Formal Analysis, Hardness and Algorithms for Extracting Internal Structure of Test-Based Problems

Wojciech Jaśkowski
wjaskowski@cs.put.poznan.pl

Krzysztof Krawiec
kkrawiec@cs.put.poznan.pl

Institute of Computing Science, Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland

Abstract
Problems in which some elementary entities interact with each other are common in computational intelligence. This scenario, typical for co-evolving artificial-life agents, learning strategies for games, and machine learning from examples, can be formalized as a test-based problem and conveniently embedded in the common conceptual framework of coevolution. In test-based problems candidate solutions are evaluated on a number of test cases (agents, opponents, examples). It has been recently shown that every test of such problem can be regarded as a separate objective, and the whole problem as multi-objective optimization. Research on reducing the number of such objectives while preserving the relations between candidate solutions and tests led to the notions of underlying objectives and internal problem structure, which can be formalized as a coordinate system that spatially arranges candidate solutions and tests. The coordinate system that spans the minimal number of axes determines the so-called dimension of a problem and, being an inherent property of every problem, is of particular interest. In this study, we investigate in-depth the formalism of coordinate system and its properties, relate them to properties of partially ordered sets, and design an exact algorithm for finding a minimal coordinate system. We also prove that this problem is NP-hard and come up with a heuristic which is superior to the best algorithm proposed so far. Finally, we apply the algorithms to three abstract problems and demonstrate that the dimension of the problem is typically much lower than the number of tests, and for some problems converges to the intrinsic parameter of the problem — its a priori dimension.

Keywords
Coevolution, Co-optimization, Games, Test-Based Problem, Interactive Domains, Pareto Coevolution, Underlying Objectives, Internal Problem Structure, NP-Hardness

[This is a preprint version. The article has been accepted for publication in Evolutionary Computation Journal.]

1 Introduction
A great part of computational intelligence research is devoted to solving problems in which some elementary entities interact with each other. Game-playing agents learn by playing against opponent strategies. Machine learning algorithms generate hypotheses and test them on examples. Evolutionary algorithms simulate the evolved designs in various environmental conditions. What these scenarios have in common is the under-
lying concept of an interaction between a solution (strategy, hypothesis, design) and a test (opponent strategy, training example, environmental conditions, respectively).

Although there exist many algorithms tailored to specific forms of such test-based problems (Bucci et al., 2004), e.g., to machine learning or game-playing, interactions between entities were historically mostly exploited within coevolutionary algorithms (Hillis, 1992; Rosin and Belew, 1997). The essence of coevolution is that the results of interactions drive the search process and substitute for an objective, external fitness measure found in traditional evolutionary algorithms. For problems for which the relations between entities are symmetric, e.g., artificial life or some games, coevolution can be modeled within a single population and is then called one-population coevolution (Luke and Wiegand, 2002; Jaśkowski et al., 2008) or competitive fitness environment (Angeline and Pollack, 1993; Luke, 1998). However, due to the inherent asymmetry of many practical problems (e.g., learning from examples) and because solutions and tests play substantially different roles in the search process and there are some arguments that their separation may be beneficial for coevolution (Bucci, 2007), it is typical for modern coevolutionary algorithms to evolve them in two populations, where tests are evaluated against solutions and vice versa. Except for the interaction phase, both populations undergo standard evolutionary operators such as selection and variation, and with time tests are expected to become more difficult for solutions while solutions more competent for solving tests. This principle of an arms-race driving the optimization process may be generalized to other non-evolutionary optimization techniques, leading to co-optimization (Service and Tauritz, 2008).

An outcome of a single interaction is usually not enough to provide an informative learning gradient and effectively guide the search process. Thus, in a typical coevolutionary algorithm multiple outcomes are aggregated, leading to an overall fitness measure, like probability of winning or accuracy of classification in the former examples. In this way, however, the individual characteristics of particular solutions and tests are inevitably lost, and the performance measure is prone to compensation – two solutions with completely different outcomes of interactions with tests can obtain the same fitness. The search algorithm has no insight into the actual, complex interactions taking place between solutions and tests.

The above-mentioned aggregation of interaction outcomes is one of the reasons for which coevolutionary algorithms often suffer from so-called pathologies. Examples of such undesired phenomena include loss of gradient, cycling (Watson and Pollack, 2001), disengagement (Cartlidge and Bullock, 2004), and inaccurate evaluation (de Jong and Pollack, 2004). In Pareto coevolution (Ficici and Pollack, 2001; Noble and Watson, 2001) proposed to overcome these drawbacks, aggregation of interaction outcomes has been abandoned in favor of using each test as a separate objective. This transforms the test-based problem into a multi-objective optimization problem and allows resorting to the well-defined concept of dominance – solution \(s_1\) is not worse than solution \(s_2\) if and only if \(s_1\) performs at least as good as \(s_2\) on all tests (see Fig. 1). Such process of converting a single-objective problem into a multi-objective one has been termed multi-objectivization (Knowles et al., 2001). Unfortunately, in real test-based problems the number of tests is usually prohibitively large or even infinite: take for instance the number of strategies in chess. Therefore, also the dimensionality of a search space in Pareto coevolution is enormous.

It was observed however, that the number of objectives in Pareto coevolution can often be reduced, since many test-based problems possess an internal structure. This structure manifests itself by the fact that sometimes it is possible to determine a group
of tests that examine the same skill or aspect of solution performance, but with different intensity. Instead of defining different objectives, such tests can be ordered with respect to difficulty and placed on a common axis identified with a single new objective. Since such an objective is not known \textit{a priori}, but must be revealed during exploration of the problem, it is referred to as an \textit{underlying objective} (de Jong and Pollack, 2004). For instance, the underlying objectives in chess could include skills of controlling the center of the board, using knights, playing endgames, etc.

The above intuition about underlying objectives and internal structure of a problem was first formalized in the notion of \textit{coordinate system} by Bucci et al. (2004), whose work our paper is based on. An important feature of coordinate system is that while \textit{compressing} the initial set of objectives, it preserves the relations between solutions and tests. Each solution is embedded in the system and the outcome of its interaction with any test can be determined given its position on all axes. As stated by Bucci et al. (2004), \textit{the structure space captures essential information about a problem in an efficient manner}.

Beyond the purely aesthetic appeal, the practical motivation for extracting the internal structure of a problem is twofold. First, there is evidence that such a structure may be exploited for the benefit of coevolutionary algorithms, for instance, by accelerating convergence or guaranteeing progress, like in Dimension Extraction Coevolutionary Algorithm (de Jong and Bucci, 2006). Second, by knowing the internal structure and underlying objectives, we can learn important properties of the problem (de Jong and Bucci, 2008). The answer to the question what are the underlying objectives of the problem can give valuable insight into problem properties and help to choose a method to solve it. The presumably most important problem property is its \textit{dimension}, i.e., the number of underlying objectives. It is hypothesized that problem dimension is highly correlated with its difficulty.

In this paper, we concentrate on the computational aspects of extracting problem structure and try to answer the questions: how to extract the underlying objectives of a problem, and, even more importantly, how to do it efficiently, so that the underlying objectives can be updated during a coevolutionary run and exploited for the sake of improving search convergence, without potentially outweighing these benefits with an enormous computational overhead. To this aim, we elaborate on the particular type of coordinate system defined by Bucci et al. (2004), formally introducing all the necessary concepts in Sections 2, 3, and 4. Our major contributions include: (i) identifying important new properties of coordinate systems and pointing out their relations to partially ordered sets (Section 3), (ii) proving the \textit{NP}-hardness of the problem of determining the dimension of the minimal coordinate system (Section 5), (iii) providing exact and approximate algorithms for building a minimal coordinate system (Section 7), and (iv) experimental demonstration of these topics on three problems (Section 8). In particular, the entire contribution (ii), a new efficient heuristic proposed in (iii), considerations about infinite games (Section 5.1) and complete evaluation set (Section 9), and the experimental part (iv) extends our previous study on this topic (Jaskowski and Krawiec, 2009).

\section{Mathematical background}

In this paper, we use basic discrete mathematics concepts concerning partially ordered sets, which we briefly introduce in this section. The definitions below follow Trotter (1992).
placed according to their performance on tests. For instance, solution $s_1$ solves test $t_1$, but fails test $t_2$.

**Figure 1:** Pareto coevolution. Each test (here $t_1$ and $t_2$) serves as a separate objective. Solutions (here $s_1$, $s_2$ and $s_3$) are embedded in the space spanned by the objectives, i.e., placed according to their performance on tests. For instance, solution $s_3$ solves test $t_1$, but fails test $t_2$.

**Definition 1.** A partially ordered set (poset, for short) is a pair $(X, P)$, where $X$ is a set and $P$ is a reflexive, antisymmetric, and transitive binary relation on $X$. We call $X$ the ground set while $P$ is a partial order on $X$.

We write $x \leq y$ in $P$ when $(x, y) \in P$ and $x \geq y$ in $P$ when $(y, x) \in P$. The notations $x < y$ in $P$ and $y > x$ in $P$ mean $x \leq y$ in $P$ and $x \neq y$. When the context is obvious, we will abbreviate $x < y$ in $P$ by just writing $x < y$.

**Definition 2.** For a poset $(X, P)$, $x, y \in X$ are comparable ($x \perp y$) when either $x \leq y$ or $x \geq y$; otherwise, $x$ and $y$ are incomparable ($x \parallel y$).

**Definition 3.** A poset $(X, P)$ is called a chain if every pair of elements from $X$ is comparable. When $(X, P)$ is a chain, we call $P$ a linear order on $X$. Similarly, we call a poset an antichain if every pair of distinct elements from $X$ is incomparable. A chain (respectively, antichain) $(X', P')$ is a maximum chain (respectively, maximum antichain) in $(X, P)$, $X' \subseteq X$, $P' \subseteq P$ if no other chain (respectively, antichain) in $(X, P)$ has more elements than it.

**Definition 4.** An $x \in X$ is called a maximal element (respectively, minimal element) in $(X, P)$, if there is no $y \in X$ such that $x < y$ (respectively, $x > y$). We denote the set of all maximal elements of a poset $(X, P)$ by max$(X, P)$, and by min$(X, P)$ the set of all minimal elements.

**Definition 5.** The width of a poset $(X, P)$, denoted as width$(X, P)$, is the number of elements in its maximum antichain.

**Theorem 6.** [Dilworth 1941] If $(X, P)$ is a poset and width$(X, P) = n$, then there exists a partition of $X = C_1 \cup C_2 \cup \cdots \cup C_n$, where $C_i$ is a chain for $i = 1, 2, \ldots, n$. We call it minimum chain partition, as it comprises the smallest possible number of chains.

Note that an important consequence of Dilworth’s theorem is that each $C_i$ contains exactly one element of the maximum antichain.

**Definition 7.** Given a poset $(X, P)$, a linear extension of $P$ is any superset of $P$ that is a linear order on $X$.

**Definition 8.** The dimension of a poset $(X, P)$, denoted dim$(X, P)$, is the smallest cardinal number\footnote{Here we follow the original paper by Dushnik and Miller [1941], since Trotter [1992] requires the dimension of a poset to be a positive integer.} for which there exists a family $\mathcal{R} = \{L_1, L_2, \ldots, L_t\}$ of linear extensions of $P$ so that $P = \bigcap \mathcal{R} = \bigcap_{i=1}^{t} L_i$. 

Evolutionary Computation Volume x, Number x
Example 9. Consider a poset \((X, P)\) with \(X = \{a, b, c, d, e, f\}\) and \(P = \{(a, c), (a, d), (b, d), (b, e), (c, f), (d, f), (a, f), (b, f)\} \cup \{(x, x) : x \in X\}\) shown below in a form of a Hasse diagram.

\[
\text{a} \quad \text{b} \quad \text{c} \quad \text{d} \quad \text{e} \quad \text{f}
\]

In \((X, P)\) some of the elements are comparable, e.g. \(a \leq f, b \leq e\); others are incomparable, e.g., \(c \parallel d, a \parallel e\), thus it is not a linear order. Examples of chains in \((X, P)\) are \(a < c \land b < d < f\). The latter is a maximum chain in \((X, P)\). \(\{a, c\}\) is an example of antichain, and \(\{c, d, e\}\) is a maximum antichain of this poset, thus \(\text{width}(X, P) = 3\). The set of maximal elements consists of \(c, f\), while \(a, b\) are minimal elements of the poset. \(\{a, c\} \cup \{b, d\} \cup \{c\}\) is an example of a minimum chain partition. \(L = a < c < b < d < e < f\) is an example of linear extension of \(P\). \(\text{dim}(X, P) = 2\), because there exists a family of two linear extensions \(L_1\) and \(L_2\), such that \(P = L_1 \cap L_2\); namely \(L_1 = a < c < b < d < f < e\) and \(L_2 = b < e < a < d < c < f\), and no smaller family with this property exists.

3 Test-Based Problem

Definition 10. A test-based problem is a formal object \(G = (S, T, G)\) that consists of a set \(S\) of solutions, a set \(T\) of tests, and an interaction function \(G : S \times T \rightarrow \mathbb{R}\).

In related work, solutions were also referred to as candidate solutions \(\text{Bucci and Pollack} 2003\), candidates \(\text{Bucci et al.} 2004\), learners \(\text{de Jong} 2004a\) or hosts \(\text{Rosin and Belew} 1993\), and tests as teachers or parasites.

In this paper, we restrict our attention to the case where the codomain of \(G\) is a binary set \(\{0, 1\}\). If \(G(s, t) = 1\), we say that solution \(s\) solves test \(t\); otherwise if \(G(s, t) = 0\), we say that \(s\) fails test \(t\). Where convenient, we will treat \(G\) as a relation and denote the fact that solution \(s\) solves test \(t\) as \(G(s, t)\) and the fact that it fails test \(t\) as \(\overline{G}(s, t)\).

Notice that in the perspective of game theory, \(G\) can be interpreted as a payoff matrix, \(S\) and \(T\) as sets of strategies, and \(G\) as a game. Thus, in this paper we will use the terms test-based problem and game interchangeably. For simplicity, we assume that both \(S\) and \(T\) are finite (\(G\) is a finite game), however, we will also raise the issue of games that are not finite.

Definition 11. A solutions failed set \(SF(t) \subseteq S\) is comprised of all solutions that fail the test \(t\). Dually, a tests solved set \(TS(s) \subseteq T\) is comprised of all tests that are solved by solution \(s\).

Notice also that \(t \in TS(s) \iff s \notin SF(t)\) for all \(s \in S, t \in T\), since both sides hold if and only if \(s\) solves \(t\).

Definition 12. Test \(t_1\) is weakly dominated by test \(t_2\), written \(t_1 \leq t_2\), when \(SF(t_1) \subseteq SF(t_2)\) for \(t_1, t_2 \in T\). Dually, solution \(s_1\) is weakly dominated by solution \(s_2\), written \(s_1 \leq s_2\), when \(TS(s_1) \subseteq TS(s_2)\) for \(s_1, s_2 \in S\).

For brevity we use the same symbol \(\leq\) for both relations, as they are univocally determined by the context. Since \(\leq\) inherits transitivity and reflexivity from \(\subseteq\), it is a
preorder in both $S$ and $T$. To make $\leq$ a partial order we need to assume that no two elements of one set are indiscernible with respect to how they interact with the elements of the other set, precisely: $\forall t_1, t_2 \in T, t_1 \neq t_2 : SF(t_1) = SF(t_2)$ and $\forall s_1, s_2 \in S, s_1 \neq s_2 : TS(s_1) = TS(s_2)$. Under this assumption $s_1 = s_2 \iff TS(s_1) = TS(s_2)$ and, dually, $t_1 = t_2 \iff SF(t_1) = SF(t_2)$; thus $(S, \leq)$ and $(T, \leq)$ are posets which eases our further arguments. In case some indiscernible objects do exist (it can happen in practice), we can merge them into one object without losing any important features of $G$.

4 Coordinate System

In the context of test-based problems, coordinate system is a formal concept revealing internal problem structure by enabling the solutions and tests to be embedded into a multidimensional space. Of particular interests are such definitions of coordinate systems, in which the relations between solutions and tests ($\leq$) are reflected in spatial arrangement of their locations in the coordinate system. Previous work suggests that this formalism can help design better coevolutionary algorithms (de Jong and Bucci, 2008) and examining properties of certain problems (de Jong and Bucci, 2006) and further investigated in (de Jong and Bucci, 2008).

There is no unique definition of coordinate system for test-based problems; currently we are aware of two formulations: by Bucci et al. (2004) and by de Jong and Bucci (2006), further investigated in (de Jong and Bucci, 2008). The difference between them lies in the way they define axes: the former defines an axis as a sequence of tests ordered by the domination relation, whereas the latter as a sequence of sets of solutions ordered by the inclusion relation. In this paper we try to analyze the coordinate system introduced by Bucci et al. (2004), so in the following by coordinate system we mean the one defined there. There are slight differences in our formulation, which, however, do not affect any important properties of coordinate system. First, in our formulation, the positions of solutions on an axis are shifted one test to the left, which is more convenient. Second, Bucci et al. worked with preordered sets, but we, as pointed out earlier, limit our discussion to posets. The reason for this simplification is merely technical, since some mathematical concepts we need were defined for posets and their extensions to preordered sets require additional effort (see, e.g., the dimension of the preordered set in chapter 4.2.1 of Bucci (2007)). Our results could be generalized to a situation where $S$ and $T$ are preordered sets, thus they are applicable to any test-based problem, however, in this paper, we stick with posets to make our presentation more comprehensible.

For convenience, we introduce a formal element $t_0$ such that $G(s, t_0)$ for all $s \in S$. Also, we define an operator ‘overline’ that augments a set of tests with $t_0$, i.e., $\overline{X} = X \cup \{t_0\}$.

**Definition 13.** A coordinate system $C$ for a game $G$ is a set of axes $(A_i)_{i \in I}$, where each axis $A_i \subseteq T$ is linearly ordered by $<$. $I$ is an index set and the size of the coordinate system, denoted by $|C|$, is the cardinality of $I$.

We interpret an axis as an underlying objective of the problem. Tests on an axis are ordered with respect to increasing difficulty ($<$ relation), so that every solution can be positioned on it according to the results of its interaction with these tests. The position of a solution is precisely determined by the position function defined below.

**Definition 14.** A position function $p_i : S \rightarrow \overline{A}_i$ is a function that assigns a test from $\overline{A}_i$ to solution $s \in S$ in the following way:

$$p_i(s) = \max\{t \in \overline{A}_i | G(s, t)\},$$

(1)
where the maximum is taken with respect to the relation $<$. The test $p_i(s)$ is the position of $s$ on the axis $A_i$.

To give an additional insight into the above definition, we show an important property of a coordinate system. Let $p_i(s) = t$. From the definition of position function $p_i$ as the maximal test $t$ for which $G(s, t)$, it follows immediately that $¬G(s, t_1)$ for each $t_1 > t$. On the other hand, tests on the axis $A_i$ are linearly ordered by the relation $<$, which means that for each $t_1, t_2 \in A_i, t_1 < t_2$ when $SF(t_1) \subseteq SF(t_2)$. Thus, according to the definition of $SF(i)$, $G(s, t_2)$ for each $t_2 < t$. Consequently, if $A_i = \{t_1 < t_2 < \cdots < t_k\}$ is an axis and $p_i(s) = t_j$, we can picture $s$’s placement on $A_i$ in the following way (Bucci et al., 2004):

\[
\begin{array}{cccccccc}
G(s, t) & 1 & 1 & \cdots & 1 & 0 & \cdots & 0 \\
A_i & t_0 & t_1 & \cdots & t_j & t_{j+1} & \cdots & t_k
\end{array}
\]  

(2)

As we can see, according to the position function, $s$ is placed in such a way that for each axis it solves all the tests on its left and fails all the tests on its right.

**Definition 15.** The coordinate system $C$ is correct for a game $G$ iff for all $s_1, s_2 \in S$

\[s_1 \leq s_2 \iff \forall_{i \in I} p_i(s_1) \leq p_i(s_2).\]  

(3)

Basically, this definition means that all relations between solutions in set $S$ have to be preserved by the coordinate system.

Notice also that in a correct coordinate system, $s_1 = s_2$ implies both $\forall_{i \in I} p_i(s_1) \leq p_i(s_2)$ and $\forall_{i \in I} p_i(s_2) \leq p_i(s_1)$, and consequently $\forall_{i \in I} p_i(s_1) = p_i(s_2)$. The proof of the converse implication is analogous. As a result, in the correct coordinate system for all $s_1, s_2 \in S$ we have

\[s_1 = s_2 \iff \forall_{i \in I} p_i(s_1) = p_i(s_2),\]  

(4)

which means that two different solutions never occupy the same position. Also

\[s_1 < s_2 \iff \forall_{i \in I} p_i(s_1) \leq p_i(s_2) \land \exists_{j \in I} p_j(s_1) < p_j(s_2),\]  

(5)

and

\[s_1 \parallel s_2 \iff \exists_{i \in I} p_i(s_1) > p_i(s_2) \land \exists_{j \in I} p_j(s_1) < p_j(s_2).\]  

(6)

Throughout the paper, we will often ask about the relationship between two solutions in a context of a test or some tests; thus the following two definitions will prove useful.

**Definition 16.** Test $t$ orders solution $s_1$ before solution $s_2$, written $s_1 <_t s_2$, if $¬G(s_1, t)$ and $G(s_2, t)$. Similarly, we will use $s_1 =_t s_2$ when $G(s_1, t) = G(s_2, t)$, and $s_1 \leq_t s_2$ when $s_1 <_t s_2$ or $s_1 =_t s_2$.

The definition above resembles the Ficici’s notion of distinctions (Ficici and Pollack, 2001). Notice also that $s_1 <_t s_2$ implies $s_2 \neq s_1$. Obviously, $\forall_{i \in I} s_1 \leq_t s_2 \iff s_1 \leq s_2$.

We will also write $s_1 \leq_C s_2$ to denote that $p_i(s_1) \leq p_i(s_2)$ holds for all $i \in I$ in $C$. Similarly, we will use $s_1 <_C s_2, s_1 =_C s_2$ and $s_1 \parallel_C s_2$.

The following simple proposition will allow us to rewrite the definition of correct coordinate system in an elegant way.

**Proposition 17.** If $C$ is a coordinate system, then for all $s_1, s_2 \in S$

\[s_1 \leq_C s_2 \iff \forall_{i \in \bigcup C} s_1 \leq_i s_2.\]  

(7)
Thus, for any \( t \) \( s_1 \leq_C s_2 \) means \( \forall t \in T p_i(s_1) \leq p_i(s_2) \). Let \( t_1 = p_i(s_1) \) and \( t_2 = p_i(s_2) \); then \( t_1 \leq t_2 \). Def. 14 implies that

\[
\forall t \leq t_1, G(s_1, t) \land \forall t > t_1, \neg G(s_1, t), \tag{8}
\]

and

\[
\forall t \leq t_2, G(s_2, t) \land \forall t > t_2, \neg G(s_2, t). \tag{9}
\]

Consider three possible positions of \( t \) on axis \( A_1 \in C \) with respect to \( t_1 \) and \( t_2 \):

1. \( t \leq t_1 \leq t_2 \) : \( G(s_1, t) \land G(s_2, t) \)
2. \( t_1 \leq t_2 < t \) : \( \neg G(s_1, t) \land \neg G(s_2, t) \)
3. \( t_1 < t \leq t_2 \) : \( \neg G(s_1, t) \land G(s_2, t) \)

Thus, for any \( t \in \bigcup C \), we have \( s_1 \leq t \leq t_2 \).

(\( \Leftarrow \)) Suppose to the contrary that \( \forall t \in \bigcup C s_1 \leq t \leq t_2 \land \neg(s_1 \leq_C s_2) \). The right operand of this conjunction implies \( \exists t \in T p_j(s_1) > p_j(s_2) \). Let \( t_1 = p_j(s_1) \) and \( t_2 = p_j(s_2) \); thus \( t_1 > t_2 \). According to the definition of position function (Def. 14), \( G(s_1, t_1) \). Similarly \( G(s_2, t_2) \), but we know that for any \( t > t_2 \), \( \neg G(s_2, t) \). Since \( t_1 > t_2 \), we have \( \neg G(s_2, t_1) \).

It follows that \( G(s_1, t_1) \land \neg G(s_2, t_1) \), thus \( s_1 > t \), \( s_2 \); since \( t \in \bigcup C \), this contradicts our initial assumption that \( \forall t \in \bigcup C s_1 \leq t_2 \).

The above proposition immediately leads to an alternative definition of correct coordinate system, which is equivalent to Def. 13:

**Definition 18.** The coordinate system \( C \) is correct for a game \( G \) if and only if for all \( s_1, s_2 \in S \)

\[
s_1 \leq s_2 \iff \forall t \in \bigcup C s_1 \leq t \leq s_2. \tag{10}
\]

**Definition 19.** A correct coordinate system \( C \) is a minimal coordinate system for \( G \) if there does not exist any correct coordinate system for \( G \) with smaller size.

**Definition 20.** The dimension \( \dim(G) \) of a game \( G \) is the size of a minimal coordinate system for \( G \).

### 4.1 Example

Let us consider an exemplary test-based problem from (de Jong and Bucci 2008), i.e.,

the misère version of the game of Nim-1-3 with two piles of sticks: one containing a single stick and one containing three sticks. The rules of this game are not important here, but an interested reader is referenced to (de Jong and Bucci 2008).

The payoff matrix of Nim-1-3 is shown in Table 1. There are a total of 144 strategies, but merging indiscernible strategies reduces the number of first player strategies to 6 (candidate solutions \( s_1, \ldots, s_6 \)) and second player strategies to 9 (tests \( t_1, \ldots, t_9 \)).

Figure 2 presents a minimal coordinate system for this game. We can see that the initial set of nine tests was “compressed” to only two underlying objectives represented by axes \( A_1 = \{ t_3 < t_8 < t_2 \} \), \( A_2 = \{ t_4 \} \). First, notice that tests on both axes are placed according to the definition of coordinate system, that is in the order of increasing difficulty (in \( A_1 \), \( t_3 \) is less difficult than \( t_8 \) that is, in turn, less difficult than \( t_2 \)).

Second, the correctness of this coordinate system can be verified by checking whether all relations between pairs of solutions are preserved (see conditions in Definition 15 or Definition 18). For instance, consider a pair \( (s_1, s_2) : s_1 < s_3 \) and \( s_1 \) is
Table 1: The payoff matrix for Nim-1-3. An empty cell means 0.

<table>
<thead>
<tr>
<th></th>
<th>(t_1)</th>
<th>(t_2)</th>
<th>(t_3)</th>
<th>(t_4)</th>
<th>(t_5)</th>
<th>(t_6)</th>
<th>(t_7)</th>
<th>(t_8)</th>
<th>(t_9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s_2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s_3)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(s_4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s_5)</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(s_6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: A minimal coordinate system for Nim-1-3

also dominated by \(s_3\) in the 2D space; on the other hand, \(s_1 \parallel s_6\) (since \(s_1 \prec_{t_8} s_6\) and \(s_6 \prec_{t_4} s_1\)) and \(s_1, s_6\) do not dominate each other also in the figure. Interestingly, only four tests out of nine were required to construct a coordinate system preserving all relations between solutions from \(S\).

Third, solutions are placed with accordance to the position function. Thus, for example, \(s_6\) is placed so that it solves \(t_9\) and \(t_8\), but fails \(t_4\) and \(t_2\), which is consistent with the relations in the original payoff matrix.

With a little effort, one could also check that in this example \(\text{width}(T, \leq) = 3\) and the minimum partition of \((T, \leq)\) consists of the following chains: \((t_9, t_3, t_6), (t_8, t_2, t_5), (t_7, t_1, t_4)\). Also, \(\dim(S, \leq) = 2\), and \(\dim(G) = 2\).

Now, let us consider removing \(t_9\) from the horizontal axis. The resulting formal object is still a coordinate system, as the ordering of remaining tests, required by Def. 13 remains intact. However, it is incorrect, because \(s_1\) shifts to the left and occupies the same position as \(s_5\). This implies \(s_1 = s_5\), which is inconsistent with the payoff matrix.

This helps understand the importance of correctness: an incorrect coordinate system does not reflect the relations between solutions and leads to essential information loss.

In the context of this example, it is also worth emphasizing that the concept of coordinate system brings a new quality to test-based problems when compared to posets \((S, \leq)\) or \((T, \leq)\). Thanks to the position function, a coordinate system explicitly involves both solutions and tests, while a poset describes the relations only within one object category (e.g., solutions), with the other one (here: tests) hidden in relation \(\leq\). By preserving all relations between solutions and tests, a correct coordinate system unequivocally determines the result of interaction between each solution and test on any axis. This information cannot be restored from the posets \((S, \leq)\) and \((T, \leq)\).

Finally, notice that if we had another solution indiscernible with, for instance, \(s_6\) \((S\) is a preordered set), its position would be the same as \(s_6\), and its existence would not change the coordinate system for this game. On the other hand, an additional test
indiscernible with, for instance, \( t_u \) is not needed in any other axis and the coordinate system is correct without it. The latter will be further generalized in Proposition 25.

### 5 Properties of Coordinate Systems

In this section, we prove several facts about the coordinate system defined above. This will allow us to better understand this mathematical object, and, eventually, will help to design algorithms constructing a coordinate system for a given test-based problem.

Let us first observe that the definition of correct coordinate system (Def. 15) does not require all tests from set \( T \) to be used in a coordinate system.

**Definition 21.** Given a game \( G = (S, T, G) \), a coordinate system \( C \) is complete if \( \bigcup C = T \).

As we have seen in Section 4.1, a correct system is not necessarily complete, however the inverse statement holds.

**Proposition 22.** Every complete coordinate system is correct.

**Proof.** If \( C \) is a complete coordinate system, then the condition (10) in Definition 18 is fulfilled, because \( \forall t \in \bigcup C s_1 \leq t s_2 \) implies \( s_1 \leq s_2 \), as \( \bigcup C = T \). Therefore, by Definition 18 \( C \) is correct. \( \square \)

Since not all tests from \( T \) are required to construct a minimal coordinate system, then it is natural to ask which tests are required and which are not. In the following we answer this question by proving that a test \( u \) can be safely removed from a correct coordinate system \( C \) if \( u \) orders only such pairs of solutions that are also ordered by other tests from \( C \); and *vice versa*, \( u \) cannot be safely removed from \( C \) if it is the only test in \( C \) that orders a pair of solutions. This is precisely expressed in the following theorem.

**Theorem 23.** Let \( C = (A_i)_{i \in I} \) be a correct coordinate system. Let \( C' \) be a coordinate system resulting from removing a test \( u \) from some axis in \( C \). \( C' \) is a correct coordinate system iff

\[
\forall s_1, s_2 \in S \ (s_1 <_u s_2 \implies \exists t \in \bigcup C s_1 <_t s_2). \tag{11}
\]

**Proof.** First, observe that \( C' \) is a coordinate system, since after removing \( u \), all axes remain linearly ordered.

(\( \Rightarrow \)) We will prove that if \( C' \) is correct then (11) is satisfied. Suppose to the contrary that \( C' \) is correct and (11) is not satisfied. This means that \( \exists s_1, s_2 \in S \) such that \( s_1 <_u s_2 \land \exists t \in \bigcup C s_1 <_t s_2 \). It follows from \( s_1 < s_2 \) that \( s_1 < s_2 \) or \( s_1 \parallel s_2 \), thus \( s_1 \not< s_2 \). On the other hand, it follows from \( \exists t \in \bigcup C s_1 <_t s_2 \) that \( s_1 \geq C' s_2 \); subsequently, since \( C' \) is correct, \( s_1 \geq s_2 \), which contradicts the earlier statement.

(\( \Leftarrow \)) Assume that (11) is satisfied. By contraposition, we have

\[
\forall s_1, s_2 \in S \ (\neg \exists t \in \bigcup C s_1 <_t s_2 \implies \neg s_1 <_u s_2), \tag{12}
\]

which, after swapping \( s_1 \) with \( s_2 \), boils down to

\[
\forall s_1, s_2 \in S \ (\forall t \in \bigcup C s_1 \leq_t s_2 \implies s_1 \leq_u s_2). \tag{13}
\]

Because \( C \) is correct, it follows from Definition 18 that for all \( s_1, s_2 \in S \)

\[
s_1 \leq s_2 \iff (\forall t \in \bigcup C s_1 \leq_t s_2) \land s_1 \leq u s_2. \tag{14}
\]

Whenever \( s_1 \leq_u s_2 \) is true, it can be ignored in the above condition. Otherwise, by (13), \( \forall t \in \bigcup C s_1 \leq_t s_2 \) must be also false. Therefore, the above reduces to
which, by Definition 18, implies that $C'$ is correct. 

An obvious consequence of the above theorem is following.

**Corollary 24.** A coordinate system $C$ is correct for game $G = (S, T, G)$ if and only if it preserves all relations between the tests in $T$, i.e.,

$$\forall s_1, s_2 \in S \ (\exists t_1 \in T \ s_1 < t_1 s_2 \implies \exists t_2 \in \bigcup C \ s_1 < t_2 s_2).$$

(16)

Let us consider a special case of Theorem 23 when $u \in A_a$ and $u \in A_b$, $a \neq b$. If we remove $u$ from one axis in $C$, $u$ will still remain in the other, thus $u \in C'$ and the condition (11) will hold regardless of the existence of other tests. Thus, Theorem 23 implies the following proposition.

**Proposition 25.** Let $C = (A_i)_{i \in I}$ be a correct coordinate system. If a test $t$ lies on two different axes, i.e., $t \in A_a$ and $t \in A_b$, $a \neq b$, we can remove it from one of them and the coordinate system will remain correct.

A coordinate system that does not contain any test lying on two axes, will be called non-redundant. Notice that removing any test from a coordinate system does not increase its size, which leads to an obvious conclusion:

**Corollary 26.** For any game, there exists a minimal coordinate system that is non-redundant. Thus, in order to find the dimension of $G$, it is enough to search the space of correct non-redundant coordinate systems.

Another important observation is that the size of a minimal coordinate system $C$ is equal to the width of the partially ordered set consisting of all tests of $C$. This is expressed formally as the following theorem

**Theorem 27.** Let $C$ be a minimal coordinate system for $G$. Let $U = \bigcup C$. Then

$$\dim(G) = \text{width}(U, \leq).$$

(17)

**Proof.** By definition of axis, each $(A_i, \leq)$, where $A_i \in C$, $i \in I$, is a chain. $C$ is a minimal coordinate system, so by Proposition 25 we can assume without loss of generality that each test occurs in at most one axis in $C$. Hence, $C$ is a chain partition of $(U, \leq)$. According to Dilworth (Theorem 6), \text{width}(U, \leq) is the number of chains in the minimum chain partition of $(U, \leq)$, thus $|C|$ is at least \text{width}(U, \leq); therefore,

$$|C| = \dim(G) \geq \text{width}(U, \leq).$$

(18)

Now, let $D = (D_i)_{i=1..n}$ be a minimum partition of $(U, \leq)$ into chains, hence, by Def. 18 for all $s_1, s_2 \in S$

$$\forall t \in \bigcup D \ s_1 \leq t s_2 \iff \forall t \in \bigcup C \ s_1 \leq t s_2 \iff s_1 \leq s_2,$$

(19)

because $\bigcup D = U = \bigcup C$. Thus, again by Def. 18 $D$ is a correct coordinate system and, as its size is $\text{width}(U, \leq)$, we get

$$\dim(G) \leq \text{width}(U, \leq).$$

(20)

Combining inequalities (18) and (20) finishes the proof.

Evolutionary Computation Volume x, Number x
Using the same reasoning as in the second part of the above proof, we could also show the correctness of a similar statement concerning correct, but not necessarily minimal, coordinate systems:

**Proposition 28.** Let $C'$ be a correct coordinate system for $G$ and $U' = \bigcup C'$. Then

$$\dim(G) \leq \text{width}(U', \leq).$$

For a brief demonstration, consider the following example. Let $G$ be a test-based problem defined by the following matrix:

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$s_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A minimal coordinate system for this game is $C = \{(t_1 < t_2) \cup (t_3 < t_4)\}$, hence $\dim(G) = \text{width}(\bigcup C) = 2$. The coordinate system $C' = \{(t_1 < t_2) \cup (t_3 < t_4) \cup (t_5)\}$ is also correct, but the poset built from all its positions, i.e., $\{(t_1 < t_2) \cup (t_3 < t_4) \cup (t_5)\}$, has width 3.

Notice that the width of a poset is monotonic regarding its ground set. Thus, we can formulate an obvious remark.

**Remark 29.** Given a poset $(X, P)$, for any $x \in X$

$$\text{width} (X \setminus \{x\}, P) \leq \text{width}(X, P).$$

The above remark together with Theorem 27 lead to the following statement.

**Corollary 30.** In order to determine the dimension of $G = (S, T, G)$ it is enough to find a poset of minimal width whose ground set is a minimal subset of $T$ producing a correct coordinate system.

It is interesting to determine the upper and lower bounds for $\dim(G)$.

**Theorem 31.** For every game $G = (S, T, G)$

$$\dim(S, \leq) \leq \dim(G) \leq \text{width}(T, \leq).$$

**Proof.** First we prove that $\dim(S, \leq) \leq \dim(G)$. Let $C = (A_1, A_2, \ldots, A_n)$ be a minimal coordinate system for $G$. We will construct a family $R = (L_1, L_2, \ldots, L_n)$ of linear orders on $S$, such that $\bigcap R = \leq$, where $\leq$ is the weak dominance relation between elements of $S$ (see Def. 12). Let $L_i$, a linear extension of $\leq$, be defined as:

$$L_i = \{(s_1, s_2) | p_i(s_1) \leq p_i(s_2)\}.$$  

Consider an ordered pair of solutions $(s_1, s_2) \in S \times S$ that is an element of $\bigcap R$. By definition of $R$ we have $(s_1, s_2) \in \bigcap R \iff \forall_i(s_1, s_2) \in L_i$ and the latter is equivalent to $\forall_i p_i(s_1) \leq p_i(s_2)$. Coordinate system $C$ is correct, thus $\forall_i p_i(s_1) \leq p_i(s_2)$ if and only if $s_1 \leq s_2$, which we can write as $(s_1, s_2) \in \leq$. Hence, $(s_1, s_2) \in \bigcap R$ is equivalent to $(s_1, s_2) \in \leq$ and we finally get $\bigcap R = \leq$. Therefore, since $C$ is a minimal coordinate system, it must hold that $\dim(S, \leq) \leq \dim(G)$.

Next, we prove that $\dim(G) \leq \text{width}(T, \leq)$. Let $C = (A_i)_{i \in I}$ be a minimal coordinate system for $G$. Let $U = \bigcup C$. Obviously, $U \subseteq T$, thus $\text{width}(U, \leq) \leq \text{width}(T, \leq)$. By Theorem 27 $\dim(G) = \text{width}(U, \leq)$, hence we have $\dim(G) \leq \text{width}(T, \leq)$.
Unfortunately, the problem of computing the lower bound $\dim(S, \leq)$ is NP-hard (Yannakakis, 1982) and the question whether there exists any polynomial-time algorithm computing a reasonable lower bound of $\dim(G)$ remains open. On the other hand, the problem of determining $\text{width}(T, \leq)$ is easy (see Section 7.2), but, as we will show, better approximations of the upper bound exist (cf. Section 7.3).

5.1 Finite and Infinite Games

Until now, we considered only finite games, i.e., games where $S$ and $T$ were finite. Here we relax this restriction and consider how the notion of coordinate system behaves when $S$ and $T$ are infinite. Let us emphasize that this distinction does not correlate with game complexity as perceived by humans: the game of chess is finite, although the number of strategies for both players is very large. On the other hand, the space of strategies in the conceptually trivial Numbers Games, such as those considered in Section 8, may be infinite (uncountable, in this case).

According to the definition of position function in Section 14, a correct coordinate system is well-defined as long as for each solution in $S$, the $\max$ operator is well-defined. For all countable games, $\max$ will work, but it may not for some uncountable games, when, for example, the set of tests solved by a solution does not have a maximal element. Thus, the definition of coordinate system needs a generalization that will cope with such cases, which should be part of further research.

Bucci et al. (2004) proved that every finite game has a dimension, thus, for every finite $S$ and $T$, there exists a minimal coordinate system. It is not entirely naive to ask whether there exist highly-dimensional games. Could it possibly be the case that $\dim(G) \leq 16$, for all games $G$? The following example settles this issue.

Example 32. Consider a game $G(n) = (S, T, G)$, such that

$$S = (s_i)_{i=1\ldots n}$$
$$T = (t_j)_{j=1\ldots n}$$
$$G(s_i, t_j) \iff i = j$$

Since all the tests are mutually incomparable and each test orders a unique pair of solutions, the dimension of $G(n)$ is $n$.

Thus, for each $n$ there exists a game with dimension $n$. Moreover, if we consider the same example, but assume that $S$ and $T$ are infinite, the dimension of $G$ is not limited by any number, so it does not exists. We will denote such a situation as $\dim(G) = \infty$.

We already know that when $S$ and $T$ are infinite, the game could have no dimension, but are all infinite game dimensionless? In the following example, we show that $G$ can have a dimension even when $S$ and $T$ are infinite.

Example 33. Consider an infinite game $G(n) = (S, T, G)$, defined in the following way:

$$S = (s_i)$$
$$T = (t_j)$$
$$G(s_i, t_j) \iff i \geq j$$

The dimension of this game is 1, because all tests can be placed on one axis $A_1 = (t_1 < t_2 < \ldots)$. Then, $p_1(s_i) = t_i$.

Corollary 34. When a game is finite, it always has a dimension; when a game is infinite, it may, or may not have a dimension.

---

Since we assumed that indiscernible elements do not exist in $T$ and $S$, if a game is infinite, both $T$ and $S$ have to be infinite; otherwise, both have to be finite.
6 Hardness of the Problem

Dimension is an important property of a test-based problem, thus it is natural to ask how hard it is to compute it for a given problem instance. Here we prove that this task is NP-hard. To this aim, let us formally define the Game Dimension Problem.

Problem 35. (Game Dimension Problem) Given a game $G = (S, T, G)$, where $S$ and $T$ are finite and a positive integer $n$, does a correct coordinate system $C$ of size $n$ or less exist for game $G$?

We will prove the NP-completeness of the above problem using the Set Covering Problem as the reference problem.

Problem 36. (Set Covering Problem) Given a universe $U = (u_i)$, a family $R = (R_j)$ of subsets of $U$, a cover is a subfamily $V \subseteq R$ of sets whose union is $U$. Given $U$ and $R$ and an integer $m$, the question is whether there exists a cover of size $m$ or less.

Denote the size of $U$ by $u$ and the size of $R$ by $r$.

The Set Covering Problem is NP-complete, which was proven by [Karp] (1972). Here we slightly narrow its domain by assuming that $\bigcup R = U$ and $u > 2$ and $r > 2$, calling it the Narrowed Set Covering Problem. The Narrowed Set Covering Problem is also NP-complete. First, the limits on $u$ and $r$ do not ease the problem. Second, the answer to the Set Covering Problem with $\bigcup R \subset U$ is no regardless of the value of $m$. It is trivial to check whether $\bigcup R = U$, thus this modification does not change the hardness of the problem, either.

Theorem 37. The Game Dimension Problem is NP-complete.

Proof. Denote our decision problem by $\pi_1$. Let $D_\pi$ denote the domain of problem $\pi$. First, notice that, according to Def. [18] in order to check whether the answer for a certain coordinate system $C$ is yes, we need $O(|S|^2 |T|)$ operations. Clearly then $\pi_1 \in NP$.

Second, we will show that $\pi_2$ is reducible to $\pi_1$ in a polynomial time, where $\pi_2$ is the Narrowed Set Covering Problem. Given an instance $I_2 \in D_{\pi_2}$, we construct $I_1 \in D_{\pi_1}$ in the following way:

1. $n = m + u + r$

2. $S = \{s_0\} \cup A \cup B \cup C \cup D$, where $A = (a_i)_{i=1 \ldots u}$, $B = (b_i)_{i=1 \ldots r}$, $C = (c_i)_{i=1 \ldots u}$, and $D = (d_i)_{i=1 \ldots r}$ are such sets that $|S| = 1 + 2r + 2u$.

3. $T = X \cup Y \cup Z$, where $X = (x_i)_{i=1 \ldots r}$, $Y = (y_i)_{i=1 \ldots u}$, and $Z = (z_i)_{i=1 \ldots r}$, are such sets that $|T| = 2r + u$

4. $G$ is defined as follows:

\[
G(s_0, y_i) \iff u_i \in R_j
\]

\[
(G(a_i, y_j) \iff i \neq j) \quad \text{and} \quad (G(b_i, z_j) \iff i \neq j)
\]

\[
(G(b_i, x_j) \iff i = j) \quad \text{and} \quad (G(c_i, y_j) \iff i = j) \quad \text{and} \quad (G(d_i, z_j) \iff i = j)
\]

It is easy to show that the construction of $I_1$ is limited from above by a polynomial function of the size of $I_2$. The following observations (see Example [39]) should help comprehend the reasons for its design:

1. Notice that $G(a_i, x_j) \iff u_i \in R_j$ means that $A$ corresponds to $U$, $X$ corresponds to $R$ and the payoff submatrix $A \times X$ describes the elements of $R$. 

14 Evolutionary Computation Volume x, Number x
2. In instance $I_1$, all tests from $T$ are mutually incomparable due to the ‘diagonal’ going through $B$, $C$, $D$ and $X$, $Y$, $Z$; therefore any correct coordinate system $C$ for game $G$ will have each of its axes contain exactly one test, thus $n = |C| = |\bigcup C|$.

3. Any correct coordinate system for game $G$ has to contain all elements from sets $Y$ and $Z$, since they are indispensable to make some pairs from set $S$ covered (see Example (39)). Notice that if the submatrix $A \times X$ contained only zeros, then the coordinate system where $\bigcup C = Y \cup Z$ would be correct, since all pairs from $S$ would be ordered. On the other hand, thanks to the intricate construction of the payoff matrix, the only pairs of elements of $S$ which must be ordered with elements of set $X$ are pairs of the form $(s_0, a_i), i = 1 \ldots u$. This will be formally shown later by analyzing all pairs of elements of $S$.

$(I_2 \Rightarrow I_1)$ Suppose that for $I_2 \in D_{e_2}$ the answer is yes. It means that there exists a cover $\mathcal{V} \subseteq \mathcal{R}$ of size $m$ such that $\bigcup \mathcal{V} = \mathcal{U}$. Under this assumption, consider a coordinate system $C$ such that

$$\bigcup C = Y \cup Z \cup \{x_j \in X | R_j \in \mathcal{V}\}.$$  \hfill (27)

We will show that $C$ is correct and its size $n = u + r + m$, thereby proving that the answer to $I_1$ is also yes.

First, notice that $|\bigcup C| = u + r + m$. Thus $n = |C| = |\bigcup C| = u + r + m$.

Second, in order to show that $C$ is correct, we will prove that condition (16) from Corollary (24) is satisfied for all pairs of solutions $s_1, s_2 \in S$. We rewrite the condition below:

$$\exists t_1 \in T s_1 <_t s_2 \implies \exists t_2 \in |C| s_1 <_{t_2} s_2.$$ \hfill (28)

Let us consider all ordered pairs:

$s_0 <_t a_i$ does not hold for any $t \in Y \cup Z$, but it holds for all $t \in Q$ such that $Q = \{x_j | G(a_i, x_j)\} = \{x_j | u_i \in R_j\}$. From $\bigcup \mathcal{V} = \mathcal{U}$ it follows that

$$\forall u_p \in \mathcal{U} \exists R_k \in \mathcal{R} u_p \in R_k \land R_k \in \mathcal{V},$$ \hfill (29)

which in particular is true for our $u_i$, thus we have

$$\exists R_k \in \mathcal{R}, u_i \in R_k \land R_k \in \mathcal{V}.$$ \hfill (30)

And since the set of tests $X$ corresponds to the set $\mathcal{R}$, we get

$$\exists x_k \in X u_i \in R_k \land R_k \in \mathcal{V}.$$ \hfill (31)

The left operand of this conjunction implies that $x_k \in Q$. On the other hand, from the right operand, it follows that $x_k \in \bigcup C$ (by (27)). Replacing the symbol $x_k$ by $t$, we finally get

$$\exists t \in X t \in Q \land t \in \bigcup C.$$ \hfill (32)

Therefore, if $s_0 <_t a_i$ then $t \in \bigcup C$, thus $\exists t \in |C| s_1 <_{t_2} s_2$ and (16) is satisfied.

$s_0 <_t b_i$ holds for $t = x_i$, but it is also true for $t \in \{z_k | k \neq i\}$, which is a non-empty set for $r > 1$; and since $Z \subseteq |C|$, (16) is satisfied.

$s_0 <_t c_i$ never holds, thus it can be ignored.
\[ s_0 <_t d_i \] holds only for \( t = z_i \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ a_i <_t s_0 \] holds only for \( t = y_i \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ a_i <_t a_j \] can hold for \( t = x_k \) for some \( k \), but it is also true for \( t = y_i \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ a_i <_t b_j \] can hold for \( t = x_k \) for some \( k \), but it is also true for \( t \in \{ z_h | h \neq j \} \), which is a non-empty set for \( r > 1 \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ a_i <_t c_j \] holds only for \( i = j \) and \( t = y_i \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ a_i <_t d_j \] holds only for \( t = z_i \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ b_i <_t s_0 \] holds for all \( t \in Y \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ b_i <_t a_j \] can hold for \( t = x_k \) for some \( k \), but it is also true for \( t \in \{ y_k | h \neq j \} \), which is a non-empty set for \( u > 1 \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ b_i <_t b_j \] holds for \( t = x_j \), but it is also true for \( t = z_i \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ b_i <_t c_j \] holds only for \( t = y_j \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ b_i <_t d_j \] holds only for \( i = j \) and \( t = z_i \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ c_i <_t s_0 \] never holds, thus it can be ignored.

\[ c_i <_t a_j \] can hold for \( t = x_k \) for some \( k \), but it also holds for \( t \in \{ y_k | h \neq i, h \neq j \} \), which is a non-empty set for \( u > 2 \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ c_i <_t b_j \] holds for \( t = x_j \), but it also holds for \( t \in \{ z_k | k \neq i \} \), which is a non-empty set for \( r > 1 \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ c_i <_t c_j \] holds only for \( t = y_j \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ c_i <_t d_j \] holds only for \( t = z_j \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ d_i <_t s_0 \] holds for all \( t \in Y \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ d_i <_t a_j \] can hold for \( t = x_k \) for some \( k \), but it is also true for \( t \in \{ y_k | h \neq j \} \), which is a non-empty set for \( u > 1 \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ d_i <_t b_j \] holds for \( t = x_j \), but it also holds for \( t \in \{ z_k | k \neq i, h \neq j \} \), which is a non-empty set for \( r > 2 \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

\[ d_i <_t c_j \] holds only for \( t = y_j \); and since \( Y \subseteq \bigcup C \), (16) is satisfied.

\[ d_i <_t d_j \] holds only for \( t \in \{ z_k | k \neq j \} \); and since \( Z \subseteq \bigcup C \), (16) is satisfied.

It has been shown above that (16) is satisfied for all pairs of tests; therefore, according to Corollary (24), \( C \) is correct, thus the answer for instance \( I_1 \) is yes.

\((I_1 \Rightarrow I_2)\) Suppose that for \( I_1 \in D_{\pi_1} \), the answer is yes. It means that there exists a correct coordinate system \( C \) of size \( n \) for game \( G = (S, T, C) \). Consider a cover \( V = \{ R_x \in \mathcal{R} | x \in \bigcup C \} \), whose size is \( |V| = |\{ x \in X | x \in \bigcup C \} | \). In order to prove that the answer to \( I_2 \) is yes, we will show that \( \bigcup V = \mathcal{U} \) and \( |V| \leq m \), where \( m \) fulfills the equation \( n = m + r + u \).
Example 39. This example shows the construction of game \( G \). Let \( U = \{1, 2, 3, 4\} \), \( R = \{R_1, R_2, R_3\} \), \( R_1 = \{1, 2\} \), \( R_2 = \{1, 3\} \), \( R_3 = \{1, 3, 4\} \). Then we can present the relation \( G \) of game \( G \) graphically (empty cells mean 0):

\[
\begin{array}{cccc}
 & R_1 & R_2 & R_3 \\
1 & 0 & 1 & 1 \\
2 & 1 & 0 & 0 \\
3 & 1 & 0 & 1 \\
4 & 1 & 1 & 0 \\
\end{array}
\]

The Game Dimension Problem is NP-complete, thus:

\textbf{Corollary 38.} Finding a minimal coordinate system for a game is NP-hard.

\textbf{Example 39.} This example shows the construction of game \( G \). Let \( U = \{1, 2, 3, 4\} \), \( R = \{R_1, R_2, R_3\} \), \( R_1 = \{1, 2\} \), \( R_2 = \{1, 3\} \), \( R_3 = \{1, 3, 4\} \). Then we can present the relation \( G \) of game \( G \) graphically (empty cells mean 0):

\[
\begin{array}{cccc}
 & R_1 & R_2 & R_3 \\
1 & 0 & 1 & 1 \\
2 & 1 & 0 & 0 \\
3 & 1 & 0 & 1 \\
4 & 1 & 1 & 0 \\
\end{array}
\]
Notice that most of the pairs to be ordered according to condition (16) from Corollary (24), can be ordered by the elements from either $Y$ or $Z$, for instance, $a_2 <_{y_2} a_3$, $a_4 <_{y_4} s_0$ or $b_1 <_{z_1} b_2$. The only pairs of elements of $S$ that must be ordered by the elements of set $X$ are pairs $(s_0, a_1)$, $(s_0, a_2)$, $(s_0, a_3)$, and $(s_0, a_4)$, which correspond to the universe $U$ of the set covering problem.

7 Algorithms

In this section, we demonstrate three algorithms that construct correct coordinate systems.

7.1 Simple Greedy Heuristic

The first algorithm (here called SIMPLEGREEDY) for coordinate system extraction was given by Bucci et al. (2004). SIMPLEGREEDY finds a correct coordinate system, but it does not guarantee finding a minimal one.

The pseudocode of SIMPLEGREEDY is shown as Algorithm 1. In the first stage, the algorithm removes from $T$ the tests that are combinations of any two other tests in the sense of solutions they fail (lines 2-7). In the second stage, it greedily finds an axis to place a test on, ensuring that at every step the axes remain linearly ordered (line 12). To this aim, it first tries to place a test on an existing axis (lines 11-17); if this is impossible, it creates a new axis (lines 18-20). Tests are considered in the ascending order with respect to the number of solutions they fail (line 9), so that the “best-performing” tests are placed at the end. Note that the poset $(A_i, \leq)$ in line 12 is a non-empty chain, so $\max(A_i, \leq)$ contains exactly one test, which is being compared with $t$.

In (Bucci et al., 2004) there was no formal proof that SIMPLEGREEDY is correct, so we provide it here.

**Proposition 40.** For a given game $G$, SIMPLEGREEDY algorithm produces a correct coordinate system $C$.

**Proof.** Observe that skipping the first stage of Algorithm 1 results in a complete coordinate system, which must be correct by Proposition 22. Thus, it is enough to show that removing a test $t$ such that $SF(t) = SF(t_1) \cup SF(t_2)$, where $t, t_1, t_2$ are distinct and $t, t_1, t_2 \in \bigcup C$, preserves the correctness of $C$. From $SF(t) = SF(t_1) \cup SF(t_2)$ it follows
Algorithm 1 SIMPLEGREEDY heuristic for extracting a minimal coordinate system.

1: procedure SIMPLEGREEDY(S, T, G)
2:   U ← T
3:   for distinct t, t₁, t₂ ∈ U do
4:     if SF(t) = SF(t₁) ∪ SF(t₂) then
5:       T ← T \ {t}
6:   end if
7: end for
8: C ← ∅
9: for t ∈ T sorted ascendingly by |SF(t)| do
10:   found ← false
11:   for Aᵢ ∈ C do
12:     if max((Aᵢ, ≤)) ≤ t then
13:       Aᵢ ← Aᵢ \ {t} ▷ Add t to existing axis
14:       found ← true
15:       break
16:   end for
17:   if !found then
18:     C ← C ∪ {{t}} ▷ Create a new axis
19:   end if
20: end for
21: return C
22: end procedure

that

s ∈ SF(t) ⇒ s ∈ SF(t₁) ∨ s ∈ SF(t₂),  (40)

and

s ∉ SF(t) ⇒ s ∉ SF(t₁) ∧ s ∉ SF(t₂).  (41)

Now, consider a pair s₁, s₂ ∈ S such that s₁ <ₚ s₂; therefore s₁ ∈ SF(t) and s₂ ∉ SF(t). From the former, by (40) and without loss of generality, it follows that s₁ ∈ SF(t₁). On the other hand, the latter implies s₂ ∉ SF(t₁) (by (41)). Therefore, ¬G(s₁, t₁) and G(s₂, t₁), which implies s₁ <ₚ t₁ s₂. Since t₁ ≠ t, test t₁ orders s₁ before s₂. Thus, according to Theorem 23 we can safely remove t and C will remain correct.

To complete the proof, notice that the tests can be removed from C in a top-down order, i.e., we remove test t₁ from the coordinate system C only when there do not exist any t₂, t₃ such that SF(t₁) ∪ SF(t₂) = SF(t₃), where t₁, t₂, t₃ are distinct elements of U \ C; otherwise we remove t₂ first. In this way we guarantee that all tests t such that SF(t) = SF(t₁) ∪ SF(t₂), where t₁, t₂ are distinct elements of T, will be removed, as it is the case in the algorithm.

SIMPLEGREEDY is fast; its worst case time complexity is \(O(|T|^3|S|)\), because of the bottleneck in lines 3-7.

7.2 The Exact Algorithm

In the following we propose an exact algorithm (EXACT for short) that constructs a minimal coordinate system for a given test-based problem and thus determines its
Algorithm 2  EXACT algorithm for extracting a minimal coordinate system.
1: procedure EXACT(S, T, G) 
2:  \( U \leftarrow \text{ALLMINIMALSUBSETS}(T, T) \) \( \triangleright \) Find all minimal correct subsets of \( T \) 
3:  \( C \leftarrow \infty \) 
4:  for \( U \in U \) do \( \triangleright \) Find a poset of minimal width 
5:    \( C' \leftarrow \text{CHAINPARTITION}(U, \leq) \) 
6:    if \( |C'| < |C| \) then 
7:      \( C \leftarrow C' \) 
8:  end if 
9: end for 
10: return \( C \) 
11: end procedure

procedure ALLMINIMALSUBSETS(L, R) 
12:  \( U \leftarrow \emptyset \) 
13:  isLeaf \( \leftarrow \text{true} \) \( \triangleright \) Is \( L \) a leaf in the recursion tree? 
14:  for \( t \in R \) do \( \triangleright \) We visit every subset \( L \) of set \( T \) at most once… 
15:    \( R \leftarrow R \setminus \{t\} \) 
16:    if CANBEMOVED(t, L) then \( \triangleright \)… taking into account correct subsets only 
17:      isLeaf \( \leftarrow \text{false} \) 
18:      \( L \leftarrow L \setminus \{t\} \) 
19:      ALLMINIMALSUBSETS(L, R) 
20:    end if 
21:  end for 
22:  \( L \leftarrow L \cup \{t\} \) 
23:  if isLeaf and \( \exists t \in L \text{CANBEMOVED}(t, L) \) then 
24:    \( U \leftarrow U \cup \{L \} \) \( \triangleright \) \( L \) is a minimal correct subset 
25:  end if 
26:  return \( U \) 
27: end procedure

procedure CANBEMOVED(u, Q) 
28:  return \( \forall s_1, s_2 \in S \ (s_1 < u \ s_2 \Rightarrow \exists t \in Q \setminus u \ s_1 < t \ s_2) \) \( \triangleright \) Condition [11] 
29: end procedure

procedure CHAINPARTITION(X, P) 
30:  return a minimal partition of poset \( (X, P) \) into chains. 
31: end procedure
dimension. As we have proven that unless $N = NP$ there does not exist any exact, polynomial-time algorithm, EXACT has exponential time complexity. Despite this, we wanted it to be as fast as possible, that is why we founded it on three results proved earlier in this paper:

1. Corollary 26 on page 11 which says that it is enough to consider the coordinate systems in which every test appears on at most one axis. We use that result to initialize the search.

2. Theorem 23 on page 10 which determines which tests can be safely removed from $T$.

3. Corollary 30 on page 12 which implies that a minimal coordinate system for $G$ must be among chain partitions of $(L, \leq)$, where $L$ is such a subset of $T$ producing a correct coordinate system that we cannot safely remove any test from $L$.

EXACT is shown in Algorithm 2 and it works as follows. In the first stage (line 2), the algorithm computes $U$, the family of all minimal correct subsets of $T$, i.e., such sets $T' \subseteq T$ that there exists such a correct coordinate system $C$ that $\bigcup C = T'$ and no $T'' \subset T'$ with this property exists. The procedure ALLMINIMALSUBSETS recursively visits the subsets of $T$, and whenever it finds a minimal one, it appends it to $U$ (line 25) and continues the search. Its recursion tree is consistent with subset inclusion, i.e., the recursive calls visit the subsets of the current set. Testing whether $t$ can be removed from $L$ (procedure CANBEREMOVED) relies on Theorem 23. By maintaining the set $R$ of the tests not yet considered for removal at a given level of recurrence (lines 15-16), the procedure never visits any subset of $T$ twice. Recursion returns when no more tests can be removed, so some parts of the search tree are never considered. Note that reaching a leaf of the recursion tree (detected using the isLeaf flag) does not guarantee that $L$ is a minimal correct subset: we still have to check individual elements of $L$ for removal (line 24). This method of visiting all subsets satisfying certain requirements is similar to the one described by de la Banda et al. (2003).

In the second stage, the algorithm computes a chain partition $C$ for every element of $U$, that is, the coordinate system (see the proof of Theorem 27), and returns the one with a minimal number of axes $|C|$. Partitioning a poset $(P, X)$, $|X| = n$ into chains can be solved in $O(n^3)$ (c.f., Mohring 1984; Felsner et al. 2003) using a max-flow computation on a bipartite network with unit capacities. This result can be further improved to $O(n^{5/2})$ with the algorithm by Hopcroft and Karp (1973) and to $O(n^{5/2} \sqrt{\log n})$ by a method introduced by Alt et al. (1991). Recognizing if the number of chains is at most $k$ is even faster: $O(n^k)$ (Felsner et al. 2003).

As the first stage of the algorithm is exponential anyway, we implemented CHAIN-PARTITION using the simplest $O(n^3)$ algorithm. As a consequence, the overall worst-case time complexity of EXACT algorithm is $O(2^{|T|} |T|^4 |S|)$. The consoling fact is that the elements of $U$ can be independently processed one by one, so it is not necessary to maintain all of them simultaneously, which results in polynomial space complexity.

7.3 Greedy Cover Heuristic

The proof of NP-hardness given in Section 6 used the Set Covering Problem as a reference problem. This inspired us to adopt an algorithm designed for this well-known problem, a classical greedy heuristic of polynomial complexity that has very good properties. Starting from an empty set $V$, in each step the heuristic adds to $V$ such an element from the family $R$ that covers the maximal number of not yet covered elements.
Algorithm 3 \textsc{GreedyCover} heuristic for extracting a minimal coordinate system.

1: \textbf{procedure} \textsc{GreedyCover}(S, T, G) \\
2: \hspace{1em} V \leftarrow \emptyset \hspace{1em} \triangleright \text{Working set of tests} \\
3: \hspace{1em} N \leftarrow \{(s_1, s_2) \mid s_1, s_2 \in S \land \exists t \in T s_1 <_t s_2\} \hspace{1em} \triangleright \text{Set of pairs not yet ordered} \\
4: \hspace{1em} \textbf{while} N \neq \emptyset \hspace{1em} \triangleright \text{Are all pairs ordered by tests from } V? \\
5: \hspace{2em} u \leftarrow \arg \max_{t \in T \backslash V} \sum \{(s_1, s_2) \in N \mid s_1 <_t s_2\} \\
6: \hspace{2em} N \leftarrow N \setminus \{(s_1, s_2) \in N \mid s_1 <_u s_2\} \\
7: \hspace{2em} V \leftarrow V \cup \{u\} \\
8: \hspace{1em} \textbf{end while} \\
9: \hspace{1em} \textbf{return} \textsc{ChainPartition}(V, \leq) \\
10: \textbf{end procedure}

from the universe \mathcal{U}. The procedure stops when all elements from the universe \mathcal{U} are covered. This heuristic was shown \cite{Johnson1974} to achieve an approximation ratio of

$$\sum_{k=1}^{s} \frac{1}{k} \leq \ln s + 1,$$  \hspace{1em} (42)

where \(s\) is the size of the largest set in \(R\). It was also proven \cite{Lund1994, Raz1997, Alon2006} to be the best possible polynomial-time approximation algorithm for this problem.

To solve the Game Dimension Problem, we need to search for a set of tests that preserves the relation for all pairs of solutions, i.e., that orders some pairs of solutions. Thus, it is easy to spot similarities between the Set Covering Problem and Game Dimension Problem. This led us to designing the \textsc{GreedyCover} heuristic (Algorithm 3).

The algorithm first uses the classical heuristic for Set Covering Problem to construct a set of tests \(V\) that order all pairs of solutions from set \(S\). Then, it computes the minimal chain partition on \((V, \leq)\).

The correctness of this heuristic results directly from Theorem 23 and the fact that elements in a chain are ordered by the \(<\) relation. Assuming that a minimal chain partition is computed using the simplest \(O(n^3)\) algorithm (this can be improved, see discussion in 7.2), \textsc{GreedyCover} has a worst case polynomial time complexity of \(O(|T|^2|S|^2 + |T|^3)\), because the loop in line 4 executes maximally \(|T|\) times and the cost of line 5 is \(O(|T||S|^2)\).

\textsc{GreedyCover} has an unknown approximation ratio (if any), but it is based on an algorithm that has low approximation ratio, which makes us hypothesize that it will perform well in practice. This intuition will be verified in the experiments in the following section.

8 Experiments

8.1 Dimension of a random test-based problem

In the first experiment, we compared Bucci’s \textsc{SimpleGreedy} with our \textsc{Exact} and \textsc{GreedyCover} algorithms on random payoff matrices of different sizes \(n = 2 \ldots 200\). A problem of size \(n\) is given by a random payoff matrix \(n \times n\) (\(n\) tests and \(n\) solutions), with each interaction outcome drawn at random with equal probability. The dimensions computed by both algorithms are shown in Fig. 3. Each data point represents the average for 30 random matrices with 95% confidence intervals.
Figure 3: The figure shows the comparison between the dimension computed by three algorithms: SIMPLEGREEDY, EXACT, GREEDYCOVER on a random problem in the function of the problem size. Clearly, SIMPLEGREEDY is inferior to both EXACT and GREEDYCOVER. On the other hand, GREEDYCOVER, which is only a heuristic, perform nearly as well as EXACT. Grey vertical bars denote 95% confidence intervals.

Figure 3 gives rise to several interesting observations. EXACT performs clearly much better than SIMPLEGREEDY, which hardly does any compression. The gap between the algorithms grows rapidly with $n$, so that SIMPLEGREEDY overestimates twice the true dimension computed by EXACT already for $n = 25$ and at least ten times for $n = 200$ (the latter case did not fit in the figure). The compression provided by EXACT is impressive, given that incomparability of almost all pairs of test becomes nearly certain for large random matrices, since the probability that a test weakly dominates another test is $\left(\frac{3}{4}\right)^n$. Note also that the logarithm-like shape of EXACT curve suggests, that the compression could be even higher for larger $n$. On the other hand, EXACT is obviously much slower than SIMPLEGREEDY — that is why we could not produce results for $n > 34$.

However, in contrast to SIMPLEGREEDY, the GREEDYCOVER heuristic seems to approximate the true problem dimension very well. For small instances ($n \leq 14$), it is not significantly worse than EXACT. For larger instances, for which we have the exact results, it overestimates the game dimension only slightly.

The dimension computed by GREEDYCOVER seems to follow a logarithmic curve, similarly to the actual problem dimension (as computed using EXACT). Since the result of GREEDYCOVER is an upper bound of problem dimension, we may hypothesize that, on average, the dimension of a test-based problem is bounded from above by a logarithm of problem size. This result may indicate that the dimension of at least some test-based problems is possible to handle and that the compression obtained by

---

Footnote:

The probability of the event $t_1 \leq s_1$, $t_2$ is $\frac{3}{4}$, because out of four possible cases, it does not occur only when $G(s, t_1) = 1$ and $G(s, t_2) = 0$. 

---

Evolutionary Computation Volume x, Number x
extracting underlying objectives may be exponential with respect to the total number of tests.

8.2 Compare-on-one

In the second experiment, we considered COMPARE-ON-ONE, an abstract game defined by de Jong and Pollack (2004), which is a variant of the Numbers Game (Watson and Pollack, 2001) widely used as a coevolutionary benchmark (Bucci et al., 2004; de Jong, 2005, 2007; Service and Tauritz, 2008). In this game, strategies are represented as non-negative real number vectors of length \( d \), which we call here the *a priori dimension* of the game. The outcome of the interaction between a solution \( s \) and a test \( t \) depends only on the dimension in which \( t \) has the highest value. Formally, first the index \( m \) to be used for comparison is defined as

\[
m = \arg \max_{i=1 \ldots d} t[i],
\]

where \( t[i] \) denotes the \( i \)-th element of vector \( t \). Then,

\[
G(s, t) \iff s[m] \geq t[m].
\]

(43)

The rules of the two-dimensional version of COMPARE-ON-ONE are visualized in Figure 4. Solution \( s_1 \) solves only the tests from the shaded area.

Figure 4: A visualization of rules of two dimensional COMPARE-ON-ONE \((d = 2)\). Here, each strategy (a solution or a test) is represented as a point in a 2D-space. According to the game definition, solution \( s_1 \) solves all tests from the gray area, thus it solves \( t_1 \), but it fails \( t_2 \).

Despite its straightforward formulation, COMPARE-ON-ONE is challenging because it has been designed to induce over-specialization: a co-evolving system of solutions and tests can easily focus on some (or even a single) underlying objectives, while ignoring the remaining ones. To make a steady progress on this problem, a coevolutionary algorithm should carefully maintain the tests that support all the underlying objectives from the very beginning of the run.

Here, we investigated how, for a given *a priori* dimension \( d \), the game dimension (computed with EXACT) changes for growing \( S \) and \( T \), \(|S| = |T| = n\), by randomly generating \( n \) test strategies and \( n \) solution strategies from a fixed \([0.0, 10.0]\) interval. Other settings of this experiment were the same as in the previous one.

The results for \( d = 2, 3, 4 \) are shown in Figure 5. The plot clearly indicates that the computed dimension of the game converges to the *a priori* dimension \( d \) of the game with growing \( n \). Also, for this game, the dimension may be reliably estimated already from a small number of interactions.

The above results made us wonder what the minimal coordinate system for this game looks like. It is easy to notice that \( t[0, 1] \) is indiscernible to \( t[0.5, 1] \) with respect
to \( S \), since only the highest value counts. Generally, \( t[a_1, \ldots, a_m, \ldots, a_d] \) is indiscernible to \( t[b_1, \ldots, b_m, \ldots, b_d] \), when \( a_m = b_m \) and \( \forall i \neq m, a_i \leq a_m \land b_i \leq b_m \). Thus, the set of tests of the form \( t[0, \ldots, a_m, \ldots, 0] \), where \( a_m > 0 \) is sufficient to construct the minimal coordinate system. The observation that \( t[0, \ldots, a_m, \ldots, 0] < t[0, \ldots, b_m, \ldots, 0] \) if and only if \( a_m < b_m \) makes it possible to define the correct coordinate system:

\[
A_i = \{ [0, \ldots, a_i, \ldots, 0], a_i \in \mathbb{R}^+ \}, \tag{44}
\]

for \( i = 1 \ldots d \), and

\[
p_i(s[s_1, \ldots, s_i, \ldots, s_d]) = t[0, \ldots, s_i, \ldots, 0]. \tag{45}
\]

Figure 6 shows the minimal coordinate system for \( d = 2 \). The tests solved by an exemplary solution \( s_1 \) are covered by a gray line.
Figure 7: A visualization of rules of two-dimensional COMPARE-ON-ALL (d = 2). Both tests and solutions are points in the d-dimensional space. s1 solves all tests from the gray area.

In most trials, the minimal coordinate systems found by EXACT were coherent with the minimal coordinate system described above, so they correctly identified the d underlying objectives of the game. Only when the number of solutions and tests n was small in proportion to d, EXACT happened to produce axes that did not correspond to such objectives.

8.3 Compare-on-all
In the last experiment, we examine another abstract game, the COMPARE-ON-ALL (de Jong and Pollack, 2004) (a.k.a. TRANSITIVE (Bucci et al., 2004)). Strategies are represented like in COMPARE-ON-ONE, but the interaction function is defined as a weak dominance relation:

\[ G(s, t) \iff \forall i s[i] \geq t[i]. \] (46)

The rules of COMPARE-ON-ALL are visualized in Figure 7.

The results (computed with EXACT) for this problem for \( d = 2, 3, 4 \) are shown in Fig. 8. Although the number of dimensions grows much slower than the problem size, this time the computed dimension clearly fails at approximating the a priori game dimension d. Even worse, it does not seem to saturate with growing n. To some extent, this corresponds to the results obtained by Bucci et al. (2004), who, using SIMPLE-GREEDY heuristic and a variant of Population Pareto Hill Climber (P-PHC) algorithm for generating solutions and tests, found out that the computed dimension was overestimating the a priori dimension, especially for large values of the latter one.

Our result demonstrates that even using an exact algorithm, it is hard to get the true dimension of this game. Let us also notice that in Bucci et al. (2004) the scale on overestimation was much lower than in our experiment (e.g., for \( d = 10 \), the game dimension was estimated to only ca. 17). The two experiments are not easily comparable because of the way in which the solutions and tests in Bucci et al. (2004) were generated (P-PHC); however, taking into the consideration the fact that the real dimension of COMPARE-ON-ALL equals its a priori dimension (see below), the better estimate found in Bucci et al. (2004) indicates that the properties of generators are of crucial importance when designing practical coevolutionary algorithms.

The minimal coordinate system for COMPARE-ON-ALL looks as the one for COMPARE-ON-ONE. Also here we need only the tests of the form \( t[0, \ldots, a_m, \ldots, 0] \), where \( a_m > 0 \): any test having more than one non-zero element is superfluous for the coordinate system because its solutions-failed set can be constructed using solutions-
Extracting Internal Structure of Test-Based Problems

Figure 8: The figure shows how the dimension for COMPARE-ON-ALL changes when increasing the number of strategies involved. Notice that for $d = 2, 3, 4$, the computed dimension does not converge to the a priori dimension of the game. Grey vertical bars denote 95% confidence intervals.

failed sets of the tests located on the axes (cf. the proof of Proposition 40 on page 18). Therefore, since also here $t[0, \ldots, a_m, \ldots, 0] < t[0, \ldots, b_m, \ldots, 0]$ if and only if $a_m < b_m$ and this relation cannot be modeled in a lower number of dimensions, the minimal coordinate system is defined by Equations 44 and 45 as for COMPARE-ON-ONE and has $d$ dimensions.

9 Relation to Complete Evaluation Set

It is interesting to notice the relation between our findings and the idea of ideal evaluation introduced by [de Jong and Pollack, 2004] which was based on a concept of complete evaluation set. The complete evaluation set is defined as a subset of $T$ that preserves all relations between solutions from $S$. Notice that the set of all tests $\bigcup C$ of a correct coordinate system is a complete evaluation set, since the condition (24) holds. By transforming the problem of determining the complete evaluation set to the Set Covering Problem, it is easy to show that also the problem of determining the minimal complete evaluation set is NP-hard.

The complete evaluation set is used in the coevolutionary algorithm DELPHI introduced in [de Jong and Pollack, 2004]. Although the authors state that DELPHI does not need to compute the minimal complete evaluation set, it is reasonable to hypothesize that approximating it could lead to better performance of DELPHI by decreasing the number of tests it has to maintain. Employing a variant of GREEDYCOVER that computes just the complete evaluation set may thus be beneficial also for DELPHI, opening the doors to practical applications of the results obtained in this paper.
10 Conclusions

Test-based problems form an important and surprisingly common class of optimization problems, which was not recognized as a separate branch of research until recently, and can be conveniently modeled and studied within the framework of coevolution. They are difficult by nature, and thus require a proper formal analysis that was the main aim of this paper. We concentrated on the notion of coordinate system, a concept introduced in [5], that allows extracting the internal structure of test-based problems by means of underlying objectives, which can be potentially exploited to maintain the progress of search in coevolutionary algorithms.

A significant part of this paper was devoted to revealing the properties of coordinate systems. Apart from determining the lower and upper bounds for problem dimension, we proved its equality to the width of the partially ordered set of tests. Moreover, we formally identified the tests that are redundant and can be safely discarded when constructing a coordinate system.

These findings allowed us to answer the question about the complexity of the problem of extracting a minimal coordinate system. Despite the fact that the problem turned out to be NP-hard, we demonstrated that it can be solved at low computational cost by means of an appropriate heuristic. Our GreedyCover algorithm is clearly superior to the SimpleGreedy, the best algorithm proposed so far in terms of approximating the true problem dimension, and similar to it in terms of computational complexity. Additionally, we carefully designed an exact algorithm which, though exponential, may be used for the problems of moderate size.

In the experimental part, we have shown that application of these algorithms leads to significant compression of the objective space of test-based problems. We demonstrated that, on average, the number of underlying objectives for an abstract random game seems to be limited from above by a logarithm of the number of tests. This indicates that the true dimension of the search space of at least some test-based problems is typically much lower than that resulting from the original idea of Pareto-coevolution that treats every test as a separate objective. If we assume the intuitive hypothesis that problem dimension is a yardstick of problem difficulty, we can conclude that test-based problems, though still hard, are less difficult than previously thought. In case of some problems (here: Compare-on-one game), we have shown that the axes of extracted coordinate systems correctly identify the true underlying skills of the game. On another problem, Compare-on-all, we have demonstrated that sometimes, even with an exact algorithm, the true dimension of a game may not be found if only samples of tests and solutions are provided. This indicates the importance of the design of generators of tests and solutions in the coevolutionary algorithms based on the idea of coordinate system.

This study focused on modeling the underlying structure of the problem; how to effectively exploit it remains an open research question. Polynomial-time complexity and good results of GreedyCover make it particularly appealing for coevolutionary algorithms, enabling to update the coordinate system online, possibly after each generation. The algorithm could be conveniently embedded into a coevolutionary framework as a part of an archive, a repository that serves as a memory for the search process by storing selected candidate solutions and tests. An archive is an indispensable component of any modern coevolutionary algorithm [9,10,11], preventing (or lessening the risk of) pathologies and providing it with gradient (i.e., maintaining desired differentiation within both populations). Solutions and tests from both populations are confronted...
with those from the archive, typically each to each, so keeping the archive small is essential to avoid excessive computational cost. The crucial question that has to be answered after each generation becomes: which solutions and test should be preserved in the archive and which can be safely discarded without forsaking monotonic progress towards a chosen solution concept (Ficici, 2004)? The proposed heuristic can serve as an oracle that allows answering this question at relatively low computational expense. Construction of such an algorithm and its experimental assessment on various problems is a natural next step we envision for this research.

Acknowledgments

This work was supported in part by Ministry of Science and Higher Education grant # N N519 441939. Wojciech Jaśkowski gratefully acknowledges a financial support from grant # N N516 188337.

References


