The Addictive Facility Location Problem

Luc Cote Dylan Fridman

March 10, 2022

1 Introduction

Consider the following problem: you are an electric car manufacturing company that can afford to place a given number of chargers throughout the world. Now, you know that your potential customers will decide whether or not to buy a car from you depending on the availability of electric chargers in their nearby area. The problem then arises of how to distribute the chargers through the world so as to maximize your profit, i.e. the number of customers you get.

If every client was satisfied by having only one charger nearby (let's say at a distance smaller than r of their home), then the problem could be reduced to Max k-Coverage. However, since the probability that a given client buys a car as a function of chargers nearby is not discrete, the problem becomes a creature of its own.

We can view this type of facility-client arrangement where a higher number of facilities increases a client's profit as a kind of addicting facility, giving rise to the problem name.

1.1 The Problem

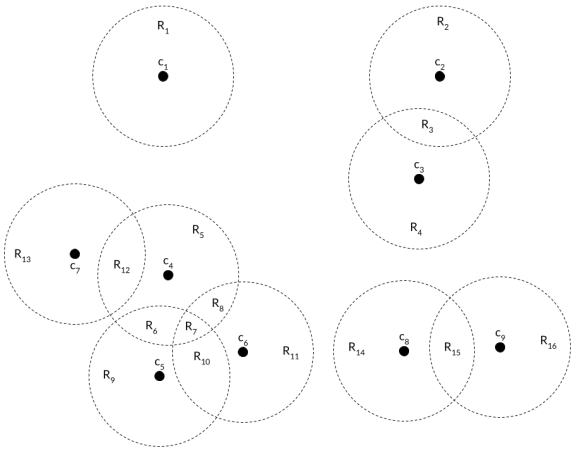

Figure 1. clients on a plane with their generated profit regions

We formalize the problem as follows. The input consists of some number of facilities k we are allowed to place, some radius r and a set of clients $\mathcal{C}$ such that each client $c \in \mathcal{C}$ itself consists of a position (x_c, y_c)

in the plane and a non-decreasing function $g_c : \mathbb{N} \to \mathbb{R}_{\geq 0}$. We interpret g_c as indicating the profit that we will make from c by placing j facilities within a distance of r.[1]

Given the input, we wish to return the positions $\mathcal{F} = \{(x_i, y_1)\}_{i=1}^{k}$ of the k facilities, so as to maximize the total profit $p(\mathcal{F})$ made from all the clients $\mathcal{C}$. Explicitly, we have

$$p(\mathcal{F}) = \sum_{c \in \mathcal{C}} g_c(j_c) \quad \text{where } j_c := |\{(x, y) \in \mathcal{F} : d((x, y), (x_c, y_c)) \leq r\}|$$

1.1.1 A note on regions

Note that we can partition the plane into a collection of regions $\mathcal{R}$ according to what clients are at distance smaller than r from any of those points. Each region $R \in \mathcal{R}$ will be given by an equivalence class of the following equivalence relation $\sim$ of points in $\mathbb{R}^2$:

$$p_1 \sim p_2 : \iff \{c \in \mathcal{C} : d(x, p_1) \leq r\} = \{c \in \mathcal{C} : d(c, p_2) \leq r\}$$

Then, $\mathcal{R} = \{[(x, y)] \sim : (x, y) \in \mathbb{R}^2\}$. Now, note that for a given region $R \in \mathcal{R}$, the profit made by placing a facility at (x, y) is the same for any $(x, y) \in R$. Thus, we can give the output to the problem as an assignment $\lambda : \mathcal{R} \to \mathbb{N}$ of facilities to regions such that $\sum_{R \in \mathcal{R}} \lambda[R] = k$ and for any $R \in \mathcal{R}$ we have $\lambda[R] = |\mathcal{F} \cap R|$. A useful fact is that $|\mathcal{R}|$ depends quadratically on $|\mathcal{C}|$ since we are on the plane.

Also, observe that we can identify $R \in \mathcal{R}$ with the set of clients that are affected by placing facilities in R. To denote such a set we write

$$\mathcal{C}(R) := \{c \in \mathcal{C} : d(c, p_0) \leq r \text{ for some } p_0 \in R\}$$

Conversly, observe that we can identify $c \in \mathcal{C}$ with the set of Regions that intersect the client's radius. To denote such a set we write

$$\mathcal{R}(c) := \{R \in \mathcal{R} : c \in \mathcal{C}(R)\}$$

We define $\lambda(c)$ as the number of facilities placed that satisfy a client c so that

$$\lambda(c) := \sum_{R \in \mathcal{R}(c)} \lambda[R]$$

So that we can directly compute the benefit of placing j facilities in some region $R \in \mathcal{R}$ for some configuration λ, we define

$$g_R(j, \lambda) := \sum_{c \in \mathcal{C}(R)} (g_c(j + \lambda(c)) - g_c(\lambda(c)))$$

Using the above notation we can now rewrite the profit that a given assignment λ gives as follows:

$$p(\lambda) := \sum_{c \in \mathcal{C}} g_c(\lambda(c))$$

For ease of algorithm description and analysis, we will use this region-based definition of the problem going forward.

[1] In this formalization of the problem, the profit made from each client is independent of the profit made from others. We do not consider fashion trends, for example.

As a final observation regarding regions, note that $\mathcal{R}$ can be ordered as a partial order given by $R_1 \leq R_2 \iff \mathcal{C}(R_1) \subset \mathcal{C}(R_2)$. Then, since g_R is non-decreasing, the maximal elements of this partial order are the only regions in which the best allocation of facilities will indeed place facilities.

2 Hardness

Claim 2.1. *The Addictive Facility Location Problem is NP-hard.*

Here we construct a mapping from the k-center problem to a special case of the Addictive Facility Location Problem. The k-center problem is known to be NP-hard [**?**] (even under a planar-constrained metric) and is defined as follows:

Problem 2.2 (k-center). *Given n points X and a distance $d(\cdot,\cdot)$ satisfying triangle inequality, we select k points $C \subseteq X$ such that the distance of any point to a point in C is minimized. Formally, we want to minimize*

$$(C) := \max_{x \in X} d(x, C) \quad \textit{where} \quad d(x, C) := \min_{y \in C} d(x, y)$$

Let us now consider the case of the addictive facility location problem where the profit of every client is given by the piecewise function

$$g_c(k_c) := \begin{cases} 1 & k_c \geq 1 \\ 0 & k_c = 0 \end{cases}$$

We note that if for a certain value of r, the solution returns $\mathcal{F}$ such that $p(\mathcal{F}) = k$ then $\mathcal{F}$ is in fact a feasible solution to the k-center problem. Using this, we can perform a binary search across the space of all radii of interest (radii which cause a new region to be formed on the plane) to find the minimum radius at which $p(\mathcal{F}) = k$, thus giving us the minimum distance feasible solution to k-center.

3 Algorithm

Due to the polynomial nature of the number of possible regions, we can enumerate the profit value that placing any number of facilities in any region will generate. We use this fact to generate our first algorithm.

3.1 Naive Greedy Algorithm

Drawing inspiration from set cover, we start with a greedy algorithm for the problem which chooses placements via a "bang for buck" style comparison of the profit values of each region-facilities pairing.

We denote ℓ to be the number of facilities we have currently placed (note: ℓ starts at 0). We consider the profit gained from placing an additional 1 through $k - \ell$ facilities on each region and choose the placement which maximizes profit per facility placed. We then repeat this action until we have placed all k facilities.

Algorithm 1 Naive Greedy Approach

$\lambda[R] \leftarrow 0 \ \forall R \in \mathcal{R}$ ▷ *tracks facilities placed at each region*
$\ell \leftarrow 0$ ▷ *tracks number of facilities already placed*
while $\ell \neq k$ **do**
    $R', j' \leftarrow R' \in \mathcal{R}, j' \in [1, k-\ell] \mid \frac{g_{R'}(j', \lambda)}{j'} \geq \frac{g_R(j, \lambda)}{j} \forall R \in \mathcal{R}, j \in [1, k-\ell]$ ▷ *find best bang for buck placement*
    $\lambda[R'] \leftarrow \lambda[R'] + j'$
    $\ell \leftarrow \ell + j'$
return λ

3.2 Issues with the Naive Greedy Approach

On the surface this seems to be a good idea: the best solution will of course have the best bang for buck across all solutions, and bang for buck can be easily compared across region placements of differing quantities. However, with a little thought, this algorithm encounters a fatal flaw: there can be a configuration of two regions: one which gives a high bang for buck for placing very few facilities and then no profit for placing additional facilities, and one which gives very little profit for placing some facilities, but for placing very many (i.e. k) facilities gives a good bang for buck (although a slightly worse bang for buck that the first region with very few facilities). This configuration will cause the algorithm to first choose to place a small number of facilities on region 1, which would disable it from placing k facilities on region 2, thus causing it to miss out on the slightly lower bang for buck, but high number of high-profit facilities from locating many facilities in region 2, and it must settle for placing its remaining facilities in low-profit configurations. With some jiggling, we can see this scenario leads to a $\frac{1}{k}$ approximation.

3.3 Greedy Algorithm with Swaps

We note that the issues in the naive greedy algorithm are caused by regions that require a large number of facility placements to generate a large profit being overshadowed by much smaller regions. Since these regions must contain many facilities, choosing even one such region (if they exist) will likely result in a good approximation. Thus, we add a final step to the greedy algorithm which looks at the value of adding all facilities to each region and swaps it with the greedy algorithm's allocation of facilities if it has a higher total profit. This leads to our adapted algorithm:

Algorithm 2 Greedy Approach with Swap

$\lambda[R] \leftarrow 0 \ \forall R \in \mathcal{R}$ ▷ *tracks facilities placed at each region*
$\ell \leftarrow 0$ ▷ *tracks number of facilities already placed*
while $\ell \neq k$ **do**
    $R', j' \leftarrow R' \in \mathcal{R}, j' \in [1, k-\ell] \mid \frac{g_{R'}(j',\lambda)}{j'} \geq \frac{g_R(j,\lambda)}{j} \forall R \in \mathcal{R}, j \in [1, k-\ell]$ ▷ *find best bang for buck placement*
    $\lambda[R'] \leftarrow \lambda[R'] + j'$
    $\ell \leftarrow \ell + j'$
$R' \leftarrow R' \in \mathcal{R}, \mid g_{R'}(k, \emptyset) \geq g_R(k, \emptyset) \forall R \in \mathcal{R}$ ▷ *find best region with all facilities added*
if $g_{R'}(k, \emptyset) \geq p(\lambda)$ **then** ▷ *make swap if better*
    $\lambda[R] \leftarrow 0 \ \forall R \in \mathcal{R}$
    $\lambda[R'] \leftarrow k$
return λ

4 Analysis of the algorithm

Conjecture 4.1. *Greedy Algorithm with Swap is a* $\frac{1}{9}$*-approximation for the Addictive Facility Location problem.*

Although we could not prove Conjecture 4.1 in its general case, we show it holds under certain conditions. The rest of the paper is devoted to showing the conjecture holds in these cases. For the analysis, let alg denote the output of the program. Also, let opt(k) denote the maximum $p(\lambda)$ for a given number of facilities k, where all the other inputs remain fixed. We begin by stating the following useful claim.

Claim 4.2. *For every* $R \in \mathcal{R}$*, we have* alg $\geq g_R(k, \emptyset)$*.*

Proof: Suppose that the swap was performed and that we picked region $R \in \mathcal{R}$. In this case, we have that alg $= \max_{R \in \mathcal{R}} g_R(k, \emptyset)$, so the claim follows. If we didn't perform the swap, it means that $p(\mathcal{F})$ before the swap is greater than any placement of k facilities onto one region by definition. □

We first prove Theorem 4.1 assuming what we call Wishful Thinking, and then show errors with this proof in the general case.

Assumption 4.3 (Wishful Thinking). *All the clients are at distance* $\geq 2r$*.*

Before we prove Theorem 4.1 under the assumption of Wishful Thinking, let us see why this is a helpful assumption. Suppose that we face a problem that doesn't satisfy the assumption so that we have clients c_1 and c_2 such that $d(c_1, c_2) \in (0, 2r]$. Then, we will have $R_0, R_1, R_2 \in R$ such that R_0 contains both c_1 and c_2, R_1 contains c_1 but not c_2, and R_2 contains c_2 but not c_1. The added complication with this is that placing a facility in R_1 or R_2 also affects R_0 by adding a facility, and viceversa. On the other hand, when we assume Wishful Thinking, the only impact on some region R by placing facilities on some other region R' is that we are decreasing the number of available facilities we have.

4.1 Assuming Wishful Thinking

Lemma 4.4. *Assuming Wishful Thinking,* $\text{alg} \geq \text{opt}(\frac{k}{2})$.

Proof: Denote by $\hat{\lambda}$ the first assignment of λ in the algorithm satisfying $\sum_{R \in \mathcal{R}} \lambda[R] \geq \frac{k}{2}$. We know that $\text{alg} \geq p(\hat{\lambda})$, so it suffices for us to prove that $p(\hat{\lambda}) \geq \text{opt}(\frac{k}{2})$. Let γ denote an optimal assignment of $\frac{k}{2}$ facilities, so that $p(\gamma) = \text{opt}(\frac{k}{2})$.

We prove that $p(\hat{\lambda}) \geq \text{opt}(\frac{k}{2})$ by arguing that for every facility placement made by γ, there is an injective mapping to a better facility placement in $\hat{\lambda}$. Take $R \in \mathcal{R}$ such that $\gamma[R] > 0$ and consider the following cases:

- Suppose $\gamma[R] = \hat{\lambda}[R]$. Then, R contributes the same to both $p(\hat{\lambda})$ and $p(\gamma)$.
- Suppose $\hat{\lambda}[R] = 0$. Then, the algorithm decided not to place any facilities in R, so it must be the case that every single pick done by the algorithm had a better bang-for-buck. This follows because the algorithm could always have chosen to pick $\lambda[R] \leftarrow \gamma(R)$ since we know that $\gamma(R) \leq \frac{k}{2}$ and before every pick the algorithm has at least $\frac{k}{2}$ facilities at its disposal.
- Suppose $\hat{\lambda}[R] < \gamma[R]$. Same argument as in the $\hat{\lambda}[R] = 0$ case: we have the option to pick $\gamma[R]$, so if we don't it means that our choice has a better bang-for-buck.
- Suppose $\hat{\lambda}[R] > \gamma[R]$. Obviously, the placement of the first $\gamma[R]$ facilities by our algorithm contributes the same to $p(\hat{\lambda})$ as the placement of $\gamma(R)$ facilities in the optimal solution contributes to $p(\gamma)$. Now, note that the bang-for-buck of placing $\hat{\lambda}[R] - \gamma[R]$ facilities after having placed $\gamma[R]$ facilities must be $\geq$ than the bang-for-buck attained by the pick that caused $\lambda[R]$ have $> \gamma[R]$ facilities. Hence, we know that the bang-for-buck of those last $\hat{\lambda}[R] - \gamma[R]$ facilities placed in R is greater than the bang-for-buck of the placement of those same facilities done by γ.

□

Proof (Conjecture 4.1 under Wishful Thinking): Let us split the assignment λ that our algorithm returns into three sub-assignments λ_1, λ_2, and λ_3 such that $\lambda = \lambda_1 \cup \lambda_2 \cup \lambda_3$ and for each region $R \in \mathcal{R}$ we have some $\lambda_i(R) = \lambda(R)$, i.e., we don't separate the facilities placed in any given region. We denote this partition of λ as $\mathcal{P}_0 := \{\lambda_1, \lambda_2, \lambda_3\}$. We'll transform this partition into a different one such that each part $\lambda_i \in \mathcal{P}_0$ satisfies that either (1) λ_i assigns its facilities to only one region or (2) λ_i assigns at most $\frac{k}{2}$ facilities. Then, by Claim 4.2 and Lemma 4.4, we have that each part contributes a profit $\leq$ alg and hence $\text{opt}(k) \leq 3 \cdot \text{alg}$.

Assign $j \leftarrow 0$. If each $\lambda_i \in \mathcal{P}_j$ assigns $\leq \frac{k}{2}$ facilities, we are done, so assume wlog that we have λ_1 assigns more than $\frac{k}{2}$ facilities. Thus, both λ_2 and λ_3 assign $< \frac{k}{2}$ facilities. If λ_1 is composed of only one region, we are done, so assume otherwise. Reassign the facilities from the region with least facilities of λ_1 to $\lambda_2 \in \mathcal{P}$ and name the resulting partition as $\mathcal{P}_{j+1}$. Repeat until λ_1 assigns no more than $\frac{k}{2}$ facilities or assigns all of its facilities to a single region.

By the end, the last partition $\mathcal{P}_j$ will satisfy that both λ_1 and λ_3 assign at most $\frac{k}{2}$ facilities, λ_1 assigns at least $\frac{k}{4}$ facilities, and λ_2 assigns at most $\frac{3}{4} \cdot k$ facilities. Now, we repeat the same process but instead of unloading λ_1 onto λ_2, we unload λ_2 onto λ_3. At the end, we know that both λ_1 and λ_2 assign at least $\frac{k}{4}$ facilities and thus λ_3 assigns at most $\frac{k}{2}$ facilities, concluding the proof.

□

4.2 Without assuming Wishful Thinking

We now proceed to attempt to remove the restriction of Wishful Thinking. We are able to prove the following weaker version of Lemma 4.4:

Lemma 4.5. $\text{alg} \geq \frac{1}{3} \cdot \text{opt}(\frac{k}{2})$.

For the sake of brevity, the proof of Lemma 4.5 has been moved to the appendix (see A).

Now, in order to show Conjecture 4.1, we would like to approach the analysis in the same way we did under Wishful Thinking and use that the facility allocations of the optimal solution can be partitioned into 3 sub-assignments such that each assignment contains either $\leq \frac{k}{2}$ facilities or assigns every facility to a single region. Then, using Claim 4.2 and Lemma 4.5 we would see this implies $\text{alg} \geq \frac{1}{9} \cdot \text{opt}$.

However, there is a bug in this proof: without Wishful Thinking, it is not true that $p(\lambda) = p(\lambda_1) + p(\lambda_2) + p(\lambda_3)$ since the profit that we get from a given region depends on the facilities we might have placed on some other region. If these two regions get assigned to different sub-assignments, we may change the total profit. We thus see we are unable to make a comparison between alg and opt in this way.

4.3 Not all hope is lost

If we recall the motivating statement of this problem, we remember the scenario of placing electric vehicle chargers across an area. It is not unreasonable to assume that a location problem such as this would be subject to the phenomenon of decreasing marginal utility, where the increase in profit from additional facilities is decremental. This leads us to an assumption of sub-linearity for the region profit functions:

Assumption 4.6 (Bounded growth).

$$z \cdot g_c(j) \geq g_c(z \cdot j) \quad \forall c \in \mathcal{C}, j \in [1, k], z \in \mathbb{Z}_+ \tag{1}$$

Proof (Conjecture 4.1 assuming Bounded Growth): With this assumption we see that Lemma 4.5 implies a $\frac{1}{6}$-approximation as $2 \cdot \text{opt}(\frac{k}{2}) \geq \text{opt(k)}$. □

5 Conclusion

Although this shows that there exists a constant factor approximation for this problem under function constraints, we have no guarantee that $\frac{1}{6}$ is the best we can do, and we suspect that it is not. One should note that Lemma 4.5 is in fact not tight and being a little more clever with an analysis could possibly lead to Lemma 4.5 being $\text{alg} \geq \frac{1}{2} \cdot \text{opt}(\frac{k}{2})$. Additionally, the proof of hardness compares this problem with the k-center problem which is known to have a PTAS [**?**], meaning it is certainly possible a PTAS exists for this problem as well.

In this paper, we show issues with our conjecture that the proposed greedy algorithm with swaps is a $\frac{1}{9}$-approximation. However, these issues do not necessarily imply this is false, and with some slight modifications to the analysis or the algorithm, it is likely possible to achieve a constant-factor approximation. Finally, this paper did not investigate non-greedy approximation techniques, and the authors believe involving other approaches, such as linear programming, may lead to a stronger approximation.

A Appendix

Proof (Lemma 4.5): Let us consider the algorithm's facility allocations at some iteration, t, of the while loop. Let the allocation the algorithm makes at iteration t be represented by the pair R_t, j_t (where R_t is the region the facilities are allocated to and j_t is the number of facilities allocated at time t), and let the number of loop iterations made by the algorithm be T. Additionally, let λ_t represent all allocations the algorithm has made at the end of iteration t and let the additional profit gained at iteration t be $\Delta_t(\lambda)$. Formally, $\Delta_t(\lambda) := p(\lambda_t) - p(\lambda_{t-1})$.

Just as in 4.4, let γ denote an optimal assignment of $\frac{k}{2}$ facilities, so that $p(\gamma) = \text{opt}(\frac{k}{2})$. Let $p_t(\gamma)$ be the total profit which adding all allocations in γ to λ_t would generate. Formally, $p_t(\gamma) := p(\lambda_t + \gamma) - p(\lambda_t))$.

Finally, we define $\hat{t}$ to be the smallest time satisfying the equation $\sum_{R \in \mathcal{R}} \lambda_{\hat{t}}[R] \geq \frac{k}{2}$. We now note that when $t \leq \hat{t}$

$$\frac{\Delta_t(\lambda)}{j_t} \geq \frac{p_t(\gamma)}{\frac{k}{2}} \tag{2}$$

as all allocations in γ can be made by the algorithm at this time, some allocation in γ must be at least as good as its overall bang for buck, and the algorithm will always make the best bang for buck allocation.

Next, $\forall t \leq \hat{t}$, when the algorithm makes the allocation R_t, j_t it has at most removed $\Delta_t(\lambda)$ from the profit functions of all overlapping regions in G due to the summation nature of overlapping regions and monotonic nature of the profit functions. Thus we get the equation

$$p_t(\gamma) \geq p_{t-1}(\gamma) - \Delta_t(\lambda) \tag{3}$$

Combining equations 2 and 3 we get

$$\frac{k}{2 \cdot j_t} \cdot \Delta_t(\lambda) \geq p_{t-1}(\gamma) - \Delta_t(\lambda) \implies \Delta_t(\lambda) \geq \frac{1}{\frac{k}{2 \cdot j_t} + 1} \cdot p_{t-1}(\gamma) \tag{4}$$

Additionally, we know $p_t(\gamma) = p(\gamma)$ when $t = 0$, thus we can use equation 3 to generate the recurrence $p_t(\gamma) \geq p(\gamma) - \sum_{t'=0}^{t}(\Delta_{t'}(\lambda))$, giving us

$$\Delta_t(\lambda) \geq \frac{1}{\frac{k}{2j_t} + 1} \cdot \left(p(\gamma) - \sum_{t'=0}^{t-1} (\Delta_{t'}(\lambda)) \right) \tag{5}$$

Finally, we note $\sum_{t=0}^{\hat{t}}(\Delta_{t'}(\lambda)) \leq \sum_{t=0}^{T}(\Delta_{t'}(\lambda)) = p(\lambda_T)$. Using equation 5 we now find

$$p(\lambda_T) \geq \sum_{t=0}^{\hat{t}} (\Delta_t(\lambda)) \geq \sum_{t=0}^{\hat{t}} \left(\frac{1}{\frac{k}{2j_t} + 1} \cdot \left(p(\gamma) - \sum_{t'=0}^{t-1} (\Delta_{t'}(\lambda)) \right) \right) \tag{6}$$

From lemma A.1 we see with some manipulation, equation 6 implies $p(\lambda) \geq \frac{1}{3} \cdot p(\gamma)$. □

Lemma A.1. *Equation 6 implies* $p(\lambda) \geq \frac{1}{3} \cdot p(\gamma)$

Proof: In order to turn the sums over $\hat{t}$ iterations into something expandable, we must shift the way we consider our allocations. Let us construct a mapping, M, of each facility allocation made by the algorithm, (R_t, j_t), to a set of j_t facility allocations of size one, $\{(R_{t.0}, 1), (R_{t.1}, 1), ..., (R_{t.j_t}, 1)\}$. We now define the profit of each of these size one facility allocations, $\Delta_{R_{t.x}}(\lambda)$, to be $\frac{\Delta_t(\lambda)}{j_t}$. Finally, we define the set of all mapped allocations up to iteration $\hat{t}$ to be $A := \cup_{t=0}^{\hat{t}}(M(R_t))$. Note that $|A| \geq \frac{k}{2}$, the number of

facilities allocated at each element in A is 1, and $\sum_{t=0}^{\hat{t}} \Delta_t(\lambda) = \sum_{a \in A} \Delta_{R_{t,x}}(\lambda) = p(\lambda)$. We now substitute this expression into equation 6

$$\begin{aligned} p(\lambda_T) &\geq \sum_{a \in A} \left(\frac{1}{(\frac{k}{2}+1)} \cdot \left(p(\gamma) - \sum_{t'=0}^{t-1} (\Delta_{t'}(\lambda)) \right) \right) \\ &\geq \frac{1}{(\frac{k}{2}+1)} \left(\frac{k}{2} \cdot p(\gamma) - \sum_{a \in A} \sum_{t'=0}^{t-1} (\Delta_{t'}(\lambda)) \right) \end{aligned} \tag{7}$$

We note that $\sum_{t'=0}^{t-1} (\Delta_{t'}(\lambda)) \leq p(\lambda_T)$ as it will exactly equal $p(\lambda_T)$ at $t-1 = T$

$$\begin{aligned} p(\lambda_T) &\geq \frac{1}{(\frac{k}{2}+1)} \left(\frac{k}{2} \cdot p(\gamma) - \sum_{a \in A} \sum_{t'=0}^{t-1} (\Delta_{t'}(\lambda)) \right) \\ &\geq \frac{1}{(\frac{k}{2}+1)} \left(\frac{k}{2} \cdot p(\gamma) - \sum_{a \in A} p(\lambda_T) \right) \\ &\geq \frac{2}{k+2} \left(\frac{k}{2} \cdot p(\gamma) - \frac{k}{2} \cdot p(\lambda_T) \right) \end{aligned} \tag{8}$$

$$(1 + \frac{k}{k+2}) \cdot p(\lambda_T) \geq \frac{k}{k+2} p(\gamma) \tag{9}$$

$$(\frac{k+2}{k} + 1) \cdot p(\lambda_T) \geq p(\gamma) \tag{10}$$

Assuming $k \geq 2$ (as when $k = 1$ it is easy to see our algorithm will always make the best choice), this gives us

$$3 \cdot p(\lambda_T) \geq p(\gamma) \tag{11}$$

Finally, we know $p(\lambda)$ will only be $\geq p(\lambda_T)$ Thus giving us

$$3 \cdot p(\lambda) \geq p(\gamma) \tag{12}$$

□