Mechanism Design Over Discrete Domains

Ahuva Mu'alem
Social and Information Sciences Laboratory
California Institute of Technology
ahumu@yahoo.com

Michael Schapira*
School of Engineering and Computer Science
Hebrew University of Jerusalem, Israel
mikesch@cs.huji.ac.il

ABSTRACT

Often, we wish to design incentive-compatible algorithms for settings in which the players' private information is drawn from *discrete* domains (e.g., integer values). Our main result is identifying discrete settings in which an algorithm can be made incentive-compatible iff the function it computes upholds a simple monotonicity constraint, known as weak-monotonicity. To the best of our knowledge, this is the first such characterization of incentive-compatibility in discrete domains (such characterizations were previously known only for inherently non-discrete domains, e.g., convex domains). We demonstrate the usefulness of this result by showing an application to the TCP-inspired congestion-control problem presented in [20].

Categories and Subject Descriptors

F.m [**Theory of Computation**]: MISCELLANEOUS

General Terms

Algorithms, Economics, Theory

Keywords

Game Theory, Mechanism Design

1. INTRODUCTION

Motivation. The economic field of mechanism design, and its algorithmic extensions [16, 17], focus on the design of algorithms that aim to achieve global objectives in settings in which the "input" is provided by self-interested strategic players. This necessitates the design of algorithms that are *incentive-compatible* in the sense that players are incentivized via payments to behave as instructed. The most natural approach to designing incentive-compatible algorithms is coming up with an algorithm *and* an explicit payment scheme that guarantees its incentive-compatibility. However, finding appropriate payments is often a difficult, setting-specific, task, which is mostly achievable for very simple types of algorithms.

A more general approach is the following: Any algorithm that interacts with selfish players and then outputs an outcome, can be regarded as computing a function, called a *social-choice function*, from the players' "input" to some outcome space. Certain properties of social-choice functions are known to imply their *implementability*, that is, the *existence* of a payment scheme that guarantees incentive-compatibility. Hence, instead of explicitly dealing with payments, the problem of designing incentive-compatible algorithms boils down to analyzing the mathematical properties of the social-choice functions computed by algorithms. This approach makes sense if these mathematical properties are simple and easy to analyze.

A simple constraint on social choice functions called "*weak-monotonicity*" has been shown to characterize the implementability of social choice functions in several interesting settings. However, all these characterizations of incentive-compatibility are known to apply only to environments in which the private information of the players is drawn from inherently non-discrete domains (for instance, convex domains).

In this paper we address the problem of finding sufficient and necessary conditions for the implementability of social choice functions in *discrete* settings. This question is motivated by the many cases in which the private information of the players is drawn from discrete domains (e.g., integers). Implementability in discrete domains is still little understood and has received but little attention in economics literature (see [13]). Characterizing implementability in discrete domains may prove to be helpful to the design of incentive-compatible algorithms.

The Setting and Related Work. We consider the standard mechanism design setting: There are n players $1 \ldots n$, and a set of outcomes O. Each player i has a private *valuation function* $v_i \in V_i$ that assigns a real value to every $o \in O$ (the higher the value of the outcome the more desirable it is). A (deterministic) social-choice function is a function that assigns an outcome o to every $v \in V$, where V denotes $V_1 \times \ldots \times V_n$. Intuitively, a social-choice function f is said to be implementable iff there is a *payment-function* $p_i : V \rightarrow R$ such that each player i never gains from "lying" about his valuation function. Formally, let V_{-j} denote the cartesian product of all V_is but V_j, and let (v_i, v_{-i}) denote the profile of valuation functions in which player i's valuation function

*Supported by grants from the Israel Science Foundation.

EC'08, July 8–12, 2008, Chicago, Illinois, USA.

is $v_i \in V_i$, and the other players' valuation functions are as specified by $v_{-i} \in V_{-i}$. Then, f is *implementable* iff there is a payment function p_i such that for every $i \in [n]$, for every $v_{-i} \in V_{-i}$, and for every $v_i, v_i' \in V_i$,

$$v_i(f(v_i, v_{-i})) - p_i(v_i, v_{-i}) \geq v_i(f(v_i', v_{-i})) - p_i(v_i', v_{-i}).$$

An algorithm is incentive-compatible (in dominant strategies) if the social-choice function it computes is implementable.

Rochet [18] has shown that any social-choice function is implementable iff a constraint called "*cycle monotonicity*" holds. Hence, it is possible to show that an algorithm is incentive-compatible by proving that the social-choice function it computes upholds cycle monotonicity. Indeed, recently Lavi and Swamy [11] have done so in the context of a simple scheduling problem (over severely restricted discrete domains). However, in general, resorting to this technique tends to be quite complicated and cumbersome. For this reason, researchers seek characterizations of implementability that are simpler to analyze (and thus more useful in the design of incentive-compatible algorithms). Bikhchandani et al. [3] proposed the simple *weak-monotonicity* constraint: A social-choice function f is said to be weakly-monotone if for every $i \in [n]$, for every $v_{-i} \in V_{-i}$, and for every $v_i, v_i' \in V_i$, such that $f(v_i, v_{-i}) = o_1$ and $f(v_i', v_{-i}) = o_2$ it holds that

$$v_i(o_1) + v_i'(o_2) \geq v_i(o_2) + v_i'(o_1).$$

Informally, f is said to be *strongly-monotone* if this inequality is always strict (see [10] for the formal definition). Observe that unlike the definition of implementability, the definition of weak-monotonicity does not involve a payment function. It is easy to show that weak monotonicity is always necessary for the implementability of a social-choice function (e.g., [3]) but that it is not always sufficient. Weak and strong monotonicity have proven to be very useful in various mechanism design settings, see e.g., [12, 2, 8, 10, 15, 5, 4].

We shall call a domain of valuation functions $V = V_1 \times \ldots \times V_n$ a *weak-monotonicity domain* if weak-monotonicity is both sufficient and necessary for implementability. [3, 9] present some canonical examples of weak-monotonicity domains. Saks and Yu [19] extended these results by showing that if each V_i is convex then V is a weak-monotonicity domain. Monderer [14] has taken this an additional step forward by essentially showing that this is true even if the closure of each V_i is convex[1]. All the aforementioned results apply to non-discrete domains only (convex domains etc.).

Our Contribution. We exhibit (in Section 2) the first family of discrete weak-monotonicity domains (to the best of our knowledge), which we term *Monge domains*. The proof that Monge domains are indeed weak-monotonicity domains takes advantage of the two dimensional version of submodularity (see [6]) that holds for this kind of domains (expressed by Monge matrices, hence the name). We highlight several properties of Monge domains that are useful from a mechanism design perspective and demonstrate their applicability in the context of the TCP-inspired congestion-control setting presented in [20] (in Section 3). In this congestion-control setting each strategic player ("flow") wants to transmit packets along a fixed path in a network in which the nodes are non-strategic routers. Routers drop packets using fair queuing [7] if the network becomes congested. The flows must decide on their transmission rates, and wish to maximize their throughput. We exploit the Monge structure of this domain to prove that the social-choice function that optimizes the max-min fairness value is implementable.

Finally (in Section 4), we present some simple discrete domains that are not weak-monotonicity domains, like integer grid domains. For these domains we show that despite the fact that weak-monotonicity does not imply the implementability of social-choice functions, strong-monotonicity is sufficient for implementability.

[1]Monderer basically shows that the closure of each V_i is convex IFF weak monotonicity characterizes implementability for *randomized* mechanisms. In this paper (as in previous works) we deal with *deterministic* mechanisms.

2. MONGE DOMAINS AND MECHANISM DESIGN

2.1 Monge Domains and Alignment

In order to define Monge domains we must first explain what Monge matrices are:

DEFINITION 1. *A matrix $A = (a_{r,c})_{r\in[R],c\in[C]}$ is a Monge matrix if for every integers $1 \leq r < r' \leq R$ and $1 \leq c < c' \leq C$ it holds that $a_{r,c} + a_{r',c'} \geq a_{r,c'} + a_{r',c}$.*

So, Monge matrices are simply matrices for which a simple inequality holds (for every two diagonals). These inequalities can be regarded as a two-dimensional version of submodularity (e.g. [6]). One of the useful properties of Monge matrices is that it is easy to verify, for any given matrix, whether it is a Monge matrix (as implied by the following proposition).

PROPOSITION 2.1. *[6] A matrix $A = (a_{r,c})_{r\in[R],c\in[C]}$ is a Monge matrix iff for every integers $1 \leq r < R$ and $1 \leq c < C$ it holds that $a_{r,c} + a_{r+1,c+1} \geq a_{r,c+1} + a_{r+1,c}$.*

That is, in order to verify whether a matrix is a Monge matrix it is sufficient to go over adjacent entries and check whether a simple inequality holds.

DEFINITION 2. *A domain $V = V_1 \times \ldots \times V_n$ is a* Monge domain *if for every $i \in [n]$ there is an order over the outcomes in O $o_1, o_2, \ldots$ and an order over the valuation functions in V_i $v_i^1, v_i^2, ...$, such that the matrix $A = (a_{r,c})$ in which $a_{r,c} = v_i^c(o_r)$ is a Monge matrix. We shall refer to A as a Monge matrix corresponding to V_i. Let $\leq_{A(O)}$ and $\leq_{A(V_i)}$ denote the orders over O and V_i, respectively (that is, $o \leq_{A(O)} o'$ if the row in A that corresponds to o comes before the row that corresponds to o', or if it is the same row. $\leq_{A(V_i)}$ is defined in an analogous way).*

REMARK 2.2. *Observe that the definition of Monge domains requires each V_i to be finite. It is easy to extend our definitions and results to V_is of infinite size.*

Alignment is a property of social-choice functions that shall play a major role in our proofs.

DEFINITION 3. *A social-choice function f is said to be* aligned *with a Monge domain V if the following holds: For every $i \in [n]$, and for every $v_{-i} \in V_{-i}$, there is a Monge matrix A that corresponds to V_i, such that for every $v_i \leq_{A(V_i)} v_i'$ it holds that $f(v_i, v_{-i}) \leq_{A(O)} f(v_i', v_{-i})$.*

Like Monge matrices, alignment also has a simple interpretation: A social-choice function is aligned with a Monge domain if its output is "non-decreasing" (when looking at the corresponding Monge matrix). This simple interpretation is explained in Figure 2.1.

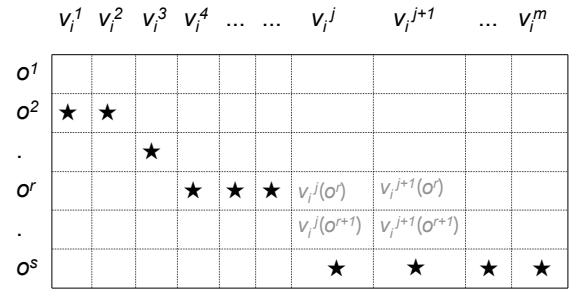

Figure 1: This example depicts an aligned social choice function f defined over a Monge domain. For a fixed v_{-i} there is a Monge matrix, such that the rows represent outcomes, the columns represent valuation functions in V_i, and $v_i^j(o^r) + v_i^{j+1}(o^{r+1}) \geq v_i^j(o^{r+1}) + v_i^{j+1}(o^r)$ for every r, j. f must be such that the outcomes chosen for the different valuations (denoted by stars) are downwards sloping

2.2 Monge Domains are Weak-Monotonicity Domains

Our main result is the following theorem:

THEOREM 2.3. *Any Monge Domain is a weak-monotonicity domain.*

We first provide an overview of the proof of Theorem 2.3: Recall that any implementable function is weakly-monotone. So, in order to prove that a Monge domain is a weak-monotonicity domain one must show that weak-monotonicity of a social-choice function implies its implementability. Our proof that any weakly-monotone social-choice function is implementable consists of 3 main steps:

- We start by proving a lemma we call the "Shifting Lemma". This lemma is of great use to us throughout the proof of Theorem 2.3.
- Using the Shifting Lemma, we show that any aligned social-choice function is implementable. We do so by proving that for any aligned social-choice function cycle-monotonicity holds (recall that cycle-monotonicity is sufficient for implementability). This intermediate step in the proof is of independent interest – it shows that one can prove the implementability of a social choice function by proving that it is aligned (a technique we shall use in our application, described in Section 3).
- After showing that any aligned social-choice function is implementable, we go on to showing that any weakly-monotone social-choice function is implementable. This too requires the use of both the Shifting Lemma and cycle-monotonicity.

PROOF. We shall now present the terminology necessary to state and prove the Shifting Lemma.

DEFINITION 4. *For any k-tuple $v_i^{1\ldots k} = v_i^1, \ldots, v_i^k$ of different valuation functions in V_i and any k-tuple $o^{1\ldots k} = o_1, ..., o_k$ of outcomes in O, we define:*

$$Value[(v_i^{1\ldots k},\ o^{1\ldots k}] = \Sigma_{r=1}^k v_i^r(o^r).$$

Let $i \in [n]$, let $v_i \in V_i$, let $v_i^{1\ldots k} = v_i^1, \ldots, v_i^k$ be a k-tuple of valuation functions in V_i, and let f be a social choice function. We denote by $f(v_i^{1\ldots k}, v_{-i})$ the k-tuple of outcomes in O $f(v_i^1, v_{-i}) \ldots f(v_i^k, v_{-i})$. Let $o^{1\ldots k} = o_1, ..., o_k$ be a k-tuple of outcomes in O. We let $\Pi[o^{1\ldots k}]$ denote the set of all k-tuples of outcomes in O that are obtained by permuting the elements in $o^{1\ldots k}$.

DEFINITION 5. *A social-choice function f is said to uphold cycle-monotonicity if for every $i \in [n]$, for every $v_{-i} \in V_{-i}$, for every positive integer $k \leq |V_i|$, for every k-tuple $v_i^{1\ldots k} = v_i^1, \ldots, v_i^k$ of different valuation functions in V_i, and for every $\pi \in \Pi[f(v_i^{1\ldots k}, v_{-i})]$, it holds that:*

$$Value[v_i^{1\ldots k}, f(v_i^{1\ldots k}, v_{-i})] \geq Value[v_i^{1\ldots k}, \pi].$$

REMARK 2.4. *Observe that weak-monotonicity is a version of cycle-monotonicity in which k equals 2.*

LEMMA 1. **[Shifting Lemma]** *Fix $r \in [n]$, and integer $k \geq 1$. Let A be a Monge matrix corresponding to V_r. Let $v_r^{1\ldots k} = v_r^1, \ldots, v_r^k$ be a k-tuple of different valuation functions in V_r such that $v_i^r \leq_{A(V_r)} \ldots \leq_{A(V_r)} v_r^k$ (since r is fixed, for simplicity we shall refer to these valuation functions as $v^{1\ldots k} = v^1, \ldots, v^k$). Let $o^{1\ldots k} = o_1, ..., o_k$ be a k-tuple of outcomes in O such that if $o_j \leq_{A(O)} o_i$ for some $i < j$ then $v^i(o_i) + v^j(o_j) = v^i(o_j) + v^j(o_i)$. Let $\pi^0 = \pi_1^0, ..., \pi_k^0$ be the permutation in $\Pi[o^{1\ldots k}]$ such that $\pi_1^0 \leq_{A(O)} \ldots \leq_{A(O)} \pi_k^0$. Then,*

$$Value[v^{1\ldots k}, o^{1\ldots k}] = Value[v^{1\ldots k}, \pi^0.]$$

PROOF. **[Sketch]** Let $1 \leq m \leq k$ be the index of the smallest element in $o_1, ..., o_k$ (according to $\leq_{A(O)}$). Consider the k-tuple of outcomes $o'^{1\ldots k} = o'_1, ..., o'_k$ such that $o'_1 = o_m$, $o'_m = o_1$, and for every $i \notin \{1, m\}$ $o'_i = o_i$. Observe that it must hold that $Value[v^{1\ldots k}, o^{1\ldots k}] = Value[v^{1\ldots k}, o'^{1\ldots k}]$ (if $m = 1$ then this is obvious, if $m > 1$ this is due to the fact that $v^1(o_1) + v^m(o_m) = v^1(o_m) + v^m(o_1)$). We shall now prove that $o'^{1\ldots k}$ too has the property that if $o'_j \leq_{A(O)} o'_i$ for some $i < j$ then $v^i(o'_i) + v^j(o'_j) = v^i(o'_j) + v^j(o'_i)$. Since, this property holds for $o^{1\ldots k}$, and $o'^{1\ldots k}$ only differs from $o^{1\ldots k}$ in (at most) two coordinates, and since o'_1 is smaller than any other outcome, it is easy to see that it suffices to handle the two following cases:

- Case 1: Consider some $1 < i < m$ such that $o_1 = o'_m \leq_{O(A)} o'_i$. Due to the fact that A is a Monge matrix it must hold that $v^i(o'_i) + v^m(o'_m) \leq v^i(o'_m) + v^m(o'_i)$. We wish to show that this inequality is an equality.

Assume, by contradiction, that

$$v^i(o'_i) + v^m(o'_m) < v^i(o'_m) + v^m(o'_i).$$

Since $o_m \leq_{A(o)} o_i$ and $i < m$, we know that

$$v^i(o'_i) + v^m(o'_1) = v^i(o_i) + v^m(o_m) =$$

$$v^i(o_m) + v^m(o_i) = v^i(o'_1) + v^m(o'_i).$$

A simple calculation (subtracting the second inequality from the first) shows that

$$v^i(o'_m) + v^m(o'_1) > v^i(o'_1) + v^m(o'_m).$$

However, since $o'_1 \leq_{A(O)} o'_m$ (o'_1 is the smallest element), and $v^i \leq_{A(V_r)} v^m$ this is a contradiction to the fact that A is a Monge matrix.

- Case 2: Consider some $i > m$ such that $o'_i \leq_{A(O)} o'_m$. Due to the fact that A is a Monge matrix it must hold that $v^m(o'_m) + v^i(o'_i) \leq v^m(o'_i) + v^m(o'_m)$. We wish to show that this inequality is an equality. Assume, by contradiction, that

$$v^m(o'_m) + v^i(o'_i) < v^m(o'_i) + v^i(o'_m).$$

Since $o_i = o'_i \leq_{A(O)} o'_m = o_1$ and $1 < i$, we know that

$$v^1(o'_m) + v^i(o'_i) = v^1(o_1) + v^i(o_i) =$$

$$v^1(o_i) + v^i(o_1) = v^1(o'_i) + v^m(o'_m).$$

A simple calculation shows that the two previous inequalities imply that

$$v^1(o'_m) + v^m(o'_i) > v^1(o'_i) + v^m(o'_m).$$

However, since $o'_i \leq_{A(O)} o'_m$, and $v^1 \leq_{A(V_r)} v^i$ this is a contradiction to the fact that A is a Monge matrix.

So, we now have a new k-tuple $o'^{1\ldots k}$, that has the same value as $o^{1\ldots k}$, that maintains the same properties as $o^{1\ldots k}$, and that has the smallest element in $o^{1\ldots k}$ as its first element. We now consider the k-tuple $o''^{1\ldots k}$, that is identical to $o'^{1\ldots k}$, with the exception that second smallest element in $o'^{1\ldots k}$ is replaced with o'_2. That is, $o''^{1\ldots k}$ has the smallest element as its first element, and the second smallest element as its second element. Similar arguments to the ones shown before imply that $o''^{1\ldots k}$ has the same value as $o'^{1\ldots k}$ and maintains the same properties. We continue shifting elements in this fashion until we reach π_o. Since the value is preserved throughout these different shifts, the lemma follows. □

We use the Shifting Lemma to prove the following lemma:

LEMMA 2. *Any social-choice function that is aligned with a Monge domain is implementable.*

PROOF. Let f be a social choice function that is aligned with a Monge domain. Assume that f is not implementable. In this case, it must be that f does not uphold cycle-monotonicity. That is, there is some $r \in [n]$, some $v_{-r} \in V_{-r}$, some k-tuple $v_r^{1\ldots k} = v_r^1, \ldots, v_r^k$ of different valuation functions in V_r, and some $\pi \in \Pi[f(v_r^{1\ldots k}, v_{-r})]$ such that $Value[v_r^{1\ldots k}, f(v_r^{1\ldots k}, v_{-r})] < Value[v_r^{1\ldots k}, \pi]$. Choose $\pi^* = \pi_1^*, ..., \pi_k^*$ be the permutation in $\Pi[f(v_r^{1\ldots k}, v_{-r})]$ for which $Value[v_r^{1\ldots k}, \pi^*]$ is maximized. So,

$$Value[v_r^{1\ldots k}, f(v_r^{1\ldots k}, v_{-r})] <$$

$$Value[v_r^{1\ldots k}, \pi] \leq Value[v_r^{1\ldots k}, \pi^*]$$

If there are some $i < j$ such that $\pi_j^* \leq_{A(O)} \pi_i^*$ it holds that $v_r^i(\pi_i^*) + v_r^j(\pi_j^*) = v_r^i(\pi_j^*) + v_r^j(\pi_i^*)$. This is because A is a Monge matrix and so $v_r^i(\pi_i^*) + v_r^j(\pi_j^*) \leq v_r^i(\pi_j^*) + v_r^j(\pi_i^*)$. If this inequality is strict then we get a permutation with a strictly higher value than π^* by substituting π_i^* and π_j^*. This would contradict the definition of π^*.

So, π^* is a permutation for which the conditions specified by the Shifting Lemma hold. Therefore, π^* has the same value as the permutation in $\Pi[f(v_r^{1\ldots k}, v_{-r})]$ in which the elements are ordered from low to high. However, this permutation is precisely $f(v_r^{1\ldots k}, v_{-r})$, because f is aligned with A, and so

$$Value[v_r^{1\ldots k}, f(v_r^{1\ldots k}, v_{-r})] = Value[v_r^{1\ldots k}, \pi^*].$$

A contradiction. □

Now, consider a social-choice function f for which weak-monotonicity holds. We shall conclude the proof of the theorem by showing that cycle-monotonicity holds for this f. Assume, by contradiction, that it does not. Then, there is some $r \in [n]$, some $v_{-r} \in V_{-r}$, some k-tuple $v_r^{1\ldots k} = v_r^1, \ldots, v_r^k$ of different valuation functions in V_r, and some $\pi \in \Pi[f(v_r^{1\ldots k}, v_{-r})]$ such that $Value[v_r^{1\ldots k}, f(v_r^{1\ldots k}, v_{-r})] < Value[v_r^{1\ldots k}, \pi]$. Let $\pi^0 = \pi_1^0, ..., \pi_k^0$ be the permutation in $\Pi[f(v_r^{1\ldots k}, v_{-r})]$ such that $\pi_1^0 \leq_{A(O)} \ldots \leq_{A(O)} \pi_k^0$. Lemma 2 implies that π^0 has the highest value out of all the permutations in $\Pi[f(v_r^{1\ldots k}, v_{-r})]$. (This is the permutation that describes the output of an aligned function. For such functions Lemma 2 shows that cycle-monotonicity holds.)

Let us denote the k-tuple of elements $f(v_r^{1\ldots k}, v_{-r})$ by $o_1, ..., o_k$. Because of the fact that A is a Monge matrix, for every $i < j$ and $o_j \leq_{A(O)} o_i$ it holds that $v_r^i(o_j) + v_r^j(o_i) \leq v_r^i(o_i) + v_r^j(o_j)$. However, the weak-monotonicity of f implies that this inequality also holds in the opposite direction: $v_r^i(o_j) + v_r^j(o_i) \geq v_r^i(o_i) + v_r^j(o_j)$. Therefore, for every $i < j$ and $o_j \leq_{A(O)} o_i$ it holds that $v_r^i(o_j) + v_r^j(o_i) = v_r^i(o_i) + v_r^j(o_j)$. This means that we can use the Shifting Lemma to show that the value of $f(v_r^{1\ldots k}, v_{-r})$ is the same as the value of π^0. Since π^0 has the highest value out of the all permutations in $\Pi[f(v_r^{1\ldots k}, v_{-r})]$ this contradicts the fact that $Value[v_r^{1\ldots k}, f(v_r^{1\ldots k}, v_{-r})] < Value[v_r^{1\ldots k}, \pi]$.

3. APPLICATION: CONGESTION CONTROL GAMES

We shall now apply the concepts and tools presented in Section 2 to the (discretized version) of the TCP-inspired congestion-control setting presented in [20]: We are given a

graph $G = (V, E)$, in which set of vertices V corresponds to non-strategic routers, and the set of edges E corresponds to physical communication links between these routers. For each edge e there is a maximum (integer) number of packets that can traverse that edge simultaneously (i.e., e's capacity), denoted by c_e. The players are n flows $1, ..., n$, each described by a fixed route r_i from a source-vertex $s_i \in V$ to a target-vertex $t_i \in V$. Each player i has a private integer value d_i that represents the number of packets it wishes to transmit.

For every vector of declared values $d'_1, ..., d'_n$, the capacity of each edge e is shared between the flows in the following recursive manner (known as fair queuing [7]): Let k_e be the number of flows whose routes go through e. If for every such flow i $d'_i \geq \frac{c_e}{k_e}$ (rounded down) then allocate a capacity of $\frac{c_e}{k_e}$ to each flow whose route traverses e. Otherwise, perform the following steps: Let d'_i be the lowest declared value of a flow that goes through e. Allocate a capacity of d'_i to i. Apply fair queuing to share the remaining capacity of $c_e - d'_i$ between the remaining $k_e - 1$ flows.

Flows are selfish and wish to maximize their throughput. The utility of flow i is the minimum of the lowest capacity share it gets (over edges in its route), and d_i. This corresponds to the number of packets it is able to transmit over its route (a flow does not care whether it is sending d_i packets or more than d_i packets). It is known (see [7, 20]) that sharing capacity via fair queuing results in an allocation of capacity shares that maximizes the max-min fairness value.

We shall now prove that it is a dominant strategy for every flow i to truthfully report its private value d_i.

CLAIM 3.1. *There is a way to define valuation functions $v_1, ..., v_n$ for the different flows $1, ..., n$ such that the congestion-control setting in [20] is a Monge domain.*

PROOF. For every flow i, we associate the value d_i with the valuation function v_i that assigns to every (integer) number of packets $1 \leq a \leq M$, where M is the maximal capacity that can traverse any edge, the value $-|d_i - a|$. Now, fix some $i \in [n]$, and some profile of declared values of the other flows d_{-i}. Consider the matrix A in which each row r corresponds to the outcome in which i is able to transmit r packets, each column s corresponds to the valuation function v_s (associated with d_s), and every entry $a_{r,s}$ in A equals $v_s(r)$. It is easy to show that A is a Monge matrix. Hence, this congestion-control setting can be embedded in a Monge domain. □

Fair queuing can be regarded as a social-choice function f that maps every vector of valuation functions $v_1, ..., v_n$ (associated with the d_is) to an outcome that specifies how the capacity is shared. This function optimizes the max-min fairness value.

CLAIM 3.2. *The max-min fairness optimizing (fair-queuing) function f is aligned with the (congestion-control) Monge domain.*

PROOF. Fix some $i \in [n]$, and some profile of declared values of the other flows d_{-i}. Let A be a Monge matrix as defined above. Observe that f is aligned with the Monge domain (A flow i cannot get a smaller capacity share by reporting a higher d_i). □

The following corollary (from Lemma 2) follows:

COROLLARY 3.3. *The fair queuing social-choice function is implementable.*

REMARK 3.4. *In fact, an examination of the form of alignment of the fair-queuing function reveals that its implementation does not require any payments (payments are constantly 0 – flows are neither required to pay the mechanism nor are they paid by the mechanism).*

4. WEAK AND STRONG MONOTONICITY

In this section we focus on two important cases of discrete domains: Integer grid domains, and 0/1 domains. We present an interesting example, given by Lan Yu [21], to show that integer grids are not weak-monotonicity domains. We then show, in contrast, that every *strongly*-monotone social choice function defined over an integer grid domain is implementable. We show that the same is true for 0/1 domains – while weak-monotonicity is insufficient to guarantee implementability, strong-monotonicity is.

4.1 Integer Grid Domains

4.1.1 Integer Grids Are Not Weak-Monotonicity Domains

Let $V = V_1 \times \ldots \times V_n$ be a domain of valuation functions defined over a set of outcomes O. We can think of every $v_i \in V_i$ as a vector in $R^{|O|}$ specifying a value for every outcome.

DEFINITION 6. *A valuation function domain is an integer grid domain if $V = Z^{|O|} \times \ldots \times Z^{|O|}$.*

That is, an integer grid domain is a domain of valuation functions that can take any combination of integer values. The next proposition, due to Lan Yu [21], shows that an integer grid is *not* a weak-monotonicity domain. In fact, this example can easily be modified to show that no *bounded* integer grid domain is a weak-monotonicity domain. By bounded integer grid, we simply mean the discrete cube $V = \{0, 1, ..., L\}^{n|O|}$ (for some positive integer L).

PROPOSITION 4.1. *[21] There is a social choice function f defined over an integer grid domain that satisfies weak-monotonicity and is not implementable.*

PROOF. Consider the following social choice function $f : V \rightarrow O$ defined for a single player domain (i.e., $V = V_1$). There are 3 distinct outcomes $O = \{a, b, c\}$ and $V = Z^3$.

$$f(v) = \begin{cases} a & \text{if } \{v(a) \geq v(c) \text{ and } v(a) \geq v(b) + 2\} \\ & \text{or } \{v(c) = v(a) + 1 \text{ and} \\ & v(c) = v(b) + 2\}, \\ c & \text{if } \{v(c) > v(a) \text{ and } v(c) > v(b)\} \\ & \text{and not } \{v(c) = v(a) + 1 \text{ and} \\ & v(c) = v(b) + 2\}, \\ b & \text{otherwise.} \end{cases}$$

LEMMA 3. *The function f is weakly-monotone.*

PROOF. In order to show that f is weakly-monotone it is enough to consider the following cases:

Case 1: Let $f(u) = c$, $f(v) = a$. Assume by contradiction that $v(a) - v(c) < u(a) - u(c)$. If $v(a) \geq v(c)$ and $v(a) \geq$

$v(b)+2$, then $0 \leq v(a)-v(c) < u(a)-u(c)$ implies that $u(a) > u(c)$. This contradicts $f(u)=c$.

If $v(c)=v(a)+1$ and $v(c)=v(b)+2$, then equivalently, $v(a)=v(c)-1=v(b)+1$. Now, $v(a)-v(c)=-1 < 0 \leq u(a)-u(c)$ implies that $u(a) \geq u(c)$. Contradicting $f(u)=c$.

Case 2: Let $f(u)=b$, $f(v)=a$. Assume by contradiction that $v(a)-v(b) < u(a)-u(b)$. If $v(a) \geq v(c)$ and $v(a) \geq v(b)+2$, then $2 \leq v(a)-v(b) < u(a)-u(b)$ implies that $u(a) > u(b)+2$. By definition of f, and by the fact that $f(u)=b \neq a,c$, we get that $u(a) < u(c)$ and $u(c) \leq u(b)$. In particular, $u(a)-u(b) < 0$ – a contradiction.

If $v(c)=v(a)+1$ and $v(c)=v(b)+2$, then equivalently, $v(a)=v(c)-1=v(b)+1$. Now, $v(a)-v(b)=1 < u(a)-u(b)$ implies that $2 \leq u(a)-u(b)$, and we can proceed as before.

Case 3: Let $f(u)=b$, $f(v)=c$. Assume by contradiction that $v(c)-v(b) < u(c)-u(b)$. Now, $f(v)=c$, so $v(a) < v(c)$, and $v(b) < v(c)$. Clearly, $0 < v(c)-v(b) < u(c)-u(b)$, and $u(b) < u(c)$. By definition of f, and by the fact that $f(u)=b \neq a,c$, $u(c) \leq u(a)$ and $u(a) < u(b)+2$. In particular, $u(c) < u(b)+2$. Therefore, $u(b) < u(c) < u(b)+2$, and so $u(c)=u(b)+1$. Now, $v(c)-v(b) < u(c)-u(b)=1$ implies that $v(c)-v(b) \leq 0$ – a contradiction. □

To prove that f is not implementable we show that cycle-monotonicity does not hold for f (recall that cycle monotonicity is necessary for implementability).

CLAIM 4.2. *f does not uphold cycle-monotonicity.*

PROOF. Consider the valuations $v=(2,1,3)$, $u=(1,0,1)$, $w=(1,1,2)$. By definition of f, $f(v)=a, f(u)=b, f(w)=c$. Now $v(a)-v(c)+w(c)-w(b)+u(b)-u(a)=-1+1-1<0$. □

REMARK 4.3. *Observe that f is not strongly-monotone: Let $v=(2,1,3)$ and $u=(1,0,1)$. $f(v)=a, f(u)=b$, and $v(a)+u(b)=2+0=1+1=v(b)+u(a)$. As we shall see next, this is no coincidence.*

4.1.2 Sufficiency of Strong-Monotonicity

THEOREM 4.4. *If $f: V \rightarrow O$ is strongly-monotone and V is an integer grid domain then f is implementable.*

PROOF. For every distinct $b,c \in O$ and v_{-i} we define:

$$\delta^i_{bc}(v_{-i}) = \min \{v'_i(b)-v'_i(c) \mid v'_i \in V_i \text{ and } f(v'_i,\, v_{-i})=b\}$$

That is, fixing v_{-i}, $\delta^i_{bc}(v_{-i})$ is the minimum possible difference between a value for b and for c assigned by the same valuation v'_i (for which f chooses b). Specifically, if $f(v)=a$, then $v_i(a)-v_i(b) \geq \delta^i_{ab}(v_{-i})$ for every $b \in A$. W.l.o.g., we shall assume a single player and thus we use the notation δ_{ab} instead of $\delta^i_{ab}(v_{-i})$ (since all monotonicity arguments involve fixing the other players, one can simplify matters by only considering single-player settings).

LEMMA 4. **[Triangle Inequality Lemma]** *Let a,b,c be three arbitrary outcomes. If there exists u,v such that $f(v)=b$ and $f(u)=c$ then: $\delta_{bc}+\delta_{ca} \geq \delta_{ba}$.*

PROOF. Assume by contradiction that: $\delta_{bc}+\delta_{ca} < \delta_{ba}$. The domain is an integer grid, and so there must be v,u such that $f(v)=b, f(u)=c$, $v(b)-v(c)=\delta_{bc}$ and $u(c)-u(a)=\delta_{ca}$. We can assume w.l.o.g that $u(c)=v(c)$ (if f is strongly-monotone, then adding a constant to v does not change the chosen outcome).

Define v' as follows: $v'(a)=u(a)$, and $v'(d)=v(d)$ for any outcome $d \neq a$. By strong monotonicity $f(v') \in \{b,a\}$, as the only changed value is the value of a. We shall show that $f(v')=b$. This is clearly the case if $u(a) \leq v(a)$. Consider the case that $v'(a)=u(a) > v(a)$. Now, if $f(v')=a$, and recall that $f(u)=c$, and $u(c)=v(c)$, so we get that $v'(a)+u(c)=v'(c)+u(a)$, thus contradicting strong monotonicity. □

Now we are ready to prove our theorem. Fix an arbitrary outcome $a \in O$ such that there exists w for which $f(w)=a$. Consider the following payment scheme: p_d is zero if d equals a, and $p_d=\delta_{da}$, otherwise. Assume by contradiction that $f(v)=b$, and $f(u)=c$, but $v(c)-p_c > v(b)-p_b$. That is, if the valuation of the player is v, he might declare u, in order to increase his utility. Clearly, $b \neq c$. By definition: $v(b)-v(c) \geq \delta_{bc}$. Now, by rearranging we get that: $\delta_{bc} \leq v(b)-v(c) < p_b-p_c=\delta_{ba}-\delta_{ca}$. Hence, we get that: $\delta_{bc}+\delta_{ca} < \delta_{ba}$, contradicting the Triangle Inequality Lemma. This completes the proof of the theorem.

4.2 0/1-Domains

DEFINITION 7. *$V=V_1 \times \ldots \times V_n$ is a 0/1 domain if $V=\{0,1\}^{|O|} \times \ldots \times \{0,1\}^{|O|}$.*

We show that, as in the case of integer grids, 0/1 domains are not weak-monotonicity domains, but strong-monotonicity is sufficient for implementability.

EXAMPLE 4.5. *We show a 0/1-domain that is not a weak-monotonicity domain. Consider a single player and 3 outcomes $O=\{a,b,c\}$, and the following social choice function: $f(0,0,0)=f(1,0,0)=f(0,0,1)=f(1,0,1)=f(1,1,1)=a$, $f(1,1,0)=f(0,1,0)=b$, and $f(0,1,1)=c$. It is not hard to verify that f is weakly-monotone. However, it is not cyclic-monotone: Let v =(0,0,1), u=(1,1,0), w=(0,1,1). Now, $f(v)=a, f(u)=b, f(w)=c$, and $v(a)+u(b)+w(c)=0+1+1<1+1+1=v(c)+u(a)+w(b)$.*

PROPOSITION 4.6. *If $f: V \rightarrow O$ is strongly-monotone, and V is the 0/1-domains then f is implementable.*

PROOF. **[Sketch]** W.l.o.g. we assume a single player setting. Suppose that there exists a negative cycle of length 3 or more (that is, a violation of cycle-monotonicity with at least 3 outcomes). Assume w.l.o.g that this cycle is: $v^1(a)+v^2(b)+v^3(c)+v^4(d) < v^1(b)+v^2(c)+v^3(d)+v^4(a)$. In a 0/1-domain, it cannot be the case that the sum in left hand side is 4. So, assume w.l.o.g that $v^1(a)=0$. Now, we shall show that it must be the case that $v^2(b)+v^3(c)+v^4(d)=3$. Assume by contradiction that $v^2(b)=0$. This contradicts strong monotonicity, as $0=v^1(a)+v^2(b) > v^1(b)+v^2(a) \geq 0$. Similarly, $v^3(c)=v^4(d)=1$. But now, if the left hand side equals 3, the right hand side must be equal to 4. By strong monotonicity: $1=v^1(a)+v^4(d) > v^1(d)+v^4(a)$, this can be true only if $v^1(d)+v^4(a)=0$, and in particular $v^4(a)=0$, a contradiction. □

5. DISCUSSION AND OPEN QUESTIONS

In this paper we have exhibited the first example of discrete weak-monotonicity domains. We have also presented

domains in which weak-monotonicity is insufficient for implementability, but strong-monotonicity does imply implementability. There are many intriguing questions that remain open:

- We have considered the notion of *strategyproofness* (incentive-compatibility in dominant strategies). It is would be interesting to understand the implications of the structure of Monge domains to other mechanism design notions. For instance: When can we get *group strategyproofness* (resilience to deviations even by *coalitions* of players)? When can we get strategyproofness *without money* (that is, when the payments are always 0, as in our TCP-related application)?

- Many examples in mechanism design fall into the category of Monge domains (other than the one presented in Section 3), e.g.,the single parameter scheduling environment discussed by Archer and Tardos [1], single-peaked voting problems, and more. These examples (like ours) all involve *single-parameter* domains (informally, domains in which the private information of each player is expressed by one real variable). However, our framework does not, at first sight, seem restricted to such domains (there is no such requirement in our model). It would be interesting to apply our techniques to a *multi-parameter* problem.

- From a computational perspective, there is much that we do not know. In particular, how hard is it to *compute* the payments given a Monge domain? The answer to this question may require, as an intermediate step, answering another interesting question: What is the form of the payments of implementable functions over Monge domains?

- We have shown that in integer-grid domains and 0/1 domains strong-monotonicity implies implementability. We do not know whether this is the case for *bounded* integer grid domains. (By bounded integer grid, we simply mean the discrete cube $V = \{0, 1, ..., L\}^{n|O|}$, for some positive integer L).

Acknowledgements

Many thanks to Lan Yu for giving us her permission to include her counter-example in this paper. We thank Shahar Dobzinski and Noam Nisan for helpful discussions. We also thank the anonymous referees for their comments

6. REFERENCES

[1] Aaron Archer and Eva Tardos. Truthful mechanisms for one-parameter agents. In *FOCS*, pages 482–491, 2001.

[2] Aaron Archer and Eva Tardos. Frugal path mechanisms. In *SODA*, 2002.

[3] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica*, 74(4):1109–1132, July 2006.

[4] George Christodoulou, Elias Koutsoupias, and Annamaria Kovacs. Mechanism design for fractional scheduling on unrelated machines. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2007.

[5] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 1163–1170. SIAM, 2007.

[6] V.G. Deineko and G. Woeginger. Some problems around travelling salesmen, dart boards, and euro-coins. *Bulletin of the European Association for Theoretical Computer Science*, 90:43–52, October 2006.

[7] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, 1989.

[8] Andrew Goldberg, Jason Hartline, Anna Karlin, and Andrew Wright. Competitive auctions, 2004. Working paper. Preliminary versions presented at SODA'01 and STOC'02.

[9] Hongwei Gui, Rudolf Muller, and Rakesh Vohra. Characterizing dominant strategy mechanisms with multi-dimensional types, 2004. Working paper.

[10] Ron Lavi, Ahuva Mu'alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions. In *FOCS*, 2003.

[11] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *EC*, 2007.

[12] Daniel Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[13] William S. Lovejoy. Optimal mechanisms with finite agent types. *Manage. Sci.*, 52(5):788–803, 2006.

[14] Dov Monderer. Monotonicity and implementability, 2007. Working paper.

[15] Ahuve Mu'alem and Michael Schapira. Setting lower bounds on truthfulness. In *SODA*, 2007.

[16] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[17] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani (eds.). *Algorithmic Game Theory*. Cambridge University Press, 2007.

[18] J.C. Rochet. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[19] Michael Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In *EC*, 2005.

[20] Michael Schapira and Aviv Zohar. Congestion-control games, 2008. Working paper.

[21] Lan Yu. Private communication, 2005.