Participatory budgeting algorithm

Adrien Fabre

2020

Set-up

- K categories (of spending)
- for each category $k \in K$, n_k possible levels of spending: $y_k^1 < ... < y_k^{n_k}$. We denote $Y_k = \{y_k^i : i \in [1; n_k]\}$, and $Y = Y_1 \times ... \times Y_K$.
- N voters: give their preferred apportionments given different budget levels: $b_1 < ... < b_m$, and the related satisfaction levels $u_1^i, ..., u_m^i$
- for budget b_j, choice of voter i is characterized by: ${}_jv^i = ({}_jv_1^i, ..., {}_jv_K^i)$, with ${}_jv_k^i \in Y_k$ under the budget constraint $\sum_k {}_jv_k^i \leq b_j$. Let us denote by ${}_jV$ the voting profile for budget level b_j: ${}_jV = ({}_jv^i)_{i:1..N}$
- (*Unused information*)* For budget b_j, voter i would obtain satisfaction $u^i = U^i({}_jv^i)$ from their favorite apportionment ${}_jv^i$
- Actual budget level: B
- informal objective: find an apportionment of budget B that best represents the choices of voters.

Proportional resolution method

Exact solution

Basic case $\exists j : b_j = B$

For each category k, define the average spending proposal $\overline{s}_k$ as $\overline{s}_k = \frac{1}{N}\sum_{i=1}^N {}_jv_k^i$. In this section, we assume that for all categories k, $\overline{s}_k \in Y_k$. If this is not the case, we need to approximate each $\overline{s}_k$ by a certain $s_k^* \in Y_k$ (see *Approximate solution* below).

The apportionment chosen is $S = (\overline{s}_1, ..., \overline{s}_K)$. It trivially respects the budget constraint. It will be useful to see S as a function of the voting profile ${}_jV$, i.e. $S =: \overline{S}({}_jV)$.

Interior case $\exists \underline{j}, \overline{j} = \underline{j} + 1 : b_{\underline{j}} < B < b_{\overline{j}}$

Take $\lambda \in (0; 1)$ such as $B = \lambda b_{\underline{j}} + (1 - \lambda) b_{\overline{j}}$, i.e. $\lambda = \frac{b_{\overline{j}} - B}{b_{\overline{j}} - b_{\underline{j}}}$.

Define S as $S(B, \lambda, {}_{\underline{j}}V, {}_{\overline{j}}V) = \lambda \overline{S}({}_{\underline{j}}V) + (1 - \lambda)\overline{S}({}_{\overline{j}}V)$

Corner case $B \notin [b_1; b_m]$

Define S as $S = \frac{B}{b_1}\overline{S}({}_1V)$ if $B < b_1$ and $S = \frac{B}{b_m}\overline{S}({}_mV)$ if $B > b_m$.

Approximate solution

Now, we have a set of target spendings $S = (s_1, ..., s_K)$ that respects the budget constraint but not necessarily the possible levels of spendings, and we want to find an apportionment $S^* = (s_1^*, ..., s_K^*) \in Y$ that fits into the possible levels of spendings, while still respecting the budget constraint. We will do so using a Knapsack algorithm.

For each k, define $s_k^- = \max_{Y_k}\{y : y \leq s_k\}$ and $s_k^+ = \min_{Y_k}\{y : y \geq s_k\}$. Further define $\widetilde{B} = B - \sum_k s_k^-$.

The Knapsack problem consists in finding the set $K^* \subset [1; K]$ that maximizes $\sum_{k \in K^*} s_k - s_k^-$ under the constraint $\sum_{k \in K^*} s_k^+ - s_k^- \leq \widetilde{B}$.

NB: other maximization programs can be used here, notably some that make use of the categories prioritized by the voters (which can be defined using other budget levels or using the satisfaction levels).

Shortcomings

- doesn't use satisfaction, i.e. doesn't try to spot which spending trigger largest jump in satisfaction, nor to weigh more the least satisfied
- doesn't use information on preferences over budget levels other than the closest from the actual budget

Implementation

```
K <- 5
Y <- list(c(0:4), c(0:3), c(0:5), c(0:1), c(0:1))
B <- 8
b <- c(4, 8)
N <- 7
V1 <- matrix(c(0, 0, 2, 1, 1,
               2, 2, 0, 0, 0,
               0, 0, 4, 0, 0,
               0, 2, 0, 1, 1,
               2, 0, 0, 1, 1,
               2, 2, 0, 0, 0,
               4, 0, 0, 0, 0), byrow=T, nrow = N, ncol = K)
V2 <- matrix(c(4, 0, 2, 1, 1,
               2, 2, 4, 0, 0,
               0, 2, 4, 1, 1,
               0, 2, 4, 1, 1,
               2, 2, 2, 1, 1,
               2, 2, 2, 1, 1,
               4, 2, 0, 1, 1), byrow=T, nrow = N, ncol = K)
V <- list(V1, V2)

# The function definitions are hidden
proportional_solution(B, b, Y, V, continuous = TRUE)
```

```
## [1] 2.0000000 1.7142857 2.5714286 0.8571429 0.8571429
```

```
proportional_solution(B, b, Y, V)
```

```
## [1] 2 2 2 1 1
```

```
proportional_solution(10, b, Y, V)
```

```
## [1] 3 2 3 1 1
```

```
proportional_solution(2, b, Y, V)
```

```
## [1] 1 1 0 0 0
```

```
proportional_solution(7, b, Y, V)
```

```
## [1] 2 1 2 1 1
```