IMPERIAL COLLEGE LONDON, DEPARTMENT OF MATHEMATICS

Scientific Computing (M3SC)

Peter J. Schmid

March 9, 2017

1 CLASS PROJECT 2: PARALLEL TASK SCHEDULING

You are the shop-floor manager of a large company and in charge of scheduling a number of tasks for multiple production lines. These tasks have given durations and dependencies, e.g., tasks 4 and 12 depend on the output of task 1, and thus in the workflow task 1 has to come before tasks 4, and 12. Tasks that do not have a dependency on other tasks can be executed in parallel, to save time. For the list of tasks in table 1.1 with specified durations and dependencies design a workflow and job schedule that minimizes the duration of the entire process while maximizing the number of processes that can be executed in parallel.

1.1 ALGORITHMIC STEPS

The following steps should get you to the optimal solution. We will be using a directed graph to represent the work flow.

1. For each job, introduce a "start" node and a "finish" node connected by a directed edge with a weight that accounts for the duration of the job.

2. Form a directed graph connecting the "start" and "finish" nodes while respecting the dependencies in the above list (table 1.1). These new connections (from the "finish" node of one job to the "start" node of the next) have an edge weight of zero.

3. Introduce a **virtual** start node that connects to *all* jobs' "start" nodes. These connections have an edge weight of zero.

job	duration	has to be completed before
0	41	1,7,10
1	51	4,12
2	50	3
3	36	
4	38	
5	45	7
6	21	5,9
7	32	
8	32	
9	49	11
10	30	12
11	19	
12	26	

Table 1.1: List of jobs, their duration and dependencies.

4. Introduce a **virtual** finish node that connects from *all* jobs' "finish" nodes. These connections have an edge weight of zero.

5. Determine the *longest* path through this graph from the virtual start node to the virtual finish node; this should give you the longest string of jobs.

6. From this longest path, iteratively determine the remaining (shorter) job sequences, until you have determined the start and finish times of all jobs.

Describe your code carefully and in sufficient detail. Produce a list of start and stop times for all jobs in the workflow (for example, add two more columns to the above table with the optimal start time and the finish time for each job) and visualize the workflow in form of a Gantt chart to display which tasks can be performed in parallel.

2 Class Project 2 (Mastery Option): Smart Image Resizing

Resizing images is an important task that has to be performed frequently on websites, smart phones and other mobile devices. Various options exist. First, the image can simply be scaled to match the dimensions of the screen; in this case, we do not lose image content, but for high aspect-ratio pictures we have to scale down substantially. Second, the image can be cropped; in this case, we lose image content, which is often not acceptable. In this exercise, we explore a third option: image resizing, where we successively eliminate image seams of "low importance" until we arrive at the desired size.

We will apply this technique for reducing the horizontal size of an image. In this case, we need to eliminate "unimportant" vertical seams. A vertical seam is defined as a path from

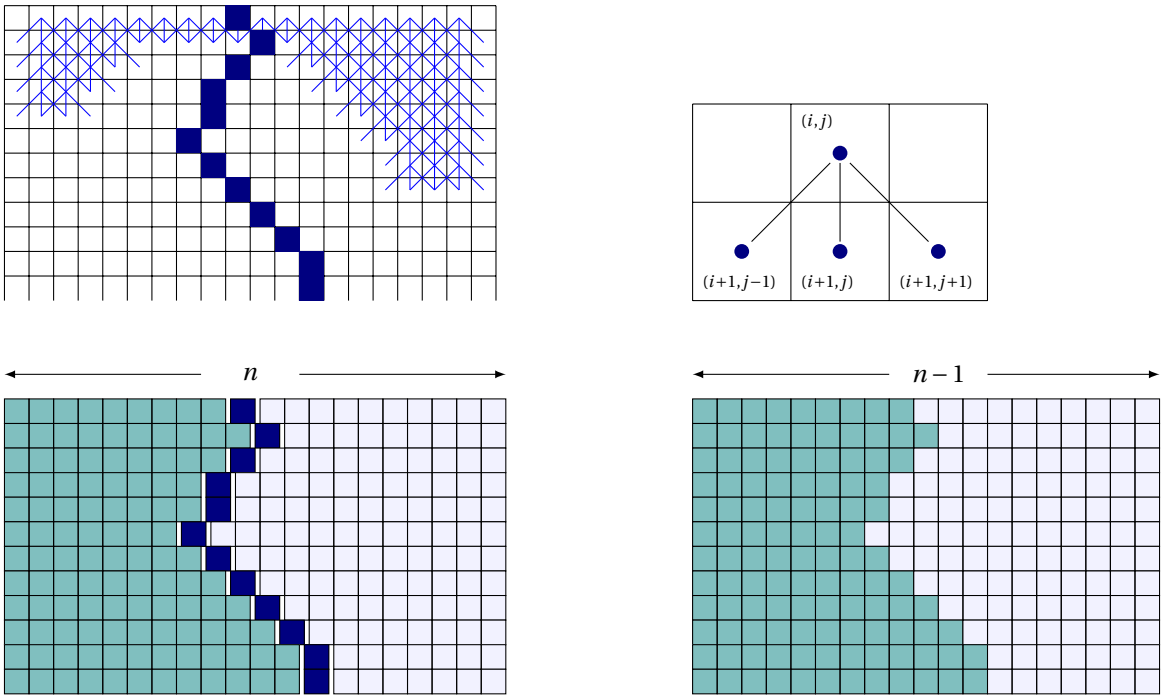

Figure 2.1: Smart image resizing. (upper left) vertical seam (in dark blue) and dependence of the minimal-importance path on the cumulative energy; (upper right) close-up of the dependence of the minimal-importance path on the cumulative energy in a lower row; (lower left) identification of the least-energetic vertical seam; (lower right) elimination of the least-energetic vertical seam and merging of the two image-halves.

the top of the image to the bottom, with one pixel per row and a horizontal shift of $-1, 0, +1$ pixel as we move from row to row (see figure 2.1 for a sketch).

For reducing the size of the picture given by the pixel matrix $I_{1:N,1:M}$, we compute an "importance map" of the image and then identify vertical seams with lowest importance which are then eliminated. The "importance map" is defined as follows. We first determine the complexity measure $C_{i,j}$ of the image by computing

$$C_{i,j} = \left|I_{i-1,j} - I_{i,j}\right| + \left|I_{i+1,j} - I_{i,j}\right| + \left|I_{i,j-1} - I_{i,j}\right| + \left|I_{i,j+1} - I_{i,j}\right|, \tag{2.1}$$

i.e., the absolute sum of the gray-scale gradients. The variable $I_{i,j}$ denotes the gray-scale of the image at position (i, j). On the edges of the image, the pixel values are extrapolated by copying the boundary values. For example, the left-edge extension is simply the left-most column of pixel values, $I_{:,0} = I_{:,1}$, and similar for the other edges. To find the minimal-importance seam, we first convert the complexity measure $C_{i,j}$ into a cumulative energy map $E_{i,j}$ using

$$E_{i+1,j} = C_{i+1,j} + \min\left\{ E_{i,j-1},\, E_{i,j},\, E_{i,j+1} \right\}. \tag{2.2}$$

Figure 2.2: Image to be reduced in width.

where we start in the top row with $E_{1,:} = C_{1,:}$. The minimal-importance path $P_{(i,j)\to(i+1,k)}$ from pixel (i, j) to the next pixel in row $(i + 1)$ is then given as

$$P_{(i,j)\to(i+1,k)} \qquad \text{with} \qquad k = \text{argmin}_{m\in\{j-1,j,j+1\}} \left\{ E_{i+1,m} \right\}, \tag{2.3}$$

i.e., we move to the pixel left below, straight below or right below, depending on the minimal value of E in these three locations. We start the seam at a minimal value of $E_{1,:}$ (note: there will be multiple minima) and traverse from the top to the bottom of the image. Once the minimal-importance vertical seam is determined, we eliminate it from the image (see figure 2.1). We continue the above procedure until enough seams are removed and the image has the desired width.

Implement the above procedure and apply it to the image '*penguins.png*' (with an original size of 600×1000 pixels, see figure 2.2). Convert the image to black-and-white and reduce the BW-image to 600×700 pixels. If necessary, add user-defined masks (patches of high values in E) to protect important parts of the image.

In your report, display (i) the original image, (ii) the complexity measure $C_{i,j}$ as an image and (iii) the reduced image.

3 General Remarks

For your report, use the LaTeX-template (which you already used for the first class-project). Describe your approach and crucial parts of your code in detail and with sufficient explanations. Interpret your findings and, if possible, present your results graphically.

Submit a pdf-file of your report and your `python`-code (including all dependencies) in form of a single zip-file to blackboard, before the submission deadline (**17:00 on 23. March 2017**).