CPSC 320 Notes, Reductions & Resident Matching

A group of residents each needs a residency in some hospital. A group of hospitals each need some number (one or more) of residents, with some hospitals needing more and some fewer. Each group has preferences over which member of the other group they'd like to end up with. The total number of slots in hospitals is exactly equal to the total number of residents.

We want to fill the hospitals slots with residents in such a way that no resident and hospital that weren't matched up will collude to get around our suggestion (and give the resident a position at that hospital instead).

1 Trivial and Small Instances

1. Write down all the **trivial** instances of RHP. We think of an instance as "trivial" roughly if its solution requires no real reasoning about the problem.

   0 hospitals, 1 hospital

2. Write down two **small** instances of RHP. Here's your first:

   $r_1 : h_1 \ h_2$ $\qquad$ $h_1 : r_1 \ r_2$

   $r_2 : h_2 \ h_1$ $\qquad$ $h_2 : r_1 \ r_2$

   each hospital has 1 slot

   And here is your second. Try to explore something a bit different with this one.

   $r_1 : h_1 \ h_2$ $\qquad$ $h_1 : r_2 \ r_1 \ r_3$ [1 slot]

   $r_2 : h_2 \ h_1$ $\qquad$ $h_2 : r_1 \ r_2 \ r_3$ [2 slots]

   $r_3 : h_1 \ h_2$

3. Although we probably would not call it *trivial*, there's a special case where all hospitals have exactly one slot. What makes this an interesting special case?

2 Represent the Problem

1. What are the quantities that matter in this problem? Give them short, usable names.

n = #residents $R = \{r_1, \dots, r_n\}$ set of residents
m = #hospitals $H = \{h_1, \dots, h_m\}$ " " hospitals
P_R : preference lists of residents $s(h)$: #slots of hospital h
P_H : " " " "

2. Rewrite one of your small instances using these names.

3. Describe using your representational choices above what a valid instance looks like:

$s(h) \geq 1$ for all $h \in H$ $n \geq m$

$\sum_h s(h) = n$

3 Represent the Solution

1. What are the quantities that matter in the solution to the problem? Give them short, usable names.

pairings (h, r) between hospitals and residents.

valid solution: set of n (hospital, resident) pairings
hospital h is in $s(h)$ pairings
each resident r is in one pairing

2. Describe using these quantities makes a solution **valid** and **good**:

2. Describe using these quantities makes a solution **valid** and **good**:

a valid solution with no instabilities.

Clicker question #1

Complete the definition of an instability for RHP: *pair (h,r) is an instability if r is not assigned to h, r prefers h to their assigned hospital, and h prefers r to…*

A. Its assigned resident

B. Its least preferred assigned resident

C. All of its assigned residents

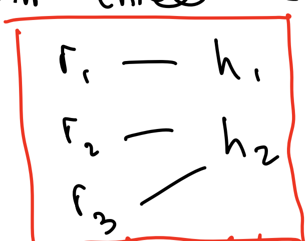
D. Not enough information to answer this question

3. Write out one or more solutions to one of your small instances using these names.

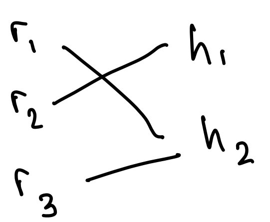
r_1 : h_1 h_2
r_2 : h_2 h_1
r_3 : h_1 h_2

h_1 : r_2 r_1 r_3 [1 slot]
h_2 : r_1 r_2 r_3 [2 slots]

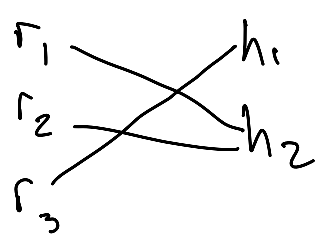
All three are valid solutions



Only good solution



Is (h_2, r_2) an instability? yes!



(h_1, r_1) is an instability

4 Similar Problems

Give at least one problem you've seen before that seems related in terms of its surface features ("story"), problem or solution structure, or representation to this one. You've probably already thought of it above!

SMP ("stable matching")

5 Brute Force?

We have a way to test if something that looks like a solution but may have an instability is stable. (From the "Represent the Solution" step.) That is, given a *valid* solution, we can check whether it's *good*.

1. Choose an appropriate variable to represent the size of an instance.

   n is #residents.
   m " #hospitals

   $n \geq m$. Choose $\underline{n}$

2. What can you say about the number of valid solutions, as a function of the instance size? Does it grow exponentially? Worse?

Recall: a valid solution is a set of n (hospital, resident) pairings, where n = #residents, where

- hospital h is in $s(h)$ pairings
- each resident r is in one pairing

Describe an instance of RHP of size n with as many valid solutions as possible.

→ An instance with n residents and hospitals has $n!$ valid solutions.

Describe an instance of RHP of size n with as few valid solutions as possible.

Clicker question #2

What is the **smallest possible number** of valid solutions for an instance of RHP with n residents?

A. $O(n!)$

B. $O\left(\frac{n!}{n}\right)$

C. $O(n)$

D. $O(1)$

There might be just one valid solution, when # hospitals = 1.

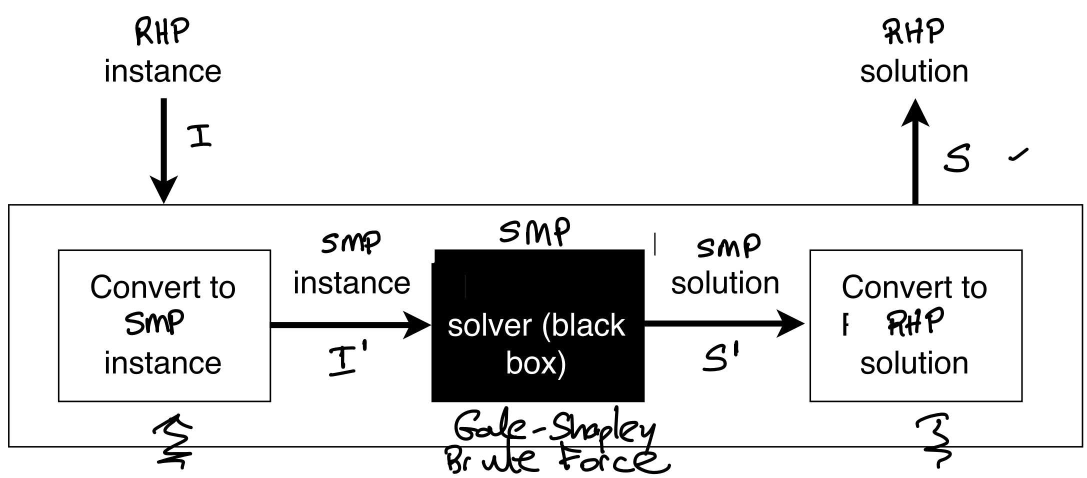
Since 1 is $O(n)$, $O(n!)$ $O(n!/n)$ actually all answers are valid.

4. Will brute force be sufficient for this problem for the domains we're interested in?

6 Promising Approach

We'll use a *reduction* for our promising approach. Informally, a reduction is simply a way of solving a new problem by leveraging an algorithm that solves an already familiar problem. Here we describe reductions somewhat formally, so you know what you are doing when proceeding informally. We need two **definitions**:

- An *instance* of a problem is simply a valid input, drawn from the space of possible inputs the problem allows. For example, the 4-element array $[5, 1, 4, 3]$ is an instance of the problem of sorting arrays of integers.
- A *reduction* from problem A to problem B provides a way to solve problem A by using an algorithm that solves B. There are two key parts to a reduction: (i) an algorithm that transforms any instance, say I, of problem A to an instance, say I', of B, and (ii) an algorithm that transforms a solution for I' back to a solution for I. (When coming up with a reduction, you don't need to design the algorithm that solves B; we think of that algorithm as a "black box" because the reduction does not depend on its details.) [1] Here's a diagram of how the parts fit together:



Your job in defining a reduction is to describe how the two white boxes work. Here we will reduce **from** RHP **to** some other problem B.

1. Choose a problem B to reduce to.

SMP

[1] Reductions can be defined more generally, where part (i) constructs many instances of B.

2. Reduction part (i) example: Transform a small instance of RHP into an instance of B.

RHP → SMP

$r_1 : h_1\ h_2$
$r_2 : h_2\ h_1$
$r_3 : h_1\ h_2$

$h_1 : r_2\ r_1\ r_3$ [1 slot]
$h_2 : r_1\ r_2\ r_3$ [2 slots]

$r_1 : h_1\ h_2^1\ h_2^2$
$r_2 : h_2^1\ h_2^2\ h_1$
$r_3 : h_1\ h_2^1\ h_2^2$

$h_1 : r_2\ r_1\ r_3$
$h_2^1 : r_1\ r_2\ r_3$
$h_2^2 : r_1\ r_2\ r_3$

3. Reduction part (ii) example: Transform a solution to your B instance into a solution to the RHP instance.

$(h_1, r_1), (h_2, r_2), (h_2, r_3)$ ← $(h_1, r_1), (h_2^1, r_2), (h_2^2, r_3)$

Clicker question #3

Let h_i be a hospital with 2 slots in RHP. In SMP, h_i should become:

A. An employer h_i

B. An employer h_i that appears twice in the employer list

C. 2 different employers h_i^1 and h_i^2

4. Generalize part (i): Design an algorithm to transform any instance I of RHP into an instance I' of SMP.

For each hospital h

- create $s(h)$ "clones" $h^1, h^2, \dots h^{s(h)}$, replacing h
- each as same ranking as h
- for each resident r, replace h in its ranking list with $h^1, h^2, \dots, h^{s(h)}$

5. Generalize part (ii): Design an algorithm to transform a solution S' for I' of B into a solution S for instance I of RHP.

Replace each pair (h^j, r) with (h, r)

Clicker question #4

In an RHP instance with m hospitals and n residents, each hospital has a preference list of length n and each resident has a preference list of length m. How long are the preference lists in the reduced SMP instance?

A. Length m for both

B. Length n for the employers and length m for the applicants

C. Length m for the employers and length n for the applicants

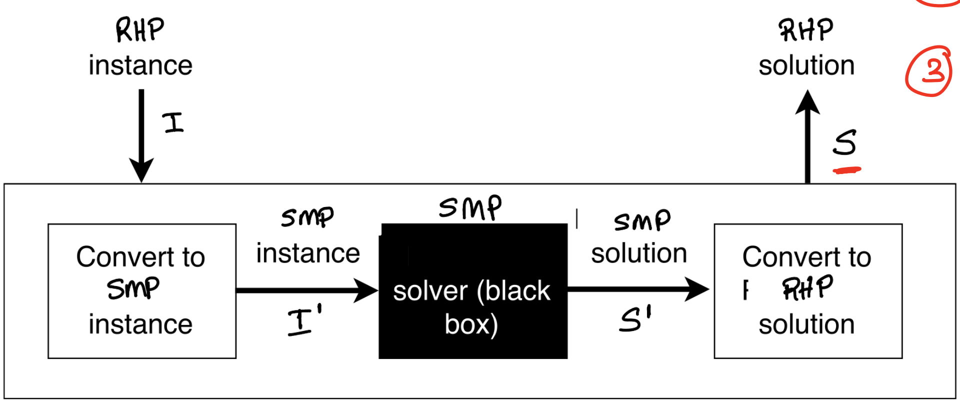
D. Length n for both

7 Proof of Correctness

Prove that your reduction produces a correct solution to the RHP instance. **Hint:** depending on your chosen reduction, you likely have a stable solution to an instance of B and need to prove that you get a correct (i.e., stable) solution to the RHP instance. You can either prove that *if B's solution is stable, RHP's solution is stable* or you can prove the contrapositive: *if RHP's solution is unstable, then B's must have been unstable as well.* (Another hint: proving the contrapositive is likely to be easier!)

2) Show S is a valid solution

3) Show that S is a good solution, i.e., has no instabilities.



1) Show I' is a valid instance of SMP

3) Suppose to contrary that S has an instability, say (h, r).
In solution S,
let h' be matched to r, let the least preferred resident matched with h be r'.

We want to show that S' has an instability.
By our conversion from S' to S, we know that in S',

- r is matched to some clone, say $(h')^j$, of h' and
- r' is matched to some clone, say h^i, of h.

We claim that (h^i, r) is an instablity of S'.
This is because

- h^i and r are not matched in S'
- r prefers h^i to its match $(h')^j$ in S' (this is because r prefers h to h' and by our conversion to instance I' it prefers any clone of h to any clone of h').
- h^i prefers r to its match r' (since h^i has the same preference ranking as h).