Heuristic Optimisation

Problem sheet 3

Brief solutions

1. Consider the following problem:

   There are given a finite number of points in the plane $P_1, \dots, P_n$, where P_i has coordinates x_i, y_i. The task is to find the distance between the pair of points P_j, P_k that are the closest to each other.

   (a) Discuss how exhaustive search could be applied to this problem.

   (b) Devise a divide and conquer algorithm to solve this problem.

Solution

(a) Consider any particular order for the points (for example the order in which they were given). Each point should be paired with all other points, but we know that $dist(P_i, P_j) = dist(P_j, P_i)$. The exhaustive search:

```
best:=dist(P1,P2)

for all points Pi, with i from 1 to n
   for all points Pj, with j from i+1 to n
        if dist(Pi,Pj) < best then
            best:= dist(Pi,Pj)
```

(b) The points can be ordered in increasing order of their x coordinates.

The **division** step:

The set of points can be divided in two subsets: the subset with the lower x coordinates $x_i \leq x_{border}$ – **P-low** and the subset with the higher x coordinates $x_i > x_{border}$ – **P-high**. x_{border} can be chosen such that the sizes of the two subsets be equal.

There are 3 cases for a solution:

- the solution pair is in **P-low**
- the solution pair is in **P-high**
- one point is in **P-low** and the other point in **P-high**

The algorithm can be applied to the two subsets. Consider d the best of the solutions to the two subproblems

The **combination** step:

We have to consider the third case for a solution, when the two points are in different subsets. It is sufficient to look at the pairs $P_i \in$ **P-low**, $P_j \in$ **P-high** with $x_{border} - x_i < d$ and $x_j - x_{border} < d$ respectively.

2. Consider the following Boolean satisfiability problem

$$\bigwedge_{k=1}^{n} \left[\left(x_{3k-2} \bigvee x_{3k-1} \right) \bigwedge x_{3k-1} \bigwedge \left(\overline{x}_{3k-2} \bigvee \overline{x}_{3k-1} \right) \bigwedge \left(\overline{x}_{3k-1} \bigvee x_{3k} \right) \right],$$

where $n \geq 2$ is an integer and we denote

$$\bigwedge_{k=1}^{n} H_k = H_1 \bigwedge H_2 \bigwedge \cdots \bigwedge H_n.$$

for any Boolean functions $H_1, H_2, \ldots, H_n$.

(a) What is the size of the search space? Justify your answer.

(b) Is it possible to use the divide and conquer method to solve this problem? Justify your answer.

(c) Solve the problem.

(d) Assume that we replace x_{3k} in the problem with x_{3k+1}, without making any other changes. Is it possible to use the divide and conquer method to solve this modified problem?

Solution

(a) There are $3n$ variables. Each variable can take two values, 0 or 1. Therefore, the size of the search space is 2^{3n}.

(b) Yes, we can use the divide an conquer method. This is justified as follows. The Boolean function can be written in the form

$$F = \bigwedge_{k=1}^{n} F_k,$$

where

$$F_k = \left(x_{3k-2} \bigvee x_{3k-1} \right) \bigwedge x_{3k-1} \bigwedge \left(\overline{x}_{3k-2} \bigvee \overline{x}_{3k-1} \right) \bigwedge \left(\overline{x}_{3k-1} \bigvee x_{3k} \right)$$

Since for every $k \neq l$ the Boolean functions F_k and F_l contain different variables, the problem F can be divided into n independent similar subproblems and the solutions of these subproblems can be combined to give a solution of the original problem.

(c) It is enough to solve only one of the F_k, because the other ones can be solved similarly. The solutions of F_k, $k = 1, \cdots, n$ can be combined into a solution of F. Let we solve

$$F_1 = \left(x_1 \bigvee x_2 \right) \bigwedge x_2 \bigwedge \left(\overline{x}_1 \bigvee \overline{x}_2 \right) \bigwedge \left(\overline{x}_2 \bigvee x_3 \right)$$

by using the GSAT algorithm. Denote

$$G_1 = x_1 \bigvee x_2,\ G_2 = x_2,\ G_3 = \overline{x}_1 \bigvee \overline{x}_2,\ G_4 = \overline{x}_2 \bigvee x_3.$$

START: $(x_1, x_2, x_3) = (0, 0, 0)$

	G_1	G_2	G_3	G_4	
flip	0	0	1	1	decrease
x_1	1	0	1	1	+1
x_2	1	1	1	0	+1
x_3	0	0	1	1	0

FLIP x_2

$(x_1, x_2, x_3) = (0, 1, 0)$

	G_1	G_2	G_3	G_4	
flip	1	1	1	0	decrease
x_1	1	1	0	0	-1
x_2	0	0	1	1	-1
x_3	1	1	1	1	1

FLIP x_3

SOLUTION OF F_1: $(x_1, x_2, x_3) = (0, 1, 1)$.

Alternative solution for $\mathbf{F_1}$: Since you are not asked how to solve the problem, a direct solution for F_1 is also acceptable. We must have $x_2 = 1$ because the second clause is x_2. With $x_2 = 1$ the third clause becomes $\overline{x}_1 \bigvee 0$, hence we must have $x_1 = 0$. With $x_2 = 1$ the fourth clause becomes $0 \bigvee x_3$, hence we must have $x_3 = 1$. Since $x_2 = 1$ the first clause is also TRUE. Hence, the solution is $(x_1, x_2, x_3) = (0, 1, 1)$.

Similarly, SOLUTION OF F_k: $(x_{3k-2}, x_{3k-1}, x_{3k}) = (0, 1, 1)$.

SOLUTION OF F: $x_{3k-2} = 0$, $x_{3k-1} = 1$, $x_{3k} = 1$, where $k \in \{1, \cdots, n\}$.

(d) Supposed we replaced x_{3k} with x_{3k+1}. For a fixed k, the clauses of F_k should be in the same group because each clause has a common variable with the next one. Moreover, the last clause of F_k and the first clause of F_{k+1} has in common the variable x_{3k+1}, hence they should belong to the same group. In conclusion, all clauses should belong to the same group and therefore it is not possible to use divide and conquer.

3. The 8-puzzle is a sliding puzzle that consists of a frame of numbered square tiles with one tile missing. The goal of the puzzle is to get from a **Start state** to a **Goal state** by making sliding moves that use the empty space.

   Show the search tree/graph expanded by best first search for the following 8-puzzle problem and using the "tiles in wrong position" heuristic:

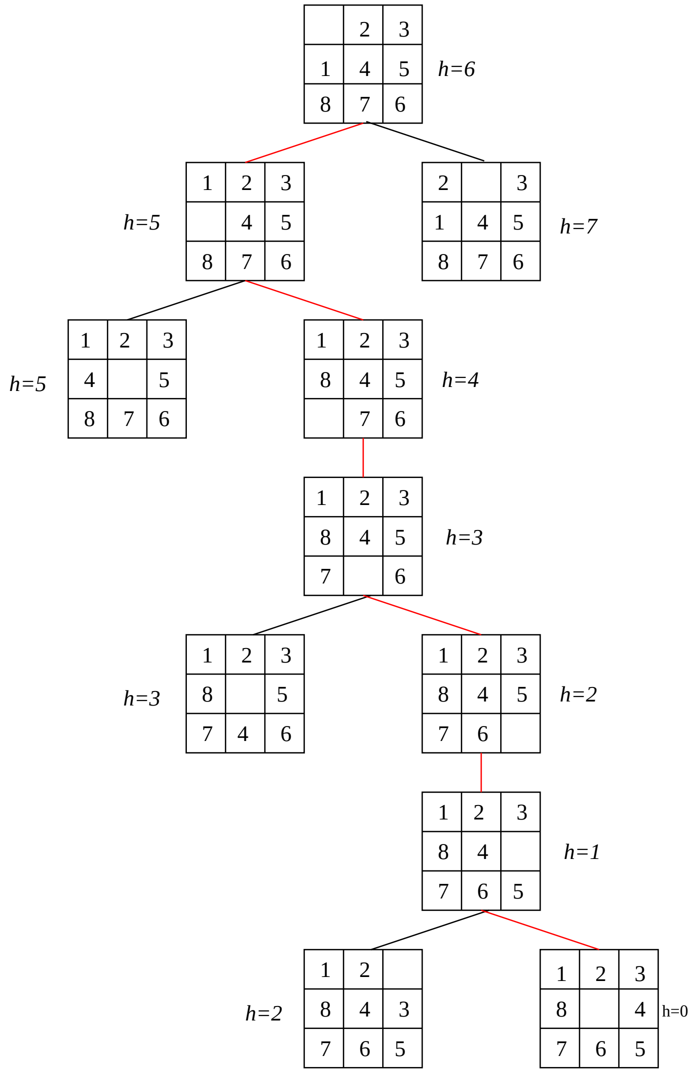
Start state

	2	3
1	4	5
8	7	6

Goal state

1	2	3
8		4
7	6	5

Solution



4. In some cases there is no good evaluation function for a problem, but there is a good comparison method, which tells whether one node is better than the other, without assigning numerical values to either. Explain whether it is possible to do best-first search based on this comparison, without a proper evaluation function. Is it possible to apply A* search using this comparison?

   Solution

   Best first search is possible: Whenever we have to choose which node from the open list to expand, we compare the possible nodes in pairs and select the one that is best.

   A* is problematic for the case when we have to choose between node 1 and node 2, $g(1) > g(2)$, node 1 better than node 2 (note that h is not given as a function, but by comparing the nodes in pairs, i.e., as guessing whether a node is better than the other).