Photo Slideshow Optimization

Rajesh Kudupudi
Virginia Tech
rajeshk19@vt.edu

Chieh Tsao
Virginia Tech
chiehtsao@vt.edu

Abstract

In the wake of recent technological advances smartphone has become an integral part of every person's life. These smartphones come equipped with cameras using which people tend to capture their memories. Given the amount of data this generates, many interesting problems which do not exist before can be solved. One of such is Photo Slideshow. It arranges some photos as a slide show such that the user is interested. This arrangement of photos is based on tags which describe things of an image. This is called an photo slide show optimization problem. In the work we formulate this problem from an optimizations standpoint and also discuss 3 approaches which can be used to solve this problem. One of the approaches is to look at this problem as a search problem in a tree and use a brute-force method to solve it. Another method prunes this tree. The last approach formulates this problem as a mixed integer programming problem and solves it. We compare these 3 approaches on the basis of time, accuracy and some other factors. We comment on these methods their merits, demerits and discuss what approach to use in different scenarios. We will present this analysis on 2 data sets one from google and other curated by us. Our code is available at code here.

1. Introduction

As the famous saying goes, "a picture is worth a thousand words". Nowadays, because of the wide-spread use of smartphones taking photos, documenting moments and capturing memories have become a major part of our life. It goes without saying that photos have become an important part of contemporary digital and cultural life. According to Google Photos, it was backing up more than 1.2 billion photos and videos per day in 2017. So, as a human it is our natural tendency to review those photos relish and live in those moments again. However, due to the vast collections of photos each human takes it is almost impossible to review all of them. And the order in which we view these photos also reminds us of a certain story. This presents an unique problem which is: How to show these photos? More precisely how do we arrange these photos as a slideshow so that user is interested. Notice that Google photos application on android smartphones complies photos in a certain fashion as presents it to the user of smartphones, as a slideshow, highlights etc... These photos are arranged such that it keeps us interested.

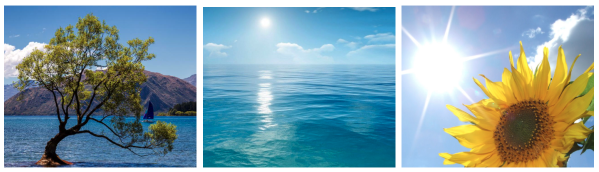
Figure 1. This is a decent slide-show of a scenery, where the images are arranged in a top to bottom fashion.

We arrange these slides (images) by computing interest factor on image descriptors or tags associated with each slide (image). In the next section of this work we define how do we quantify interest between 2 slides (images). Due to the advancements in Deep Learning and Convolutional Neural Networks [1] it has been feasible to extract different descriptors (tags) which describe an image with high reliability. These descriptors can be strings or any unique identifiers which explain different aspects of an image.

Our goal here is to arrange the images in such a way that user is interested and user wants to view the next image in the slideshow. We present 3 different approaches to solve this problem of photo slideshow optimization. The first 2 approaches are based on search and the last approach involves formulating this problem as a integer programming problem with binary constraints on variables.

2. Datasets

We intend on working with 2 datasets. One is provided by Google Hashcode Archive Challenge [2]. Our problem statement is loosely based on this challenge. So, Google has already created a descriptors for each image and stored them in a text file. This text file which is our dataset consists of number of images ($> 10k$), and one or more tags which are associated with each image.

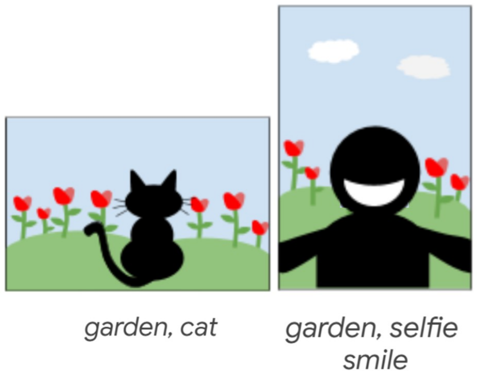

Figure 2. Two slides, slide S1 which contains a cat, and S2 which contains a person smiling

We also create a new database using images from our smartphone's gallery. We individually label each image depending on the content of the images. We use these labels or tags and create a slide show. We present a few of these slideshows in the later part of the report.

3. Interest Factor

Computing the interest factor is the integral part of our algorithm to solve Photo Slideshow problem. The interest factor is computed on 2 adjacent images in a slideshow. It is based on how interesting the transitions between each pair of subsequent slides (neighbors) are. It is formulated with these things in mind: The transitions between any 2 neighboring slides should have something in common, this is to preserve continuity (the two slides should not be completely different). But they should be different enough such that the audience going through the slideshow is still interested.

For two subsequent slides S_i and $S_i + 1$, the interest factor is minimum of:

- The number of common tags between the slides S_i and S_{i+1}.
- The number of tags in slide S_i but not in S_{i+1}.
- The number of tags in slide S_{i+1} and not in S_i.

The interest factor for 2 slides S_1 and S_2 shown in figure 2 is shown below. Here:

- The number of common tags is 1.
- The number of tags in slide S_1 but not in S_2 is 1.
- The number of tags in slide S_2 but not in S_1 is 2.

The calculation of interest factor is shown as a graphic in fig(3). So interest factor for any 2 slides can be expressed

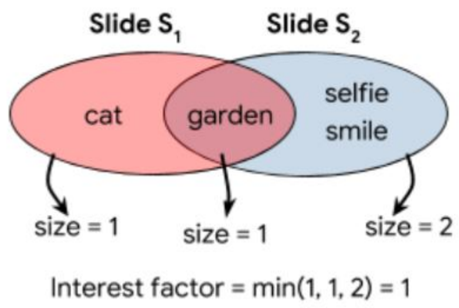

Figure 3. Interest factor of slides S1, S2 shown in figure 2

as: Let T_i and T_j be the set of tags associated with S_i and S_j. $len(T_i)$ corresponds to length of the set T_i

$$\textit{Interest factor}(S_i, S_j) = \min(len(T_i) - len(T_i \cap T_j),\\ len(T_i \cap T_j), len(T_j) - len(T_i \cap T_j))$$

For slideshow of S slides the total interest factor will be sum of interest factor of neighboring slides.

4. Our Approaches

We can formulate this problem of photo slide-view optimization as an arrangement problem, where we arrange all the N or some of the N photos in an order (We solve it for arranging all the N photos but our approach can be easily extended if we need only some photos) such that it maximizes the sum of total interest factors between all the neighboring photos. The initial optimization objective can be written as:

$$\max_{\tau_i}\left(\sum_{k=1}^{N-1} \textit{Interest Factor}(S_k^{\tau_i}, S_{k+1}^{\tau_i})\right) \tag{1}$$

Here τ_i is the arrangement or sequence. We need to find an arrangement such that it maximizes the term above. In the expression above $S_k^{\tau_i}$ is the kth image in the arrangement τ_i. We present 3 approaches in that order to solve this problem. In approach 1, 2 we treat this as a search problem where we have to find the sequence τ_i such that it maximizes the optimization objective above.

4.1. Search

We can think of the problem at hand as a tree where each node corresponds to a slide and connections between every node is the interest factor between these slides. We need to find a path between all the nodes without cycles such that every node is visited once and only connected to 2 nodes (barring the start and the end node which are only connected to one node). In our first approach we pick a starting slide (node) randomly and greedily select the next slide, and arrange the other slides in a stack such that we can visit them

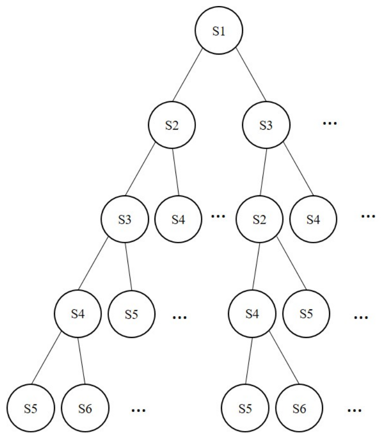

Figure 4. This Tree has slides as nodes and edge weights are based on interest factor between 2 slides.

later. We keep exploring in a depth first manner and finally when there are no more nodes to explore we remember the path and store the sum of interest factors (score) along the path. Then we backtrack along this path (using stack) and explore the other nodes to find any other path which maximizes the objective, if we find any other path which does this we replace our maximum path with this path and we replace even the score.

This approach searches all the paths in the tree and only then it can give a best solution. The time complexity of this backtracking approach is $O(N!)$. This is an inefficient approach so our computer might run out of time before finding a solution. Since we are exploring greedily we might have a decent solution, which might not be optimal. Another comment on approach 1 is if we choose to create a slide-show of length (3-5) or some finite number (given 100 images as input, which is often the case in many of our photo galleries) then we think this approach might be able to give us a good solution. We call this approach **bruteforce-search approach, (Approach-1)**.

4.2. Minor improvements to Approach 1

We present a minor modification to algorithm presented above to improve it. If we look closely at the tree on which our algorithm is operating on, we can actual prune some of the nodes. If we look into the sub-tree below nodes S_4 at depth 3 in the tree above, they are similar. If all the nodes under node S_2 at depth 1 are explored exhaustively and we remember the solution for all the subtrees (dynamic programming) then the subtree below $S_1- > S_3- > S_2- > S_4$ is similar to subtree below $S_1- > S_2- > S_3- > S_4$, where the former has already been explored. Same holds true for the adjacent node at depth 3, $S_1- > S_3- > S_2- > S_5$. This pruning is a minor improvement which can be incorporated in approach 1. We call this approach **search-with-pruning approach, (Approach-2)**.

5. Optimizations based solution

The Approach 1 is inefficient and a brute force way of solving the problem of photo slide-show optimization. So our next approach involves formulating this objective in (1) as a known optimization problem with constraints, such that we can use any known optimization algorithm to solve it from there. We were able to formulate this photo slide-show optimizations as an integer programming problem with binary constraints on the variables.

Before we dive into the details of our formulation we would like to introduce a few matrices. First matrix we talk about is an interest factor matrix (C) which is an NXN matrix where N is the total number of images to be arranged.

$$\text{C} = \begin{bmatrix} 0 & c_{12} & c_{13} & ... & c_{1N} \\ c_{21} & 0 & c_{23} & ... & c_{2N} \\ .... \\ c_{N1} & c_{N2} & c_{N3} & ... & 0 \end{bmatrix}$$

This is an interest factor matrix where each element represents interest factor between 2 slides. c_{12} represents the interest factor between slides 1 and 2. There are some interesting properties about this matrix which are any $c_{ii} = 0$, for $1 \leq i \leq N$. Because interest factor with the same slide is always 0. So all the diagonal elements are 0. And another interesting fact about this matrix is: This C matrix is symmetric with respect to its diagonal. This is because:

$$\textit{Interest Factor}(i,j) = \textit{Interest Factor}(j,i)$$

We define another matrix an indicator matrix X which has the same dimensions NXN. Each element in this matrix can only take values either 0 or 1.

$$\text{X} = \begin{bmatrix} 0 & x_{12} & x_{13} & ... & x_{1N} \\ x_{21} & 0 & x_{23} & ... & x_{2N} \\ .... \\ x_{N1} & x_{N2} & x_{N3} & ... & 0 \end{bmatrix}$$

An element $x_{ij} = 1$ means that slide j follows slide i. So our intuition behind creating this matrix is, any slide in our arrangement can have one slide after it. Some more important properties of this matrix are: Its diagonal elements are also 0 and it is also a symmetric matrix wrt its diagonal elements. The first property is true because we cannot reuse an image twice. So $x_{ii} = 0$ for all $1 \leq i \leq N$.
Optimization objective:

$$max_X \sum_i^N \sum_j^N c_{ij} * x_{ij} \tag{2}$$

Constraints:

$$x_{ij} = \{0,1\} \; \forall i,j \in 1..N \tag{3}$$

$$\sum_{i}^{N} x_{ij} = 1 \; \forall j \in 1..N \tag{4}$$

$$\sum_{j}^{N} x_{ij} = 1 \; \forall i \in 1..N \tag{5}$$

$$x_{ii} = 0 \; \forall i \in 1..N \tag{6}$$

$$x_{ij} + x_{ji} = 1 \; \forall i, j \in 1..N \tag{7}$$

Note: Our formulation here is slightly different from the one presented in our proposal. Some of the arrangements/sequences, were not expressible using the previous formulation. These are explored in more detail in appendix section.

This is a binary programming problem with constraints. By solving this problem we would be maximizing the interest factor. The diagonal elements in matrix X need not be considered as they do not affect our optimization objective as the corresponding c's are always 0. We can see here that we created X as a matrix (for our interpretability) but we are solving it as a vector. The binary constraints (3) limits all the optimization variables to take either 0,1. The (4), (5) constraints limit the row sum and column sum to 1, this is to ensure that each slide is followed by only 1 slide. The constraint in (7) is to avoid **sub-cycles** which is explored in detail inside the results section. We call this **binary-programming approach, (Approach-3)**.

One important fact about our slide show is $S_1 - > S_2 - > S_3 - > S_4$ will have the same combined interest factor score as $S_4 - > S_3 - > S_2 - > S_1$. One caveat with this method which is, we will not have a starting point while interpreting the matrix X. We expect that after solving the binary programming problem we will get a cycle. One good fix for this is, we need to break the cycle where the interest factor is lowest which gives us 2 open slides, either of them can be chosen as start or end.

5.1. Implementation

We used CVXOPT's mixed integer linear programming solver (ilp) for solving the binary-programming approach's optimization objective. A brief summary of ilp is provided below.

$$\min_x c^{'*} x$$
subject to: $G^* x \leq h$

$$A^* x = b$$

x[I] are all integer
x[B] are all binary

We have unrolled our cost matrix C into a vector, same with x, discarding the diagonal elements in both cases. We have expressed all the variables x as binary. We have formulated all our constraints row-sum (4) , column-sum (5) and the (7) in matrix A and vector b. G, h was a dummy matrix as we had no inequality constraints.

6. Evaluation metrics

6.1. Timing

We compare the time taken by all the 3 approaches, (bruteforce-search approach, search-with-pruning approach, and binary-programming approach) and plot it as a function of number of slides in the slideshow (N).

6.2. Interest-Factor Error (IFE)

Our optimization objective is finding a sequence which maximizes sum of interest factor between adjacent slides in the sequence (1). So for the optimal arrangement generated by all the 3 approaches we compute their respective interest factors. We carried out majority of our experiments on Google Hashcode Archive Dataset [2], which did not have results or (optimal solutions for each set of images) to compare against. Since bruteforce-search approach goes through all the available sequences we believe that it finds out the optimal solution. We use this as a ground truth to compute the error. We compute the error of Approach-2 and Approach-3 with Approach-1. For a given number of images (N), the Interest Factor Error (IFE) in arranging all the images is:

$$\text{IFE(N)} = |IF(\tau_{N,2}) - IF(\tau_{N,1})| \tag{8}$$

Here IF stands for interest factor. $\tau_{N,2}$ corresponds to the best arrangement by approach 2 (search with pruning) given N images to be arranged as a slideshow. This τ i similar to the one in (1). In other words IFE can be expressed as the absolute value of difference between solution of Approach 1 and any other Approach. Similarly IFE can be calculated for Approach-3 binary programming approach with respect to approach-1. (Lower the IFE the better).

6.3. Recall

IFE might not be a good representative of our Approach's performance. Sometimes we might have a very low IFE between 2 arrangements, but arrangements can be completely different. This pushes the need for a metric which can be used to compare 2 sequences. A simple straight forward comparison of 2 sequences might not be a good way because, interest factor is commutative. One good metric might be checking the ratio of common neighbors in between 2 sequences (a ground truth sequences and a sequence to evaluate) to neighbors in ground truth sequence. Our optimal solution again here refers to the solution obtained by brute force-search (Approach 1).

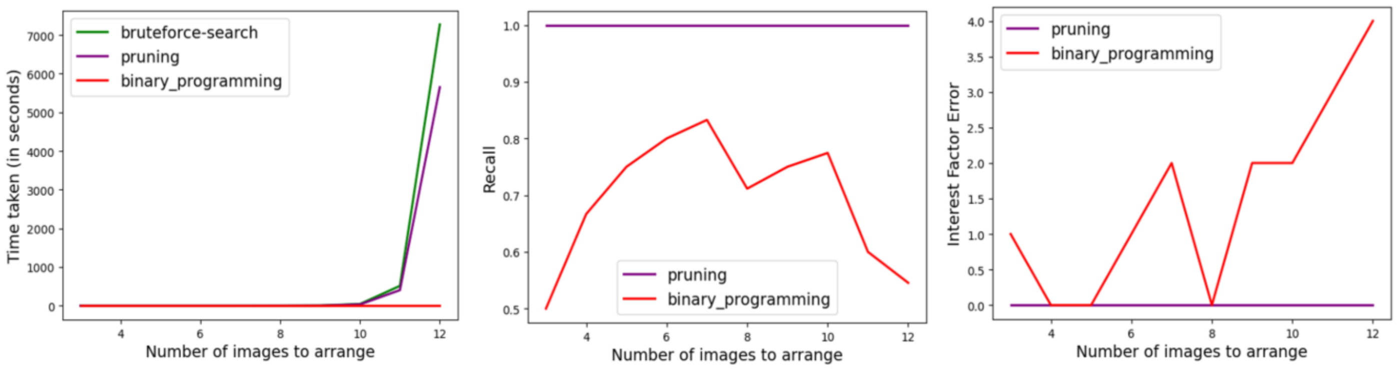

Figure 5. Quantitative metrics: Time, Recall and Interest Factor Error (IFE) are plotted against Number of slides in the slideshow (N). In the first graph from the left Time taken to create the slideshow is plotted against N for all the 3 approaches. Approach1- Brute force serach approach, Approach2- Search with Pruning, Approach3- Binary Programming. The center graph plots recall against N. For this graph it is assumed that solution from Approach1 is ideal and recall for Approach2, Approach3 is computed with respect to Approach1. The last graph the right most one, plots Interest Factor Error against N. Similar to the previous graph here also IFE is computed with respect to Approach1's solutions.

$$Recall(N) =$$

$$\frac{len(Set(Neighbors(\tau_{N,2})) \cap Neighbors(\tau_{N,1}))}{len(Neighbors(\tau_{N,1}))}$$

Here $\tau_{N,1}$ is a sequence, the best sequence of arrangement following Approach 1. $len(Neighbors(\tau))$ this corresponds to total number of neighbors in that sequence τ. The numerator corresponds to the size of set common neighbors in the two arrangements $\tau_{N,1}$ and $\tau_{N,2}$. The higher the recall the better.

7. Quantitative results

We quantitatively evaluate all the 3 approaches, with the evaluation metrics presented above.

7.1. Timing Analysis

In the first graph from the left in fig(5), we compare the time taken by bruteforce search (Approach 1), search-with-pruning (Approach 2) and binary programming (Approach 3) to arrange N images into a slideshow. Here $N \in 3, ..., 12$. We tried out only 10 different values of N because of computational and time constraints. To arrange 12 images Approach 1 and Approach 2 look into 469 million arrangements (12!), which took approximately 7272.5 seconds (2 hours) and 5649 seconds (1.5 hours) respectively for these approaches. For arranging 13 images it would be 6 billion arrangements which would be infeasible to run on our machines. On the other hand binary programming based Approach 3 was able to arrange 12 images in under 2 milliseconds. We were surprised to see how fast binary programming based approach was able to arrange slides. We wanted to test binary programming approach alone to see how many images can it arrange in limited time. We observed that to arrange 280 slides as a slideshow, binary programming based approach took under 30 seconds. We think this is really good. We did not present that results here because we had nothing to compare it against. From the timing analysis we conclude that binary programming based approach is the fastest. We can also observe from the fig(5) that Approach 2 (search-with-pruning) was faster than bruteforce search based Approach 1, which implies that pruning does save some time but not very substantial.

7.2. Recall and Interest Factor Error Analysis

In fig(5) we plot the Interest Factor Error and Recall metrics as a function of number of images to arrange (N), for Approach 2 (search-with-pruning) and Approach 3 (binary programming) with Approach 1 (bruteforce search) as the ground truth. As expected in fig(5) the pruning search (Approach 2) does not differ at all from the ground truth (Approach 1) which is expected. This leads to recall=1, IFE =0 for Approach 2. But the recall and error for Approach 3 (binary programming based approach) was not good. Although it is much faster Approach 3 has a average recall of 70%, and an IFE of around 2.0. We anticipated that recall would be a bit low because our binary programming solver gives us a cyclic arrangement of slides (not a sequence as discussed in Section 5). In a cyclic arrangement all the slides to be arranged are present with no start and end slides. We would be breaking this cycle at the points where the interest factor between adjacent slides was lowest to get a sequence of slides. But this one factor alone should not degrade the results by this margin.

After investigating and interpreting all of the results generated by binary programming solver for different values of N, we were able to conclude that the reason for low re-

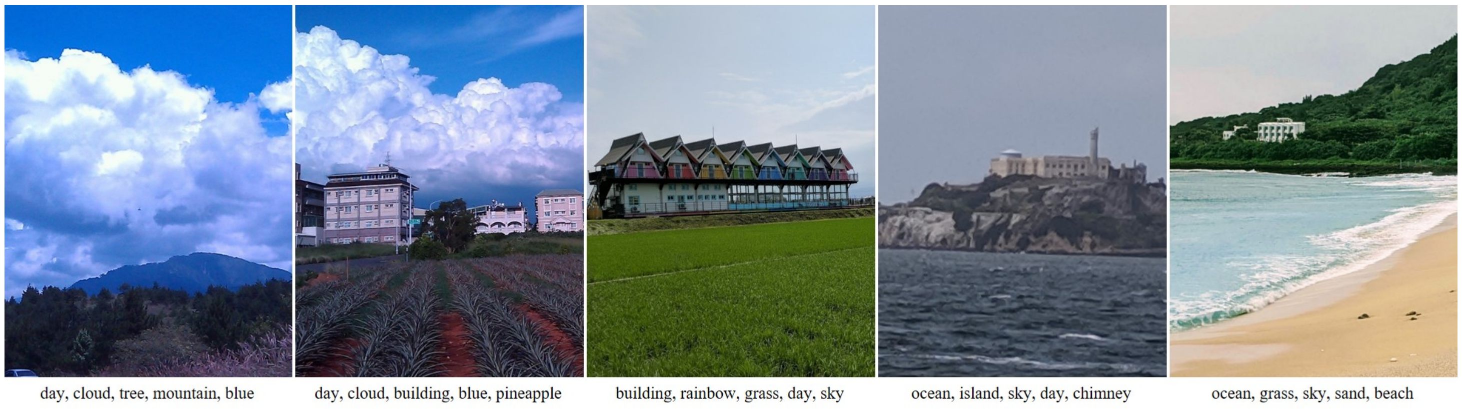
day, cloud, tree, mountain, blue
day, cloud, building, blue, pineapple
building, rainbow, grass, day, sky
ocean, island, sky, day, chimney
ocean, grass, sky, sand, beach

Figure 6. Slideshow created with a dataset curated by us. The tags corresponding to each slide is shown below the slide. The slideshow is arranged from left to right. The total interest factor is 8 for this slideshow

call was **sub-cycles**. (We decided to call them that). A sub-cycle does not contain all the images in which are supposed to be in a slideshow. Suppose we have 7 images to arrange as a slideshow $\{S_1, S_2, ...S_7\}$ a sub cycle can be $S_1 -> S_3 -> S_1$, and rest of the slides forming other sub-cycle. Whenever these kinds of sub-cycles were present our recall was low. So in order to deal with these kinds of sub-cycles we introduced a the constraint (7) in our problem formulation (this constraint was non existent in previous formulations). This fix helped in a few cases. But in many cases after adding this constraint our binary programming solver was unable to return us any feasible solution. We think this is because: the constraints (4), (5) add at most N constraints each, they constrain row sum and column sum. But the (7) adds almost $N^2/2$ constraints. In general more the constraints the difficult it is to find a feasible solution.

Moreover, (7) does not handle sub-cycles which have more than 2 elements. $S_1 -> S_4 -> S_3 -> S_1$ is another sub-cycle in a slideshow of 7 slides which will not be handled by our constraints. We tried doing many different things but were not able to solve this in the given time frame. In-spite of all of these we feel that a recall of around 70% and a decent slideshow, given the speed at which they are computed for even larger values of N.

8. Qualitative results

To verify our approach in real world cases, and to observe what kind of slideshow we get, we decided to use the photos in our phone gallery. Firstly, we choose 30 photos randomly in our phone gallery. Secondly, We manually gave each photo 5 tags that is most related to this photo. Next, we used **bruteforce-search approach** to exhaustively iterate all the permutations in 30 photos to find what is the most interesting slideshows in only 5 photos.

This is what Google gallery slideshow shows us. They choose few photos in thousands of photos in our phone gallery and make these photos in slideshow. If this slideshow is interesting enough, we may want to buy it's paid version. We can see fig(6), the adjacent photos have something in common also something different between each other, making is as an interesting slideshow.

More slideshows with different interest factor can be found in the appendix.

9. Conclusion

In this report we examined and explained the problem of photo slideshow optimization. We proposed 3 different approaches to solve the problem. First 2 approaches were based on search (brute force search, search-with-pruning) and another approach uses binary programming. Although the first 2 approaches gave us good results, it was not feasible computationally to create a slideshow of large number of slides. The binary programming approach was much faster, but in many cases ended up giving sub-optimal solutions and it needs some improvement. We demonstrated the ability to generate good slideshows on images in our phone gallery like Google Photos. Going forward, with the feedback from Dr. Guoqiang Yu, we would like to look into linear programming formulation of travelling sales man problem. We would like to explore if we can take some inspiration from that formulation and adapt it for solving our problem.

10. References

1. Kaggle, Hash Code Archive - Photo Slideshow Optimization, 2019, https://www.kaggle.com/c/hashcode-photo-slideshow/overview
2. Wikipedia, Backtracking, https://en.wikipedia.org/wiki/Backtracking
3. Dynamic Programming, http://web.mit.edu/15.053/www/AMP-Chapter-11.pdf

11. Appendix

11.1. Original Formulation

We kind of have many constraints in our binary programming problem which might create some issues. Can we rewrite this optimizations problem taking advantage of properties of these matrices X, C?

We think we definitely can. If $x_{ij} = 1$ then we do not need another indicator variable to tell us x_{ji}. If i is a neighbor to j then it is given that j is a neighbor to i and unnecessary computation to solve for it can be avoided. Interest factor matrix- C which is symmetric about the diagonal can be rewritten as a lower-triangular matrix.

$$C = \begin{bmatrix} 0 & 0 & 0 & ... & 0 \\ c_{21} & 0 & 0 & ... & 0 \\ .... & & & & \\ c_{N1} & c_{N2} & c_{N3} & ... & 0 \end{bmatrix}$$

$$X = \begin{bmatrix} 0 & 0 & 0 & ... & 0 \\ x_{21} & 0 & 0 & ... & 0 \\ .... & & & & \\ x_{N1} & x_{N2} & x_{N3} & ... & 0 \end{bmatrix}$$

We also constrain our X matrix in a similar fashion, lower triangular matrix, which gets rid of symmetric constraint on X. Now our **optimization objective does not change from before**, but we can change some of the constraints.

Constraints:

$$x_{ij} = \{0, 1\} \; \forall i, j \in 1..N$$

$$\sum_{i}^{N} x_{ij} = 1 \; \forall j \in 1..N$$

$$\sum_{j}^{N} x_{ij} = 1 \; \forall i \in 1..N$$

$x_{ij} = 0$ if ij in upper triangle or diagonal

11.2. Problems with this formulation

In this formulation looking at the X matrix, we are unknowingly fixing the first and last elements of our arrangement. To demonstrate this let us consider a problem in which we have to create a slideshow of 5 images. X and C would be (5x5). X = $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ x_{21} & 0 & 0 & 0 & 0 \\ x_{31} & x_{32} & 0 & 0 & 0 \\ x_{41} & x_{42} & x_{43} & 0 & 0 \\ x_{51} & x_{52} & x_{53} & x_{54} & 0 \end{bmatrix}$

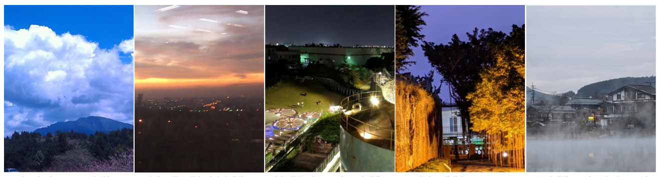
day, cloud, tree, mountain, blue
dusk, yellow, light, dark, building
night, light, playground, tree, building
dusk, wall, light, tree, grass
water, building, spring, cloud, mountain

Figure 7. Slideshow with interest factor 6

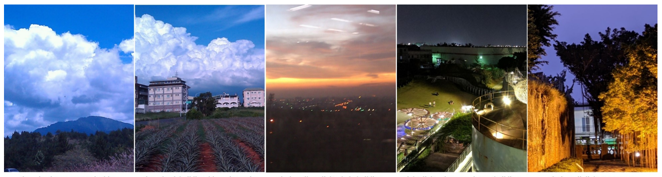
day, cloud, tree, mountain, blue
day, cloud, building, blue, pineapple
dusk, yellow, light, dark, building
night, light, playground, tree, building
dusk, wall, light, tree, grass

Figure 8. Slideshow with interest factor 7

This formulation with the constraints in place is not capable of expressing a sequence like $S_1- > S_5- > S_4- > S_2- > S_3$. A connection between S_1, S_5 can be expressed as $x_{51} = 1$, now S_5, S_4 cannot be expressed because if $x_{54} = 1$ it violates our row sum constraint. Before starting the optimization itself in this formulation we are fixing the first and last elements of the sequence which is not a good thing to do.

11.3. Few more Qualitative results

These are presented above.