Computational Social Choice: Autumn 2012

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Plan for Today

Today we will discuss the problem of fairly allocating a number of *indivisible goods* to a group of agents. Specific topics:

- Complexity of finding socially optimal allocations
- Compact representation languages for utility functions
- Algorithms for finding socially optimal allocations

For general background on these topics, see my lecture notes and the "MARA Survey".

U. Endriss. *Lecture Notes on Fair Division*. ILLC, University of Amsterdam, 2010.

Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-Aguilar and P. Sousa. Issues in Multiagent Resource Allocation. *Informatica*, 30:3–31, 2006.

Notation and Terminology

We consider the case of allocating *indivisible* (as well as non-sharable, single-unit, static) goods to agents with cardinal preferences.

We shall work in this framework:

- Set of *agents* $\mathcal{N} = \{1, \ldots, n\}$ and finite set of indivisible *goods* $\mathcal{G}$.
- An *allocation* A is a partitioning of $\mathcal{G}$ amongst the agents in $\mathcal{N}$.
  Example: $A(i) = \{g_5, g_7\}$ — agent i owns goods g_5 and g_7
- Each agent $i \in \mathcal{N}$ has got a *utility function* $u_i : 2^{\mathcal{G}} \to \mathbb{R}$.
  Example: $u_i(A) = u_i(A(i)) = 577.8$ — agent i is pretty happy

An *allocation problem* is a triple $\langle \mathcal{N}, \mathcal{G}, \mathcal{U} \rangle$, where $\mathcal{U}$ is a set of utility functions ($+$ possibly the *initial allocation* A_0).

Allocation Procedures

We can distinguish two approaches:

- This lecture: in the *centralised approach*, we need to devise an optimisation algorithm to compute an allocation meeting our fairness and efficiency requirements.
- Next lecture: in the *distributed approach*, allocations emerge as agents implement a sequence of local deals. What can we say about the properties of these emerging allocations?

Discussion: advantages and disadvantages of either approach (simplicity, trust towards the centre, ...)

Social Welfare

Recall that we have seen a number of criteria, most of them based on various social welfare orderings, that can be used to define what constitutes an optimal allocation.

Specifically, *utilitarian social welfare* is defined as follows:

$$\mathrm{SW}_{\mathrm{util}}(A) \quad = \quad \sum_{i \in \mathcal{N}} u_i(A)$$

How hard is it to find an allocation that is optimal in this sense?

Welfare Optimisation

How hard is it to find an allocation with maximal social welfare?

Rephrase this *optimisation problem* as a *decision problem:*

> Welfare Optimisation (WO)
> **Instance:** $\langle \mathcal{N}, \mathcal{G}, \mathcal{U} \rangle$ and $K \in \mathbb{Q}$
> **Question:** Is there an allocation A such that $\mathrm{SW}_{\mathrm{util}}(A) > K$?

Unfortunately, the problem is intractable:

Theorem 1 Welfare Optimisation *is NP-complete.*

Proof: NP-membership: we can check in polytime whether a given allocation A really has social welfare $> K$. NP-hardness: next slide. ✓

This seems to have first been stated by Rothkopf *et al.* (1998).

M.H. Rothkopf, A. Pekeč, and R.M. Harstad. Computationally Manageable Combinational Auctions. *Management Science*, 44(8):1131–1147, 1998.

Proof of NP-hardness

By reduction to Set Packing (known to be NP-complete):

Set Packing
Instance: Collection $\mathcal{C}$ of finite sets and $K \in \mathbb{Q}$
Question: Is there a collection of disjoint sets $\mathcal{C}' \subseteq \mathcal{C}$ s.t. $|\mathcal{C}'| > K$?

Given an instance $\mathcal{C}$ of Set Packing, consider this allocation problem:

- Goods: each item in one of the sets in $\mathcal{C}$ is a good
- Agents: one for each set in $\mathcal{C}$ + one other agent (called agent 0)
- Utilities: $u_C(S) = 1$ if $S = C$ and $u_C(S) = 0$ otherwise; $u_0(S) = 0$ for all bundles S

That is, every agent values "its" bundle at 1 and every other bundle at 0. Agent 0 values all bundles at 0.

Then every set packing corresponds to an allocation (with SW $= |\mathcal{C}'|$). *Vice versa*, for every allocation there is one with the same SW corresponding to a set packing (give anything owned by agents with utility 0 to agent 0). ✓

Representation of Preferences

In our proof, we have glossed over the *representation* of preferences:

- To be precise, we have proved NP-hardness wrt. *the number of pairs of agents and bundles with non-zero value*, corresponding to the number of sets involved in SET PACKING.
- Observe that this number itself may already be very high (exponential in the number of goods).
- In other words, we have proved NP-completeness wrt. an *explicit representation* of the utility functions involved.

But the result is very robust: we also get NP-completeness for more sophisticated preference representation languages. Some of those results are reviewed by Chevaleyre et al. (2006).

Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-Aguilar and P. Sousa. Issues in Multiagent Resource Allocation. *Informatica*, 30:3–31, 2006.

Welfare Optimisation under Additive Preferences

Sometimes we can reduce complexity by restricting attention to problems with certain types of preferences.

A utility function $u : 2^{\mathcal{G}} \to \mathbb{R}$ is called *additive* if for all $G \subseteq \mathcal{G}$:

$$u(G) \quad = \quad \sum_{g \in \mathcal{G}} u(\{g\})$$

The following result is almost immediate:

Proposition 1 Welfare Optimisation *is polynomial in case all individual preferences are additive.*

Proof: To compute an allocation with maximal social welfare, simply give each item to (one of) the agent(s) who value it the most. ✓

This works, because we have $\sum_{i \in \mathcal{N}} \sum_{g \in \mathcal{G}} u_i(\{g\}) = \sum_{g \in \mathcal{G}} \sum_{i \in \mathcal{N}} u_i(\{g\})$.
So the same restriction does not help for, say, the egalitarian or Nash CUF.

More Complexity Results

To give you a rough idea of what kind of complexity results there are out there, here are some representative examples (omitting precise statements regarding the representation of preferences employed):

- Finding an allocation with maximal *egalitarian* social welfare is NP-hard, even when all valuations are additive.
- Checking whether a given allocation is *Pareto efficient* is coNP-complete.
- Checking whether an *envy-free* allocation exists is NP-complete; checking whether an allocation that is both Pareto efficient and envy-free exists is even Σ_2^p-complete (NP with NP-oracle).

References to these results may be found in the "MARA Survey".

Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-Aguilar and P. Sousa. Issues in Multiagent Resource Allocation. *Informatica*, 30:3–31, 2006.

Preference Representation Languages

Example: Allocating 10 goods to 5 agents means $5^{10} = 9765625$ allocations and $2^{10} = 1024$ bundles for each agent to think about.

So we need to choose a good *language* to compactly represent preferences over such large numbers of alternative bundles, e.g.:

- Logic-based languages (weighted goals)
- Bidding languages for combinatorial auctions (OR/XOR)
- Program-based preference representation (straight-line programs)
- CP-nets and CI-nets (for ordinal preferences)

The choice of language affects both *algorithm design* and *complexity*.

See our *AI Magazine* article for references and an introduction to the problem of preference modelling in combinatorial domains.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Preference Handling in Combinatorial Domains: From AI to Social Choice. *AI Magazine*, 29(4):37–46, 2008.

Criteria for Choosing a Language

At least the following questions should be addressed when choosing a representation language:

- *Cognitive relevance:* How close is a given language to the way in which humans would express their preferences?
- *Elicitation:* How difficult is it to elicit the preferences of an agent so as to represent them in the chosen language?
- *Expressive power:* Can the chosen language encode all the preference structures we are interested in?
- *Succinctness:* Is the representation of (typical) structures succinct? Is one language more succinct than the other?
- *Complexity:* What is the computational complexity of related decision problems, such as comparing two alternatives?

Weighted Goals

To exemplify the research agenda in preference representation, let us look at one framework in some detail.

Identify $\mathcal{G}$ with a set of propositional variables $PS = \{X_1, \ldots, X_p\}$: $X_i =$ "I get item g_i". So need to model utility functions $u : 2^{PS} \to \mathbb{R}$.

Let $\mathcal{L}_{PS}$ be the propositional language defined over PS.

Use formulas in $\mathcal{L}_{PS}$ to express *goals*. Give each goal a numerical *weight*. A *goalbase* $G = \{(\varphi_i, w_i)\}_i$ is a set of weighted goals.

Each goalbase G generates a utility function u_G:

$$u_G(M) = \sum_{(\varphi,w)\in G[M]} w \quad \text{where } G[M] = \{(\varphi, w) \in G \mid M \models \varphi\}$$

That is, the utility of a model M is the sum of the weights of the formulas it satisfies.

A Family of Languages

Recall: $\{(\varphi_i, w_i)\}_i$ induces $u : M \mapsto \sum\{w_i \mid M \models \varphi_i\}$ (multiset)

By imposing different restrictions on formulas and/or weights we can design different representation languages.

Regarding *formulas*, we may consider restrictions such as:

- *positive* formulas (no occurrence of $\neg$)
- *clauses* and *cubes* (disjunctions and conjunctions of literals)
- *k-formulas* (formulas of length $\leqslant k$), e.g. 1-formulas = literals
- combinations of the above, e.g. k-pcubes

Regarding *weights*, interesting restrictions would be $\mathbb{R}^+$ or $\{0, 1\}$.

If $H \subseteq \mathcal{L}_{PS}$ is a restriction on formulas and $H' \subseteq \mathbb{R}$ a restriction on weights, then $\mathcal{L}(H, H')$ is the language conforming to H and H'.

Properties

We are interested in the following types of questions:

- Are there restrictions on goalbases such that the utility functions they generate enjoy natural structural properties?
- Are some goalbase languages more succinct than others?
- What is the computational complexity of reasoning about preferences expressed in a given language?

The results on the following slides are from Uckelman et al. (2009). More details in Joel Uckelman's PhD thesis.

J. Uckelman, Y. Chevaleyre, U. Endriss, and J. Lang. Representing Utility Functions via Weighted Goals. *Mathematical Logic Quarterly*, 55(4):341–361, 2009.

J. Uckelman. *More Than the Sum of its Parts: Compact Preference Representation over Combinatorial Domains*. PhD thesis, ILLC, University of Amsterdam, 2009.

Expressive Power

We first give an example for a language that is fully expressive:

Proposition 2 (Expressivity of pcubes) *$\mathcal{L}(pcubes, \mathbb{R})$, the language of positive cubes, can express all utility functions.*

Proof: Let $u : 2^{PS} \to \mathbb{R}$ be any utility function. Define goalbase G:

$(\top, w_\top)$ with $w_\top = u(\emptyset)$
(p, w_p) with $w_p = u(\{p\}) - w_\top$
$(p \wedge q, w_{p,q})$ with $w_{p,q} = u(\{p, q\}) - w_p - w_q - w_\top$ ...
$(\bigwedge P, w_P)$ with $w_P = u(P) - \sum_{Q \subset P} w_Q$

Clearly, G thus defined will generate the function u. ✓

Observe that the proof also demonstrates that $\mathcal{L}(pcubes, \mathbb{R})$ has a *unique* way of representing any given function.

$\mathcal{L}(cubes, \mathbb{R})$, for example, is also fully expressive but *not unique:*
$\{(p \wedge q, 5), (p \wedge \neg q, 5), (\neg p \wedge q, 3), (\neg p \wedge \neg q, 3)\} \equiv \{(\top, 3), (p, 2)\}$

Expressive Power: Modular Functions

A function $u : 2^{PS} \to \mathbb{R}$ is *modular* if for all $M_1, M_2 \subseteq 2^{PS}$ we have:

$$u(M_1 \cup M_2) = u(M_1) + u(M_2) - u(M_1 \cap M_2)$$

Here's a nice characterisation of the modular functions:

Proposition 3 (Expressivity of literals) *$\mathcal{L}(literals, \mathbb{R})$ can express all modular utility functions, and only those.*

Proof sketch: Modular functions are like *additive* functions, i.e., functions u with $u(M) = \sum_{x \in M} u(\{x\})$, except that modularity also permits non-zero values for $\emptyset$.

Easy to see that $\mathcal{L}(atoms, \mathbb{R})$ expresses exactly the additive functions.

So $\mathcal{L}(atoms \cup \{\top\}, \mathbb{R})$ expresses exactly the modular functions.

To see that adding negation does not increase expressive power, observe that $G \cup \{(\neg\varphi, w)\} \equiv G \cup \{(\top, w), (\varphi, -w)\}$. ✓

Relative Succinctness

If two languages can express the same class of utility functions, which should we use? An important criterion is *succinctness*.

Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages that can define all utility functions belonging to some class $\mathcal{U}$.

We say that $\mathcal{L}'$ is at least as succinct as $\mathcal{L}$ ($\mathcal{L} \preceq \mathcal{L}'$) over $\mathcal{U}$ if there exist a mapping $f : \mathcal{L} \to \mathcal{L}'$ and a *polynomial* function p such that for all expressions $G \in \mathcal{L}$ with the corresponding function $u_G \in \mathcal{U}$:

- $G \equiv f(G)$ (i.e., they represent the same function: $u_G = u_{f(G)}$);
- and $size(f(G)) \leqslant p(size(G))$ (polysize reduction).

$\mathcal{L}$ is *less succinct* than $\mathcal{L}'$ ($\mathcal{L} \prec \mathcal{L}'$) iff $\mathcal{L} \preceq \mathcal{L}'$ and not $\mathcal{L}' \preceq \mathcal{L}$.
Equivalence ($\sim$) and *incomparability* ($\bowtie$) are defined accordingly.

If left implicit, then $\mathcal{U}$ is the intersection of the ranges of $\mathcal{L}$ and $\mathcal{L}'$.

The Effect of Negation

We have seen that positive cubes are fully expressive. Hence, $\mathcal{L}(pcubes, \mathbb{R})$ and $\mathcal{L}(cubes, \mathbb{R})$ have the same expressivity.

Proposition 4 (Succinctness) $\mathcal{L}(pcubes, \mathbb{R}) \prec \mathcal{L}(cubes, \mathbb{R})$.

Proof: Clearly, $\mathcal{L}(pcubes, \mathbb{R}) \preceq \mathcal{L}(cubes, \mathbb{R})$, because any positive cube is also a cube.

Now consider u with $u(\emptyset) = 1$ and $u(M) = 0$ for all $M \neq \emptyset$:

- $G = \{(\neg p_1 \wedge \cdots \wedge \neg p_n, 1)\} \in \mathcal{L}(cubes, \mathbb{R})$ has *linear* size and generates u.
- $G' = \{(\bigwedge X, (-1)^{|X|}) \mid X \subseteq PS\} \in \mathcal{L}(pcubes, \mathbb{R})$ has *exponential* size and also generates u.

But there can be not better way of expressing u in pcubes, because we have seen that pcube representations are *unique*. ✓

Computational Complexity

Other interesting questions concern the complexity of reasoning about preferences. Consider the following decision problem:

$\textsc{Max-Utility}(H, H')$
Instance: Goalbase $G \in \mathcal{L}(H, H')$ and $K \in \mathbb{Z}$
Question: Is there an $M \in 2^{PS}$ such that $u_G(M) \geqslant K$?

Some basic results are straightforward:

- $\textsc{Max-Utility}(H, H')$ is *in NP* for any $H \subseteq \mathcal{L}_{PS}$ and $H' \subseteq \mathbb{Q}$, because we can always check $u_G(M) \geqslant K$ in polynomial time.
- $\textsc{Max-Utility}(\mathit{forms}, \mathbb{Q})$ is *NP-complete*, certainly if we do not assume that formulas are satisfiable (reduction from $\textsc{Sat}$).

More interesting questions would be: are there either (1) "large" sublanguages for which $\textsc{Max-Utility}$ is still polynomial, or (2) "small" sublanguages for which it is already NP-hard?

Two Complexity Results

Proposition 5 $\textsc{Max-Utility}(k\text{-}cubes, \mathbb{Q})$ *is NP-complete, even for* $k = 2$ *and assuming all formulas are satisfiable.*

Proof: By reduction from $\textsc{Max2Sat}$ (NP-hard): "Given a set of 2-clauses, is there a satisfiable subset with cardinality $\geqslant K$?". Given a $\textsc{Max2Sat}$ instance, for each clause $p \vee q$ create a goal $(\neg p \wedge \neg q, -1)$. Add $(\top, N)$, where N is the number of clauses. Answer YES for the $\textsc{Max2Sat}$ instance *iff* maximal utility is $\geqslant K$. ✓

Proposition 6 $\textsc{Max-Utility}(literals, \mathbb{Q})$ *is in P.*

Proof: Assuming that G contains every literal exactly once (possibly with weight 0), making p true *iff* the weight of p is greater than the weight of $\neg p$ results in a model with maximal utility. ✓

Integer Programming

Solving fair division problems requires sophisticated algorithm design. However, for medium-size problems sometimes (highly optimised) off-the-shelf tools based on *integer programming* can be sufficient.

IP tools can solve problems presented in the following form:

Maximise *objective function* $\sum_{i=1}^{n} w_i \cdot x_i$ subject to

- constraints of the form $\sum_{i=1}^{n} a_i \cdot x_i \leqslant b$
- and all variables x_i taking non-negative integer values.

A *mixed* IP also allows for non-integer variables.

In principle, any problem in NP can be modelled as an integer program. The challenge is to find a (good) translation.

An easy introduction is available in the book by Shoham and Leyton-Brown (2009).

Y. Shoham and K. Leyton-Brown. *Multiagent Systems*. CUP, 2009.

Maximising Utilitarian Social Welfare

Let us design an IP algorithm to maximise *utilitarian social welfare* when utilities are modelled as *positively weighted cubes* $[= \mathcal{L}(cubes, \mathbb{R}^+)]$.

IP variables:

- $x_{ij} \in \{0, 1\}$: "agent i gets item j"
- $y_{ik} \in \{0, 1\}$: "the kth goal of agent i is satisfied"

Let w_{ik} be the weight of the kth goal of agent i. Note that social welfare is given by $\sum w_{ik} \cdot y_{ik}$ (sum of weights of satisfied goals).

Hence, our objective is to *maximise* $\sum w_{ik} \cdot y_{ik}$ *subject to* two constraints:

(1) Each item j is allocated to (exactly) one agent: $\sum_{i \in \mathcal{N}} x_{ij} = 1$ for all j

(2) For a goal (a cube) to be true, all its conjuncts need to be true:

- $y_{ik} \leqslant x_{ij}$ for every agent i, goal k (within i's goalbase), and item j such that item j is a *positive literal* in goal k
- $y_{ik} \leqslant 1 - x_{ij}$ for every agent i, goal k (within i's goalbase), and item j such that item j is a *negative literal* in goal k

Maximising Egalitarian Social Welfare

An algorithm using (mixed) *integer programming* for maximising *egalitarian* social welfare with utilities represented in the *XOR-language:*

XOR-language means: Agent i submits n_i atomic bids $\langle B_{ij}, u_{ij} \rangle$ with $B_{ij} \subseteq \mathcal{G}$ and $u_{ij} \in \mathbb{R}^+$. Then utility of $B \subseteq \mathcal{G}$ is $\max\{u_{ij} \mid B_{ij} \subseteq B\}$.

IP variables: $x_{ij} \in \{0, 1\}$ ("agent i gets jth bundle"); $y \geqslant 0$ $(= \mathrm{SW}_{\mathrm{egal}})$

IP algorithm: *maximise y subject to* three constraints:

(1) Each good gets allocated to at most one agent:

$$(\forall k \leqslant |\mathcal{G}|) \quad \sum_{i \in \mathcal{N}} \sum_{j=1}^{n_i} [k \in B_{ij}] \cdot x_{ij} \quad \leqslant \quad 1, \quad \text{where } [k \in B_{ij}] \in \{0, 1\}$$

(2) Each agent receives at most one bundle specified in their XOR-bid:

$$(\forall i \in \mathcal{N}) \quad \sum_{j=1}^{n_i} x_{ij} \quad \leqslant \quad 1$$

(3) Egalitarian social welfare is at most equal to any individual utility:

$$(\forall i \in \mathcal{N}) \quad y \quad \leqslant \quad \sum_{j=1}^{n_i} u_{ij} \cdot x_{ij}$$

Summary

We have discussed the problem of finding an allocation of goods to agents that satisfies a suitable fairness criterion.

- *Complexity:* computing a socially optimal allocation is typically a hard problem, unless we are willing to make stromg restrictive assumptions about the nature of preferences
- *Algorithms:* still, algorithm design for fair division is often feasible
- *Representation:* the choice of language used to model preferences (i.e., usually utility functions) is central

As an aside, we have discussed problems in *preference representation* that are not only relevant to fair division:

- *Expressive power*
- *Succinctness*
- *Complexity*

What next?

Next we will review the *distributed approach* to the fair allocation of indivisible goods in detail.