WOAS WORLD ORGANIZATION OF APPLIED SCIENCES JOURNALS

International Journal of Information Technology and Applied Sciences
ISSN (2709-2208)
Int. J. Inf. Tec. App. Sci. 2, No.4 (October-2020)
https://doi.org/10.52502/ijitas.v2i4.13

An algorithm to calculates an allocation maximizing the leximin order on the utility profiles of the agents

Sylvain Bouveret, Michel Lemaître

[1] *Office National d'Études et de Recherche Aérospatiales, Centre de Toulouse, France*
E-mail address: sylvain.bouveret@onera.fr, michel.lemaitre@onera.fr

Abstract: Allocating a limited set of resources equitably and efficiently to agents each with their own preferences is a general problem of considerable significance. Many examples of this problem are commonly found, among which we can cite the construction of schedules, the sharing of communication networks, the management of airport resources involving several companies, the sharing of airspace between different users, sharing of satellite resources. In the context of constraint programming, we propose an algorithm solving the following problem: allocate in an equitable and efficient way a finite set of objects to agents each having their own utilities, under admissibility constraints. The algorithm calculates an allocation maximizing the leximin order on the utility profiles of the agents. We also describe the field of application that motivated this work: the sharing of satellite resources. We extract from it a simple and precise problem of fair allocation, which serves as a basis, thanks to a generator of test sets, for the evaluation of the proposed algorithm. Two implementations of the algorithm are compared, one in "pure" constraint programming, with Choco, the other in mixed linear programming with Cplex.

Keywords: Machine Learning, detection systems.

1. INTRODUCTION

Allocating a limited set of resources equitably and efficiently to agents each with their own preferences is a general problem of considerable significance. Many examples of this problem are commonly found, among which we can cite the construction of schedules, the sharing of communication networks, the management of airport resources involving several companies, the sharing of airspace between different users, sharing of satellite resources.

In this article, we approach this problem with four restrictive assumptions, but which still leave a very wide field of application:

1) the resources are discrete, finite, and come down to distinct, indivisible objects in finite number.
2) An agent preference on eligible allocations is expressed numerically.
3) We are looking for fair and efficient allocations - the meaning of these words will be clarified and discussed later.
4) The search for a satisfactory allocation is carried out centrally by an "arbitrator" who is supposed to be fair and impartial and obeys principles accepted by all agents.

In other words, we are not interested here in allocation or negotiation procedures distributed among agents. This marked economic problem affects several active research fields: Operational Research (OR), Artificial Intelligence (AI), Microeconomics, Social Choice theory. Our contribution draws on these different areas. From the last two we borrow the idea of utility to translate numerical preferences, and the comparison by the leximin order to reflect the requirement of fairness and efficiency. OR and AI provide us with the framework for constraint programming, in which we provide a simple, centralized algorithm for finding leximin-optimal allocations. Numerical preferences and utilities Let be a finite set of admissible alternatives S, in which an arbiter must choose an alternative involving n agent, each having its own preferences. The most classic model of this situation is that of welfarism (see for example [13, 18]). According to this model, which we adopt here, the arbitrator's decision elements are entirely contained in the data, for each agent and for each alternative, of his level of "well-being". This level is measured, in the cardinal version of the model, by a numerical index measuring the individual utility ui (s) of agent i for the alternative s. It is assumed that the individual utilities are comparable between agents (they are given on a common scale of utilities). To each alternative s therefore corresponds a utility profile hu1 (s), . . ., one (s) i and the comparison between two alternatives is made on the sole basis of the two associated profiles.

A convenient way to compare individual utility profiles is to aggregate each into a collective utility index representing the collective welfare of the agent firm. Thus, to each alternative s ∈ S will correspond a collective utility uc (s) = g (u1 (s),..., Un (s)), where g is a well-chosen aggregation function. An optimal decision is one that maximizes this collective utility.

2. FORMEL SETTING

The constraint programming framework is widely used in solving combinatorial problems as diverse as scheduling, planning, and frequency allocation problems. This paradigm is based on the notion of a constraint network. A constraint network is formed by a set of variables X = {x1,..., xp}, of a set of domains D = {dx1,. . . , dxp}, where dxi is a finite set of possible values for xi (we

suppose that dxi ⊂ N, and denote by xi = min (dxi) and xi = max (dxi)), and a set of constraints C. Each constraint C ∈ C specifies a set of allowed tuples R (C) over a set of variables X (C). An instantiation v of a set S of variables is an application which to any variable x ∈ S associates a value v (x) of its domain dx. If S = X, this instantiation is complete, otherwise, it is partial. If S ′ (S, the projection of an instantiation of S on S ′ is the restriction of this instantiation to S ′ and is denoted v ↓ S ′ . An instantiation is consistent if and only if it does not violate any constraint. a constraint network, the problem of existence of a complete instantiation consistent with this constraint network is called the Constraint Satisfaction Problem (CSP) [16] and is NP-complete. solution of the CSP There is a variation of the CSP in optimization problem (resulting from the max-CSP extension of constraint satisfaction problems), in which a variable o plays the role of objective variable. of this optimization problem is a coherent complete instantiation bv of the constraint network such that bv (o) = max {v ′ (o) | v ′ coherent complete instantiation}.

Let - → x = hx1,. . . , xni a vector of integers; we denote - → x ↑ = hx ↑ 1,. . . , x ↑ nor the non-descending ordered version of this vector. We define the leximin order on the integer vectors.

3. PROPOSED ALGORITHM

The principle of Algorithm 1 is to iteratively calculate each component of the vector corresponding to the ordered values of the leximin-optimal instantiation of - → u. For this, we introduce in lines 3 and 4 a vector of optimization variables, the role of each variable yi being to calculate the value of the index i of the leximin-optimal (we will note m = min {ui | 1 ⩽ i ⩽ n} and M = max {ui | 1 ⩽ i ⩽ n}). To each iteration i of loop 6..10, we add a cardinality constraint corresponding to the component being calculated (line 7), and we calculate (line 8) the maximum value ofSupervised learning Supervised learning responds to this need to integrate expert knowledge. Indeed, a supervised detection model is constructed from labeled data provided by the expert: mild events, but also malicious events to guide the detection model. The learning algorithm will automatically look for the points allowing to characterize each of the classes or to discriminate them to build the detection model. Once the detection model is learned on a training dataset, it can be applied automatically to detect malicious events.

Thanks to supervised learning, the security operator supervising the detection system can easily participate in improving the detection model based on the alerts that he analyzes. Indeed, false alerts can be reinjected to correct the detection model and thus avoid generating the same false alerts in the future. The real alerts can also be fed back into the model to let it follow the evolution of the threat. Thus, security experts do not give control of the detection system to an automatic model, but they actively supervise it to improve its performance over time [16].

Also, supervised learning is guided by malicious examples provided by the expert, which reduces the rate of false positives compared to the detection of anomalies. Supervised methods are therefore to be preferred when labeled data are available to train the detection model. However, these methods must be applied taking into account the operational constraints of the detection systems. The detection model must be able to process data in real time, and the false positive rate must remain below a certain threshold to prevent the security operator from being overwhelmed by false alerts.

Finally, the administrator must have confidence in the model to put it into production, and the operator must be able to understand the alerts generated. In the rest of the paper, we give a methodology so that machine learning meets these constraints, and so that it can be integrated into detection systems to better detect new threats. the variable yi such that the current constraint network (which corresponds to the initial network added to the variables yk and the cardinality constraints of the preceding iterations) has a solution. The variable yi is set at this optimal value (line 9) for all subsequent iterations. Line 10 safely restricts the domain of the next variable yi + 1; however, the tests show that it does not significantly influence the computation times, certainly because the constraint propagation is able to filter very quickly this part of the domain of yi + 1.

Moreover, for all j i + 1, dvi + 1 (-! U) ↑ j dvi + 1 (yj) (otherwise at least one of the AtLeast constraints is violated). By noting that an allowable allocation for (X ′ , D ′ , C ′) at iteration i + 1 is also admissible at iteration i (because between two successive iterations we only add a constraint and reduce the domain of a variable), we deduce that we cannot have dvi + 1 (-! u) ↑ j> dvi + 1 (yj). Indeed, in this case, since dvi + 1 (yj) = b vj (yj) (for j <i + 1, the domain of yj is a singleton), dvi +

1 would have been strictly better than b vj for yj at iteration j, and if maximize is correct, it is not possible.

Algorithme 1 : Algorithme de recherche d'une solution leximin-optimale d'un réseau de contraintes.

entrées : un réseau de contraintes $(\mathcal{X}, \mathcal{D}, \mathcal{C})$; un vecteur $\langle u_1, \ldots, u_n \rangle$ de variables de $\mathcal{X}$
sortie : Une solution du problème [LEXIMIN-OPTIMAL] ou «Incohérent»

si $\mathbf{solve}(\mathcal{X}, \mathcal{D}, \mathcal{C}) =$ *«Incohérent»* **alors**
  retourner *«Incohérent»*
$\mathcal{X}' \leftarrow \mathcal{X} \cup \{y_1, \ldots, y_n\}$;
$\mathcal{D}' \leftarrow \mathcal{D} \cup \{[\![m, M]\!], \ldots, [\![m, M]\!]\}$;
$\mathcal{C}' \leftarrow \mathcal{C}$;
pour $i \leftarrow 1$ **à** n **faire**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{AtLeast}(\{u_1 \geq y_i, \ldots, u_n \geq y_i\}, n - i + 1)\}$;
  $\widehat{v} \leftarrow \mathbf{maximize}(y_i, (\mathcal{X}, \mathcal{D}, \mathcal{C}))$;
  $d_{y_i} \leftarrow \{\widehat{v}(y_i)\}$;
  $d_{y_{i+1}} \leftarrow [\![\widehat{v}(y_i), M]\!]$
retourner $\widehat{v}_{\downarrow \mathcal{X}}$

So 8j i + 1, dvi + 1 (-! U) ↑ j = dvi + 1 (yj), which proves the first equality. The extension of b vs which affects the value b vs (-! U) ↑ i + 1 to yi + 1 is feasible at iteration i + 1 (it satisfies, in addition to the other constraints, the AtLeast constraint of the iteration i + 1). So dvi + 1 (yi + 1) b vs (-! U) ↑ i + 1. If we had dvi + 1 (yi + 1)> b vs (-! U) ↑ i + 1, then the projection of dvi + 1 on X would be a solution of this constraint network such that 8j <i + 1 , dvi + 1 (-! u) ↑ j = b vs (-! u) ↑ j and dvi + 1 (-! u) ↑ i + 1> b vs (-! u) ↑ i + 1, so a solution leximin-greater than b vs, which is not possible. We therefore have dvi + 1 (yi + 1) = b vs (-! U) ↑ i + 1, which completes proving (Hi + 1). By induction, we therefore have: (1) cvn is a solution of the constraint network at iteration n, so a fortiori, its projection on X is a solution and (2) for all i, cvn (-! U) ↑ i = b vs (-! u) ↑ i, so cvn (-! u) and b vs (-! u) are leximin indifferent. So the instantiation returned by the algorithm is indeed a solution of the [Leximin-] problem. If constraint programming lends itself particularly well to the implementation of this algorithm, the cardinality meta-constraint used in the algorithm is also expressed in the field of linear programming [10, p.11] thanks to the introduction of n variables 0 ‑ 1 {1,..., n}. The meta-constraint AtLeast ({x1 ⩾ y,..., Xn ⩾ y}, k) is equivalent to the set of linear constraints {x1 + 1y ⩾ y,. . . , xn + ny ⩾ y, Pn i = 1 i ⩽ n - k}.

4. APPLICATION TO A PROBLEM OF SHARING SATELLITE RESOURCES

We now describe the application that motivated this work, and which we used to experiment and evaluate the proposed algorithm in a realistic situation.

A. Description of the application

The application concerns the joint operation, by several agents (countries, international organizations...), Of a constellation of Earth observation satellites. The mission of this type of satellite is, as shown in Figure 1, to acquire photographs of the Earth, in response to requests for photographs filed by agents. These agents deposit, at the limit of the satellite visibility corridor, photographs in the process of being acquired orbit photographs not acquired photographs acquired Figure 1 - Acquisition of a photograph by an Earth observation satellite. from a common planning center, photography requests valid for a given day. The overall planning of the shots of all the satellites in the constellation is organized by successive time intervals, generally 1 day. The planning center therefore determines, among the requests concerning a given day, the set of requests that will be satisfied, that is to say all the photographs that will be acquired on that day by the constellation. This set of satisfied requests constitutes a daily allocation of requests to agents. The physical operating constraints and the large number of requests concerning certain areas generate conflicts between requests. It is therefore generally impossible to simultaneously satisfy all the requests filed for a given day. In other words, only a subset of the requests will be able to be satisfied. All these constraints define the set of eligible allowances. Here are some orders of magnitude regarding the real problem. The agents are between 3 and 6. Several hundred applications are candidates each day, among which 100 to 200 will be satisfied. The demands of an agent are of unequal importance. Each agent translates the relative importance of its requests by associating to each a weight, which is a positive or zero number 4, and implicitly corresponds to additive preferences: given two sets of requests of an agent whose sum of the weights is identical, the agent concerned is indifferent to obtaining one or the other set of requests. The individual utility of an allowance for an agent is the sum of the weights of its demands satisfied by the allowance. A utility standardization device - details of which do not concern this article - is used in order to make individual utilities comparable. Here we implicitly consider normalized utilities (and weights). Not all agents contributed equally to the funding of the constellation; the "right of return on investment" provided for each is therefore different. There are different ways of taking into account these different rights. We translate this inequality here by consumption constraints (in addition to the eligibility constraints): each agent is entitled to a maximum consumption of resources per day, this maximum being different for each agent. Beyond taking

these unequal rights into account, the allocation of requests to agents must be fair. Quite different solutions to this problem have been proposed in [15], then in [6]. To meet the fairness requirement, one of the proposed sharing protocols is to choose an allocation that maximizes the leximin order on the individual utility profiles.

B. Fair allocation problem

We have drawn from this application a simplified problem of equitable allocation of objects to agents. It takes the form of an extension of the Leximin-Optimal problem described in section 2. The objects correspond to the demands of our application. The conflicts between demands are modeled in an approximate but suitable way in the form of "generalized volume constraints", linear. Note that in this problem (1) all the objects will not necessarily be allocated, and (2) the same object can be allocated to several agents 5. Here is the formal description of this fair allocation problem. First the data:

- A is a set of agents.
- O is a set of objects to be allocated to agents.
- wio, positive or zero number, is the weight assigned to the object o by agent i;
- ro is the resource consumed by the object o.
- rmax is the maximum consumption of resources authorized for agent i;
- This is the set of generalized volume constraints.
- vco is the volume of the object o in the constraint of generalized volume c.
- vmaxc is the maximum volume in the generalized volume constraint c.

We now define the following variables:

5. What is possible in our application.

- xio = 1 if the object o is allocated to agent i, and 0 otherwise, o ∈ O, i ∈ A. The possible assignments of the variables xio represent the set of possible allocations (among which are the admissible ones).
- ui = Po∈O xio • wio is the individual utility of agent i, i ∈ A;
- so = maxi∈A xio = 1 if the object o is allocated to at least one agent, and 0 otherwise. The problem is to find an allocation x maximizing the leximin order on the huiii ∈ A utility profiles, under the following admissibility constraints:
- wio = 0 ⇒ xio = 0 (objects of zero weight are not allocated to an agent having assigned this weight) 6;
- Po∈O xio • ro ⩽ rmaxi, for all i ∈ A (resource consumption constraints);
- Po ∈ O so • vco ⩽ vmaxc, for all c ∈ C (generalized volume constraints).

5. CONCLUSIONS AND PERSPECTIVES

We have studied the general problem of equitably allocating a set of indivisible goods to agents, in the presence of any admissibility constraints, each agent having its own utility function on the eligible allocations. We

We have translated equity by a Paretoeeffective particularization of the maximin on the utilities of agents: the notion of leximin-optimal allocation, which potentially applies to all multi-agent allocation problems for which the notion of equity has a strong meaning. The main contribution of this article is the proposal of an algorithm for calculating a leximin-optimal allocation, in a constraint programming framework (PPC), based on the use of the cardinality meta-constraint AtLeast. The general framework of PPC is particularly interesting here, because it allows to separately describe the algorithm dedicated to leximin, and on the other hand the utility functions of the agents and the admissibility constraints specific to each potential application. In a context where many of the eligibility constraints are rarely set in stone and evolve with the life of the application, tools like PPC allow you to adapt quickly without changing everything. This algorithm is precisely proposed in the context of a real application: the sharing of satellite resources. From this application we extracted a simplified multiagent allocation problem for which we built a parameterized random generator of instances. This allowed us to test two implementations of the proposed algorithm, in PPC (with Choco [14]), and in PLNE (with Cplex [12]). The results obtained show a clear advantage in favor of PLNE, which can be explained by the specificity of this framework compared to PPC in our case. Several avenues remain to be explored to make the PPC implementation more efficient: heuristics for choosing the variables to instantiate and for choosing the values of the domains, and procedures for filtering the domains of Variables (faster detection of inconsistent or sub-optimal solutions) among others. The article is one algorithmic approach to the problem among others, perhaps more effective. Even if we have only considered exact methods here, we can use, for more difficult instances, incomplete methods like those of [20], or [21, 1] mixing linear programming and taboo search. We hope that our instance generator (online) will allow teams interested in the problem to propose and validate other approaches.

Among the possible consequences of this work, let us quote:

- the experimental comparison of the proposed algorithm with the other algorithms mentioned in section 5.
- the study of more flexible and general approaches to equity, replacing the leximin order by a

parameterized collective utility function making compromises between egalitarianism and classical utilitarianism.

- the application of the algorithm to other practical fields, such as the construction of time schedules or the sharing of airport resources. The authors thank the reviewers for their insightful comments, Jérôme Lang and Jean-Michel Lachiver for their stimulating discussions on the subject, and Simon de Givry for his advice on Cplex.

REFERENCES

[1] N. Bianchessi, J.-F. Cordeau, J. Desrosiers,G. Laporte,and V. Raymond. A heuristic for the multisatellite,multi-orbit and multi-user management of earth observation satellites. European Journal of Operational Research, available online February 2006. doi :10.1016/j.ejor.2005.12.026.

[2] N. Bleuzen-Guernalec and A. Colmerauer. Narrowing a block of sortings in quadratic time. InProc. of CP'97, pages 2–16, Linz, Austria, 1997. D. Dubois, H. Fargier, and H. Prade. Refinements of the maximin approach to decision-making infuzzy environment. Fuzzy Sets and Syst., 81 :103– 122, 1996.

[3] D. Dubois and P. Fortemps. Computing improvedoptimal solutions to max-min flexible constraintsatisfaction problems. European Journal of OperationalResearch, 1999.

[4] M. Ehrgott. Multicriteria Optimization. Number 491 in Lecture Notes in Economics and Mathematical Systems. Springer, 2000.

[5] H. Fargier, J. Lang, M. Lemaître, and G. Verfaillie. Partage équitable de ressources communes. (1) Un modèle général et son application au partage de ressources satellitaires. (2) Éléments de complexité et d'algorithmique. Technique et Science Informatiques, 23(9) :1187–1238, 2004.

[6] H. Fargier, J. Lang, and T. Schiex. Selecting preferred solutions in fuzzy constraint satisfaction problems. In Proc. of EUFIT'93, Aachen, 1993.

[7] A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Multiset ordering constraints. In Proc. of IJCAI'03, February 2003.

[8] M. R. Garey and D. S. Johnson. Computers and Intractability, a guide to the theory of NPcompleteness. Freeman, 1979.

[9] R. S. Garfinkel and G. L. Nemhauser. Integer Programming. Wiley-Interscience, 1972.

[10] P. Van Hentenryck, H. Simonis, and M. Dincbas. Constraint satisfaction using constraint logic programming. A.I., 58(1-3) :113–159, 1992.

[11] ILOG. Cplex 10.0. http://www.ilog.com/products/cplex/.

[12] R. L. Keeney and H. Raiffa. Decisions with Multiple Objectives : Preferences and Value Tradeoffs. John Wiley and Sons, 1976.

[13] F. Laburthe. CHOCO: Implémentation du noyau d'un système de contraintes. In Actes des JNPC-00, Marseille, France, 2000. http://sourceforge.net/projects/choco.

[14] M. Lemaître, G. Verfaillie, and N. Bataille. Exploiting a Common Property Resource under a Fairness Constraint : a Case Study. In Proc. Of IJCAI-99, pages 206–211, Stockholm, 1999.

[15] U. Montanari. Networks of constraints : Fundamental properties and applications to picture processing. Information Sciences, 7 :95–132, 1974.

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

[17] H. Moulin. Fair division and collective welfare. MIT Press, 2003.

[18] G. Pesant and J-C. Régin. SPREAD : A balancing constraint based on statistics. In Proc. of CP'05, Sitges, Spain, 2005.

[19] M. Vasquez and J.-K. Hao. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. Journal of Computational Optimization and Applications, 20(2) :137–157, 2001.

[20] M. Vasquez and J.K. Hao. A Hybrid Approach for the 0–1 Multidimensional Knapsack Problem. In Proc. of IJCAI-01, volume 1, pages 328–333, August 2001.

[21] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on Systems, Man, and Cybernetics, 18 :183–190, 1988.