MOUNIR AFIFI

FOR: EPISTEME

GOLD BARS TO SHEFFIELD

Contents

Brief Summary and Motivation:

The problem can be summarized as follows: a bank wants to transport a number of gold bars from their current location to Sheffield. They have to pay a tax upon entry to every town or village crossed, but not upon leaving. The tax at villages is one gold bar regardless of the amount transported, whereas at towns the tax is evaluated as the rounded up value of the twentieth of the amount at the entry. Given a fixed number of gold bars that must reach Sheffield, how much should be carried to begin with while paying the least amount of tax?

The solution of the problem therefore consists in minimizing the cost by choosing the minimal path from source to destination. The obvious method would be to check every single path, calculate the cost, and choose the smallest value. We can improve upon this idea by understanding the problem more deeply, from a mathematical standpoint, and finding some formulas or heuristic rules to minimize the search set.

Assumptions:

In order to understand our problem, we have to explicitly lay down all the basic assumptions about its variables and aspects. Some of these assumptions were explicit in the problem description, and other ones could be deduced from its wording.

1. The number of villages and towns each cannot exceed 26, since they are represented only as Latin alphabet letters. In general, this may not necessarily be true, but considering our problem specifications, it is.
2. A map cannot be empty or contain a single node (village or town). This would be a trivial problem of no practical importance.
3. A road must link two distinct nodes (villages or towns).
4. The number of gold bars cannot be equal to zero, since the bank would have no interest in transporting 0 gold bars to another location, and paying taxes for it!
5. The number of gold bars cannot exceed 1000.
6. The problem must always have a solution, that is, there must always be a path from source to destination.
7. A path must be non-cyclic (source and destination are not the same node) and non-repetitive (does not pass through the same node twice), otherwise it would be by default a wasteful path.

Preliminary calculations:

Based on these assumptions we can make some preliminary calculations, to evaluate the scope of our problem.

- The total number of possible nodes in a map would be 52, including the starting position and the destination.
- The largest map would therefore have $\frac{52*51}{2} = 1326$ roads.
- The total number of non-repetitive paths on this map would be

$$\sum_{k=2}^{52} C(k, 52) * C(2, k)$$

where $C(k, n)$ is the number of combinations formed by randomly selecting k elements from a total of n elements.

The $C(k, 52)$ component corresponds to choosing k nodes from the total 52.

The $C(2, k)$ component corresponds to choosing a non-cyclic non-repetitive path that links all k nodes together, i.e. choosing 2 nodes among the k nodes which would be disconnected.

The formula evaluates to

$$1326*2^{50}$$

If we fix the starting and ending positions of the paths, this would be like choosing one combination of 2 nodes out of 52 possible nodes, which is equal to one combination out of 1326 possible combinations, thus the total number of paths on a map with fixed starting and ending positions would be 2^{50}.

The sheer size of this number shows that it would be senseless to evaluate the cost of every path in order to find the minimal path, and therefore we need a method to decrease the number of paths to be checked.

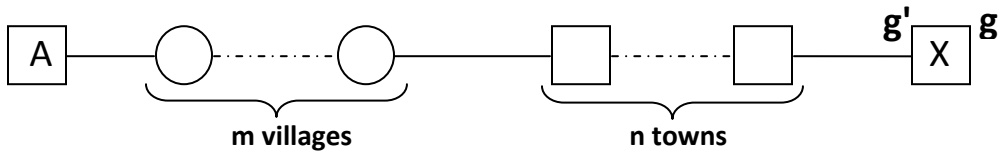
The Min-Max formulas:

In order to eliminate as many paths from the search set as possible, we must find some general rule or rules that can be applied in every case, using general parameters such as the number of villages, the number of towns and the number of gold bars to be delivered. For every path, we define the measure of the path to be its cost relative to the gold bar count. A path p_1 is smaller than another path p_2, if the cost of p1 is smaller than the cost of p2. But since calculating this cost is the aim of the our whole endeavor in the first place, not to mention that the gold bar count is not a fixed parameter, we must find another way to determine or approximate the cost indirectly, with less calculations.

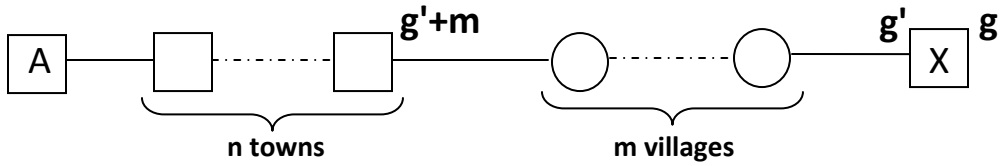
We define the pseudo-measure of a path to be the pair (m,n), such that m is the number of villages on the path, and n is the number of towns. Intuitively, we would expect that if two paths p_1 and p_2 have pseudo-measures (m_1,n_1) and (m_2,n_2) respectively, such that $m_1 \leq m_2$ and $n_1 \leq n_2$, then $p_1 \leq p_2$.

However, this is not necessarily always true, as the exact order in which the villages and towns are placed can affect the value of the true measure (the cost). Consider for example these two paths, with the same pseudo-measure (m,n):

case 1:



case2:



In the first figure (case 1), the m villages are clustered near the starting position, followed by the n towns, while the order is inverted in the second figure (case 2).

Let *g* be the number of gold bars to be delivered. We will posit *g'* as the number of gold bars before taxation at the destination. Considering that the amount to transport for *n* towns will be *g'* for case 1, and *g'+m* for case 2, we can estimate that the total number of gold bars to transport at A will be larger for case 2 compared to case 1. In fact, if we consider all possible combinations of m villages and n towns, we can intuitively predict that case 2 represents the maximal case, and case 1 represents the minimal case. Let us formulate this more rigorously.

Let c_k be the cost at town k (starting from the destination) for case 1, and d_k be the cost at town k for case 2. Considering boundary conditions on c_k and d_k at each town, we find:

$$g'\sum_{i=1}^{k}(\frac{1}{19})^i \le c_k \le g'\sum_{i=1}^{k}(\frac{1}{19})^i + \sum_{i=0}^{k-1}(\frac{1}{19})^i$$

Thus:

$$g'\frac{1}{18}(1 - (\frac{1}{19})^k) \le c_k \le g'\frac{1}{18}(1 - (\frac{1}{19})^k) + \frac{19}{18}(1 - (\frac{1}{19})^k)$$

By analogy for case 2 we get:

$$(g'+m)\frac{1}{18}(1 - (\frac{1}{19})^k) \le d_k \le (g'+m)\frac{1}{18}(1 - (\frac{1}{19})^k) + \frac{19}{18}(1 - (\frac{1}{19})^k)$$

Let $C_{m,n}$ be the total cost for case 1, and $D_{m,n}$ be the total cost for case 2. We have:

$$C_{m,n} = m + \sum_{k=1}^{n} c_k$$

$$D_{m,n} = m + \sum_{k=1}^{n} d_k$$

Thus:

$$m+g'\frac{1}{18}(n - \sum_{k=1}^{n}(\frac{1}{19})^k) \le C_{m,n} \le m+g'\frac{1}{18}(n - \sum_{k=1}^{n}(\frac{1}{19})^k) + \frac{19}{18}(n - \sum_{k=1}^{n}(\frac{1}{19})^k)$$

$$m+(g'+m)\frac{1}{18}(n - \sum_{k=1}^{n}(\frac{1}{19})^k) \le D_{m,n} \le m+(g'+m)\frac{1}{18}(n - \sum_{k=1}^{n}(\frac{1}{19})^k) + \frac{19}{18}(n - \sum_{k=1}^{n}(\frac{1}{19})^k)$$

=>

$$m+g'\frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n) \le C_{m,n} \le m+(g'+19)\frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n)$$

$$m+(g'+m)\frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n) \le D_{m,n} \le m+(g'+m+19)\frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n)$$

Let's call these "the Min-Max inequalities". To compare $C_{m,n}$ and $D_{m,n}$, we have to compare their cross boundaries (the lower boundary of $C_{m,n}$ with the upper boundary of $D_{m,n}$ and vice versa).

To evaluate the condition for $C_{m,n} \le D_{m,n}$ we set X as follows:

$$X = m+g'\frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n) + \frac{19}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n) - m-(g'+m)\frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})^n))$$

We are trying to find the condition on m and n, such that $X \leq 0$.

$$X = \frac{19}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right) - \frac{m}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right)\right)\right)$$

$$X = \left(\frac{19}{18} - \frac{m}{18}\right)\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right)\right)$$

$X \leq 0 \Leftrightarrow$ { $n = 0$ and $m \leq 19$ (this is a trivial case however, since the path contains only villages)
or
$n \geq 1$ and $m \geq 19$ (i)

To evaluate the condition for $C_{m,n} \geq D_{m,n}$ we set Y as follows:

$$Y = m + (g' + m)\frac{1}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right) + \frac{19}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right) - m - g'\frac{1}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right)\right)\right)$$

We are trying to find the condition on m and n, such that $Y \leq 0$.

$$Y = \frac{m}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right) + \frac{19}{18}\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right)\right) = \left(\frac{m}{18} + \frac{19}{18}\right)\left(n - \frac{1}{18}\left(1 - \left(\frac{1}{19}\right)^n\right)\right)$$

We have $Y \geq 0$, $\forall m \geq 0$, $\forall n \geq 1$

Since the implication $Y \leq 0 \Rightarrow C_{m,n} \geq D_{m,n}$ works only in one direction, we cannot draw any conclusions from this result.

If we have a random configuration of m villages and n towns on a given path, we should expect a cost $S_{m,n}$ such that $C_{m,n} \leq S_{m,n} \leq D_{m,n}$ (ii) at least under the condition (i), which is a sufficient but not necessary condition. This is because villages have a "screening effect" on towns which tends to increase when we increase the number of villages that are clustered together on the side of the destination, and tends to decrease otherwise. For $m \leq 18$, it's likely that we would need to use brute force on every possible value of m, n and g' to check whether (ii) holds true, which is feasible but would be very time- and CPU-consuming.

Applications:

The above formulas have several important applications that would help reduce the complexity of our program.

Maximum value of gold bars:

If we have the maximum value of gold bars, we would be able to calculate the minimal path on a map without having to store the total gold bar count for each path in a list, and then calculate the minimum of the list, which would require more memory and CPU.

This is feasible because we have set upper boundaries on the values of m, n and g, as we explained in the Assumptions section. We simply need to calculate the maximum value of g', and then replace the values in the formula for the upper boundary of $D_{m,n}$.

We have:

$$g_{max} = 1000$$

The value of g'_{max} is either 1001 (if the destination is a village) or 1053 (if the destination is a town), so we take the second value.

$$B_{max} = m+(g'_{max} +m) \frac{1}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})n) +\frac{19}{18}(n - \frac{1}{18}(1 - (\frac{1}{19})n)$$

Where $m = 26$, $n = 26$, and $g'_{max} = 1053$

We get:

$$B_{max} \approx 1608,61$$

Since the maximum value would be an integer, we take $B_{max} = 1608$. This is larger than the value used for "lowest" in "cheapest" function, which was 1535, due to a mistake in calculations, but it should not affect the result in most cases, since we would only approach such values if g is close to 1000 and if the shortest path on the map has a number of villages and towns close to 26 each, with most of the villages clustered near the destination.

Comparison rules:

In this section, we will represent the paths with their pseudo measures. In other words, the notation $(m,n) \le (m',n')$ means "the path whose pseudo-measure is (m,n) has lower cost compared to the path whose pseudo-measure is (m',n')".

1. The path (0,0) is the smallest possible path. This means that if a map has a direct road between the source and destination, the lowest cost would be the tax at the destination.
2. $(1,0) \le (0,1)$, since the tax at a town will at minimum equal that of a village.
3. $m \le m' \Rightarrow (m,0) \le (m',n')$. This can be proven easily for $m \ge 19$ by subtracting $C_{m,0} = m$ from $D_{m',n'}$ in the second Min-Max inequality for $n \neq 0$. The case n = 0 is trivial and holds true as well. For $m \le 18$, this is also easy to prove without the use of the inequalities (we would use the congruence of the gold bar count modulo 20). This is a generalization for the first rule.
4. $n < n' \Rightarrow (0,n) \le (0,n')$. The use of the Min-Max inequalities in this case will not be useful, but it is also obvious that adding more towns will increase the cost, if there are no villages.
5. $n < n' \Rightarrow (1,n) \le (1,n')$. This is also easy to prove with congruence, since the cost of (1,n) will at maximum be 1+the cost of (0,n) (if the gold bar count before the village is not congruent to 1 modulo 20), and the cost of (1,n') will at minimum be the cost of (0,n') (if the congruence is otherwise), and since according to rule 4, $n < n' \Rightarrow (0,n) \le (0,n')$, we get $(1,n) \le (1,n')$.
6. By recursion, we can similarly show that $\forall m \ge 0$, $n < n' \Rightarrow (m,n) \le (m,n')$. This is a rule that was only deduced after the deadline, and thus has not been used in the program.
7. $m \le m'$ and $m' - m \ge 3 \Rightarrow (m,1) \le (m',1)$. This rule works in general and not only for $m \ge 19$ since it derives from inequalities that are similar to the Min-Max inequalities, but in the general case of random distribution of towns and villages (which is easier to tackle since we have only one town). This rule was not used in the program. Instead we used rule 8, which is slightly more powerful, but can only be confidently applied for $m \ge 19$.
8. $m \le m'$ and $m' - m \ge 1 + \frac{m'-1}{20}$ and $m \ge 19 \Rightarrow (m,1) \le (m',1)$.
9. This rule: $m \le m'$ and $m' - m \le \frac{m+1}{20} - 1$ and $m \ge 19 \Rightarrow (m,1) \ge (m',1)$, is theoretically true, but is in fact mostly useless and even practically wrong considering that the comparison function will return 1 in such cases. In fact:

   $$m \le m' \text{ and } m' - m \le \frac{m+1}{20} - 1 \Rightarrow m' = m, \text{ if } 19 \le m \le 26.$$

10. It is tempting to think that $\forall m \ge 0$, $\forall n \ge 0$, $m < m'$ and $n < n' \Rightarrow (m,n) \le (m,n')$. This in fact is likely true in most, and possibly all cases, but is harder to show rigorously. Using the same method we used to prove rule 7, we can show that $m \le m'$ and $m' - m \ge k(n) \Rightarrow (m,n) \le (m',n)$, with k(n) a function of n, and combined with rule 6, we will get $m \le m'$ and $n < n'$ and $m' - m \ge k(n) \Rightarrow (m,n) \le (m',n')$. This rule has also not been used in the program.