Disclaimer: Following are few samples question. Do not depend on the following question for exam preparation. These are given for you to understand general question patterns.

Chapter 4:

Book Exercise: Q4.1, Q4.3, 4.13

Solution: Check Solution Manual

Question: Solve genetic algorithm for 1 generation for 8 queen problem where you need to show all the stages of genetic algorithm.

Note: Question may contain any other problem description in a similar setting.

Answer: Check class lecture

Question: You may need to solve a problem using hill climbing or simulated annealing.

Answer: Check lecture notes

Chapter 5:

Book Exercise: Q5.1, Q5.4, Q5.6, Q5.9, Q5.12, Q5.13

Solution: Check solution manual

Question: A game tree will be given and you need to perform alpha beta pruning

Answer: Check lecture notes

Chapter 6

Question: Q6.1, Q6.2, Q6.3, Q6.4 , Q6.7, Q6.8, Q6.9

Solution: Check solution manual

Question: Problem related to crypto arithmetic, graph node coloring.

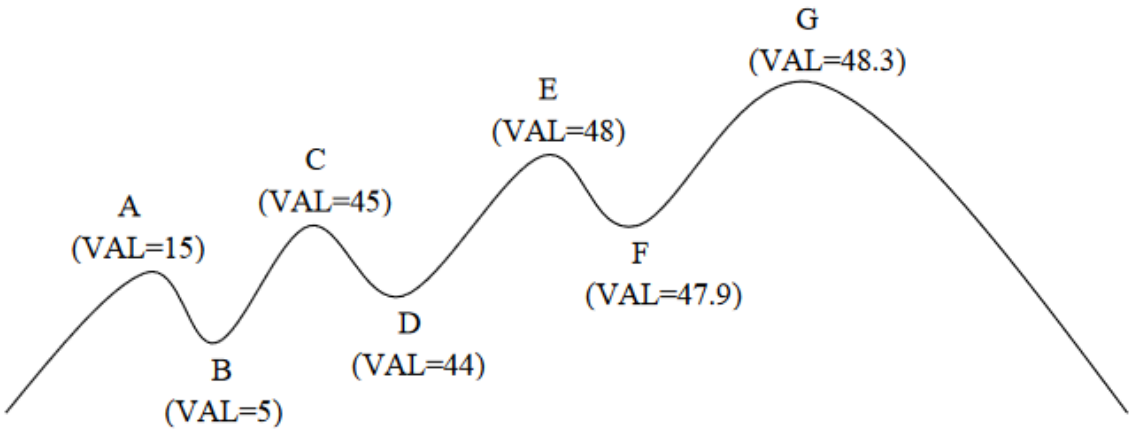
Answer: Check lecture note

Chapter 4

Simulated Annealing:

Problem: Perform Simulated Annealing search to maximize value in the following search space.

Recall that a good move (increases value) is always accepted (P = 1.0); a bad move (decreases value) is accepted with probability P = $e^{\Delta VAL/T}$, where ΔVAL = VAL(Next) − VAL(Current) and T is temperature.



Use this temperature schedule:

Time Step	*1–100*	*101–200*	*201–300*
Temperature (T)	*10*	*1.0*	*0.1*

This table of values of e may be useful:

x	*0.0*	*−1.0*	*−4.0*	*-4.3*	*−40.0*	*−43.0*
e^x	*1.0*	*≈0.37*	*≈0.018*	*≈0.014*	$\approx 4.0*10^{-18}$	$\approx 2.1*10^{-19}$

Solution:

Following are the possible moves in the search. The first one is

Time	*From*	*To*	*T*	*ΔVAL*	*ΔVAL/T*	*P*
57	*A*	*B*	*10*	*−10*	*−1*	*0.37*
78	*C*	*B*	10	−40	−4	≈0.018
132	*C*	*B*	1.0	−40	−40	$\approx 4.0*10^{-18}$
158	*C*	*D*	1.0	−1	−1	≈0.37
194	*E*	*D*	1.0	−4	−4	≈0.018
194	*E*	*B*	1.0	−43	−43	$\approx 2.1*10^{-19}$
238	*E*	*D*	0.1	−4	−40	$\approx 4.0*10^{-18}$
263	*E*	*F*	0.1	−0.1	−1	≈0.37
289	*G*	*F*	0.1	−0.4	−4	≈0.018
289	*G*	*D*	0.1	−4.3	−43	$\approx 2.1*10^{-19}$

6.6 Show how a single ternary constraint such as “A + B = C” can be turned into three binary constraints by using an auxiliary variable. You may assume finite domains. (Hint: Consider a new variable that takes on values that are pairs of other values, and consider constraints such as “X is the first element of the pair Y .”) Next, show how constraints with more than three variables can be treated similarly. Finally, show how unary constraints can be eliminated by altering the domains of variables. This completes the demonstration that any CSP can be transformed into a CSP with only binary constraints.

Solution:

6.6 The problem statement sets out the solution fairly completely. To express the ternary constraint on A, B and C that $A + B = C$, we first introduce a new variable, AB. If the domain of A and B is the set of numbers N, then the domain of AB is the set of pairs of numbers from N, i.e. $N \times N$. Now there are three binary constraints, one between A and AB saying that the value of A must be equal to the first element of the pair-value of AB; one between B and AB saying that the value of B must equal the second element of the value of AB; and finally one that says that the sum of the pair of numbers that is the value of AB must equal the value of C. All other ternary constraints can be handled similarly.

Now that we can reduce a ternary constraint into binary constraints, we can reduce a 4-ary constraint on variables A, B, C, D by first reducing A, B, C to binary constraints as shown above, then adding back D in a ternary constraint with AB and C, and then reducing this ternary constraint to binary by introducing CD.

By induction, we can reduce any n-ary constraint to an $(n-1)$-ary constraint. We can stop at binary, because any unary constraint can be dropped, simply by moving the effects of the constraint into the domain of the variable.

Problem:

6.4 Give precise formulations for each of the following as constraint satisfaction problems:

a. Rectilinear floor-planning: find non-overlapping places in a large rectangle for a number of smaller rectangles.

b. Class scheduling: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes that he or she can teach.

c. Hamiltonian tour: given a network of cities connected by roads, choose an order to visit all cities in a country without repeating any.

Solution:

6.4 **a.** For rectilinear floor-planning, one possibility is to have a variable for each of the small rectangles, with the value of each variable being a 4-tuple consisting of the x and y coordinates of the upper left and lower right corners of the place where the rectangle will be located. The domain of each variable is the set of 4-tuples that are the right size for the corresponding small rectangle and that fit within the large rectangle. Constraints say that no two rectangles can overlap; for example if the value of variable R_1 is $[0, 0, 5, 8]$, then no other variable can take on a value that overlaps with the $0, 0$ to $5, 8$ rectangle.

b. For class scheduling, one possibility is to have three variables for each class, one with times for values (e.g. MWF8:00, TuTh8:00, MWF9:00, ...), one with classrooms for values (e.g. Wheeler110, Evans330, ...) and one with instructors for values (e.g. Abelson, Bibel, Canny, ...). Constraints say that only one class can be in the same classroom at the same time, and an instructor can only teach one class at a time. There may be other constraints as well (e.g. an instructor should not have two consecutive classes).

c. For Hamiltonian tour, one possibility is to have one variable for each stop on the tour, with binary constraints requiring neighboring cities to be connected by roads, and an AllDiff constraint that all variables have a different value.

Problem:

6.2 Consider the problem of placing k knights on an n × n chessboard such that no two knights are attacking each other, where k is given and k ≤ n2.

a. Choose a CSP formulation. In your formulation, what are the variables?

b. What are the possible values of each variable?

c. What sets of variables are constrained, and how?

d. Now consider the problem of putting as many knights as possible on the board with-out any attacks. Explain how to solve this with local search by defining appropriate ACTIONS and RESULT functions and a sensible objective function.

Solution:

6.2

a. Solution A: There is a variable corresponding to each of the n^2 positions on the board.
Solution B: There is a variable corresponding to each knight.

b. Solution A: Each variable can take one of two values, {occupied,vacant}
Solution B: Each variable's domain is the set of squares.

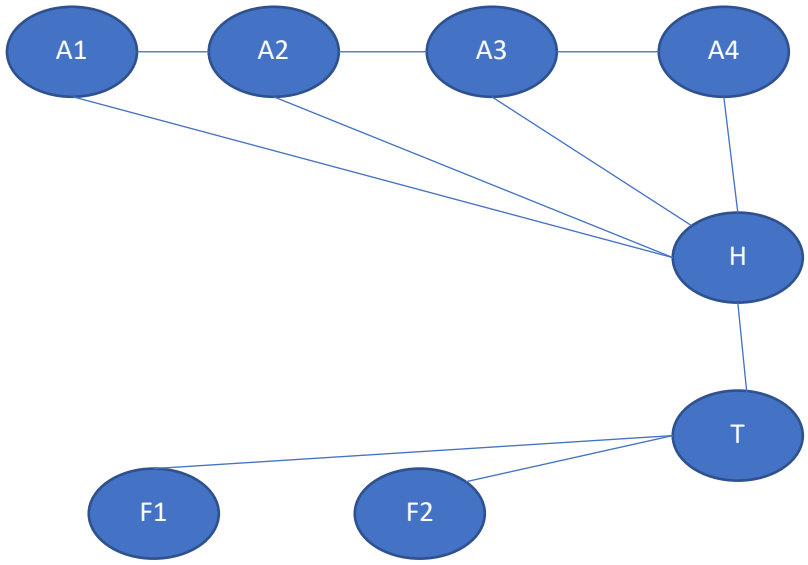
c. Solution A: every pair of squares separated by a knight's move is constrained, such that both cannot be occupied. Furthermore, the entire set of squares is constrained, such that the total number of occupied squares should be k.
Solution B: every pair of knights is constrained, such that no two knights can be on the same square or on squares separated by a knight's move. Solution B may be preferable because there is no global constraint, although Solution A has the smaller state space when k is large.

d. Any solution must describe a *complete-state* formulation because we are using a local search algorithm. For simulated annealing, the successor function must completely connect the space; for random-restart, the goal state must be reachable by hillclimbing from some initial state. Two basic classes of solutions are:
Solution C: ensure no attacks at any time. Actions are to remove any knight, add a knight in any unattacked square, or move a knight to any unattacked square.
Solution D: allow attacks but try to get rid of them. Actions are to remove any knight, add a knight in any square, or move a knight to any square.

Problem:

6.8 Consider the graph with 8 nodes A1, A2, A3, A4, H, T, F1, F2. Ai is connected to Ai+1 for all i, each Ai is connected to H, H is connected to T, and T is connected to each Fi. Find a 3-coloring of this graph by hand using the following strategy: intelligent backtracking, the variable order A1, H, A4, F1, A2, F2, A3, T, and the value order R, G, B.

Solution:



a) A1 = R.
b) H = R conflicts with A1.
c) H = G.
d) A4 = R.
e) F1 = R.
f) A2=R conflicts with A1, A2=G conflicts with H, so A2 = B
g) F2 = R
h) A3 = R conflicts with A4, A3 = G conflicts with H. A3 = B conflicts with A2. So back tack. Conflicts set is {A2, H, A4}, so jump to A2. Add {H, A4} to A2's conflict set.
i) A2 has no more values, so backtrack. Conflict set is {A1, H, A4} so jump back to A4. Add {A1,H} to A4's conflict set.
j) A4 = G conflicts with H so A4=B

k) F1=R
l) A2 = R conflict with A1, A2 = G conflicts with H, so A2 = B
m) F2=R
n) A3 = R
o) T = T conflicts with F1 and F2. T = G conflicts with G, so T = B
p) success