Fair division of indivisible goods under risk

Charles Lumet **Sylvain Bouveret** **Michel Lemaître**
Onera DTIM. 2, avenue Édouard Belin
31055 Toulouse Cedex 4 – FRANCE
first.last@onera.fr

Abstract

We study the problem of fairly allocating a set of indivisible goods to a set of agents having additive preferences. More precisely, we consider the problem in which each object can be in two possible states: good or bad. We further assume that the actual object state is not known at allocation time, but that the decision-maker knows the probability for each object to be in each state. We propose a formal model of this problem, based on the notions of *ex-ante* and *ex-post* fairness, and we propose some algorithms aiming at computing optimal allocations in the sense of *ex-post* egalitarianism, the efficiency of these algorithms being tested on random instances.

1 Introduction

The problem of allocating a set of indivisible goods to a set of agents arises in a wide range of applications including, among others, auctions, divorce settlements, frequency allocation, airport traffic management, fair and efficient exploitation of Earth observing satellites [8]. In many such real-world problems, one needs to find *fair* solutions where fairness refers to the need for compromises between the agents (often antagonistic) objectives.

While most works (see *e.g.* [4] for a survey on multiagent resource allocation) on fair division typically assume that the agents are able to evaluate their preferences (ranking, utility function) over the sets of objects at stake before the beginning of the allocation process, it might not always be the case, and the actual value (or state) of some given objects may depend on exogenous factors and not be known by the agents beforehand. This is the case for example in the fair share of a constellation of Earth observing satellites [8], as the weather conditions on a given area, which are only known with a given probability when the allocation is decided, can dramatically reduce the quality of the observation, and, in the end the utility of an observation for an agent.

Uncertainty (or, more precisely, risk) issues in collective decision making have been studied for example by Myerson [10] and more recently Gajdos and Tallon [5]. However, to the best of our knowledge, this problem has never been considered from a computational point of view, except within the combinatorial auctions framework, when one wants to minimize the influence in terms of revenue of potential bids withdrawals [7]. Our work aims at bridging this gap.

In this article, we make three main assumptions. (i) The allocation is *centralized*, that is, it is decided and computed by a central benevolent authority, according to the agents' individual preferences. (ii) Each object can only be in two possible conditions (good or bad). The actual condition of each object is only known with a given probability when the allocation is decided, but is known for sure when the objects are actually allocated to the agents. (iii) The agents have non exogenous additive preferences over the objects. In other words, the preferences of each agent are represented by a set of weights, standing for the utility (or satisfaction) she enjoys for each single object. The utility of an agent for a subset of objects S is then given by the sum of the weights of all the objects in S that are in good condition (we assume that a bad object has absolutely no value for the agent who receives it).

Even if this framework seems restrictive, we advocate that it is worth studying for the following reasons. Firstly, the additivity assumption is very natural as soon as preferences over sets of objects have to be represented in a compact way. Secondly, in many real-world problems, uncertainty can be defined "object-wise" and thus can be very naturally modeled as we suggest. Finally, as we show in this paper, despite its apparent simplicity, our framework raises non trivial computational issues.

This article is structured as follow. In Section 2, we introduce our framework for fair division of indivisible goods under risk. In Section 3, we mainly focus on the computation of optimal or good ex-post egalitarian allocations and we propose two algorithms to solve this problem. Finally, we compare the efficiency of these algorithms on random instances in Section 4.

2 Framework

2.1 Model

In the following, we use lower case bold font to represent vectors and upper case bold font to represent matrices.

A finite set of indivisible *objects* $\mathcal{O} = \{1, \ldots, l\}$ must be allocated to a finite set of *agents* $\mathcal{A} = \{1, \ldots, n\}$. An *allocation decision* (or simply *allocation*) is a vector of shares $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_n \rangle$ where $\pi_i \subseteq \mathcal{O}$, and $j \in \pi_i$ iff object j

has been given to agent i. The set of feasible decisions is $\mathcal{D} = \{\boldsymbol{\pi},\ i \neq i' \Rightarrow \pi_i \cap \pi'_i = \emptyset\}$. We further denote by $\pi_0 = \mathcal{O} \setminus \bigcup_{i \in \mathcal{A}} \pi_i$ the set of non allocated objects.

Each object can be either in *good* condition or in *bad* condition. The objects conditions are known only after the allocation has been made, but the decision-maker is nevertheless given probabilistic information: to each object $j \in \mathcal{O}$, is attached a binary random variable X_j which can take value in $\{good, bad\}$. We assume the existence of a vector $\mathbf{p} \in [0;1]^l$ giving each object probabilities $p_j = \mathbb{P}(X_j = good)$, and $\overline{p}_j = 1 - p_j = \mathbb{P}(X_j = bad)$. Variables $X_j,\ j \in \mathcal{O}$ are assumed to be independent.

Each state of nature in the problem is therefore characterized by the set of objects in good condition (the other ones being in bad condition). Let $\mathcal{S} = \{1, \ldots, k\}$ be the set of the possible *states of nature* (where $k = 2^l$) ; to each state s of this set, one can relate the set $good(s) \subseteq \mathcal{O}$ of objects in good condition when state of nature s happens. $\mathcal{S}$ is provided with a probability distribution, fully characterized by coefficients p_j.

$$\forall s \in \mathcal{S},\ \Pr(s) = \prod_{j \in good(s)} p_j \prod_{j \notin good(s)} \overline{p}_j \tag{1}$$

Computing an acceptable allocation for such a problem requires the decision-maker to know about the tastes of the agents for the objects. These *preferences* are numerically expressed by the agents in the form of *utility functions*, which, for each state s, map each decision $\boldsymbol{\pi}$ to a numerical value $u_{i,s}(\boldsymbol{\pi})$ conveying the attractiveness of the decision for the agent i if this state of nature happens. This utility is built upon the specification of *weights* for each agent to each object ; the weight w_{ij} represents the intensity of agent i's preference for object j ; we assume that an agent utility for a decision and a state of nature are given by the sum of the weights of the objects *in good condition* received by said agent: agents have *additive* preferences over the objects, and each object in bad condition gives no extra utility to the agent it is allocated to.

$$\forall i \in \mathcal{A}, \forall s \in \mathcal{S},\ u_{i,s}(\boldsymbol{\pi}) = \sum_{j \in good(s) \cap \pi_i} w_{ij} \tag{2}$$

Let us now define an instance of the problem studied in this article.

Definition 1 (Resource allocation problem under risk) *An instance of a resource allocation problem under risk is a tuple $(\mathcal{A}, \mathcal{O}, \mathbf{p}, \mathbf{W})$, where $\mathcal{A} = \{1, \ldots, n\}$ is a set of agents, $\mathcal{O} = \{1, \ldots, l\}$ is a set of objects, $\mathbf{p} \in [0;1]^l$ expresses the probability for each object to be in good condition, and $\mathbf{W}$ is the n-lines l-columns matrix of weights given to the objects by the agents.*

Table 1 shows an example of a resource allocation problem under risk, with the probabilities of each possible state of nature (line 2) and utility profiles associated with a given decision (lines 3 and 4).

2.2 The *timing effect*

For a given state of nature, a decision quality depends on the level of satisfaction of all the agents. A classical way to define this quality is to *aggregate* the agents utility vector with a commutative and increasing *collective utility function* $\mathfrak{M} : (\mathbb{R}^+)^n \to \mathbb{R}^+$, which measures social welfare. Two classical choices are $\mathfrak{M} = \sum$ and $\mathfrak{M} = \min$, which have been at the root of classical utilitarianism on the one hand, and egalitarianism on the other hand. The latter promotes equity, since best decisions are those which satisfy the most the poorest agent, whereas the former promotes a kind of efficiency which aims at giving objects to the agents producing the most utility, without any concern for equity. A general survey on collective utility functions can be found in [9]. In the following we will write $\mathfrak{M}_{i \in \mathcal{A}} u_i$ for $\mathfrak{M}(\mathbf{u})$.

In the same manner, we aggregate agent utilities in the different states of nature using the classical expected utility (even if other choices could be made).

In order to map a unique numerical value to each decision, and depending on whether aggregation is first made over states of nature and then over agents or the other way around, we obtain two different functions [6; 10] : $acu : \mathcal{D} \to \mathbb{R}^+$, defined in (3), is called *ex-ante* collective utility and $pcu : \mathcal{D} \to \mathbb{R}^+$, defined in (4) is called *ex-post* collective utility.

$$\forall \boldsymbol{\pi} \in \mathcal{D},\ acu(\boldsymbol{\pi}) = \mathop{\mathfrak{M}}_{i \in \mathcal{A}} \left(\sum_{s \in \mathcal{S}} \Pr(s) \cdot u_{i,s}(\boldsymbol{\pi}) \right) \tag{3}$$

$$\forall \boldsymbol{\pi} \in \mathcal{D},\ pcu(\boldsymbol{\pi}) = \sum_{s \in \mathcal{S}} \Pr(s) \cdot \left(\mathop{\mathfrak{M}}_{i \in \mathcal{A}} u_{i,s}(\boldsymbol{\pi}) \right) \tag{4}$$

Harsanyi [6] shows that the only aggregation functions for which *ex-post* and *ex-ante* utilities coincide are linear or affine, which entails that, on the contrary, each equity-prone collective aggregation function will give different *ex-ante* and *ex-post* utilities. There therefore exists a conflict – known as *timing effect* – between the *ex-post* approach on the one hand, which considers the expected social welfare and the *ex-ante* approach on the other hand, which considers the social welfare measured with expected utilities.

2.3 *Ex-ante* versus *ex-post* utility

Even if no link exists *a priori* between *ex-post* and *ex-ante* utilities for a given decision, one can show that, under some mild assumption on the collective aggregation function, the *ex-ante* collective utility is always greater than the *ex-post* one.

This is especially true in the egalitarian case, where Proposition 1 is a direct application of the triangular inequality for function min.

Proposition 1 *Let $\mathfrak{M} = \min$ be the egalitarian collective aggregation operator. Then, the following inequality stands:*

$$\forall \boldsymbol{\pi} \in \mathcal{D},\ \ pcu(\boldsymbol{\pi}) \leq acu(\boldsymbol{\pi}) \tag{5}$$

3 Computing *ex-ante* and *ex-post* optimal allocations

In this section, we will deal with the problems of finding an allocation maximizing *ex-ante* and *ex-post* utilities. In the following, we will restrict to the classical egalitarian criterion –

s	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{1,3\}$	$\dots$	$\{2,3,4\}$	$\{1,2,3,4\}$	$\mathfrak{E}(\mathbf{u})$
$\Pr(s)$	0.016	0.004	0.016	0.004	0.064	0.016	0.064	$\dots$	0.256	0.064	—
$u_{1,s}$	0	10	0	0	7	10	10	$\dots$	7	17	9.4
$u_{2,s}$	0	0	8	4	0	8	4	$\dots$	12	12	8.4
$\mathfrak{M}(\mathbf{u})$	0	0	0	0	0	8	4	$\dots$	7	12	8.4 / 6.448

Table 1: Utility profile and *ex-ante* and *ex-post* utility computation for a problem with 2 agents, 4 objects, probabilities $\mathbf{p} = \langle 0.8, 0.5, 0.5, 0.2\rangle$, weights $\mathbf{w_1} = \langle 10, 2, 4, 7\rangle$ and $\mathbf{w_2} = \langle 3, 8, 4, 10\rangle$, decision $\boldsymbol{\pi} = \langle\{1,4\},\{2,3\}\rangle$, $\mathfrak{M} = \min$. Here, $pcu(\boldsymbol{\pi}) = 6.448$ and $acu(\boldsymbol{\pi}) = 8.4$ (see Section 2.2).

that is, $\mathfrak{M} = \min$ – which is worthy of attention in this context, as it represents exactly the expected utility of the poorest agent.

Ex-ante collective utility *Ex-ante* collective utility is defined by Equation (3) ; introducing some "expected weights" $\tilde{w}_{ij} = p_j w_{ij}$, the expression can be simplified: $\forall \boldsymbol{\pi} \in \mathcal{D}$, $acu(\boldsymbol{\pi}) = \mathfrak{M}_{i\in\mathcal{A}}\, \tilde{u}_i(\boldsymbol{\pi})$ where $\tilde{u}_i(\boldsymbol{\pi}) = \sum_{j\in\pi_i} \tilde{w}_{ij}$.

Thus, since the $\tilde{w}_{ij}$ coefficients can be computed in mere linear-time, the problem of finding an *ex-ante* optimal allocation can be reduced to a classical risk-free resource allocation problem with additive preferences, known as the Santa Claus problem [1]. Since this problem has already been tackled in litterature, we focus in the following on the *ex-post* optimization problem.

Ex-post collective utility A basic algorithm for *computing* the *ex-post* collective utility, directly applying formula (4), requires the computation of the collective utility in each possible state (*i.e* each column in Table 1), that is, the enumeration of an exponential number of values. Clearly, computing the *ex-post* collective utility of a given decision is in #P, but we do not know yet if it is complete for this class (even if we strongly believe it).[1]

However, as soon as all the objects allocated to an agent are in bad condition, the utility of this agent is zero, and so is the collective utility, whatever states the remaining objects are in. Algorithm 1, which computes the *ex-post* collective utility for a given decision, is based on this remark: it quickly "eliminates" such states of nature, whose enumeration is unnecessary. A function SORT is used in this algorithm in the following manner: $\text{SORT}(\mathbf{u}, f)$ returns a vector $\mathbf{u}^{\uparrow}$ which is a permutation of the values of $\mathbf{u}$, such that $i < i' \Rightarrow f(u^{\uparrow}_i) \leq f(u^{\uparrow}_{i'})$.

The *optimization* problem is tackled with both exact and approximate algorithms.

The exact approach is based on a classic *branch and bound* algorithm. Efficiency of such an algorithm highly depends on its ability to quickly detect poor allocations in order to "cut" significant parts of the search tree. A cut must be based on an easy-to-compute function which maximizes the value to be optimized.

Ex-post utility computation is time-consuming, and is therefore not used as a cut strategy, but only to assess complete allocations.

[1] Of course, computing an optimal allocation is even harder.

Algorithm 1: EXPOST function: *ex-post* collective utility computation

Data: A complete allocation $\boldsymbol{\pi}$
Result: *Ex-post* collective utility $pcu(\boldsymbol{\pi})$

$\boldsymbol{\pi}^{\uparrow} \leftarrow \text{SORT}(\langle \pi_1, \dots, \pi_n\rangle, \mathcal{X} \mapsto |\mathcal{X}|)$;
return BRANCH($\langle 0, \dots, 0\rangle$, 1, $\boldsymbol{\pi}^{\uparrow}$, 1);

Function BRANCH($\mathbf{u}$, pr, $\langle \rho_1, \dots, \rho_n\rangle$, i)

Data: A utility vector $\mathbf{u}$, a number $pr \in [0;1]$, a vector of shares $\boldsymbol{\rho}$, an agent i
Result: *Ex-post* collective utility

if $\rho_i = \emptyset$ **then**
  if $i = n$ **then**
    return $\min(\mathbf{u}) \times pr$;
  else
    if $u_i = 0$ **then return** 0;
    return BRANCH($\mathbf{u}$, pr, $\boldsymbol{\rho}$, $i+1$);
else
  $j \leftarrow$ arbitrary object in ρ_a;
  $\boldsymbol{\rho'} \leftarrow \langle \dots, \rho_{i-1}, \rho_i \setminus \{j\}, \rho_{i+1}, \dots\rangle$;
  $\mathbf{u'} \leftarrow \langle \dots, u_{i-1}, u_i + w_{ij}, u_{i+1}, \dots\rangle$;
  return BRANCH($\mathbf{u}$, $pr \cdot \overline{p}_j$, $\boldsymbol{\rho'}$, i) + BRANCH($\mathbf{u'}$, $pr \cdot p_j$, $\boldsymbol{\rho'}$, i);

Instead, we use inequality (5) and choose function $\overline{acu}$ as upper bound ; $\overline{acu}$ represents the *ex-ante* utility of a virtual decision which would allocate to *all the agents* the set of objects (denoted π_0) that are not yet allocated by the current decision $\boldsymbol{\pi}$:

$$\overline{acu}(\boldsymbol{\pi}) = \min_{i\in\mathcal{A}}\Big(\sum_{j\in\pi_i} \tilde{w}_{ij} + \sum_{j\in\pi_0} \tilde{w}_{ij}\Big)$$

Even though $\overline{acu}$ is clearly a rough upper bound, this value remains fast to compute.

At this point, it seemed interesting to look for a intermediate function, which would be a better upper bound than $\overline{acu}$ and faster to compute than pcu. The idea is to compute utility in an *ex-post* manner for a subset Ω of objects, and in an *ex-ante* manner for the other ones ; we introduce in this sense the *mixed utility*, denoted $mu_{i,s}$ for a given agent i and a given state of nature s.

Algorithm 2: Stochastic greedy

Data: A risky fair division problem instance.
Result: A good allocation, according to *ex-post* collective utility

$Stock \leftarrow \emptyset$;
$\boldsymbol{\pi}^\star = \langle \pi_1^\star, .., \pi_l^\star \rangle \leftarrow \langle \emptyset, \ldots, \emptyset \rangle$;
$pcu^\star \leftarrow 0$;
$i \leftarrow 0$;
while *given time has not elapsed* **do**
  $\boldsymbol{\pi} \leftarrow$ BUILDALLOCATION() ;
  if $acu(\boldsymbol{\pi}) \geq pcu^\star$ **then**
    $pcu_{app} \leftarrow$ EXPOSTA($\boldsymbol{\pi}$) ;
    if $pcu_{app} > \min_{\boldsymbol{\pi} \in Stock}$(EXPOSTA($\boldsymbol{\pi}$)) **then**
      STORE($\boldsymbol{\pi}$) ;
  $i \leftarrow i + 1$;
  if $i = nbStorage \times nbBeforeExactComputation$ **then**
    for $\boldsymbol{\pi} \in Stock$ **do**
      $pcu \leftarrow$ EXPOST($\boldsymbol{\pi}$) ;
      if $pcu > pcu^\star$ **then**
        $\boldsymbol{\pi}^\star \leftarrow \boldsymbol{\pi}$;
        $pcu^\star \leftarrow pcu$;
    $Stock \leftarrow \emptyset$;
    $i \leftarrow 0$;
return $\boldsymbol{\pi}^\star$;

Procedure BUILDALLOCATION()

$\mathbf{u} = \langle u_1, .., u_n \rangle \leftarrow \langle 0, \ldots, 0 \rangle$;
$\boldsymbol{\pi} = \langle \pi_1, .., \pi_n \rangle \leftarrow \langle \emptyset, \ldots, \emptyset \rangle$;
while $\exists j \in \pi_0$ **do**
  $\mathring{i} \leftarrow \text{argmin}_{i \in \mathcal{A}}(alter(u_i))$;
  $\mathring{j} \leftarrow \text{argmax}_{j \in \pi_0}(alter(w_{\mathring{i}j}))$;
  $\pi_{\mathring{i}} \leftarrow \pi_{\mathring{i}} \cup \mathring{j}$;
  $u_{\mathring{i}} \leftarrow u_{\mathring{i}} + w_{\mathring{i}\mathring{j}}$;
return $\boldsymbol{\pi}$;

$$mu_{i,s}(\boldsymbol{\pi}, \Omega) = \sum_{\substack{j \in \Omega \cap good(s) \\ j \in \pi_i}} w_{ij} + \sum_{\substack{j \notin \Omega \\ j \in \pi_i}} \tilde{w}_{ij}$$

The mixed utility represents the utility of an agent which considers that objects outside Ω are for sure in good condition and which assigns them weights $\tilde{w}_{ij}$. The *mixed collective utility* is defined by Equation (6) as the *ex-post* collective utility from individual mixed utilities.

$$mcu(\boldsymbol{\pi}, \Omega) = \sum_{s \in \mathcal{S}} \Pr(s) \cdot \min_{i \in \mathcal{A}} mu_{i,s}(\boldsymbol{\pi}, \Omega) \quad (6)$$

Note that individual mixed utilities are independent from the states of the objects outside Ω. The expected value computation in Equation (6) can therefore boil down to the formula (7), where for s s.t. $good(s) \subseteq \Omega$, one denotes $\Pr(s, \Omega) = \prod_{j \in good(s)} p_j \prod_{j \in \Omega \setminus good(s)} \overline{p}_j$ the probability for objects in Ω to be in the state specified by s, whatever states the other objects are in. The number of states of nature to list is halved for each object outside Ω, which shows the algorithmic point of mixed collective utility.

$$mcu(\boldsymbol{\pi}, \Omega) = \sum_{\substack{s \in \mathcal{S} \\ good(s) \subseteq \Omega}} \Pr(s, \Omega) \cdot \min_{i \in \mathcal{A}} um_{i,s}(\boldsymbol{\pi}, \Omega) \quad (7)$$

We can prove that mixed collective utility lies between *ex-post* and *ex-ante* collective utilities (proof omitted due to lack of space).

Proposition 2 (Mixed collective utility) *For all decision $\boldsymbol{\pi} \in \mathcal{D}$ and for all subset $\Omega \subseteq \mathcal{O}$, one has:*

$$acu(\boldsymbol{\pi}) \geq mcu(\boldsymbol{\pi}, \Omega) \geq pcu(\boldsymbol{\pi}) \quad (8)$$

Our *branch and bound* algorithm uses the upper bound function $\overline{acu}$ to cut within the body of research: the function mcu is used only when a complete allocation has been made, to avoid unnecessary *ex-post* collective utility computations.

Dynamic heuristics are used as suggested by [2] : each object will be firstly allocated to the poorest agent (i.e. the one whose expected utility is currently the lower) ; when a new object has to be allocated, the one preferred by the currently poorest agent is chosen among those still left.

The approximate algorithm (Algorithm 2) is based upon a greedy stochastic algorithm [3]. As soon as a complete allocation has been built, an approximate *ex-post* collective utility computation is made by EXPOSTA, in order to decide if the allocation will be stored or not. The approximate computation is made using the mixed collective utility or the Monte-Carlo method (the latter being based on a sequence of random draws in the space of states of nature). A fixed number $nbStorage$ of promising allocations is stored within the course of the algorithm ; if an allocation is better – as far as the approximate computation can tell – than the worst currently stored, the function STORE saves this new allocation (and the other one is deleted if the storage capacity is reached). As soon as $nbStorage \times nbBeforeExactComputation$ allocations have been made, an exact *ex-post* collective utility computation occurs for each stored allocation, and only the best one is kept.

During the building of an allocation, we use randomly biased heuristics, introducing function $alter : \mathbb{R} \rightarrow \mathbb{R}$, such that $\forall y \in \mathbb{R}$, $alter(y) = y \cdot (1 + \phi X)$, where ϕ is a positive real parameter and X a standard normal random variable.

4 Results

Algorithms introduced in this article are implemented using Java and run on random instances, where weights w_{ao} are uniformly drawn in $\{0, 1, .., 99\}$, and probabilities p_j uniformly in $[0; 1]$.

Table 2 and Figure 1 show the results of the exact search algorithm. Four configurations are tested: the algorithm is firstly run with a cut based upon $\overline{acu}$ function only (case (a)), then by using dynamic heuristics (case (b)), next by introducing Algorithm 1 for *ex-post* collective utility computation (case (c)), and finally by adding mixed collective utility cuts (case (d)). Figure 1 shows efficiency of configuration (d),

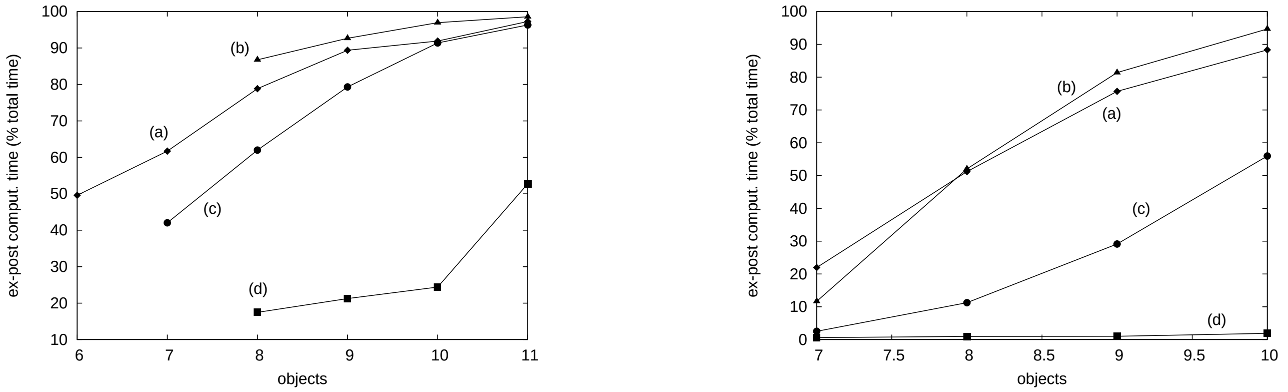


Figure 1: Exact resolution. Duration of *ex-post* collective utility computations, as a percentage of total execution time, for 5 (left) and 7 (right) agents (mean over 100 instances)

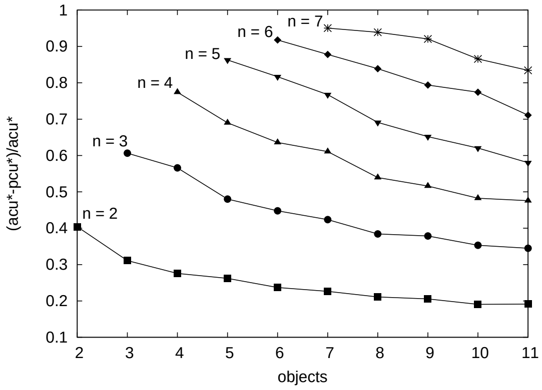


Figure 2: Timing effect influence. The ratio $(acu^\star - pcu^\star)/acu^\star$ varies with the number of objects, for different numbers of agents..

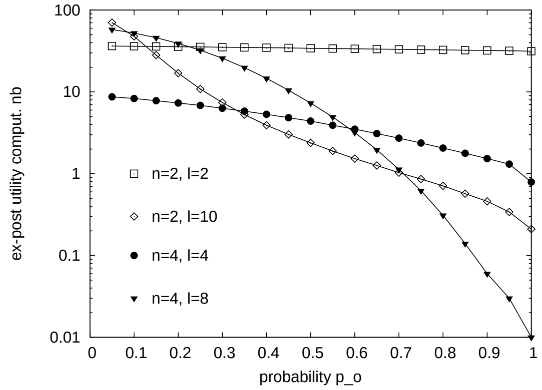
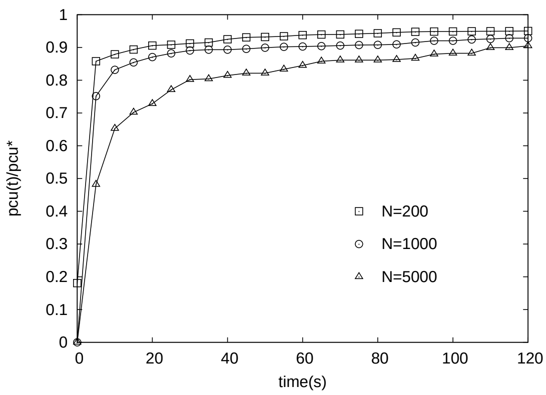
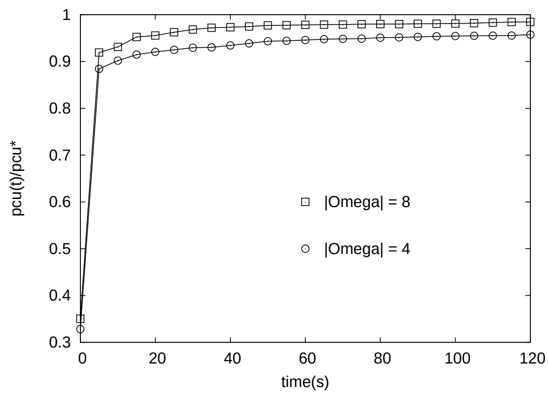


Figure 3: Probabilities influence. Number of totally explored search tree branches, as a percentage of the total number of branches (mean over 100 instances for different numbers of agents and “equaly likely” objects)



(a) Monte-Carlo approximation, for different numbers of draws.



(b) Mixed collective utility approximation, for different sizes of the Ω set.

Figure 4: Approached resolution. Evolution of the best *ex-post* collective utility with time, for two approximation methods (means over 100 instances involving 5 agents and 12 objects).

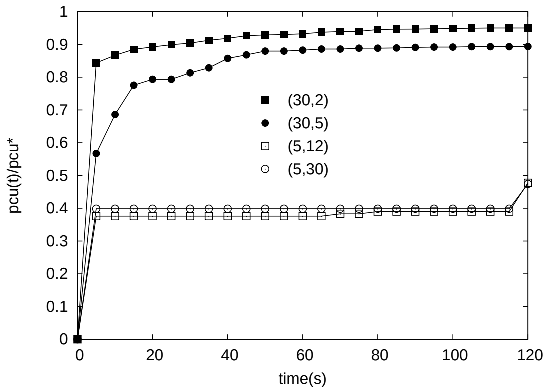


Figure 5: Approached resolution with Monte-Carlo method. *Ex-post* collective utility evolution for different couples $(nbStorage, nbBeforeExactComputation)$ (means over 100 instances involving 5 agents and 12 objects)

in which use of mixed collective utility produces good cuts, and therefore deeply reduces the number of *ex-post* collective utility computations made during the algorithm.

n	l	(a)	(b)	(c)	(d)
5	≤ 9	100	100	100	100
5	10	49	52	89	100
5	11	1	1	10	52
5	≥ 12	0	0	0	0
7	≤ 8	100	100	100	100
7	9	27	47	100	100
7	10	0	1	19	32
7	≥ 11	0	0	0	0

Table 2: Exact resolution. Number of instances solved in 30 seconds (over 100 random instances).

Figure 2 shows influence of the *timing effect*. Relative difference between *ex-post* and *ex-ante* collective utilities increases when the number of agents increases or when the number of objects decreases.

The algorithm efficiency highly depends on probabilities $\mathbf{p}$, which is clearly illustrated by Figure 3. Because of higher proximity between *ex-ante* and *ex-post* collective utilities when the probabilities p_j are closer to 1, cutting strategies are more efficient in this case.

Algorithm 2 is tested on 100 instances ($n = 5$, $l = 12$), for a duration of 2 minutes[2]. Figure 4 illustrates the influence of the approximation methods parameters ; Figure 5 shows the importance of functionnal parameters: the best solution quality significantly increases with the number of allocations stored during the run.

5 Conclusion

In this article, we have introduced a simple model for resource allocation problems under risk. We have shown that, under reasonable hypothesis, *ex-ante* collective utility optimization could be reduced to risk-free optimization, but that *ex-post* optimization seemed to be far more complex. We have proposed the mixed collective utility as groundwork for the building of both an exact and an approximate algorithm.

Algorithms introduced in this article are a first attempt at solving risky resource allocation problems and can most probably be improved. Further work has to be made to characterize the complexity of the *ex-post*-related problems. Moreover, the *ex-post* egalitarian framework shows its limits when $l \leq n$ due to the drawning effect induced by function $\min$. We plan next to extend the model, in order to work with other collective utility aggregations such as the leximin ordering, consider preferential and/or probabilistic dependences between objects, and to embrace a more general notion of risk.

[2]Exact resolution of problems of this size takes 5 to 10 minutes.

Acknowledgements We wish to thank Jérôme Lang and Dídac Busquets for their numerous remarks during the genesis of the present article. Authors would also like to thank the anonymous reviewers for their comments and suggestions.

References

[1] N. Bansal and M. Sviridenko. The santa claus problem. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 31–40. ACM, 2006.

[2] S. Bouveret and M. Lemaître. Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, 173(2):343–364, 2009.

[3] J. L. Bresina. Heuristic-Biased Stochastic Sampling. In *AAAI-96*, pages 271–278, Portland, OR, 1996.

[4] 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.

[5] T. Gajdos and J.-M. Tallon. Fairness under uncertainty. *Economics Bulletin*, 4(18):1–7, 2002.

[6] J. C. Harsanyi. Cardinal welfare, individualistic ethics, and interpersonal comparisons of utility. *Journal of political economy*, 63:309–321, 1955.

[7] A. Holland and B. O'Sullivan. Robust solutions for combinatorial auctions. In *Proc. of the 6th ACM Conf. on Electronic Commerce*, pages 183–192. ACM, 2005.

[8] M. Lemaître, G. Verfaillie, and N. Bataille. Exploiting a common property resource under a fairness constraint: a case study. In *IJCAI-99*, pages 206–211, Stockholm, Sweden, July 1999.

[9] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[10] R. B. Myerson. Utilitarianism, egalitarianism, and the timing effect in social choice problems. *Econometrica*, 49(4):883–897, 1981.