

Formal Analysis and Algorithms for Extracting Coordinate Systems of Games

Wojciech Jaśkowski and Krzysztof Krawiec

Abstract—A two-player game given in the normal form of payoff matrix may be alternatively viewed as a list of the outcomes of binary interactions between two sets of entities, solutions and tests. The internal structure of such interactions may be characterized by an appropriately constructed coordinate system, which spatially arranges the solutions with respect to coordinates identified with tests, while preserving their mutual relations as given by the matrix. Of particular interest are coordinate systems of minimal size that give rise to the notion of dimension of a game. Following [1], we investigate such coordinate systems and relate their features to properties of partially ordered sets (posets), mostly to poset width and poset dimension. We propose an exact algorithm for constructing a minimal correct coordinate system and prove its correctness. In the experimental part, we compare the exact algorithm to the heuristics proposed in [1] on a sample of random payoff matrices of different sizes to demonstrate that the heuristics heavily overestimates the size of the minimal coordinate system. Finally, we show how the game dimension relate to the *a priori* dimension of a game.

I. INTRODUCTION

It is quite common in computational intelligence that system's change (learning, optimization) is driven by the outcomes of interactions between some elementary entities. Game-playing agents often learn by playing against opponent strategies. Machine learning algorithms generate hypotheses and test them on training examples. Evolutionary algorithms guide the search by simulating the evolved designs in various environmental conditions. What these scenarios have in common is the underlying concept of a binary interaction between a *solution* (strategy, hypothesis, design) and a *test* (opponent strategy, training example, environmental conditions, respectively).

Typically, a single interaction is not informative enough to explain the underlying phenomenon or, in particular, to guide the system's change. Thus, the outcomes of individual interactions are usually aggregated, leading to an overall performance measure, like winning probability, accuracy of classification, or fitness in the above examples. In this way, however, the individual characteristics of particular solutions and tests are inevitably lost. Also, using a fixed set of tests brings the risk of biasing the method, known as overfitting in machine learning. Recent results in coevolutionary algorithms show that this has not have to be the case: it is possible to infer and model the intrinsic structure of a set of interactions by analyzing their mutual relations. There is growing evidence that such structure may be exploited for the

benefit of the learning method, for instance by accelerating its convergence.

The form of structurization that seems to be especially appealing relies on the concept of a *coordinate system*. This paper we elaborate on the coordinate system defined by Bucci [1], revisited in Section IV. Our major contribution consists in identifying some of its properties and providing an exact algorithm for creation of a minimal coordinate system, presented in Section V. To this aim, we employ the powerful formalism of posets [2], introduced in Section II, and formalize the notion of game used in this paper in Section III.

II. MATHEMATICAL BACKGROUND

Definition 1: *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 $\max(X, P)$, while by $\min(X, P)$ the set of all minimal elements.

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

Theorem 1: (Dilworth [3]) If (X, P) is a poset and $\text{width}(X, P) = n$, then there exists a partition of $X = C_1 \cup C_2 \cup \dots \cup C_n$, where C_i is a chain for $i = 1, 2, \dots, n$. We call it *minimum chain partition*, as it comprises the smallest possible number of chains.

Note that an important consequence of Dilworth theorem is that each C_i contains exactly one element of the maximum

W. Jaśkowski and K. Krawiec are with the Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60965 Poznań, Poland; email: {wjaskowski, kkrawiec}@cs.put.poznan.pl.

antichain.

Definition 6: Given a poset (X, P) , a *linear extension* of P is any superset of P that is a linear order on X .

Definition 7: The *dimension* of a poset (X, P) , denoted $\dim(X, P)$, is the smallest cardinal number¹ t for which there exists a family $\mathcal{R} = \{L_1, L_2, \dots, L_t\}$ of linear extensions of P so that $P = \bigcap \mathcal{R} = \bigcap_{i=1}^t L_i$.

Example 1: Consider a poset (X, P) with $X = \{a, b, c, d, e, f\}$ and $P = \{(a, c), (a, d), (b, d), (b, e), (c, f), (d, f), (a, f), (b, f)\} \cup \{(x, x) : x \in X\}$. Some of its elements are incomparable (e.g., $c \parallel d$), thus it is not a linear order. Examples of chains in (X, P) are $a < c$ and $b < d < f$. $\{a, e\}$ is an example of antichain, and $\{c, d, e\}$ is a maximum antichain of this poset, thus $\text{width}(X, P) = 3$. However, $\dim(X, P) = 2$, 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.

III. BASIC DEFINITIONS

Definition 8: In this paper by *game* we will mean a formal object $\mathcal{G} = (S, T, g)$ that consists of a finite set S of *solutions* (a.k.a. *candidate solutions*, e.g., [5]), a set T of *tests*, and an *interaction function* $g : S \times T \rightarrow \mathbb{R}$. We limit ourselves 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, we say that s *fails* test t .

Notice that in the perspective of game theory, g can be interpreted as a payoff matrix, and S, T as sets of strategies. For simplicity, we assume that both S and T are finite (\mathcal{G} is a finite game).

Definition 9: *Solutions failed set* $SF(t) \subseteq S$ comprises of all solutions that fail the test t . Dually, *tests solved set* $TS(s) \subseteq T$ comprises 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 10: 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: $\nexists t_1, t_2 \in T, t_1 \neq t_2 : SF(t_1) = SF(t_2)$ and $\nexists s_1, s_2 \in S, s_1 \neq s_2 : TS(s_1) = TS(s_2)$. 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 what 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 \mathcal{G} .

¹Here we follow the original paper by Dushnick [4], since [2] requires the dimension of poset to be a positive integer.

IV. COORDINATE SYSTEM

In the context of games, a coordinate system is a formal concept that enables the solutions (and sometimes also the 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 designing better coevolutionary algorithms [6] and examining properties of certain problems and games [7].

There is no unique definition of coordinate system for a game; currently we are aware of two formulations: by Bucci *et al.* [1] and by de Jong and Bucci [6], [7]. When investigating these and other definitions of coordinate systems, there are several aspects that should be considered: i) size of minimal coordinate system for various games; ii) computational complexity of the algorithm that builds the minimal coordinate system from interactions; iii) the existence of efficient heuristics for that purpose.

In this paper we try to analyze the historically first definition of coordinate system introduced by Bucci *et al.* in [1], so in 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 the coordinate system. First, Bucci *et al.* worked with preordered sets, but we, as pointed out earlier, limit our discussion to posets. Second, in our formulation, the positions of solutions with respect to an axis are shifted one test to the left, which is more convenient.

For convenience, we introduce a formal element t_0 such that $g(s, t_0) = 1$ for all $s \in S$. Also, we define an operator ‘overline’ that augments a set of tests with t_0 , i.e., $\bar{X} = X \cup \{t_0\}$.

Definition 11: The *coordinate system* \mathcal{C} for a game \mathcal{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 $|\mathcal{C}|$, is the cardinality of I .

Definition 12: *Position function* $p_i : S \rightarrow \bar{A}_i$ is a function that assigns a test from \bar{A}_i to solution $s \in S$ in the following way:

$$p_i(s) = \max\{t \in \bar{A}_i \mid g(s, t) = 1\},$$

where the maximum is taken with respect to the relation $<$. The test $p_i(s)$ is the *position* of s on the axis \bar{A}_i .

Definition 13: The coordinate system \mathcal{C} is *correct* for a game \mathcal{G} iff for all $s_1, s_2 \in S$

$$s_1 \leq s_2 \iff \forall_i p_i(s_1) \leq p_i(s_2)$$

Lemma 1: Notice that in a correct coordinate system $s_1 = s_2$ implies both $\forall_i p_i(s_1) \leq p_i(s_2)$ and $\forall_i p_i(s_2) \leq p_i(s_1)$, and consequently $\forall_i p_i(s_1) = p_i(s_2)$. The proof of converse implication is similar. In result, in the correct coordinate system for all $s_1, s_2 \in S$ we have

$$s_1 = s_2 \iff \forall_i p_i(s_1) = p_i(s_2). \quad (1)$$

This also leads to

$$s_1 < s_2 \iff \forall_{i \in I} p_i(s_1) \leq p_i(s_2) \wedge \exists_{j \in I} p_j(s_1) < p_j(s_2). \quad (2)$$

To make the notation more convenient we will write $s_1 \leq_C s_2$ to denote that $p_i(s_1) \leq p_i(s_2)$ holds for $i \in I$ in \mathcal{C} . Similarly, we will use $s_1 <_C s_2$, $s_1 =_C s_2$ and $s_1 \parallel_C s_2$.

Let $p_i(s) = t$. From the definition of position function p_i as the maximal test t for which $g(s, t) = 1$, it follows immediately that $g(s, t_1) = 0$ 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 $SF(t_1) \subset SF(t_2)$. Thus, according to the definition of $SF(t)$, $g(s, t_2) = 1$ for each $t_2 < t$. Consequently, if $A_i = \{t_1 < t_2 < \dots < t_{k_i}\}$ is an axis and $p_i(s) = t_j$, we can picture s 's placement on A_i in the following way ([1]):

$$\begin{array}{cccccccc} g(s, t) & 1 & 1 & \dots & 1 & 0 & \dots & 0 \\ \bar{A}_i & t_0 & t_1 & \dots & t_j & t_{j+1} & \dots & t_{k_i} \end{array}$$

Definition 14: A correct coordinate system \mathcal{C} is a *minimum coordinate system* for \mathcal{G} if there does not exist any correct coordinate system for \mathcal{G} with smaller size.

Definition 15: The *dimension* $\dim_B(\mathcal{G})$ of a game \mathcal{G} is the size of the minimum coordinate system for \mathcal{G} .

A. Example

Let us consider an exemplary game from [7], i.e., the misère version of game of Nim [1 3] with two piles of sticks: one containing a single stick and one containing three sticks. The exact rules of this game are not important here, but an interested reader is referenced to [7].

Table I
THE PAYOFF MATRIX FOR NIM [1 3]. AN EMPTY CELL MEANS 0.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
s_1	1		1	1		1	1		1
s_2									
s_3	1	1	1	1	1	1	1	1	1
s_4	1	1	1				1	1	1
s_5	1			1			1		
s_6							1	1	1

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

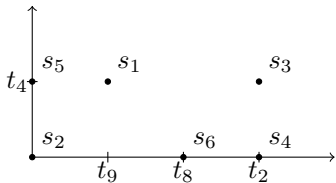


Figure 1. The minimal coordinate system for the game shown in Table I

Figure 1 presents the minimal coordinate system for this game. The initial set of nine tests was compressed to only two underlying objectives (axes) composed of only four tests. Note also, that $\text{width}(T, \leq) = 3$ and the poset partition consists of the following chains: (t_9, t_3, t_6) , (t_8, t_2, t_5) , (t_7, t_1, t_4) . On the other hand, $\dim(S, \leq) = 2$.

V. PROPERTIES OF COORDINATE SYSTEM

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 construct algorithms constructing coordinate system for a given game.

Let us observe that the Definition 13 does not state that all tests from T have to be used somewhere in \mathcal{C} .

Definition 16: Coordinate system $\mathcal{C} = (A_i)_{i \in I}$ system is *complete* if $\bigcup_{i \in I} A_i = T$.

Neither a correct coordinate system nor a minimum one does have to be complete. However, see the following proposition.

Proposition 1: Every complete coordinate system is correct.

Proof: Let us use a more compact notation of $t_1^i = p_i(s_1)$ and $t_2^i = p_i(s_2)$. We have to prove that $s_1 \leq s_2 \iff \forall_{i \in I} t_1^i \leq t_2^i$ for all $s_1, s_2 \in S$.

(\Leftarrow) Assume $\forall_{i \in I} t_1^i \leq t_2^i$. Let us consider an axis A_i , containing both t_1^i and t_2^i . We know that s_1 solves all tests $t \in A_i$ such that $t \leq t_1^i$ and s_2 solves all tests $t \in A_i$ such that $t \leq t_2^i$. Since the coordinate system is complete, $\bigcup_{i \in I} \{t \in A_i | t \leq t_1^i\} = TS(s_1)$ and, similarly, $\bigcup_{i \in I} \{t \in A_i | t \leq t_2^i\} = TS(s_2)$. $t_1^i \leq t_2^i$ implies that all tests on axis T_i that are solved by s_2 are also solved by s_1 and this argument holds for any $i \in I$. Thus $TS(s_1) \subseteq TS(s_2)$; therefore, by the definition, $s_1 \leq s_2$, as we set out to show.

(\Rightarrow) Assume to the contrary that $s_1 \leq s_2$ and it is false that $\forall_{i \in I} t_1^i \leq t_2^i$. Thus, there exists such axis A_i that $t_2^i < t_1^i$. It follows that t_1^i is solved by s_1 , but it is not solved by s_2 . However, from $s_1 \leq s_2$ we know that $TS(s_1) \subseteq TS(s_2)$, thus if t_1^i is solved by s_1 then it is also solved by s_2 for all $s_1, s_2 \in S$. The contradiction completes the proof. ■

Proposition 2: Let $\mathcal{C} = (A_i)_{i \in I}$ 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 the axes and the coordinate system will remain correct.

Proof: Let \mathcal{C}' be a coordinate system resulting from removing t from A_a and p'_i be a position function after removing t from A_a . First, observe that \mathcal{C}' is a coordinate system, since $A_a \setminus \{t\}$ is totally ordered as A_a is. Now, we will show that \mathcal{C}' is correct.

Let $\bar{A}_a = \{t_0 < \dots < t' < t < \dots < t_{k_a}\}$. Removing t from A_a will produce the following definition of p'_i :

$$p'_i(s) = \begin{cases} t' & i = a \wedge p_i(s) = t \\ p_i(s) & \text{otherwise} \end{cases} \quad (3)$$

\mathcal{C} is correct, thus $s_1 \leq s_2 \iff \forall_i p_i(s_1) \leq p_i(s_2)$ for all $s_1, s_2 \in S$. To prove that \mathcal{C}' is correct we have to show that $s_1 \leq s_2 \iff \forall_i p'_i(s_1) \leq p'_i(s_2)$ for all $s_1, s_2 \in S$.

(\Rightarrow) Suppose that $s_1 \leq s_2$. Thus, $\forall_i p_i(s_1) \leq p_i(s_2)$ and from (3) immediately follows that $\forall_{i \neq a} p'_i(s_1) \leq p'_i(s_2)$ and it remains to prove that $p'_a(s_1) \leq p'_a(s_2)$. Having in mind (3) and that $p_a(s_1) \leq p_a(s_2)$, consider four possible cases for $p_a(s_1)$ and $p_a(s_2)$:

- 1) $p_a(s_1) \neq t$ and $p_a(s_2) \neq t$. Thus $p'_a(s_1) = p_a(s_1)$ and $p'_a(s_2) = p_a(s_2)$. So from $p_a(s_1) \leq p_a(s_2)$ it follows $p'_a(s_1) \leq p'_a(s_2)$.
- 2) $p_a(s_1) = t$ and $t < p_a(s_2)$. Thus $p'_a(s_1) = t'$ and $p'_a(s_2) = p_a(s_2)$. From $p'_a(s_1) = t' < t < p'_a(s_2)$ it follows $p'_a(s_1) < p'_a(s_2)$.
- 3) $p_a(s_1) < t$ and $p_a(s_2) = t$. Thus $p'_a(s_1) = p_a(s_1) < t$, so $p'_a(s_1) \leq t'$ and $p'_a(s_2) = t'$; that implies $p'_a(s_1) \leq p'_a(s_2)$.
- 4) $p_a(s_1) = t$ and $p_a(s_2) = t$. Thus $p'_a(s_1) = t'$ and $p'_a(s_2) = t'$, so $p'_a(s_1) = p'_a(s_2)$.

In all cases $p'_a(s_1) \leq p'_a(s_2)$, therefore, $s_1 \leq s_2$ implies $\forall_i p'_i(s_1) \leq p'_i(s_2)$.

(\Rightarrow) Suppose that $\forall_i p'_i(s_1) \leq p'_i(s_2)$. Thus, from (3) we have $\forall_{i \neq a} p_i(s_1) \leq p_i(s_2)$ and it remains to prove that $p_a(s_1) \leq p_a(s_2)$. Having in mind (3) and that $p'_a(s_1) \leq p'_a(s_2)$, consider four possible cases for $p'_a(s_1)$ and $p'_a(s_2)$:

- 1) $p'_a(s_1) \neq t'$ and $p'_a(s_2) \neq t'$. Thus $p_a(s_1) = p'_a(s_1)$ and $p_a(s_2) = p'_a(s_2)$. So from $p'_a(s_1) \leq p'_a(s_2)$ it follows $p_a(s_1) \leq p_a(s_2)$.
- 2) $p'_a(s_1) = t'$ and $t' < p'_a(s_2)$. Thus $p_a(s_1) \in \{t' < t\}$ and $p_a(s_2) = p'_a(s_2)$. From $t' < p_a(s_2)$ it follows that $t \leq p_a(s_2)$. Thus, $p_a(s_1) \leq p_a(s_2)$.
- 3) $p'_a(s_1) < t'$ and $p'_a(s_2) = t'$. Thus $p_a(s_1) = p'_a(s_1)$ and $p_a(s_2) \in \{t' < t\}$. From $p_a(s_1) < t'$ and $t' \leq p_a(s_2)$, it follows $p_a(s_1) < p_a(s_2)$.
- 4) $p'_a(s_1) = t'$ and $p'_a(s_2) = t'$. There are four subcases:
 - a) $p_a(s_1) = t' \wedge p_a(s_2) = t'$. Thus, $p_a(s_1) = p_a(s_2)$.
 - b) $p_a(s_1) = t \wedge p_a(s_2) = t$. Thus, $p_a(s_1) = p_a(s_2)$.
 - c) $p_a(s_1) = t' \wedge p_a(s_2) = t$. Thus, $p_a(s_1) < p_a(s_2)$.
 - d) $p_a(s_1) = t \wedge p_a(s_2) = t'$. From $t' < t$ it follows $p_a(s_2) < p_a(s_1)$. It means that s_1 solves t and s_2 does not. We know, however, that t lies on two axes: A_a and A_b . Thus it must be also that $p_b(s_2) < p_b(s_1)$. But, earlier we stated that $\forall_{i \neq a} p_i(s_1) \leq p_i(s_2)$. Thus this case cannot happen.

In all cases $p_a(s_1) \leq p_a(s_2)$, thus we have $\forall_i p_i(s_1) \leq p_i(s_2)$. Since \mathcal{C} is correct, the latter implies $s_1 \leq s_2$, what finishes the proof. \blacksquare

Notice that removing any test from \mathcal{C} does not increase the size of \mathcal{C} . Proposition 2 implies that for every correct coordinate system \mathcal{C} , there is a coordinate system \mathcal{C}' such that every test appears on at most one axis in \mathcal{C}' and $|\mathcal{C}'| \leq |\mathcal{C}|$.

Corollary 1: In order to find the dimension for \mathcal{G} it is enough to search the space of such a coordinate systems \mathcal{C} that every test from T appears on at most one axis in \mathcal{C} .

For convenience, denote $\cup \mathcal{C} = \bigcup_{i \in I} A_i$ for $\mathcal{C} = (A_i)_{i \in I}$.

Definition 17: We will say that test t orders solution s_1 before solution s_2 , written $s_1 <_t s_2$, if $g(s_1, t) = 0$ and $g(s_2, t) = 1$. Notice that $s_1 <_t s_2$ implies $s_2 \not<_t s_1$.

Theorem 2: Let $\mathcal{C} = (A_i)_{i \in I}$ be a correct coordinate system and let $U = \bigcup_{i \in I} A_i$. Let \mathcal{C}' be a coordinate system resulting from removing a test t from some axis in \mathcal{C} . \mathcal{C}' is a correct coordinate system if and only if

$$\forall_{s_1, s_2 \in S} (s_1 <_t s_2 \implies \exists_{t_1 \in U/\{t\}} s_1 <_{t_1} s_2) \quad (4)$$

Proof: First, observe that \mathcal{C}' is a coordinate system since after removing t all axes remain linearly ordered.

(\Rightarrow) We will prove that if \mathcal{C}' is correct then (4) is satisfied. Suppose to the contrary that \mathcal{C}' is correct and (4) is not satisfied. It means that $\exists s_1, s_2 \in S$ such that $s_1 <_t s_2 \wedge \nexists_{t_1 \in U/\{t\}} s_1 <_{t_1} s_2$. From $s_1 <_t s_2$ it follows $s_1 < s_2$ or $s_1 \parallel s_2$, thus $s_1 \not\geq s_2$. On the other hand, from $\nexists_{t_1 \in U/\{t\}} s_1 <_{t_1} s_2$ follows that $s_1 \geq_{\mathcal{C}'} s_2$ and, since \mathcal{C}' is correct, $s_1 \geq s_2$, what contradicts earlier statements.

(\Leftarrow) We will prove that if (4) is satisfied then \mathcal{C}' is correct. According to the definition of correctness (Definition 13), it is enough to prove that for each pair (s_1, s_2) the relation between s_1 and s_2 in \mathcal{C} will be preserved in \mathcal{C}' , i.e., $s_1 \leq_{\mathcal{C}'} s_2 \iff s_1 \leq_{\mathcal{C}} s_2$.

First let us prove that removal of t does not affect the \leq relation, i.e., $\nexists s_1, s_2 \in S$ such that $s_1 \leq_{\mathcal{C}} s_2 \implies s_1 \parallel_{\mathcal{C}'} s_2 \vee s_1 \geq_{\mathcal{C}'} s_2$. For the right hand side of this implication to be true, there would have to exist an axis $A'_j \in \mathcal{C}'$ such that $p_j(s_2) < p_j(s_1)$. This implies the existence of $t' \in A'_j$ such that $p_j(s_2) < t'$ and $p_j(s_1) = t'$. From the definition of position function (12) it follows that $g(s_1, t') = 1$, and from (12) and the linear ordering of tests on A'_j it follows that $g(s_2, t') = 0$. This in turn implies that $t' \in TS(s_1) \wedge t' \notin TS(s_2)$, and thus $TS(s_1) \not\subseteq TS(s_2)$, which together with $s_1 \leq_{\mathcal{C}} s_2$ contradicts the correctness of \mathcal{C} . Thus, $s_1 \leq_{\mathcal{C}} s_2 \implies s_1 \leq_{\mathcal{C}'} s_2$.

Now let us prove that the \parallel relation is preserved, i.e., $s_1 \parallel_{\mathcal{C}} s_2 \implies s_1 \parallel_{\mathcal{C}'} s_2$ despite removing t . Let us consider two cases for the premise of (4):

- 1) $s_1 <_t s_2$: Given $s_1 \parallel_{\mathcal{C}} s_2$, there must exist at least one axis $A_i \in \mathcal{C}$ such that $p_i(s_1) < p_i(s_2)$ and at least one axis $A_j \in \mathcal{C}$ such that $p_j(s_1) > p_j(s_2)$. Because of the linear ordering, the A_j axes cannot contain t , so the ordering of s_1 and s_2 on such axes remains unchanged under the removal of t . For the A_i axes, after removing t , either $p_i(s_1) < p_i(s_2)$ or $p_i(s_1) \geq p_i(s_2)$ holds. In the former case, ordering of s_1 and s_2 on A_i is preserved in \mathcal{C}' . In the latter case not, but according to (4) there exists t_1 such that $s_1 <_{t_1} s_2$, thus there must exist an axis A_k such that $t_1 \in A_k$ and $p_k(s_1) < p_k(s_2)$. Thus, the resulting coordinate system \mathcal{C}' will contain at least one axis such that $p_i(s_1) < p_i(s_2)$ and at least one axis for which $p_j(s_1) > p_j(s_2)$, hence $s_1 \parallel_{\mathcal{C}'} s_2$.
- 2) $\neg s_1 <_t s_2$. For each axis $A_i \in \mathcal{C}$ containing t ($t \in A_i$) there are three possibilities:
 - a) If $p_i(s_1) = p_i(s_2)$, then removal of t from A_i cannot change the ordering of s_1 and s_2 , whatever their relative position w.r.t. t .

- b) If $p_i(s_1) > p_i(s_2)$, then removing t from A_i can affect the ordering of s_1 and s_2 (i.e., cause $p_i(s_1) \leq p_i(s_2)$) only if $p_i(s_1) = t$. This however implies that $s_2 <_t s_1$, which boils down to Case 1.
- c) If $p_i(s_1) < p_i(s_2)$, then from $\neg s_1 <_t s_2$ it follows that either $t \leq p_i(s_1)$ or $p_i(s_2) < t$. In both cases, there exists $t' \in A_i$, $t' \neq t$ such that $p_i(s_2) = t'$, thus $g(s_2, t') = 1$, and $g(s_1, t') = 0$. Therefore, after removing t , t' will still guarantee preserving the order of s_1 and s_2 .

In this way we proved that $s_1 \leq_C s_2 \implies s_1 \leq_{C'} s_2$ and $s_1 \parallel_C s_2 \implies s_1 \parallel_{C'} s_2$. Because for any pair (s_1, s_2) either $s_1 \leq_{C'} s_2$ or $s_1 \geq_{C'} s_2$ or $s_1 \parallel_{C'} s_2$, the converse implication has to hold too, so $s_1 \leq_C s_2 \iff s_1 \leq_{C'} s_2$, which concludes the proof. ■

Proposition 3: Let $\mathcal{C} = (A_i)_{i \in I}$ be a minimal coordinate system for \mathcal{G} . Let $U = \cup \mathcal{C}$. Then

$$\dim_B(\mathcal{G}) = \text{width}(U, \leq) \quad (5)$$

Proof: By definition of axis, each (A_i, \leq) , where $A_i \in \mathcal{C}$, $i \in I$, is a chain. \mathcal{C} is a minimal coordinate system, so by Proposition 2, we can assume without loss of generality that each test occurs in at most one axis in \mathcal{C} . Hence, \mathcal{C} is a chain partition of (U, \leq) . According to Dilworth theorem, $\text{width}(U, \leq)$ is the number of chains in the minimum chain partition of (U, \leq) , thus $|\mathcal{C}|$ is at least $\text{width}(U, \leq)$; therefore, $|\mathcal{C}| = \dim_B(\mathcal{G}) \geq \text{width}(U, \leq)$.

Let $\mathcal{D} = (D_i)_{i=1 \dots n}$ be a minimum partition of (U, \leq) into chains, hence $U = \cup \mathcal{D}$ and $n = \text{width}(U, \leq)$. Let $\mathcal{E} = (E_i)_{i=1 \dots m}$ ($m \geq n$) be such a complete coordinate system for \mathcal{G} that $D_i \subseteq E_i$ for $i = 1 \dots n$. \mathcal{E} is complete, thus it is also correct (by Proposition 1). Let us remove some tests from the axes of \mathcal{E} in such a way that \mathcal{E} will remain correct. Namely, for all $i \in 1 \dots m$, remove every test $t \in E_i$ from E_i such that $t \notin D_i$. If such a t occurs on more than one axis of \mathcal{E} , we can safely remove it by Proposition 2; otherwise, notice that $t \notin U$, since $t \notin D_i$ and, according to Theorem 2, it can also be safely removed (if it could not be removed, then it would be an element of U , because \mathcal{C} is correct). After this process is completed, the axes $E_i = \emptyset$ where $i > n$ are empty, because $D_i = \emptyset$, so they can be removed from \mathcal{E} . Then, $m = n$, $E_i = D_i$ for all $i = 1 \dots n$, hence $\mathcal{D} = \mathcal{E}$ are correct. We have proved that the minimum chain partition of U is a correct coordinate system for \mathcal{G} , thus $\dim_B(\mathcal{G}) \leq |\mathcal{E}| = n = \text{width}(U, \leq)$, what finishes the proof. ■

It is interesting to determine the upper bound for $\dim_B(\mathcal{G})$.

Proposition 4: For every game \mathcal{G}

$$\dim_B(\mathcal{G}) \leq \text{width}(T, \leq) \quad (6)$$

Proof: Let $\mathcal{C} = (A_i)_{i \in I}$ be a minimal coordinate system for \mathcal{G} . Let $U = \cup \mathcal{C}$. Obviously, $U \subseteq T$, thus $\text{width}(U, \leq) \leq \text{width}(T, \leq)$. By Proposition 3 $\dim_B(\mathcal{G}) = \text{width}(U, \leq)$, hence we have $\dim_B(\mathcal{G}) \leq \text{width}(T, \leq)$. ■

Algorithm 1 The heuristics for extracting coordinate system.

```

1: procedure HEURISTICS( $S, T, g$ )
2:    $U \leftarrow T$ 
3:   for distinct  $t, t_1, t_2 \in U$  do
4:     if  $SF(t) = SF(t_1) \cup SF(t_2)$  then
5:        $T \leftarrow T \setminus \{t\}$ 
6:     end if
7:   end for
8:    $\mathcal{C} \leftarrow \emptyset$ 
9:   for  $t \in T$  sorted by  $|SF(t)|$  do
10:     $found \leftarrow false$ 
11:    for  $T_i \in \mathcal{C}$  do
12:      if  $\max((T_i, \leq)) < t$  then
13:         $T_i \leftarrow T_i \cup \{t\}$   $\triangleright$  Add  $t$  to existing axis
14:         $found \leftarrow true$ 
15:      break
16:    end if
17:  end for
18:  if  $!found$  then
19:     $\mathcal{C} \leftarrow \mathcal{C} \cup \{\{t\}\}$   $\triangleright$  Create a new axis
20:  end if
21: end for
22: return  $\mathcal{C}$ 
23: end procedure

```

VI. ALGORITHMS

In this section, we show two algorithms constructing correct coordinate systems.

A. The Heuristics

The first algorithm for coordinate system extraction was given by Bucci *et al.* [1]. This heuristic does not guarantee finding the minimal coordinate system.

The pseudocode of the algorithm is shown as Algorithm 1. In the first stage, the algorithm removes from T the tests that are combinations of 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 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 embedded at the end. Note that the poset (T_i, \leq) in line 12 is a non-empty chain, so $\max(T_i, \leq)$ contains exactly one test, which is being compared with t .

In [1] there was no formal proof that the heuristics is correct, so we provide it here.

Proposition 5: For a given game \mathcal{G} , the Algorithm 1 produces a correct coordinate system \mathcal{C} .

Proof: Observe that skipping the first stage of the Algorithm 1 results in a complete coordinate system, which must be correct by Proposition 1. 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 $t, t_1, t_2 \in \cup \mathcal{C}$, preserves the

correctness of \mathcal{C} . From $SF(t) = SF(t_1) \cup SF(t_2)$ it follows that

$$s \in SF(t) \implies s \in SF(t_1) \vee s \in SF(t_2) \quad (7)$$

$s \in SF(t_1)$ or $s \in SF(t_2)$ implies $s \in SF(t)$, thus

$$s \notin SF(t) \implies s \notin SF(t_1) \wedge s \notin SF(t_2) \quad (8)$$

Now, consider a pair $s_1, s_2 \in S$ such that $s_1 <_t s_2$; therefore $s_1 \in SF(t)$ and $s_2 \notin SF(t)$. From the former, by (7) and without loss of generality, it follows that $s_1 \in SF(t_1)$. On the other hand, the latter implies $s_2 \notin SF(t_1)$ (by (8)). Therefore, $g(s_1, t_1) = 0$ and $g(s_2, t_1) = 1$, what implies $s_1 <_{t_1} s_2$. Since $t_1 \neq t$, test t_1 orders s_1 before s_2 . Thus, according to Theorem 2, we can safely remove t and \mathcal{C} will remain correct.

To complete the proof, notice that the tests can be removed from \mathcal{C} in a top-down order, i.e., we remove a test t_1 from the coordinate system \mathcal{C} only when there do not exist any t_2, t_3 such that $SF(t_1) \cup SF(t_2) = SF(t_3)$, where t_1, t_2, t_3 are distinct elements of $\cup \mathcal{C}$; otherwise we remove t_3 first. In this way we guarantee that all tests t such that $SF(t) = SF(t_1) \cup SF(t_2)$, where t, t_1, t_2 are distinct elements of T , will be removed, as it is the case in the algorithm. ■

B. The Exact Algorithm

In following we propose an exact algorithm that constructs a minimal coordinate system for a given game and thus determines its dimension. Shown in Algorithm 2, it has exponential complexity, and is founded on three results proved in this paper:

- 1) Corollary 1, which says that it is enough to consider the coordinate systems in which every test appears on at most one axis
- 2) Theorem 2 that determines which tests can be safely removed from T .
- 3) Proposition 3, which implies that when we cannot remove any test from T , finding the chain partition of (T, \leq) gives the dimension of \mathcal{G} .

The algorithm first computes \mathcal{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 \mathcal{C} that $\cup \mathcal{C} = T'$ 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 \mathcal{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 (CANBEREMOVED) relies on Theorem 2. By maintaining the set R of 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 and some parts of the search tree are effectively never considered. Note that reaching a leaf of 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 is similar to the one described in [8].

Algorithm 2 The exact algorithm.

```

1: procedure EXACT( $S, T, g$ )
2:    $\mathcal{U} \leftarrow$  AllMinimalSubsets( $T, T$ )
3:    $\mathcal{C} \leftarrow \infty$ 
4:   for  $U \in \mathcal{U}$  do
5:      $\mathcal{C}' \leftarrow$  ChainPartition( $U, \leq$ )
6:     if  $|\mathcal{C}'| < |\mathcal{C}|$  then
7:        $\mathcal{C} \leftarrow \mathcal{C}'$ 
8:     end if
9:   end for
10:  return  $\mathcal{C}$ 
11: end procedure
12:
13: procedure ALLMINIMALSUBSETS( $L, R$ )
14:   $isLeaf \leftarrow true$ 
15:  for  $t \in R$  do
16:     $R \leftarrow R \setminus \{t\}$ 
17:    if CanBeRemoved( $t$ ) then
18:       $isLeaf \leftarrow false$ 
19:       $L \leftarrow L \setminus \{t\}$ 
20:      AllMinimalSubsets( $L, R$ )
21:       $L \leftarrow L \cup \{t\}$ 
22:    end if
23:  end for
24:  if  $isLeaf$  and  $\nexists t \in L$  CanBeRemoved( $t$ ) then
25:    yield  $L$ 
26:  end if
27: end procedure
28:
29: procedure CANBEREMOVED( $t$ )
30:  Works according to Eq. (4)
31: end procedure
32:
33: procedure CHAINPARTITION( $(X, P)$ )
34:  Returns a set of chains  $\mathcal{C}$ 
35: end procedure

```

In the second stage, the algorithm computes chain partition \mathcal{C} for every element of \mathcal{U} that is the coordinate system (see the proof of Proposition 3), and returns the one with minimal number of axes $|\mathcal{C}|$. Partitioning a poset (P, X) , $|X| = n$ into chains is can be solved in $O(n^3)$ (c.f. [9], [10]) using a max-flow computation on a bipartite network with unit capacities. This result can be further improved to $O(n^{5/2})$ with a Hopcroft and Kart algorithm [11] and to $O(n^{5/2}/\sqrt{\log n})$ by a method introduced in [12]. Recognizing if the number of chains is at most k is even faster: $O(n^2 k^2)$ [10].

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

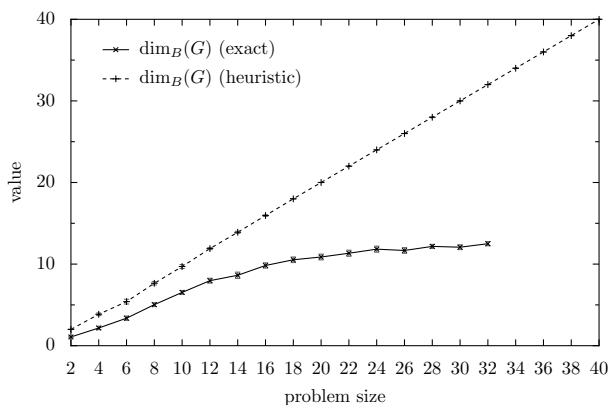


Figure 2. Heuristics vs. exact algorithm on a random game

VII. EXPERIMENTS

A. Dimension of a random game

In the first experiment, we compared the heuristics with our exact algorithm (Exact for short) on random games of different sizes $n = 2 \dots 40$. 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 of coordinate systems produced by both algorithms are shown in Fig. 2. Each data point represents the average dimension for 20 random matrices for each problem size. The 95% confidence intervals were also plotted.

Figure 2 gives rise to several interesting observations. The exact algorithm performs clearly much better than the heuristics, which hardly does any compression. The gap between the algorithms grows rapidly with n , so that the exact algorithm is nearly three times better for $n = 32$. The compression provided by the exact algorithm is impressive, given that incomparability of all pairs of test becomes almost certain for large random matrices. Note also that the logarithm-like shape of Exact curve suggests, the compression could be even higher for larger n . On the other hand, the heuristics is obviously much faster than the exact algorithm, that is why we could not produce results for $n > 32$.

B. Compare-on-one

In the second experiment we considered the compare-on-one game defined in [13]. In this game, strategies are represented as real-number vectors of length d , which we call here the *a priori dimension* of the game. The interaction function for solution s and test t is defined in two steps. First, t determines the vector position m to be used for comparison as $m = \arg \max_i t[i]$, where $t[i]$ denotes the i -th element of vector t . Then,

$$g(s, t) = \begin{cases} 1 & \text{if } s[m] \geq t[m] \\ 0 & \text{otherwise} \end{cases}$$

This time, along with the exact coordinate system dimension $\dim_B(G)$, we calculated the width of the poset (T, \leq) and dimension of the poset (S, \leq) . The width was computed

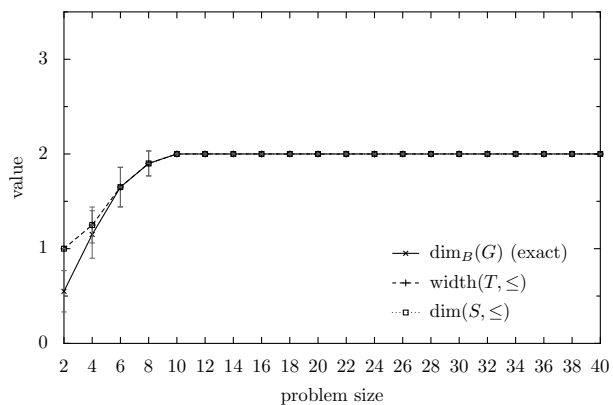


Figure 3. Compare-on-one, dimension $d = 2$

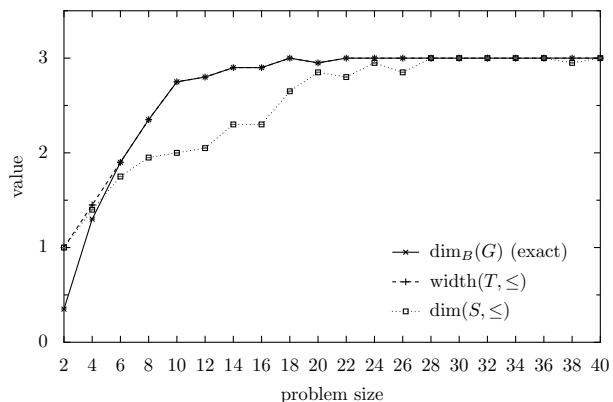


Figure 4. Compare-on-one, dimension $d = 3$

using the max-flow algorithm on a bipartite network with unit capacities, while for poset dimension we used the algorithm from [14]. We investigated how, for a given d , the above three properties of coordinate systems change for a growing size of S and T , $|S| = |T| = n$, by randomly generating n test strategies and n solution strategies. Other settings of this experiment were the same as in the previous one.

The results for $d = 2$ and $d = 3$ are shown in Figures 3 and 4. Clearly, the game dimension approaches *a priori* dimension d with growing n . This solidifies the finding of [1], where only the heuristics for \dim_B was used. Interestingly, for for this game also both width and poset dimension seem to converge towards d .

C. Compare-on-all

In the last experiment we examined the compare-on-all game [13]. Strategies are represented as vectors as in compare-on-one, but the interaction function is defined as

$$g(s, t) = \begin{cases} 1 & \text{if } \forall_i s[i] \geq t[i] \\ 0 & \text{otherwise} \end{cases}$$

Notice that this game is transitive.

The results for this problem are shown in Figs. 3 ($d = 2$) and 4 ($d = 4$). This time, \dim_B fails at approximating the

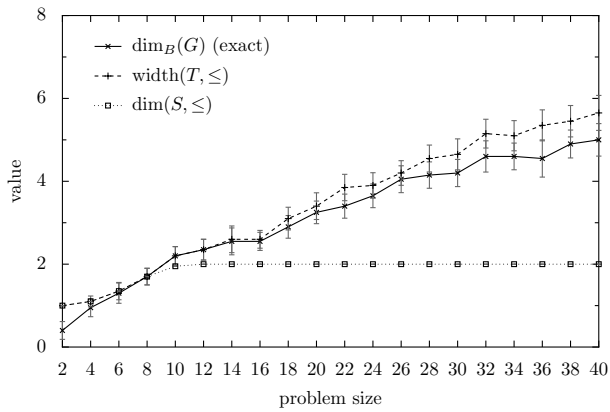


Figure 5. Compare-on-all, dimension $d = 2$

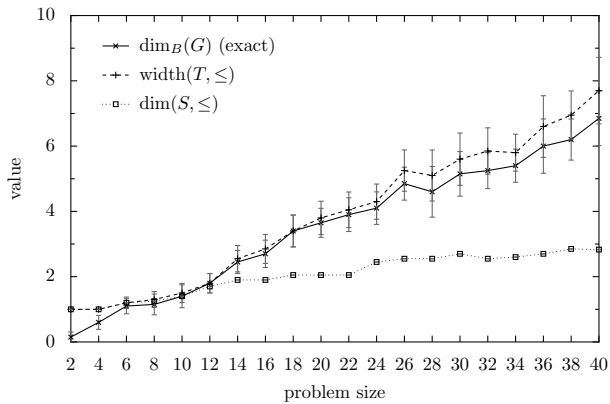


Figure 6. Compare-on-all, dimension $d = 4$

a priori game dimension d . Even worse, it does not seem to saturate with growing n .

In [1] it was found that for this problem the heuristics algorithm was constantly overestimating the *a priori* dimension of this game. There could be two possible reasons for it: i) the algorithm was only an heuristics; ii) this definition of a coordinate system does not fit the definition of the *a priori* dimension for this game. Basing on our results, we can reject the first hypothesis and accept the second one.

VIII. SUMMARY

The performance gap that spans the heuristics method [1] and the exact one clearly deserves a better approximate algorithm. The theoretical results presented here suggest the possibility of devising a better heuristics, i.e., such that more effectively exploits the trade-off between the quality of approximating $\dim_B(\mathcal{G})$ and time complexity. For such a new heuristics, it would be highly desirable to have an upper bound on estimated $\dim_B(\mathcal{G})$.

The objective of this study was to structure a game by embedding its elements into a correct coordinate system of minimal dimension. In this pursuit, we abstracted from any particular search problem and definition of solution concept. Incorporating the exact algorithm or an effective heuristics in tasks mentioned in Introduction, like learning a game

strategy, machine learning from examples, or evolutionary design, could benefit it by making the search more well-informed. An example of interesting consequence of making a search algorithm aware of interaction structure is the possibility of discarding those interactions that turn out to be superfluous as they do not determine solution's position in the minimal coordinate system. This and related concepts of exploiting coordinate systems of games definitely deserve further investigation.

ACKNOWLEDGMENTS

This work was supported in part by Ministry of Science and Higher Education grant # N N519 3505 33. Wojciech Jaśkowski gratefully acknowledges a scholarship from Sectoral Operational Programme 'Human Resources Development', Activity 8.2 co-financed by the European Social Fund European Union and the Government of Poland.

REFERENCES

- [1] A. Bucci, J. B. Pollack, and E. de Jong, "Automated extraction of problem structure," in *Genetic and Evolutionary Computation - GECCO-2004, Part I*, ser. Lecture Notes in Computer Science, K. D. et al., Ed., vol. 3102. Seattle, WA, USA: Springer-Verlag, 26-30 Jun. 2004, pp. 501-512. [Online]. Available: <http://link.springer.de/link/service/series/0558/bibs/3102/31020501.htm>
- [2] W. Trotter, *Combinatorics and partially ordered sets: Dimension theory*. Johns Hopkins University Press, 1992.
- [3] R. Dilworth, "A decomposition theorem for partially ordered sets," *Annals of Mathematics*, pp. 161-166, 1950.
- [4] B. Dushnik and E. W. Miller, "Partially ordered sets," *American Journal of Mathematics*, vol. 63, no. 3, pp. 600-610, 1941. [Online]. Available: <http://www.jstor.org/stable/2371374>
- [5] A. Bucci and J. B. Pollack, "A mathematical framework for the study of coevolution," in *Foundations of Genetic Algorithms 7*, K. A. De Jong, R. Poli, and J. E. Rowe, Eds. San Francisco: Morgan Kaufmann, 2003, pp. 221-236.
- [6] E. D. de Jong and A. Bucci, "DECA: dimension extracting coevolutionary algorithm," in *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, M. C. et al., Ed. Seattle, Washington, USA: ACM Press, 2006, pp. 313-320. [Online]. Available: <http://doi.acm.org/10.1145/1143997.1144056>
- [7] E. de Jong and A. Bucci, "Objective Set Compression. Test-Based Problems and Multiobjective Optimization," in *Multiobjective Problem Solving from Nature: From Concepts to Applications*, J. K. et al., Ed. Berlin: Springer, 2008, pp. 357-376.
- [8] M. G. de la Banda, P. J. Stuckey, and J. Wazny, "Finding all minimal unsatisfiable subsets," in *PPDP '03: Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declarative programming*. New York, NY, USA: ACM, 2003, pp. 32-43.
- [9] R. Möhring, "Algorithmic aspects of comparability graphs and interval graphs," *Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Its Applications*, pp. 41-102, 1984.
- [10] S. Felsner, V. Raghavan, and J. Spinrad, "Recognition algorithms for orders of small width and graphs of small Dilworth number," *Order*, vol. 20, no. 4, pp. 351-364, 2003.
- [11] J. Hopcroft and R. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs," *SIAM Journal on Computing*, vol. 2, p. 225, 1973.
- [12] H. Alt, N. Blum, K. Mehlhorn, and M. Paul, "Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5} \log n)$," *Information Processing Letters*, vol. 37, no. 4, pp. 237-240, 1991.
- [13] E. D. de Jong and J. B. Pollack, "Ideal Evaluation from Coevolution," *Evolutionary Computation*, vol. 12, no. 2, pp. 159-192, Summer 2004.
- [14] J. Yáñez and J. Montero, "A poset dimension algorithm," *J. Algorithms*, vol. 30, no. 1, pp. 185-208, 1999.