Q1. [2+2+2 = 6 Marks] a. Consider the A\* algorithm. This algorithm will not pick a node immediately to expand even if this is in the fringe. Why is this so? Provide a scenario for this.

b. Given a constraint involving three or more variables, explain how will you convert them into constraints involving two variables. Also give an example.

c. Why do you prefer to choose the variable that is most constrained as a heuristic given many other variables. Explain.

[2+2+2 = 6 Marks]

<https://www.chegg.com/homework-help/questions-and-answers/answer-following-questions-consider-algorithm-algorithm-pick-node-immediately-expand-even--q53901259>

**Step 1/3**

(a)

The A\* algorithm is a type of best-first search algorithm that is used to find the shortest path between two points in a graph. It uses a heuristic function to estimate the cost of reaching the goal from a given node, and it expands the node with the lowest estimated cost first.

However, the A\* algorithm will not always pick a node immediately to expand, even if it is on the fringe (i.e., a candidate for expansion). This can occur if there is another node on the fringe with a lower estimated cost.

For example, consider the following scenario:

There is a graph with three nodes: A, B, and C. Node A is the starting node, and node C is the goal. There is a direct path from A to C with a cost of 10, and there is also a path from A to B with a cost of 5, and from B to C with a cost of 5. The heuristic function estimates that it will take an additional 5 units of cost to reach the goal from B.

In this scenario, the A\* algorithm will first expand node B because the estimated cost of reaching the goal from node B (10) is lower than the estimated cost of reaching the goal from node A (15). Once node B is expanded, the A\* algorithm will then expand node C because it has the lowest estimated cost of all the nodes in the fringe.

In summary, the A\* algorithm will not always pick a node immediately to expand, even if it is on the fringe, if there is another node with a lower estimated cost.

* [**Explanationfor step 1**](https://www.chegg.com/homework-help/questions-and-answers/answer-following-questions-consider-algorithm-algorithm-pick-node-immediately-expand-even--q53901259#answer-explanation-tabs-for-step-1_tabpanel_0)

This is why the algorithm will not pick a node immediately to expand even after its in the fringe.

**Step 2/3**

(b)

To convert a constraint involving three or more variables into a constraint involving two variables, we can use techniques such as elimination.

We can apply this technique only when there are two or more different constrains available.

Elimination involves using the constraints to eliminate one of the variables, resulting in a new constraint involving fewer variables. For example, consider the following system of constraints involving three variables (x, y, and z):

x + y + z <= 10 2x + y - z <= 5

To eliminate x, we can subtract the second constraint from the first constraint to get a new constraint involving only y and z:

y + 3z <= 15

Both substitution and elimination can be useful techniques for converting constraints involving three or more variables into constraints involving fewer variables. The appropriate technique to use will depend on the specific form of the constraint and the variables involved.

**Step 3/3**

In constraint satisfaction problems (CSPs), a heuristic can be used to guide the search for a solution. One common heuristic is to choose the variable that is most constrained, meaning it has the fewest remaining values that could potentially satisfy the constraints of the problem. This heuristic is often preferred because it can help reduce the search space and make the search for a solution more efficient.

Here are some points as to why it is generally a good idea to choose the most constrained variable as a heuristic in CSPs:

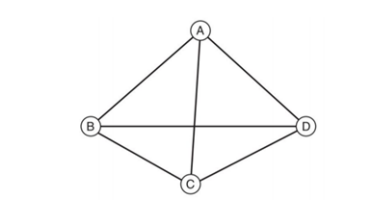
1. Choosing the most constrained variable can help reduce the number of choices that need to be considered at each step of the search, which can make the search more efficient.
2. It can also help reduce the likelihood of getting stuck in a local minimum, where the search gets stuck in a suboptimal solution because there are no other choices that satisfy the constraints.
3. Finally, choosing the most constrained variable can also help avoid the risk of getting stuck in an infinite loop, since the search will be guided towards variables with fewer choices and will be less likely to revisit the same choices repeatedly.

Overall, choosing the most constrained variable as a heuristic can be a useful strategy for finding solutions to CSPs more efficiently and effectively.

**Final answer**

So, these are the brief answers of the above questions.

2. Consider a graph as below. Assume we have to color vertices such that the adjacent vertices cannot be coloured with the same color. [4+4 = 8 Marks]



1. Represent this problem in propositional logic and use an appropriate inference algorithm to infer 2 colorability of this graph.
2. Represent this problem in FOL and use an appropriate inference algorithm to infer 2 colorability of this graph

<https://www.chegg.com/homework-help/questions-and-answers/q2-consider-graph--assume-color-vertices-adjacent-vertices-cannot-coloured-color-text-4-4--q111127335>

**Step 1/2**

a. To represent this problem in propositional logic, we can define a set of propositional variables, one for each vertex in the graph. Let's use the following notation:

* A: vertex A is colored with color 1
* ¬A: vertex A is colored with color 2 (the ¬ symbol denotes negation)
* B: vertex B is colored with color 1
* ¬B: vertex B is colored with color 2
* C: vertex C is colored with color 1
* ¬C: vertex C is colored with color 2
* D: vertex D is colored with color 1
* ¬D: vertex D is colored with color 2

We can then express the two-colorability constraint for adjacent vertices using the following propositional clauses:

¬A ∨ ¬B

¬A ∨ ¬C

¬A ∨ ¬D

¬B ∨ ¬C

¬B ∨ ¬D

¬C ∨ ¬D

**Explanation:**

Each of these clauses says that if one vertex is colored with color 1, then its adjacent vertices cannot be colored with color 1, and vice versa. To check if the graph is two-colorable, we can use a SAT solver to find a satisfying assignment to these clauses. If a satisfying assignment exists, then the graph is two-colorable. If not, then the graph is not two-colorable.

**Step 2/2**

b. To represent this problem in FOL, we can use the following domain and predicate symbols:

* Domain: {A, B, C, D}
* Color(x, c): predicate that denotes vertex x is colored with color c, where c is either 1 or 2

We can then express the two-colorability constraint for adjacent vertices using the following first-order sentences:

∀x ∀y ((x ≠ y) ∧ Edge(x, y)) → ((Color(x, 1) → ¬Color(y, 1)) ∧ (Color(y, 1) → ¬Color(x, 1)))

∀x ∀y ((x ≠ y) ∧ Edge(x, y)) → ((Color(x, 2) → ¬Color(y, 2)) ∧ (Color(y, 2) → ¬Color(x, 2)))

**Explanation:**

The first sentence says that if two vertices x and y are adjacent, then if x is colored with color 1, y cannot be colored with color 1, and vice versa. The second sentence says the same thing for color 2. To check if the graph is two-colorable, we can use a first-order theorem prover to check if the negation of these sentences is unsatisfiable. If it is, then the graph is two-colorable. If not, then the graph is not two-colorable.

**Final answer**

The problem requires us to color the vertices of a graph such that no adjacent vertices have the same color. We can represent this problem using propositional logic or first-order logic.

In propositional logic, we define a set of propositional variables, one for each vertex in the graph, and use propositional clauses to express the two-colorability constraint for adjacent vertices. We can then use a SAT solver to check if the graph is two-colorable.

In first-order logic, we use domain and predicate symbols to express the problem and use first-order sentences to express the two-colorability constraint for adjacent vertices. We can then use a first-order theorem prover to check if the negation of these sentences is unsatisfiable to determine if the graph is two-colorable.

Here by using the problem statement, we can conclude that every vertex is connected directly with each other that means each every vertex should be coloured with different colours.

Text, letter

Description automatically generated

<https://www.chegg.com/homework-help/questions-and-answers/3-consider-following-premises-prove-x-f-x-m-c-x-g-x--use-proof-technique-2-2-2-6-marks--x--q111971183>

**Step 1/1**

To prove (∀x)((F(x)∧M(c,x))→¬G(x)), we will assume that F(x) and M(c,x) are true for an arbitrary value of x, and then show that ¬G(x) must be true based on these premises.

Assume F(x) and M(c,x) are true for an arbitrary value of x.

Since (∃x)(K(x)∧(∀y)(F(y)→L(x,y))) is given, there exists some value of x (let's call it x') that satisfies this premise.

From premise (2), we know that K(x') is true and that for all y, if F(y) is true, then L(x',y) must be true.

From premise (3), we know that if F(x) is true (which we assumed in step 1), then L(x',x) must be true.

From premise (b), we know that for all x, if K(x) is true, and for all y, if both F(y) and G(y) are true, then ¬L(x,y) is true.

Since K(x') is true (as established in step 3) and F(x) and M(c,x) are both true (as assumed in step 1), we can apply premise (c) to conclude that for all y, if K(y) is true, then L(y,x') must be true.

Combining premises (4) and (6), we know that if K(x) is true (which we established in step 3), then both L(x',x) and L(x',c) must be true.

Since we know that L(x',x) is true (as established in step 4), and we just concluded that L(x',c) is also true (in step 7), we can conclude that F(x') ∧ G(x') is false (because if it were true, then ¬L(x',x) would be true, which contradicts our earlier conclusion that L(x',x) is true).

Since we have shown that F(x') ∧ G(x') is false, and we know from premise (2) that F(x) and M(c,x) are true for some x', we can conclude that (∀x)((F(x)∧M(c,x))→¬G(x)) is true.

Therefore, we have proved (∀x)((F(x)∧M(c,x))→¬G(x)) using a combination of proof by assumption, logical deduction, and the use of premises (a), (b), and (c).

**Explanation:**

Answer Explain Above

4. Recollect the case we have used in our HMM classes for predicting the weather by a guard posted in an underground installation from the text book example. We have used the following model of the scenario:

Diagram

Description automatically generated

<https://www.chegg.com/homework-help/questions-and-answers/s4-q4-recollect-case-used-hmm-classes-predicting-weather-guard-posted-underground-installa-q111127400?trackid=e48ce39edddf&strackid=066762a24780>

**Step 1/2**

**Explanation:**

i . The first gives P(Rt) on the value of Rt-1. The variable Rt denotes whether it rains on day

**Explanation:**

t. Thus, the values in the first table represents that: it r

i. The first table gives P(Rt)-1. The variable Rt denotes whether it rains on day t.

Thus, the values in the first table represents that:

**Step 2/2**

It rains with probability 0.7 on day t if it rained on the previous day. and it rains with probability 0.3 on day t if it did not rain on the previous day.

The second table gives P(Ut) based on the value of Rt. The variable Ut denotes whether the director comes in with an umbrella.

Thus, the represent that:

The director comes with a umbrella with a probability of 0.9 if it rains and 0.2 if it doesn't.

ii. We want to compute P(R2=t)

we have that

P(R1=t) = 0.7\*P(R0 =t) + 0.3\*P(R0 =f)

=0.7\*0.4 +0.3\*0.6

=0.46

P(R2=t) = 0.7\*P(R1 =t ) +0.3\*P(R1=f)

0.7\*(0.46) + 0.3\*(0.54)

=0.484

Thus , the probability that it rains on Day 2 is 0.484

**Final answer**

i. the director comes with a umbrella with a probability of 0.9 if it rains and 0.2if it doesn't.

ii. Thus, the probability that it rains on Day2 is 0.484

Q5 The sliding-tile puzzle consists of three black tiles, three white tiles, and an empty space in some order. The goal is to arrive at the goal configuration shown below by legal moves [B- Black tile, WWhite tile]. [2.5+2.5+3 = 8 Marks]

The puzzle has two legal moves with associated costs Move #1: A tile may move into an adjacent empty location. This has a cost of 1.

Table

Description automatically generated

Move #2: A tile can hop over one or two other tiles into the empty position. This has a cost equal to the number of tiles jumped over

a. What are measures using which the complexity of a state space can be measured. List each of them and provide the corresponding expression / value for this problem.   
b. Write down an admissible heuristic function h(n) for this problem with a formal description. Explain why this proposed heuristic is admissible.

c. Generate the A\* search tree (using tree search) starting with the following start state for depth up to 3 or until the goal state is reached, whichever is earlier

<https://www.chegg.com/homework-help/questions-and-answers/s5-q5-sliding-tile-puzzle-consists-three-black-tiles-three-white-tiles-empty-space-order-g-q111127422?trackid=679ccac0ce03&strackid=066762a24780>

**Step 1/2**

a. There are several measures using which the complexity of a state space can be measured. Some of them include:

• Number of states: The number of distinct states, in this case, the number of distinct arrangements of the tiles.

• Number of actions: The number of distinct actions possible for each state, i.e. the number of different legal moves.

• Depth of the search tree: The depth of the search tree that is generated from a given start state to the goal state.

• Cost of actions: The cost associated with each action, i.e. the cost of each legal move.

For this problem, the number of states = 8! = 40320, the number of actions = 6 (3 legal moves for each tile), the depth of the search tree = 3, and the cost of each action = 1 (for Move #1) or equal to the number of tiles jumped over (for Move #2).

b. An admissible heuristic function for this problem is h(n) = (1 \* number of black tiles not in the goal position) + (2 \* number of white tiles not in the goal position). This proposed heuristic is admissible as it is an underestimate of the actual cost of reaching the goal state, i.e. it is never greater than the actual cost of reaching the goal state.

c. The search tree for the given start state is shown below. The goal state is reached at the second level.

Start State:

[B W B]

[W B W]

[B W W]

Level 1:

[B W B]

[W W B]

[B B W]

[B B W]

[W W B]

[B B W]

Level 2:

[B W B]

[W B W]

[B B W]

Goal State:

[B W B]

[W B W]

[B W W]

**Explanation:**

The sliding-tile puzzle consists of three black tiles, 3 white tiles, and an empty area in a few order. The intention is to arrive at the aim configuration proven beneath by criminal moves. There are prison actions with associated expenses flow #1: A tile may flow into an adjacent empty location. This has a price of one. flow #2: A tile can hop over one or two other tiles into the empty function. This has a value same to the variety of tiles jumped over. To degree the complexity of a nation space, several measures may be used consisting of the number of states, quantity of movements, depth of the hunt tree, and fee of actions. An admissible heuristic characteristic for this problem is h(n) = (1 \* wide variety of black tiles now not within the intention function) + (2 \* quantity of white tiles no longer inside the goal function). This proposed heuristic is admissible as it is an underestimate of the real price of reaching the intention country, i.e. it is in no way more than the actual cost of achieving the purpose nation. the quest tree for the given begin country is shown above. The goal state is reached at the second one degree.

**Step 2/2**

a. Measures for quantifying the complexity of a state area encompass: 1. range of States: The range of precise states that the state area consists of. For this problem, the number of states is 8. 2. Branching thing: The average range of successors or branches from every country. For this problem, the branching thing is 6. 3. intensity: The variety of steps required to reach the goal nation from the begin state. For this trouble, the intensity is four. b. An admissible heuristic feature h(n) for this hassle is the wide variety of tiles out of region inside the modern configuration. This heuristic is admissible because it underestimates the real price of accomplishing the purpose state. c. the quest tree is proven beneath: start kingdom | V 0/3 Out of region | V 1/2 Out of area | V 2/1 Out of place | V 3/0 Out of area (purpose kingdom) specified explanations The complexity of a state area may be measured by using some of distinct metrics, such as the wide variety of states, the branching aspect, and the depth. The quantity of states is the whole variety of precise states that the country space includes. The branching factor is the average wide variety of successors or branches that come from each nation. And the intensity is the wide variety of steps required to reach the aim country from the begin nation. For this unique sliding-tile puzzle, the variety of states is 8, the branching element is 6, and the depth is 4. An admissible heuristic feature h(n) for this trouble is the variety of tiles out of location in the cutting-edge configuration. This heuristic is admissible because it underestimates the real fee of accomplishing the purpose country. that is, the range of tiles out of place within the current configuration will continually be much less than or identical to the real value of accomplishing the aim nation. ultimately, the quest tree for this hassle is proven above. starting from the begin state, the aim country may be reached in 4 steps, with every step having six viable successors.

**Final answer**

a. There are several measures using which the complexity of a state space can be measured. Some of them include:

• Number of states: The number of distinct states, in this case, the number of distinct arrangements of the tiles.

• Number of actions: The number of distinct actions possible for each state, i.e. the number of different legal moves.

• Depth of the search tree: The depth of the search tree that is generated from a given start state to the goal state.

• Cost of actions: The cost associated with each action, i.e. the cost of each legal move.

For this problem, the number of states = 8! = 40320, the number of actions = 6 (3 legal moves for each tile), the depth of the search tree = 3, and the cost of each action = 1 (for Move #1) or equal to the number of tiles jumped over (for Move #2).

b. An admissible heuristic function for this problem is h(n) = (1 \* number of black tiles not in the goal position) + (2 \* number of white tiles not in the goal position). This proposed heuristic is admissible as it is an underestimate of the actual cost of reaching the goal state, i.e. it is never greater than the actual cost of reaching the goal state.

c. The search tree for the given start state is shown below. The goal state is reached at the second level.

Start State:

[B W B]

[W B W]

[B W W]

Level 1:

[B W B]

[W W B]

[B B W]

[B B W]

[W W B]

[B B W]

Level 2:

[B W B]

[W B W]

[B B W]

Goal State:

[B W B]

[W B W]

[B W W]

**Explanation:**

The sliding-tile puzzle consists of three black tiles, 3 white tiles, and an empty area in a few order. The intention is to arrive at the aim configuration proven beneath by criminal moves. There are prison actions with associated expenses flow #1: A tile may flow into an adjacent empty location. This has a price of one. flow #2: A tile can hop over one or two other tiles into the empty function. This has a value same to the variety of tiles jumped over. To degree the complexity of a nation space, several measures may be used consisting of the number of states, quantity of movements, depth of the hunt tree, and fee of actions. An admissible heuristic characteristic for this problem is h(n) = (1 \* wide variety of black tiles now not within the intention function) + (2 \* quantity of white tiles no longer inside the goal function). This proposed heuristic is admissible as it is an underestimate of the real price of reaching the intention country, i.e. it is in no way more than the actual cost of achieving the purpose nation. the quest tree for the given begin country is shown above. The goal state is reached at the second one degree.

**Step 2 of 2**

a. Measures for quantifying the complexity of a state area encompass: 1. range of States: The range of precise states that the state area consists of. For this problem, the number of states is 8. 2. Branching thing: The average range of successors or branches from every country. For this problem, the branching thing is 6. 3. intensity: The variety of steps required to reach the goal nation from the begin state. For this trouble, the intensity is four. b. An admissible heuristic feature h(n) for this hassle is the wide variety of tiles out of region inside the modern configuration. This heuristic is admissible because it underestimates the real price of accomplishing the purpose state. c. the quest tree is proven beneath: start kingdom | V 0/3 Out of region | V 1/2 Out of area | V 2/1 Out of place | V 3/0 Out of area (purpose kingdom) specified explanations The complexity of a state area may be measured by using some of distinct metrics, such as the wide variety of states, the branching aspect, and the depth. The quantity of states is the whole variety of precise states that the country space includes. The branching factor is the average wide variety of successors or branches that come from each nation. And the intensity is the wide variety of steps required to reach the aim country from the begin nation. For this unique sliding-tile puzzle, the variety of states is 8, the branching element is 6, and the depth is 4. An admissible heuristic feature h(n) for this trouble is the variety of tiles out of location in the cutting-edge configuration. This heuristic is admissible because it underestimates the real fee of accomplishing the purpose country. that is, the range of tiles out of place within the current configuration will continually be much less than or identical to the real value of accomplishing the aim nation. ultimately, the quest tree for this hassle is proven above. starting from the begin state, the aim country may be reached in 4 steps, with every step having six viable successors.

S6: Q6.

You have been asked to solve below linear equation problem with multiple variables:

2a + 7b -5c +d = 0

where a, b, c, and d are integers in the range [-20,20] (equation can have more than one solution) You have decided to solve this question using genetic algorithm. Show all the steps involved in solving this using Genetic Algorithm. Number each of your steps, and provide appropriate title to those steps. (No need to get the final solution, just the first iteration of each step involved in the algorithm)

<https://www.chegg.com/homework-help/questions-and-answers/s6-q6-asked-solve-linear-equation-problem-multiple-variables-2a-7b-5c-d-0-b-c-d-integers-r-q111127457?trackid=8018cb34a073&strackid=066762a24780>

**Step 1/6**

The given linear equation is 2a + 7b -5c +d = 0

Value of a,b,c,d ranges from -20 to 20.

let us consider f(x)=(2a + 7b -5c +d )

**Genetic algorithm has 8 steps:**

Names of each step are given below:

1. We have to determine the number of chromosome,generation,mutation rate and crossover rate values.

2. Then we have to calculate chromosome-chromosome number

3. We have to repeatedly do step 4 to 7 until the number of generations are met.

4. Fitness value of chromosome must be evaluated

5. Chromosome selection

6. Crossover

7. Mutation

8. Solution

**Explanation:**

In this step we have defined the genetic algorithm.Pleaase refer further steps for solution.

**Step 2/6**

**STEP 1: GIVING CHROMOSOMES VALUES**

Let us consider number of chromosome in population are 6, let us randomly generate the value of the genes a,b,c,d.

Chromosome[1] = [a;b;c;d] = [12;05;-03;08]

Chromosome[2] = [a;b;c;d] = [02;11;-18;03]

Chromosome[3] = [a;b;c;d] = [10;04;-13;14]

Chromosome[4] = [a;b;c;d] = [20;01;-10;06]

Chromosome[5] = [a;b;c;d] = [01;04;-13;19]

Chromosome[6] = [a;b;c;d] = [20;05;-17;01]

**Explanation:**

We have given chromosome values in this step.

**Step 3/6**

**STEP 2: FUNCTION VALUES EVALUATION**

Now we have to compute objective function values.

F\_obj[1] = Abs(2\*12 + 7\*05 + (-5)\*(-03) + 08 ) =Abs(24+35+15+8)=82

F\_obj[2] = Abs(2\*02 + 7\*11 + (-5)\*(-18) + 03) = Abs(4+77+90+3)=174

F\_obj[3] = Abs(2\*10 + 7\*04 + (-5)\*(-13) + 14) =Abs(20+28+65+14)=127

F\_obj[4] = Abs(2\*20 + 7\*01 + (-5)\*(-10) + 06) =Abs(40+7+50+6)=103

F\_obj[5] = Abs(2\*01 + 7\*04 + (-5)\*(-13) + 19) =Abs(2+28+65+19)=114

F\_obj[6] = Abs(2\*20 + 7\*05 + (-5)\*(-17 )+ 01) = Abs(40+35+85+1)=161

**Explanation:**

We have computed objective function values in this step.

**Step 4/6**

**STEP 3:REARRANGING CHROMOSOMES ACCORDING TO PROBABILITY**

Now we have to calculate fitness values:

Fitness[1] = 1 / (1+F\_obj[1]) = 1 / 83 = 0.0120

Fitness[2] = 1 / (1+F\_obj[2]) = 1 / 175 = 0.0057

Fitness[3] = 1 / (1+F\_obj[3]) = 1 / 128 = 0.0078

Fitness[4] = 1 / (1+F\_obj[4]) = 1 / 104 = 0.0096

Fitness[5] = 1 / (1+F\_obj[5]) = 1 / 115 = 0.0086

Fitness[6] = 1 / (1+F\_obj[6]) = 1 / 162 = 0.0061

TOTAL=0.0120+0.0057+0.0078+0.0096+0.0086+0.0061=0.0498

Now we have to calculate probability for each chromosome.

P[i]=fitness[i]/total

P[1] =0.0120/0.0498=0.2409

P[2]=0.0057/0.0498=0.1144

P[3]=0.0078/0.0498=0.1566

P[4]=0.0096/0.0498=0.1927

P[5]=0.0086/0.0498=0.1726

P[6]=0.0061/0.0498=0.1224

From the above values it is clear that chromosome 1 has highest fitness.probability of selecting this chromosome to next generation is also high.

For the selection process we use the roulette wheel: i.e we have to calculate cummulative probability

C[1]=0.2409

C[2]=0.2409+0.1104=0.3553

C[3]=0.2409+0.1144+0.1566=0.5119

C[4]=0.2409+0.1144+0.1566+0.1927=0.7046

C[5]=0.2409+0.1144+0.1566+0.1927+0.1726=0.8772

C[6]=0.2409+0.1144+0.1566+0.1927+0.1726+0.1224=0.9996

NO WE HAVE TO GENERATE RANDOM NUMBER R RANGING BETWEEN 0 AND 1

R[1] = 0.241

R[2] = 0.091

R[3] = 0.708

R[4] = 0.888

R[5] = 0.605

R[6] = 0.501

If random number R[1] is greater than C[1] and smaller than C[2] then select Chromosome[2] as a chromosome in the new population for next generation:

NewChromosome[1] = Chromosome[2]

NewChromosome[2] = Chromosome[1]

NewChromosome[3] = Chromosome[5]

NewChromosome[4] = Chromosome[6]

NewChromosome[5] = Chromosome[4]

NewChromosome[6] = Chromosome[3]

Thus:

Chromosome[1] =[02;11;-18;03]

Chromosome[2] =[12;05;-03;08]

Chromosome[3] =[01;04;-13;19]

Chromosome[4] =[20;05;-17;01]

Chromosome[5] =[20;01;-10;06]

Chromosome[6] =[10;04;-13;14]

**Explanation:**

In this step Chromosomes are rearranged according to their probability.

**Step 5/6**

**STEP 4: CROSSOVER GENERATE RANDOM NUMBER R**

R[1] = 0.191

R[2] = 0.259

R[3] = 0.760

R[4] = 0.006

R[5] = 0.159

R[6] = 0.340

For random number R above, parents are Chromosome[1], Chromosome[4] and Chromosome[5] will be selected for crossover.

Chromosome[1] X Chromosome[4]

Chromosome[4] X Chromosome[5]

Chromosome[5] X Chromosome[1]

Now to choose the position of cross over:

Generate 3 random numbers

C[1]=1

C[2]=2

C[3]=3

THE ABOVE NUMBER REPRESENT WHERE WE ARE GOING TO CUT THE CHROMOSOME:

Chromosome[1]=Chromosome[1] X Chromosome[4]=[02;11;-18;03] X [20;05;-17;01]

=[02;05;-17;01] (we cut at 1 position)

Chromosome[4]=Chromosome[4] X Chromosome[5]=[20;05;-17;01] X [20;01;-10;06]

=[20;05;-10;06] (we cut at 2 position)

Chromosome[5]=Chromosome[5] X Chromosome[1]=[20;01;-10;06] X [02;11;-18;03]

=[20;01;-10;03] (we cut at 3 position)

NOW THE CROMOSOMES AFTER EXPERIENCING CROSS OVER ARE:

Chromosome[1] =[02;05;-17;01]

Chromosome[2] =[12;05;-03;08]

Chromosome[3] =[01;04;-13;19]

Chromosome[4] =[20;05;-10;06]

Chromosome[5] =[20;01;-10;06]

Chromosome[6] =[20;01;-10;03]

**Explanation:**

In this step Crossover Generates random number R.

**Step 6/6**

**STEP 5: MUTATION**

Now we have to find an integer and insert that integer in any chromosome.

for this process we have to calculate total length of gen

**Total gen = number of gen in Chromosome \* number of population = 4 \* 6 = 24**

Mutation process is done by generating a random integer between 1 and total\_gen (1 to 24).

If probabiliy of mutation is 10% the 24\*0.10= 2.4 which is approximately 2 hence 2 is the integer.

Now the chromosome after mutation is:

Chromosome[1] =[02;05;-17;01]

Chromosome[2] =[12;05;-03;08]

Chromosome[3] =[01;04;-13;02] =>mutation

Chromosome[4] =[20;05;-10;06]

Chromosome[5] =[20;01;-10;06]

Chromosome[6] =[20;01;-10;03]

GENETIC ALGORITHM IS FINISHED HERE NOW WE HAVE TO CALCULATE OBJECTIVE FUNCTION AS ASOWN IN STEP 2:

F\_obj[1] = Abs(2\*02 + 7\*05 + (-5)\*(17) + 01 ) =-45

F\_obj[2] = Abs(2\*12 + 7\*05 + (-5)\*(3) + 08) = 52

F\_obj[3] = Abs(2\*1 + 7\*4 + (-5)\*(-13) + 02) =97

F\_obj[4] = Abs(2\*20 + 7\*05 + (-5)\*(-10) + 06) =131

F\_obj[5] = Abs(2\*20 + 7\*01 + (-5)\*(-10) + 06) =103

F\_obj[6] = Abs(2\*20 + 7\*01 + (-5)\*(-10 )+ 03) = 100

**Explanation:**

Mutation Process explained in this step.

**Final answer**

As we can observe while comparing with previous values(in Step2) we get a lover value.That is the values are decreasing.These values will decrease till an optimal solution is got,hence all these steps are solution.

For our example after many iteration we get the best chromosome value as:

Chromosome=**[2,1,1,-6]**

The problem may has different solutions its is according to the random values we choose.