CSCI5240 – Combinational Search and Optimization with Constrains
Fall 2015
Assignment 3

1155070976 ZHANG Wanying (Sophie)

1.
The CSP model can be illustrate as following under these parameters: totally, there are **m movers, n items**, and we have two arrays **requiremovers[n]** and **requiretime[n]**, where requiremovers[i] is an integer of how many movers are needed of moving item[i], and requiretime[i] is an integer of how much time is needed for moving item[i]. The time of finishing all the jobs must within a **maxtime**. All these are given in data.in.

Variable:
All the following statement satisfy: for any integer i ∈[1..n], j ∈[1..m]

assignment[1..n][1..m]: assignment[i][j]means whether the mover[j] is assigned with moving item[i]

start[1..n]: start[i]means the begin time of moving item[i]
duration[1..n]: duration[i]means the duration time of moving item[i]
end[1..n]: end[i]means the end time of moving item[i]

fatigue[1..n][1..m]: fatigue[i][j]means if the mover[j] is assigned with moving item[i], fatigue[i][j] is the fatigue value of him due to his previous work.

MaxFatiItem[1..n]: MaxFatiItem[i] is max fatigue value of movers who moving item[i]

MaxItemAssignMover[1..m]: MaxItemAssignMover[j] is the max amount of items can be assigned to mover[j].

Domain:
D(assignment) = {0,1}
for any integer i ∈[1..n], j ∈[1..m]
assignment[i][j] = 0 means the mover[j] is NOT assigned with moving item[i]
assignment[i][j] = 1 means the mover[j] is assigned with moving item[i]

for any integer i ∈[1..n]
D(start[i]) = [0..maxtime-requretime[i]]
D(duration[i]) = [requretime[i] .. requretime[i]+n-1]
D(end[i]) = [maxtime-requretime[i] .. maxtime]

D(fatigue) = [0..n-1] the fatigue value of a mover is range from 0 to n-1, because a

mover at most moving n items
D(MaxFatiItem) = D(fatigue) = [0..n-1]

D(MaxItemAssignMover) = [0..n]: a mover at least moving 0 items and at most moving n items.

Constraint:

1. For i ∈[1..n], $\sum_{j\in[1,m]} assignment[i][j] = requiremovers[i]$

2. one worker cannot do two things at same time

```
for  i ∈[1..n], j ∈[1..m]
    if assignment[j][i]==1 && assignment[k][i]==1
        start[j] >= end[k] || start[k] >= end[j]
```

3. for i ∈[1..n], start[i] + duration[i] = end[i]

4. cumulative(m, start, duration, end, requiremovers)

5. if mover[i] assigned with item A and item B,

```
        if start[A] < start[B] then fatigue[A][i] < fatigue[B][i] and vice versa.
        It can be written as following,
    For i ∈[1..m] j∈[1..n-1] and k∈[j..n]
        if assignment[j][i]=1 && assignment[k][i]=1 && start[j] < start[k]
            fatigue[j][i]< fatigue[k][i]
        if assignment[j][i]=1 && assignment[k][i]=1 && start[j] > start[k]
            fatigue[j][i]> fatigue[k][i]
```

6. for j∈[1..m], $\sum_{i\in[1,n]} assignment[i][j] = MaxItemAssignMover[i]$

7. if the mover[j] not assigned with item[i], then fatigue[i][j]=0 , it can be written as:
   if(assignment[i][j]==0) fatigue[i][j]=0

8. The max-fatigue value of a mover is smaller than the amount of items he moved.

```
For i ∈[1..m] j∈[1..n-1] and k∈[j..n]
    fatigue[j][i] < MaxItemAssignMover[i]
```

2

Following table shows difference of runtime and fails among Model 1, Model 2 and Model 3

Run-time(sec)	L(2,7)	L(2,8)	L(3,9)
Model 1	0.011	0.023	0.012
Model 2	0.021	0.084	0.418
Model 3 Search 1	0.023	0.054	0.053
Model 3 Search 2	0.027	0.112	0.398
Model 3 Search 3	0.019	0.045	0.072

Failures	L(2,7)	L(2,8)	L(3,9)
Model 1	276	1230	926
Model 2	423	2144	6409
Model 3 Search 1	276	1214	923
Model 3 Search 2	423	2144	6409
Model 3 Search 3	276	1214	923

Comparing Model 1 and Model 2, we can found that both the run-time and failures of Model one is smaller than Model two. If we channel X and Y in Model 3, we can found that in all three search methods of Model 3, the runtime is smaller than Model 2 and larger than Model 1.

Comparing branching methods of Model 3, we can draw a simple conclusion, if we branch X only, the failures is the same as Model 1, if we branch Y only, the failures is the same as Model 2. And if we branch X and Y both, we can find the failures is the smaller one, which is the same as we only branch X in this particular problem.

Following table shows difference of runtime and fails among Model 3 and Model 4

Run-time(sec)	L(2,7)	L(2,8)	L(3,9)
Model 3 Search 1	0.023	0.054	0.053
Model 3 Search 2	0.027	0.112	0.398
Model 3 Search 3	0.019	0.045	0.072
Model 4 Search 1	0.014	0.029	0.041
Model 4 Search 2	0.017	0.045	0.272
Model 4 Search 3	0.014	0.033	0.049

Failures	L(2,7)	L(2,8)	L(3,9)
Model 3 Search 1	276	1214	923
Model 3 Search 2	423	2144	6409
Model 3 Search 3	276	1214	923
Model 4 Search 1	171	780	676
Model 4 Search 2	273	1317	4395
Model 4 Search 3	171	780	676

Comparing Model 3, and Model 4, by using the same search method, we can found that both the run-time and failures of Model 3 is larger than Model 4. It's intuitive, because in Model 4, we add symmetry breaking constraint to push the search space of model four is smaller than the search space of model three.

3.

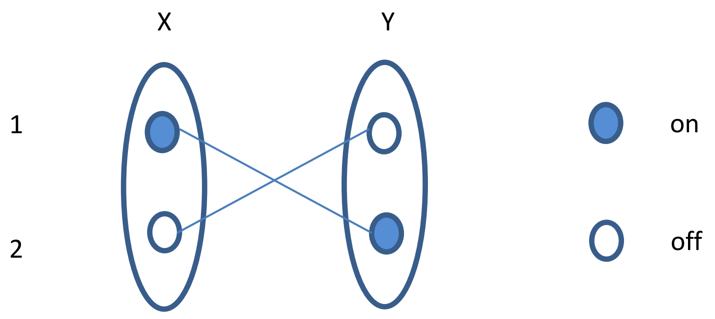
Variable: X, Y

Domain: D(X) = {1, 2}, D(Y) = {1, 2}

Constraint: X = Y

Start Genet Network:

State 1:



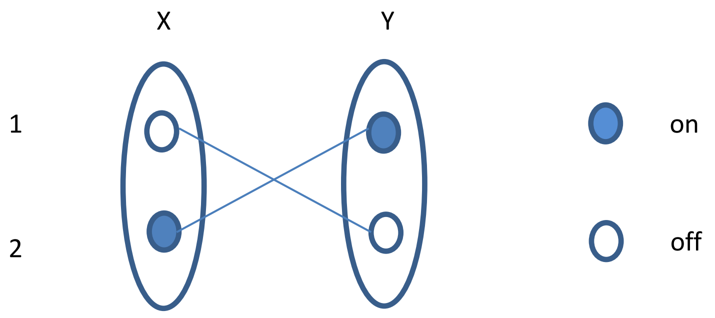
We can call the node X1 where the variable is X and the value is one which is the node on the left top. Also, we can define node X2, Y1, Y2.

At start, in state 1, we make X1 and Y2 on. So the input can be calculated as following:

$$\text{Input}(X1) = -1$$
$$\text{Input}(X2) = 0$$
$$\text{Input}(Y1) = 0$$
$$\text{Input}(Y2) = -1$$

Because the output of state 1 is changed, we can jump to State 2, where node X2 and Y1 is on.

State 2:

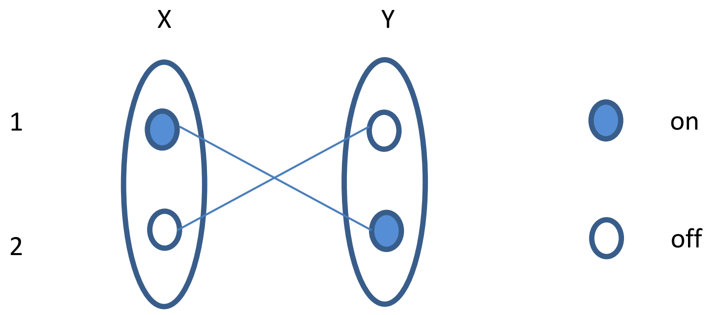


At state 2, input can be calculated as following:

$$\text{Input}(X1) = 0$$
$$\text{Input}(X2) = -1$$
$$\text{Input}(Y1) = -1$$
$$\text{Input}(Y2) = 0$$

Because the output of state 2 is changed, we can jump to State 3 , where node X1 and Y2 is on.

State 3:



At State 3, we can find that the state 3 is the same as state 1.
Thus, the genet network will repeat state 1, state 2, state 1, state 2....
That's the oscillation.

4.

element (I, [3, 3, 1, 3], X)
gcc ([X_1, X_2, X_3, X_4] , (1, 0, 1), (2, 0 ,1), (3, 1, 1), (4, 2, 2))
precede ([X_1, X_2, X_3, X_4], 3, 2)
nvalues ([X_1, X_2, X_3, X_4], 3)

5.

We can first illustrate an example of this question:

$$\text{Min } 10 - Y - Z \quad \text{subject to}$$
$$X = 3 - Y$$
$$W = 5 - Y$$
$$T = 4 + 2Y - 2Z$$
$$X \geq 0, Y \geq 0, Z \geq 0, T \geq 0, W \geq 0$$

(a) Explain why the picked equation x = b + cy + · · · must have c < 0 in line 3.

In this example, if we pick up a variable Y in objective function, and then we pick the equation T = 4 + 2Y – 2Z, where c = 2 > 0. After rewrite the equation into Y= 0.5T + Z -2, we can find that the equation is not in basic solved form, because the constant on the right hand side is -2, not a non-negative one.
Further, if we continue to use this equation Y= 0.5T + Z -2 to substitute Y in objective function, the objective function will become 12 – 0.5T – 2Z, it's useless, because this step did not decrease the amount of variables in objective function with negative coefficient. (The objective function originally have 2 variables with negative coefficient Y and Z, however, after substitution, it still have 2 variables with negative coefficient T and Z)

(b) Explain why the picked equation x = b + cy + · · · must have a minimal −b/c term among all equations with a negative c in line 3.

In this example, if we pick up a variable Y in the objective function and then we have two choices:

1. Choose X= 3 – Y (where –b/c is minimal)
2. Choose W = 5 –Y

If we choose W= 5 –Y, after rewrite the equation, Y = 5 – W, then we will encounter a problem when we try to use Y = 5 – W to substitute Y in X= 3 – Y. If we do such a substitution, X= 3 – Y will become X= -2 + W and this violate the requirements of Basic Solved Form which demands each constant on the right hand side of equations is non-negative. However, at beginning, if we choose X= 3 – Y (where –b/c is minimal), we will not destroy Basic Solved Form.

(c) Explain why a variable y of negative coefficient should be chosen from the objective in line 2.

Because it's a minimum problem, in each iteration, we want to eliminate a variable with negative coefficient by substituting that variable using an equation. For example, after substituting variable Y in objective function, we have a new objective function: min 13 + X – Z, where X is a parameter and X ≥ 0. And finally we want our objective function is something like min 8 + 2X + 0.5T, where X and T are parameters and X ≥ 0, T ≥ 0.

(d) Explain why when there is no longer a variable y with negative coefficient in the objective, the optimal is found.

In this example, finally, we get the objective function: min $8 + 2X + 0.5T$, where $X \geq 0$, $T \geq 0$. It's intuitively, min $8 + 2X + 0.5T = 8$ (where $X = 0$, $T = 0$). If there is still a variable with negative coefficient in objective function, e.g. min $13 + X - Z$, where Z is a variable and $Z \geq 0$ and we have $T = 4 + 2Y – 2Z$, we cannot get an optima from the objective function, because we don't know when Z=? , we can get the optima one.

(e) Explain why when there is a variable y with negative coefficient in the objective but no equations with a negative c, the optimization problem has no optima in line 10.

If finally, we get such an objective function min $8 + 2X + 0.5T – Q$, where $X \geq 0$, $T \geq 0$, $Q \geq 0$, there is no optimal solution, because there is not any constraint on Q, if $Q = \infty$, then the objective function can get $-\infty$. Thus, there is not any optimal solution under this condition.