National University of Singapore

School of Computing

CS4234 - Optimisation Algorithms

(Semester 1: AY2016/17)

Thursday, 24 November 2016

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections.
   It comprises TWELVE (12) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** or **on free page 12** in this booklet.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard *) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g. run Push-Relabel on G', etc.
7. Write your Student Number in the box below:

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	44		
B	56		
Total	100		

A (P?-)easy Questions (44 marks)

A.1 Push-Relabel (10 marks)

Please run Push-Relabel algorithm on the flow graph in Figure 1 with the following strategy: "At each step, choose the vertex with excess at maximum height and if ties, the vertex with the smallest vertex number.", "if we can push excess flow of a vertex u through more than one edge, we push that excess flow from u to neighboring vertex v that has the smallest vertex number", and "whenever we do relabel of u, we set the height of u, i.e. $h(u)$, to be 1+min$(h(v) : (u, v)$ in the residual graph)".

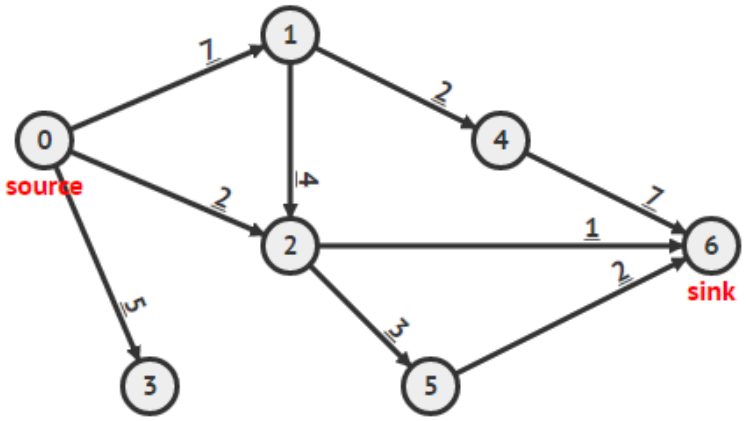

Figure 1: Perform Push-Relabel algorithm until you perform the **first** non-saturating push.

Now to check that you really use Push-Relabel algorithm instead of another Max Flow algorithm, please **stop** the algorithm after you perform the **first** non-saturating push, draw the **residual graph** (draw back edges as curvy lines), and write down the **label** (the height) + the **excess** of each vertex at this point of time in empty Figure 2 below (up to 7 marks only if the next sub-question is correct).

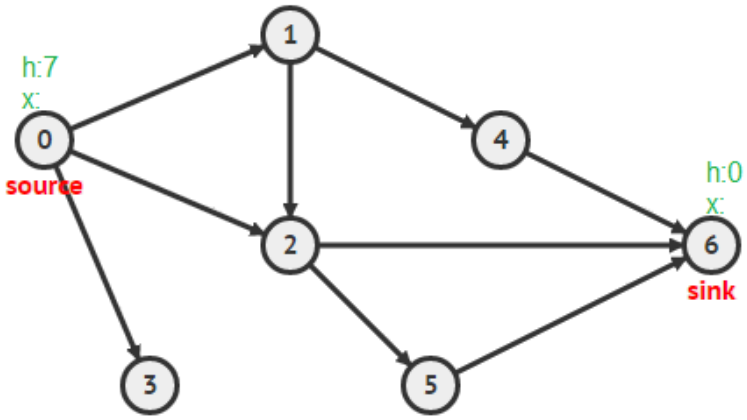

Figure 2: Write down your answer here.

To facilitate quicker grading, please mention the *unique* edge where the **first** non-saturating push happens (and you stop the Push-Relabel afterwards) (3 marks):

A.2 Manual Graph Matching (10 marks)

Solve the Max-Cardinality-Matching (MCM) problem on the unweighted undirected graph G with $V = 20$ vertices, $E = 21$ edges, and several connected components shown in Figure 3. Please highlight the edges that form the MCM by circling those edges directly in Figure 3 (up to 5 marks only if the next sub-question is correct).

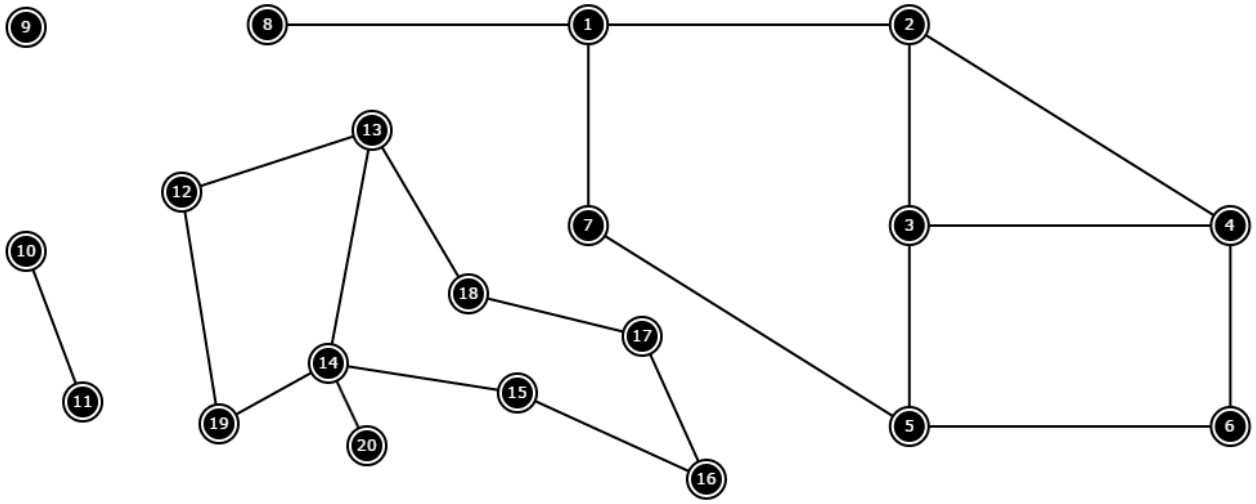

Figure 3: The input graph G; please circle the edges that form the MCM

To facilitate quicker grading, please write down the size (cardinality) of the MCM (i.e. the number of edges that you circled in Figure 3) (3 marks):

Additionally, is the MCM above a perfect matching? If yes, list down the $V/2$ pairs of matched edges below. If no, **add just one edge to** G so that we are able to find perfect matching on G (2 marks).

A.3 Statements About SLS (24 marks)

For each statement below about Stochastic Local Search (SLS) algorithm, determine if it is True/False/It depends and give a short explanation (3 marks per question).

1. We can run an SLS algorithm (the first 'S' = Stochastic) for an NP-hard Combinatorial Optimization Problem (COP) instance for an **extremely long time**, e.g. $\approx \infty$, and still unable to prove that the best found solution of that run is the Global Optima (GO) for that COP instance.

2. **All** SLS algorithms, if run for **extremely long time**, e.g. $\approx \infty$, will **always** encounter a GO of a COP instance during its long search run although it cannot stop immediately after encountering such GO (see the previous statement).

3. SLS algorithms that use larger neighborhood is **always** better than SLS algorithms that use smaller neighborhood.

4. It may be possible to provide an approximation ratio for an SLS algorithm even when we only run the SLS algorithm for a finite amount of time. Current Computer Scientists are just not yet able to prove the approximation ratio of an SLS algorithm yet.

5. We can easily take a state-of-the-art SLS algorithm S for another (NP-)hard COP C and use it directly for our new COP D.

6. Hybrid SLS algorithms (that combines two, or more, simpler SLS algorithms) is **always better** than its individual SLS algorithm working individually on its own.

7. If we have a COP whereby the typical fitness landscapes of its instances are of 'Big Valley' type with high Fitness-Distance Correlation (FDC) coefficient, it is much better to focus our SLS algorithm on intensification strategies than diversification strategies.

8. Tabu Search (TS) algorithm is a better SLS algorithm than Simulated Annealing (SA).

Section A Marks = ___ + ___ + ___ = ___

B (NP?-)hard Questions (56 Marks)

B.1 Updating Max-Flow Value (16 Marks)

Suppose that you **have just run** your best implementation of the $O(V^3)$ Push-Relabel Max Flow algorithm *for more than 2 hours* to compute the s-t MAX-FLOW value of a flow graph $G = (V, E)$ (with $V = 10\,000$). Now you are left with the residual graph R that has no more s-t flow anymore and also the valid assignments of flow f of the Max Flow on every edge in E. You are happy, but...

B.1.1 Misread Edge Capacity — Version 1 (6 Marks)

However, to your horror, you realize that you misread the original capacity of **an edge** by 1 unit, i.e. edge (u, v)'s capacity is not e but actually $e + 1$. Design a patching solution instead of re-running the $O(V^3)$ Push-Relabel Max Flow algorithm on the updated flow graph G (re-running the Push-Relabel algorithm will not work as this final assessment will be over by then) (4 marks)!

What is the time complexity of your patching solution? (2 marks)

B.1.2 *Misread Edge Capacity — Version 2 (10 Marks)

Same as above, but with one *small* change: edge (u, v)'s capacity is not e but actually $e - 1$ (notice the sign). What is your patching solution for this case (8 marks)?

What is the time complexity of your patching solution? (2 marks)

B.2 Early Termination of Push-Relabel Algorithm (10 Marks)

Someone suggests that we can optimize the performance of Push-Relabel algorithm for MAX-FLOW problem by *not* processing vertices that still have excess (no more push or relabel operation on those vertices) when their heights are $\geq n$ *if we only need the* s-t MAX-FLOW *value* of the flow graph.

B.2.1 It is actually correct (6 marks)

Show that this idea is actually correct by explaining succinctly on what will happen to the excess flow in those non-processed vertices (i.e. vertices with heights $\geq n$) if we run Push-Relabel algorithm as per normal (i.e. until all vertices have no more excess flow)? (6 marks).

B.2.2 Getting the s-t Min-Cut solution (4 marks)

Now if we use that early termination technique mentioned above, show how to find the set of edges that constitute the s-t MIN-CUT of that flow graph? (3 marks)

What is the time complexity of that MIN-CUT finding algorithm when the early termination of Push-Relabel algorithm is used? (1 mark)

B.3 Mr. Kwan (20 Marks)

Mr. Kwan is a postman who wants to deliver a bag of mails to houses in an unnamed Chinese city. The houses in that city are located along the streets and no house is located at a junction. Streets in that city are one-way and Kwan is a very law-abiding citizen and will never walk that one-way street in the opposite direction. Mr. Kwan wants to design a route whereby he starts from a starting junction 0 (his favorite landmark in the city), go through *every* one-way street in that city *at least once*, and returns to that starting junction 0 again to deliver the mails. Kwan knows that his government has designed the city well enough so that there will always be at least one such route. Kwan just want to know what is *the minimum number of streets* that he has to traverse to accomplish his objective. There are V junctions and E streets in that Chinese city (graph G).

B.3.1 Complete Search (3 marks)

To kick start your understanding of this Combinatorial Optimization Problem (COP), please do a complete search on the small instance of this problem shown in Figure 4.

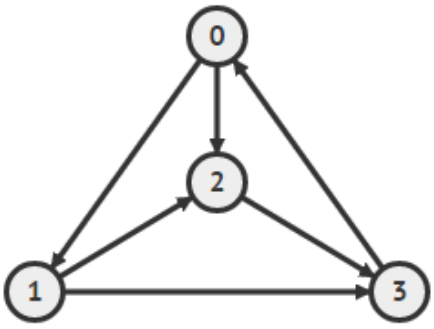

Figure 4: Small instance of this problem.

The minimum number of streets that Mr. Kwan has to traverse to accomplish his objective is optimal route taken by Mr. Kwan is _____ streets/edges (3 marks).
Just to ensure that you do not put random answer, please write down the optimal route here:
$0 \rightarrow$ _____ $\rightarrow 0$.

B.3.2 Complexity Class (2 marks)

Is this COP NP-hard? (circle one)

YES	NO

B.3.3 Special Case (5 marks)

Suppose that the city has one more additional property, i.e. the number of streets that goes into a junction is *always equal* to the number of streets that goes out from that junction, i.e. there is a "flow conservation" property. For example, imagine that there is no junction 2 (and all edges associated with it) in Figure 4 above and all the other 3 junctions $\{0, 1, 2\}$ will each have 1 street that goes in/out of those junctions so Mr Kwan's route will be something like $0 \rightarrow 1 \rightarrow 3 \rightarrow 0$.

Does this additional property simplify the COP?
If yes, design a polynomial time algorithm to solve this problem (3 marks).

What is the time complexity of your polynomial time algorithm? (2 marks)

B.3.4 *General Case (10 marks)

Now solve the general case of this COP using *any algorithm that you have learned in class* and analyze the time complexity of your solution! If you need more writing space for this (NP?-)hard question, please use the last empty page 12 of this paper.

B.4 Delta Evaluation for QAP (10 Marks)

The delta evaluation technique has been shown to be very useful for achieving good-performing SLS algorithms. The idea is generic enough: To compute the objective value of a neighboring candidate solution, one does not need to recompute from scratch by definition (usually with higher time complexity) but rather compute the delta changes instead (usually with much lower time complexity) as Local Search, as the name implies, only does local changes to the candidate solution.

In class, we have discussed $O(1)$ delta evaluation of 2-opt swap edges local move for TRAVELLING-SALESMAN-PROBLEM (current objective value - 2 deleted edges + 2 added edges) compared to $O(n)$ computation from scratch. You have also been exposed to $O(n)$ delta evaluation of 1-bit flip local move for LOW-AUTOCORRELATION-BINARY-SEQUENCE Problem (by analyzing the effect of that bit flip to values of $C(k)$) compared to $O(n^2)$ computation from scratch.

In class, we have also briefly discussed another NP-hard COP: QUADRATIC-ASSIGNMENT-PROBLEM (QAP). The formal problem description of QAP is as follows: Given two $n \times n$ matrices $A = (a_{ij})$ (typically contains flow information between facilities) and $B = (b_{ij})$ (typically contains distance information between facilities), find an assignment (a permutation) s of $\{0, 1, 2, ..., n\text{-}1\}$ over all possible permutations in the search space which minimizes the objective function $g(s) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{s_i s_j} \times b_{ij}$. As you can see, computation from scratch is clearly $O(n^2)$. Now, please design a faster-than-$O(n^2)$ delta evaluation for swap 2 locations local move for QAP analyze its time complexity!

B.4.1 Manual Computation (6 Marks)

Please use this example to help you design a fast delta evaluation.
You are given two 4×4 matrices A (left) and B (right) below:

0	8	3	2
0	0	0	1
0	2	0	0
2	0	0	0

0	1	8	9
1	0	2	1
8	2	0	5
9	1	5	0

Now, assuming that $s = \{0, 1, 2, 3\}$, compute $g(s)$ (2 marks):

Then, swap the first two facilities, i.e. we now have $s' = \{1, 0, 2, 3\}$ now and compute $g(s')$ (2 marks):

What if we swap the middle two facilities, i.e. we have $s'' = \{0, 2, 1, 3\}$; compute $g(s'')$ (2 marks):

B.4.2 *The Actual Delta Evaluation (4 Marks)

Now based on the manual computation that you have performed earlier (you can do more), design (any) delta evaluation for swap 2 facilities local move for QAP that is faster-than-$O(n^2)$ (4 marks)!

Section B Marks = ___ + ___ + ___ + ___ = ___

– End of this Paper, All the Best, You can use this Page 12 for extra writing space –