Taking into Account Expected Future Bids in ePolicy Optimisation Problem

Nic Wilson and Lars Kotthoff*

July 29, 2014

1 Introduction

This report considers a particular optimisation problem that has arisen in the e-Policy project (http://www.epolicy-project.eu), which corresponds to a variation of the classic knapsack problem, where the items are received in large batches.

We assume that we will receive a set of bids, where each bid is of the form $B_j = (w_j, v_j)$, and where w_j is the cost, or grant requested, and v_j is the power which the bidder is offering to produce. We have a given total budget, and the objective is to choose a subset of the bids with maximal total power, subject to the constraint that the total cost of chosen bids is less than the budget. Thus, if C is the set of bids chosen (accepted), then we are maximising $\sum_{B_j \in C} v_j$ subject to the constraint that $\sum_{B_j \in C} w_j$ is no more than the budget. If we receive all the bids at the same time, this is an instance of the classic knapsack problem (cf. Deliverable e-Policy Deliverable 5.2).

The complication is that in the e-Policy optimisation problem we do not receive all the bids at the same time (nor do we receive bids one at a time, as in an online knapsack problem). We accept a collection of bids each year, over a number of years, and we choose the bids we will accept for each year.

A difficulty in solving this form of online problem is the division of the allocation of budget between the years. For instance, if we place no limit to the budget we use in the first year, then we may well use the whole budget then, which will prevent us from accepting any very attractive bids in later years. We can instead divide the budget allocation equally between the years, but this can also be sub-optimal, if, for example, we receive many more very attractive bids in the first year than expected. The idea behind our approach here is to take into account any information we might have about the future bids we expect to receive, and use this to determine whether a current bid is worth accepting, or if we expect to be able to do better later.

*Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Ireland, email: nic.wilson@insight-centre.org, lars.kotthoff@insight-centre.org

2 Basic Algorithm

Here we describe the basic structure of our algorithm, as applied to the bids received in a particular year. The idea is to try to choose the bids that have maximal ratio of power to cost, taking into account the expected future bids.

We will use a function V that, given remaining budget R, estimates the total value (total power) $V(R)$ we expect to achieve in later years. A single year's bids $\mathcal{B}$ consists of a set of pairs of the form $B_j = (w_j, v_j)$, where w_j is cost to the budget, and v_j is the value (power gained). We first sort $\mathcal{B}$ in such a way that it has decreasing values of $\frac{v_j}{w_j}$, breaking ties by increasing value of w_j, i.e., for $i, j \in \{1, \ldots |\mathcal{B}|\}$, if $i \leq j$ then $\frac{v_i}{w_i} \geq \frac{v_j}{w_j}$, and if $\frac{v_i}{w_i} = \frac{v_j}{w_j}$ then $w_i \leq w_j$.

Suppose at any point in the algorithm (applied to that year) we are considering bid $B_j = (w_j, v_j)$ and suppose we have remaining budget E. If we accept bid B_j we expend cost w_j to gain value (power) v_j.

Suppose we stop without accepting bid B_j. Then the expected extra total value we will obtain using the future years' bids is $V(E)$. If, on the other hand, we accept bid B_j and stop then, the expected extra total value we will obtain using the future years' bids will be $V(E - w_j)$. Thus it is better to (a) accept bid B_j and then stop, rather than (b) stopping now, rejecting bid B_j, if and only if $v_j + V(E - w_j) > V(E)$, i.e., $v_j > V(E) - V(E - w_j)$. This is the idea behind the test in the algorithm which determines whether to accept bid B_j.

Let E be the current remaining budget as we receive the current year's bids. At each point, R will represent the remaining budget.

Basic Algorithm

$R := E$
Total_Value := 0
for $j = 1, \ldots, |\mathcal{B}|$,
  if $v_j > V(R) - V(R - w_j)$ **then**
    (* We accept bid $B_j = (w_j, v_j)$ *)
    Accepted_Bids := Accepted_Bids $\cup \{B_j\}$
    Total_Value := Total_Value + v_j.
    $R := R - w_j$
    end (* **if** *)
  end (* **for** *)

Naturally, the final value of variable Accepted_Bids will contain the collection of accepted bids, with total power equalling (the final value of) Total_Value, which equals the sum of v_j over all j such that B_j is in Accepted_Bids.

Roughly speaking, we are accepting bids whose ratio $\frac{v_j}{w_j}$ is greater than we would expect to obtain from the worst bids we would expect to be accepting from future years.

3 Estimating value $V(R)$

The basic algorithm involves the key test of whether $v_j > V(R) - V(R - w_j)$. In this section we discuss our approach for estimating $V(R)$ (and hence also $V(R) - V(R - w_j)$). Recall that $V(R)$ is the expected additional power we can achieve using the bids in future years, given the remaining budget R.

An important consideration is the efficiency of determining if $v_j > V(R) - V(R - w_j)$. The size of the collection of bids $\mathcal{B}$ will often be very substantial, and the basic algorithm loops over every element of $\mathcal{B}$. We thus may need that the test $v_j > V(R) - V(R - w_j)$ is not too expensive.

3.1 Estimating $V(R)$ based on collection $\mathcal{S}$ of future bids

First let's consider how we could proceed if we knew the collection $\mathcal{S}$ of bids we will receive in the future. Let $V^{\mathcal{S}}(R)$ be our estimate of $V(R)$ for this situation. For reasons of efficiency, we simplify in a couple of ways. We define $V^{\mathcal{S}}(R)$ to be equal to the maximum total power achievable given set of bids $\mathcal{S}$ and total budget R, where fractions of bids are allowed. (If a fraction p between 0 and 1 of bid (w_j, v_j) is accepted then it costs pw_j to the budget, and gives us an additional pv_j of power.) Thus firstly, we are ignoring the fact that the future bids $\mathcal{S}$ will be split between years; and we only know which bids we receive one year at a time; and secondly, we approximate the situation by allowing fractional bids. The second assumption will not likely make much difference to the result, apart from in some exceptional circumstances (and the fact that $V^{\mathcal{S}}(R)$ will then depend more smoothly on R than if we were not to allow fractional bids, may in fact help the accuracy of the overall algorithm). The first assumption will tend to mean we are somewhat overestimating $V(R)$ by $V^{\mathcal{S}}(R)$.

Simple implementation of function $V^{\mathcal{S}}(R)$

Let S be a collection of bids, each bid consisting of a pair (w_j, v_j). We first order S in such a way that it has decreasing values of $\frac{v_j}{w_j}$, breaking ties by increasing value of w_j, i.e., for $i, j \in \{1, \ldots |S|\}$, if $i \leq j$ then $\frac{v_i}{w_i} \geq \frac{v_j}{w_j}$, and if $\frac{v_i}{w_i} = \frac{v_j}{w_j}$ then $w_i \leq w_j$.

For real value (of budget) R, the value $V^{\mathcal{S}}(R)$ is the total power we achieve by choosing, using a greedy algorithm, a subset S' of S subject to the constraint that the sum of w_j in the chosen set S' is no more than R.

function $V^{\mathcal{S}}(R)$
$A := R$
Total_Value := 0
for $j = 1, \ldots, |S|$,
    if $w_j \leq A$ **then**
        Total_Value := Total_Value + v_j.
        $A := A - w_j$

end (* **if** *)
end (* **for** *)
Total_Value := Total_Value + $v_j(w_j - A)/w_j$
return Total_Value

3.2 Estimating $V(R)$ based on a distribution over future bids

Now, we consider the case where we have an estimate Q of the distribution governing future bids, and that we expect N future bids for some natural number N (which may well tend to decrease over the years).

We use a Monte-Carlo simulation algorithm involving a number M of trials. For each $i = 1, \ldots, M$, we create a random collection of bids $\mathcal{S}_i$ of cardinality N, drawn independently with distribution Q.

We then estimate $V(R)$ as $\frac{1}{M}\sum_{i=1}^{M} V^{\mathcal{S}_i}(R)$, where $V^{\mathcal{S}_i}(R)$ is defined below.

Generating the distribution Q

A simple approach is to let Q be the distribution of bids we have seen so far. So, for the first year, Q is just the collection of bids submitted in the first year. For the second year, Q is the collection of bids submitted in the first two years.

If we have additional information about what Q might be like, we could add extra elements to take this into account, or use some weighted average between the bids we have so far, and some guessed distribution.

If the distribution changes over time, so that e.g., the bids for the second year are much less attractive than those for the first year, then this should be taken into account. We might, for example, give more weight to the most recent bids.

4 More Detailed Implementation of the Algorithm

In Section 2 we described the basic algorithm, and in Section 3 we described how a sampling approach can be used an approximation in the implementation of the function $V(R)$. Here we describe a more efficient method of implementing our method.

Recall that a single year's bids $\mathcal{B}$ consists of a set of pairs of the form $B_j = (w_j, v_j)$, where w_j is cost to the budget, and v_j is the value (power gained). We first sort $\mathcal{B}$ in such a way that it has decreasing values of $\frac{v_j}{w_j}$, breaking ties by increasing value of w_j, i.e., for $i, j \in \{1, \ldots |\mathcal{B}|\}$, if $i \leq j$ then $\frac{v_i}{w_i} \geq \frac{v_j}{w_j}$, and if $\frac{v_i}{w_i} = \frac{v_j}{w_j}$ then $w_i \leq w_j$.

Let E be the current remaining budget as we receive the current year's bids. At each point, R will represent the remaining budget.

The idea is to use an approximate form for $V^{\mathcal{S}}(R)$ for each sample collection $\mathcal{S}$ of bids. $V^{\mathcal{S}}(R)$ is the maximum total power one can get for cost R, *if we allow fractional bids*. That is, the total power obtained by an optimal solution of the fractional knapsack problem (i.e., the continuous knapsack problem) based on the set of bids $\mathcal{S}$.

The sample collection $\mathcal{S}$ of bids

To distinguish the costs and powers of the bids in the sample from those in the current inputs, we use a slightly different notation.

Each bid in $\mathcal{S}$ is of the form $(\bar{w}_i, \bar{v}_i)$, where $\bar{w}_i$ is the cost of the bid, and $\bar{v}_i$ is the power offered. As discussed earlier, we order the bids in decreasing order of their power to cost ratio so that if $i \leq j$ then $\bar{v}_i/\bar{w}_i \geq \bar{v}_j/\bar{w}_j$. (Lets say also that if $i \leq j$ and $\bar{v}_i/\bar{w}_i = \bar{v}_j/\bar{w}_j$ then $\bar{w}_i \leq \bar{w}_j$, but the correctness of the algorithm does not depend on this.)

$V^{\mathcal{S}}(R)$ is the maximum total power one can get for cost R (for this sampled collection $\mathcal{S}$), if we allow fractional bids. It can be expressed as follows. Let $A_j = \sum_{i=1}^{j} \bar{w}(i)$, and let k be maximal such that $A_k \leq R$, so that $A_k \leq R < A_{k+1}$. Then $V^{\mathcal{S}}(R)$ is equal to $\sum_{i=1}^{k} \bar{v}(i) + (R - A_k)\bar{v}(k+1))/\bar{w}(k+1)$. This is implemented efficiently using function `evaluateV`(R, k) defined below.

4.1 Algorithm using a single sampled collection of bids $\mathcal{S}$

To aid clarity we first present the algorithm when we only use a single sampled collection $\mathcal{S}$ of bids. This corresponds to the case of $M = 1$ in the more general algorithm below in Section 4.2.

Suppose the sample $\mathcal{S}$ contains N bids. We can use linear arrays for the sample bids, and also the partial sums.

- `sv`$[i]$ is equal to $\bar{v}_i$, and
- `sw`$[i]$ equals $\bar{w}_i$.

We also use arrays for the partial sums:

- `sum-sv`$[j]$ represents $\sum_{i=1}^{j} \bar{v}(i)$;
- `sum-sw`$[j]$ represents $\sum_{i=1}^{j} \bar{w}(i)$.

The following function[1] returns the value of $V^{\mathcal{S}}(R)$. When the function is called in the algorithm, the parameter k has been arranged to be such that k be maximal such that $A_k \leq R$, i.e., `sum-sw`$[k] \leq R$.

function `evaluateV`(R, k)
return `sum-sv`$[k]$ + ((R − `sum-sw`$[k]$) * `sv`$[k+1]$)/`sw`$[k+1]$

[1] We are assuming here that not all bids in $\mathcal{S}$ could be accepted given budget R, i.e., that `sum-sw`$[N] > R$. If `sum-sw`$[N] \leq R$ then we set `evaluateV`(R, k) to be `sum-sv`$[N]$.

The following is a more detailed implementation of the basic algorithm (see Section 2 above), but using only one sampled collection $\mathcal{S}$ of bids.

Variable k stores the maximal integer such that $\sum_{i=1}^{k} \bar{w}(i) \leq R$, where R is the remaining budget at each point. Thus, k (once the for-j loop starts) only changes when a bid $B_j = (w_j, v_j)$ is accepted, which is when w_j is taken off the value of R. Because R is decreasing, k is decreasing. Variable h is looped in order to determine the value `VRwj` of $V^{\mathcal{S}}(R - w_j)$.

Algorithm based on a single sample collection of bids

$R := E$
Total_Value := 0
    (* The next few lines generate the initial values of k and `VR` *)
$k := N$
while `sum-sw`$[k] > R$ **do** $k := k - 1$;
    (* k is now the maximal integer such that $\sum_{i=1}^{k} \bar{w}(i) \leq R$, *)
`VR` := `evaluateV`(R, k)
    (* `VR` is the value of $V^{\mathcal{S}}(R)$, i.e., currently $V^{\mathcal{S}}(E)$ *)
for $j = 1, \ldots, |\mathcal{B}|$,
    $h := k$
    while `sum-sw`$[h] > R - w_j$ **do** $h := h - 1$;
    `VRwj` := `evaluateV`$(R - w_j, h)$ (* `VRwj` is the value of $V^{\mathcal{S}}(R - w_j)$ *)
    if $v_j >$ `VR` $-$ `VRwj` **then**
        (* We accept bid $B_j = (w_j, v_j)$ *)
        Accepted_Bids := Accepted_Bids $\cup \{B_j\}$
        Total_Value := Total_Value $+ v_j$.
        $R := R - w_j$
        `VR` := `VRwj`
        $k := h$
        end (* **if** *)
    end (* **for** *)

Discussion

The algorithm (like the basic algorithm) uses the loop variable j to go through the bids in $\mathcal{B}$. A crude implementation would independently also loop over the elements of $\mathcal{S}$ in the computation of $V^{\mathcal{S}}(R)$. This last algorithm avoids this excessive looping by first of all pre-computing the values of the partial sums `sum-sw`$[k]$ and `sum-sv`$[k]$, and, secondly, updating k incrementally. The

algorithm still has the smaller loop for h (for each value of j), but typically this will just involve a small number of different values of h for each value of j.

There are, however, exceptional collections $\mathcal{S}$ where many values of h will be required for some values of j; this happens when $\bar{w}_i$ is very small for all the worst (lowest/last) values of $\bar{v}_i/\bar{w}_i$. For such cases, a cleverer implementation can be used, involving a sufficient condition for the antecedent $v_j > V(R) - V(R - w_j)$ of the if-statement. However, this should not usually be necessary.

4.2 Version of algorithm using mean over M sampled collections of bids

Now we extend, in the obvious way, the algorithm in Section 4.1 making use of M sampled collections of bids. The idea is as in Section 4.1 above, but we need a parameter s to range from 1 to M to index the computation with respect to each sample of bids. We thus have collection of N bids $\mathcal{S}_s$, for each $s = 1, \ldots, M$.

Each bid in $\mathcal{S}_s$ is of the form $(\bar{w}_i^s, \bar{v}_i^s)$, where $\bar{w}_i^s$ is the cost of the bid, and $\bar{v}_i^s$ is the power offered. As discussed earlier, we order the bids in decreasing order of their power to cost ratio so that if $i \leq j$ then $\bar{v}_i^s/\bar{w}_i^s \geq \bar{v}_j^s/\bar{w}_j^s$.

We now have to index all arrays also by the sample number s.

$\texttt{sv}[s, i]$ is equal to $\bar{v}_i^s$, and $\texttt{sw}[s, i]$ equals $\bar{w}_i^s$.

- $\texttt{sum-sv}[s, j]$ represents $\sum_{i=1}^{j} \bar{v}^s(i)$;
- $\texttt{sum-sw}[s, j]$ represents $\sum_{i=1}^{j} \bar{w}^s(i)$.

The following function returns the value of $V^{\mathcal{S}_s}(R)$. The parameter $k[s]$ is arranged to be such that $k[s]$ be maximal such that $A_{k[s]} \leq R$, i.e., $\texttt{sum-sw}[s, k[s]] \leq R$.

function $\texttt{evaluateV}(s, R, k[s])$

return $\texttt{sum-sv}[s, k[s]] + ((R - \texttt{sum-sw}[s, k[s]])$ * $\texttt{sv}[s, k[s] + 1])/\texttt{sw}[s, k[s] + 1]$

Algorithm using M collections of bids

$R := E$
Total_Value := 0
(* The next few lines generate the initial values of $k[s]$ and $\texttt{VR}[s]$ *)
for $s = 1, \ldots, M$
$k[s] := N$
while $\texttt{sum-sw}[s, k[s]] > R$ **do** $k[s] := k[s] - 1$;
(* $k[s]$ is now the maximal integer such that $\sum_{i=1}^{k[s]} \bar{w}^s(i) \leq R$, *)
$\texttt{VR}[s] := \texttt{evaluateV}(s, R, k[s])$
end (* **for** s *)
for $j = 1, \ldots, |\mathcal{B}|$,
for $s = 1, \ldots, M$
$h[s] := k[s]$
while $\texttt{sum-sw}[s, h[s]] > R - w_j$ **do** $h[s] := h[s] - 1$;
$\texttt{VRwj}[s] := \texttt{evaluateV}(s, R - w_j, h[s])$
end (* **for** s *)
if $v_j > \frac{1}{M} \sum_{s=1}^{M} (\texttt{VR}[s] - \texttt{VRwj}[s])$ **then**
(* We accept bid $B_j = (w_j, v_j)$ *)
Accepted_Bids := Accepted_Bids $\cup \{B_j\}$
Total_Value := Total_Value + v_j.
$R := R - w_j$
for $s = 1, \ldots, M$
$\texttt{VR}[s] := \texttt{VRwj}[s]$
$k[s] := h[s]$
end (* **for** s *)
end (* **if** *)
end (* **for** *)

5 Preliminary implementation and evaluation

We implemented the algorithm for a single sample collection of bids (cf. Section 4.1) for the simplest case where the sample $\mathcal{S}$ is equal to the complete set of bids received during a time step.

We ran a set of preliminary experiments to compare this approach to the pure optimisation approach. We used the budgets levels €1,000,000, €2,000,000, €3,000,000, €4,000,000, €5,000,000, €6,000,000, €7,000,000, €8,000,000, €9,000,000, €10,000,000, €20,000,000, €30,000,000, €40,000,000, €50,000,000, €60,000,000,

€70,000,000, €80,000,000, €90,000,000, €100,000,000, €200,000,000, €300,000,000, €400,000,000, €500,000,000, €600,000,000, €700,000,000, €800,000,000, €900,000,000, and €1,000,000,000. Both national and regional incentives were enabled and first come, first serve budget distribution used for the Emilia-Romagna region.

Figure 1 compares the total installed capacity at the end of each simulation. The results achieved by both approaches are very close, with the partial allocation approach achieving higher installed capacity in the majority of cases. There are cases where lower installed capacity is achieved as well. These results demonstrate the uncertainty inherent in this approach – we do not know future bids and can only make assumptions what they may be. Such assumptions may turn out to be wrong.

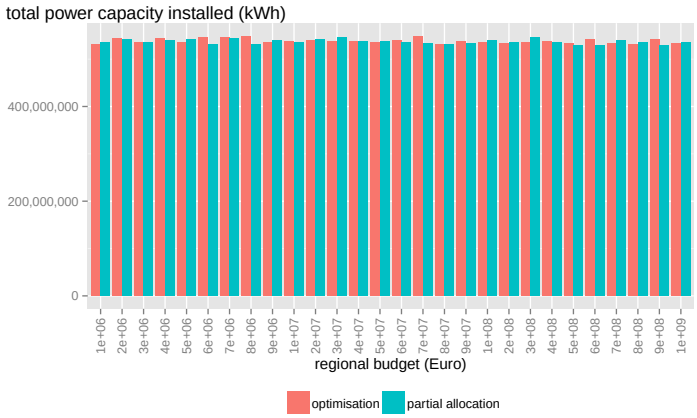

Figure 1: Total power capacity over regional budget for optimisation and partial allocation.

Nevertheless the results show the promise of the approach. We are able to improve on results that are provably optimal for each single time step by taking the entire simulation into account. This is very desirable, as it enables the policy maker to achieve more with the same budget.

Figure 2 presents the same comparison for the total cost incurred. Again the total cost is very similar for both approaches (except for the three largest budgets) and there are cases where the partial allocation approach spends less than the optimisation approach, but achieves a higher installed capacity. This again underlines the promise of the approach – we are able to exploit cases where not funding a bid not but a better one later will yield an overall better result.

For the largest three budgets, the overall cost of the partial allocation approach is much lower than that of the optimisation approach, even though the achieved installed capacities are similar. Indeed, in two out of three cases, the installed capacity achieved by the partial allocation approach is slightly higher.

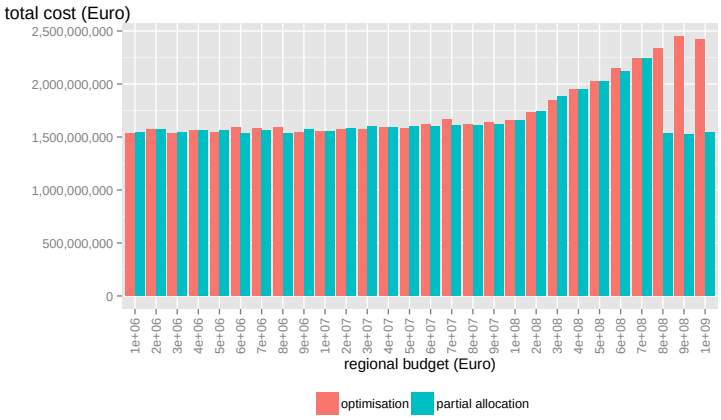

Figure 2: Total cost over regional budget for optimisation and partial allocation.

It is unclear why this is happening and what dynamics contribute to this phenomenon. Further research is needed and more work required to make it feasible to include this approach in the ePolicy system. The aim of the preliminary implementation and evaluation here is to highlight the promise of the approach as a potential direction for future research, not to present an alternative system ready for deployment.