides selection using a coordinate system extracted from the population.

The algorithm presented here guarantees that the coordinate systems that are produced will be of minimum size, and thus have a number of objectives equal to the evaluation dimension. While this algorithm is of theoretical interest in that it can be used to identify minimal-dimensional objective sets for small test problems, application to problems of practical interest will typically be computationally infeasible. Therefore, a final open question is how objective set compression may be performed efficiently when approximate solutions are acceptable. An example of an approximate dimension extraction algorithm has been given in [Bucci et al., 2004], but more accurate and more efficient algorithms would form valuable extensions of the work presented here.

References

[Bouton, 1902] Bouton, C. L. (1901–1902). Nim, a game with a complete mathematical theory. *The Annals of Mathematics, 2nd Ser.*, 3(1-4):35–39.

[Bucci and Pollack, 2003] Bucci, A. and Pollack, J. B. (2003). Focusing versus intransitivity: Geometrical aspects of coevolution. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation Conference - GECCO 2003*, volume 2723 of *Lecture Notes in Computer Science*, pages 250–261. Springer.

[Bucci and Pollack, 2005] Bucci, A. and Pollack, J. B. (2005). On identifying global optima in cooperative coevolution. In Hans-Georg Beyer et al., editor, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 1, pages 539–544, Washington DC, USA. ACM Press.

[Bucci et al., 2004] Bucci, A., Pollack, J. B., and De Jong, E. D. (2004). Automated extraction of problem structure. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation Conference – GECCO 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 501–512. Springer.

[De Jong and Bucci, 2006] De Jong, E. and Bucci, A. (2006). DECA: Dimension extracting coevolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-06*, pages 313–320.

[De Jong and Pollack, 2004] De Jong, E. D. and Pollack, J. B. (2004). Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192.

[Deb and Saxena, 2006] Deb, K. and Saxena, D. K. (2006). Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 3353–3360. IEEE Press.

[Ficici and Pollack, 2001] Ficici, S. G. and Pollack, J. B. (2001). Pareto optimality in coevolutionary learning. In Kelemen, J. and Sosík, P., editors, *Sixth European Conference on Artificial Life (ECAL 2001)*, pages 316–325. Springer.

[Goldman and Kearns, 1995] Goldman, S. A. and Kearns, M. J. (1995). On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31.

[Grundy, 1939] Grundy, P. M. (1939). Mathematics and games. *Eureka*, 2:6–8. Reprinted in Eureka 27 (1964) 9-11.

[Hillis, 1990] Hillis, D. W. (1990). Co-evolving Parasites Improve Simulated Evolution in an Optimization Procedure. *Physica D*, 42:228–234.

[Knowles et al., 2001] Knowles, J. D., Watson, R. A., and Corne, D. W. (2001). Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In Zitzler, E., Deb, K., Thiele, L., Coello, C. C., and Corne, D., editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 268–282. Springer, Berlin.

[Noble and Watson, 2001] Noble, J. and Watson, R. A. (2001). Pareto Coevolution: Using Performance Against Coevolved Opponents in a Game as Dimensions for Pareto Selection. In L. Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 493–500, San Francisco, CA. Morgan Kaufmann Publishers.

[Potter and Jong, 2000] Potter, M. A. and Jong, K. A. D. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

[Saxena and Deb, 2007] Saxena, D. K. and Deb, K. (2007). Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 772–787.

[Singmaster, 1996] Singmaster, D. (1996). Chronology of recreational mathematics. http://anduin.eldar.org/ problemi/singmast/recchron.html.

[Sprague, 1936] Sprague, R. P. (1935–1936). Über mathematische kampfspiele. *Tôhoku Mathematical Journal*, 41:438–444.

[Watson and Pollack, 2001] Watson, R. and Pollack, J. B. (2001). Coevolutionary dynamics in a minimal substrate. In L. Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, San Francisco, CA. Morgan Kaufmann Publishers.