2. THE PROPOSED ADAPTIVE SYSTEM

According to Holland [39], the study of adaptation involves the study of both the adaptive system and its environment. In general terms, it is a study of how systems can generate procedures enabling them to adjust efficiently to their environments. If adaptability is not to be arbitrarily restricted at the outset, the adapting system must be able to generate any method or procedure capable of an effective definition. Therefore, adaptation is based upon differential selection of supervisory programs. That is, the more "successful" a supervisory program, in terms of the ability of its problem-solving programs to produce solutions, the more predominant it is to become (in numbers) in a population of supervisory programs.

It is to be noted that in the earliest adaptive systems writings, Holland [40] only hinted at the importance of crossover or other recombinant genetic operators. The first direct written acknowledgment of the importance of these other operators came three years later [28].

Given a set of rules to start with, the adaptive rule-based system applies various genetic algorithms [28], [32], [73] to generate and test new rules. Each solution reached by the system is evaluated, and credit is assigned to each rule and the rule set.

The first attempt towards development of an adaptive system that combines symbolic representation and genetic based machine learning was by
Stackhouse and Zeiglar [73] in the learning of a robot navigation problem. The standard rule \(<\text{IF} \text{ condition} \text{ THEN} \text{ action} >\) is used. The system operates in a loop consisting of the following steps which are performed each generation.

1. A solution to all problems is attempted using a subset of the current rule set.
2. Each solution reached is evaluated and credit is assigned to each rule and the rule set.
3. The current rule set has the "weeding" criteria applied to it. Low performers are removed.
4. A few new rules are generated from, and placed into the current rule set.

Zeiglar and Stackhouse [73] followed the Michigan approach of GBML for their system in which the genetic operators used are applied on the rules. Recently, some developments suggesting that it might be possible to combine the GBML approaches, Pitt's approach and Michigan's approach so it may be more powerful and interesting ways [19], [35].

In this chapter we present an adaptive system which is based on a combination of GBML (Michigan and Pitt approaches) and symbolic machine learning approaches and on a more flexible structure that handles the variable precision logic in which the HCPRs system is used as knowledge representation.
2.1. System Overview

The basic cycle of the system is divided into two phases. The first phase is to construct two Working HCPR-tree Sets (WS-I and WS-II) which contain sequence of solution paths for all those problems which are solvable using the ad hoc master set. In the second phase genetic algorithms play the main role in improving the capability of these working sets by generating new rules and HCPR-trees, thereby enabling the working sets to solve the unsolved problems. The competition between the two working sets in the second phase for solving more number of problems also plays an important role to generate faster a proper HCPR-tree set that would solve most of the problems thereby expediting the learning process. A general overview system flow chart is shown in Fig. 2.1.

2.2. Genetic Operators For HCPRs system

Development of genetic operators has been an active area of research and several genetic operators have been suggested in the past [13], [16], [20], [30], [31], [41], [67], [71] for use in genetic algorithms and GBML systems. Since genetic operators are one of the most important components of genetic algorithms, further investigation and application of these operators and development of new advanced genetic operators continue to be an active research area [9]. A number of genetic operators suitable for HCPRs system are suggested below:
Fig. 2.1. A general overview system flow chart
2.2.1. Set Crossover

This is performed by selecting one HCPR-tree set from TRS-I (TRansit Set I) and another from TRS-II (TRansit Set II) (TRS-I and TRS-II are sets containing HCPR-tree sets and explained latter in the algorithm), which are then cut and crossed (Fig.2.2).

2.2.2. Inversion

The inversion operator applied to a set of HCPR-trees first specifies two HCPR-trees indicating the beginning and the end of the sequence of HCPR-trees to be inverted and then inverts the chosen sequence of HCPR-trees resulting in a new form (ordering) of HCPR-tree set (Fig.2.3).

2.2.3. AND Algorithm for Independent HCPRs

The AND algorithm has been stated by Stackhouse and Zeiglar in [73] for the standard rules system. Here an extension of AND algorithm to HCPRs system is proposed. AND algorithm combines two independent HCPRs chosen at random into a new higher order HCPR as shown in the following example.

Example: Consider the following HCPRs corresponding to HCPR-trees HCPR_1 and HCPR_2 as shown in Fig.2.4. (notice that in this figure and in the figures that follow [p1] [p1,...] etc represents the condition part of the related HCPR). HCPRs corresponding to HCPR_1 are

\[ \text{HCPR}_1 \]
\[ \text{HCPR}_2 \]
Fig. 2.2. The set crossover operation (here H-i denotes HCPR_i).
Fig. 2.3. Inversion operation.
(1)  [A1]  IF [p1]
      UNLESS [ ]
      GENERALITY [ ]
      SPECIFICITY [A11,A12]

(2)  [A11] IF [p2]
      UNLESS [ ]
      GENERALITY [A1]
      SPECIFICITY [ ]

(3)  [A12] IF [p3]
      UNLESS [ ]
      UNLESS [k1]
      GENERALITY [A1]
      SPECIFICITY [A121]

(4)  [A121] IF [p4]
      UNLESS [ ]
      GENERALITY [A12]
      SPECIFICITY [ ]

HCPRs corresponding to HCPR₂ :

(a)  [B1] IF [p5]
      UNLESS [ ]
      GENERALITY [ ]
      SPECIFICITY [B11,B12]
(b) [B11] IF [p7]
   UNLESS [ ]
   GENERALITY [B1]
   SPECIFICITY [ ]

(c) [B12] IF [p8]
   UNLESS [k1]
   GENERALITY [B1]
   SPECIFICITY [B121,B122]

(d) [B121] IF [p9]
   UNLESS [ ]
   GENERALITY [B12]
   SPECIFICITY [ ]

(e) [B122] IF [p10]
   UNLESS [ ]
   GENERALITY [B12]
   SPECIFICITY [ ]

Assuming that the following two HCPRs are chosen at random one from each HCPR-tree

(3) [A12] IF [p3]
   UNLESS [k1]
   GENERALITY [A1]
   SPECIFICITY [A121]
Fig. 2.4. Two HCPR-trees chosen at random.
Simultaneously, the process of the AND algorithm is applied to the specificity part of the above HCPRs resulting in a HCPR-tree shown in Fig.2.5.

2.2.4. Scaling Algorithm for a Linear System

In a linear system, the condition C repeated N times should produce the sequence of N identical actions A [73]. Hence from the HCPR

\[ [A] \quad \text{IF} \quad [C] \quad \text{UNLESS} \quad [k] \]
Fig. 2.5. The resultant HCPR-tree when AND algorithm is applied to the HCPR-trees in Fig. 2.4.
A new HCPR can be generated:

\[(N*A) \text{ IF } (N*C)\]
\[\text{UNLESS } [K]\]
\[\text{GENERALITY } [G]\]
\[\text{SPECIFICITY } [X1,X2,X3]\]

2.2.5. HCPRs Crossover

The crossover suggested in [73] for standard rule structure can be extended to HCPR system as explained below. Given two HCPRs

\[(A1) \text{ IF } [p1,p2]\]
\[\text{UNLESS } [k1]\]
\[\text{GENERALITY } [A1]\]
\[\text{SPECIFICITY } [A111,A112]\]

and

\[(B1) \text{ IF } [p9,p5,p3]\]
\[\text{UNLESS } [k2]\]
\[\text{GENERALITY } [B1]\]
\[\text{SPECIFICITY } [B121]\]

the application of crossover would generate the following new HCPRs (by swapping the condition parts in the above HCPRs):
2.2.6. Mutation

Two types of mutation operators are proposed for the HCPR system.

(a) HCPR mutation

(b) HCPR-tree mutation

In biology, two classes of gene mutations are recognized, point mutation and intragenic deletion [24]. The point mutation is altering the genes (single gene). This kind of mutation has been widely used by the researchers of the genetic algorithms [28], [31], [32], [44]. Intragenic deletion is an extensive deletion within the gene. In the most extreme case, all the information of gene is lost. Based on intragenic mutation HCPR mutation is suggested that would remove redundant or unrelated conditions from the HCPR depending on some mutation probability value. Following example illustrates the usefulness of this operator:
Let us consider the HCPR

**John is at home**  IF  [It is sunday, It is raining, John has a car]  

UNLESS [John is sleeping, John is doing overtime]

GENERALITY [ ]

SPECIFICITY [ ]

in which [John has a car] is a redundant condition and [John is sleeping] is unrelated censor condition. Application of HCPR mutation operator may produce a better rule.

**John is at home**  IF  [It is sunday, It is raining]  

UNLESS [John is doing overtime]

GENERALITY [ ]

SPECIFICITY [ ]

Under HCPR-tree mutation, the children of two siblings chosen at random from a HCPR-tree are exchanged as shown in Fig.2.6.

2.2.7. **Fusion**

Fusion as a genetic operator is a combination of two HCPR-trees to generate one or more new HCPR-trees:

\[
\text{Fusion}(\text{HCPR}_i, \text{HCPR}_j) \rightarrow \text{HCPR}_k
\]
Fig. 2.6. HCPR-tree mutation: Here B and D are two siblings chosen at random.
such that \( cr(HCPR_k) = cr(HCPR_i) + cr(HCPR_j) \),

where \( HCPR_i \) and \( HCPR_j \) are the parents, and \( HCPR_k \) is the offspring.
\( cr(HCPR_i) \) is the credit of the \( i \)th HCPR-tree. Here the two HCPR-trees \( HCPR_i \) and \( HCPR_j \) are related satisfying the following condition :

\[ R_i \cap R_j \neq \emptyset , \text{ where } R_i \text{ and } R_j \text{ are the sets representing condition (properties) parts of the HCPRs corresponding to the root nodes in } HCPR_i \text{ and } HCPR_j \text{ respectively.} \]

Under this condition the following two cases can arise:

**case a.** \( R_i \subseteq R_j \) or \( R_j \subseteq R_i \)

**case b.** \( R_i \nsubseteq R_j \) and \( R_j \nsubseteq R_i \)

The Fusion operator can be considered as a type of crossover operator because it is similar to the standard crossover genetic operator in the sense that a newly generated HCPR-tree gets its characteristics from both the parents which are chosen at random. Clearly, the offspring \( HCPR_k \) bears a structural relationship to both the parents \( HCPR_i \) and \( HCPR_j \).

The different choices of the two parameters of the Fusion\((X,Y)\) operator give rise to the following four possibilities (sub HCPR-tree in the following means a randomly chosen subtree from HCPR-tree):

(i) \( X = HCPR_i \), \( Y = HCPR_j \)

(ii) \( X = \text{sub } HCPR_i \), \( Y = HCPR_j \)

(iii) \( X = HCPR_i \), \( Y = \text{sub } HCPR_j \)
(iv) \( X = \text{sub HCPR}_i, \ Y = \text{sub HCPR}_j \)

When Fusion is applied under the choice(i), the parents are removed and the offspring is retained whereas in the remaining possibilities ((ii) to (iv)) the parents as well as the offsprings are retained.

During the Fusion operation, how to randomly generate sub HCPR-trees and perform credit revision are the two main issues. These issues are discussed below.

2.2.7.1. Random generation of sub HCPR-trees

Given a HCPR-tree, The random generation of a sub HCPR-tree involves the following steps.

A. A node in a given HCPR-tree is chosen at random. The subtree corresponding to this node as root would form the sub HCPR-tree.

B. The number of firings of the chosen node (root of the sub HCPR-tree) remains the same.

C. The chosen node (node of sub HCPR-tree) inherits the properties of its ancestors and a share of the credit of its ancestors is added to its credit.

Fig.2.7 shows the scheme used for calculating the share for the chosen node, where \( K_i, A_i, \) and \( D_i \) are HCPRs, and \( F_1, F_2, \) and \( F_3 \) are the factors corresponding to the HCPR-tree with roots at \( K_i, A_i, \) and \( D_i \) respectively such that
Fig. 2.7. Computation of credit share to be assigned to the randomly chosen node. Assuming $D_1$ as the randomly chosen node, the share of the credit of its ancestors (here only the Root node) is computed as

$$\text{share of } D_1 = \frac{F_3}{F_1 + F_2 + F_3} * \text{cr(Root)}$$

where $F_1$, $F_2$ and $F_3$ are the factors defined by equations (1), (2) and (3) respectively.
\[ F_1 = \sum_{i=1}^{4} F(K_i) \times np(K_i) \]  
(1)

\[ F_2 = \sum_{i=1}^{6} F(A_i) \times np(A_i), \text{ and} \]  
(2)

\[ F_3 = \sum_{i=1}^{6} F(D_i) \times np(D_i), \]  
(3)

where \( F(X) \) is the number of firings of the HCPR \( X \) and \( np(X) \) is the number of properties in the condition part of the HCPR \( X \).

### 2.2.7.2. Credit Revision During Fusion

#### A. Assuming that \( R_i \subseteq R_j \) (case a)

By applying Fusion(HCPR\(_i\),HCPR\(_j\)), the randomly generated HCPR\(_k\) gets two new HCPRs together with old HCPRs. The old HCPRs get their previous credit and number of firings. For the two new HCPRs, credit and number of firings are to be assigned. The number of firings of the first new HCPR (the root node of HCPR\(_k\)) is

\[ F(\text{root HCPR}_k) = F(\text{root HCPR}_i) + F(\text{root HCPR}_j), \]  
(4)

where \( F(\text{root HCPR}_k) \) and \( F(\text{root HCPR}_j) \) are the number of firings of the root nodes of HCPR\(_i\) and HCPR\(_j\) respectively. The credit of (root HCPR\(_k\)) is
computed as:

\[ \text{cr(root HCPR}_k) = \text{cr(root HCPR}_j) + \text{np(root HCPR}_k) \times \text{AVG}_j \]  \hspace{1cm} (5)

where

\[ \text{AVG}_j \] : the average credit for each property at the root of 
HCPR\(_j\) tree and equals to 
\[ \left[ \frac{\text{cr(root HCPR}_j)}{\text{np(root HCPR}_j)} \right] \]

\[ \text{cr(root HCPR}_j) \] : credit of the root node of HCPR\(_j\)

\[ \text{cr(root HCPR}_j) \] : credit of the root node of HCPR\(_j\)

\[ \text{np(root HCPR}_j) \] : number of properties in the condition part of the 
HCPR corresponding to the root node of HCPR\(_j\)

\[ \text{np(root HCPR}_k) \] : number of properties in the condition part of the 
HCPR corresponding to the root node of HCPR\(_k\)

The number of firings of the second new HCPR (a child of root HCPR\(_k\), say E) 
is given by

\[ F(E) = F(\text{root HCPR}_j) \]  \hspace{1cm} (6)

The credit of this node is also computed using

\[ \text{cr(E)} = \text{AVG}_j \times \text{np(E)} \]  \hspace{1cm} (7)

It is to be noted that if \( R_j = R_j \), no new HCPRs are created in the newly 
generated HCPR\(_k\) tree and the set representing condition part of the HCPR 
corresponding to the root of HCPR\(_k\), \( R_k \) is \( R_j \) (\( = R_j \)). The new credit and the 
number of firings are computed using (4) and (5).
B. Assuming that \( R_i \neq R_j \) and \( R_j \neq R_i \) (case b)

In this case the generated HCPR\(_k\) tree gets three new HCPRs (the root node and its two children) together with the old HCPRs. The number of firings and the credit for the old HCPRs do not change. The number of firings of the root node is computed using (4) while its credit is computed using the following formula:

\[
\text{cr(root HCPR}_k) = \text{np(root HCPR}_k) \times [\text{AVG}_i + \text{AVG}_j]
\] (8)

Formulas (6) and (7) are used for computing the number of firings and the credit for the two children HCPRs. Fig.2.8 to 2.10 illustrate various possibilities of Fusion operator and the corresponding credit revision computations.

2.2.8. Fission

Fission is a process which is applied in general after Fusion to restructure a HCPR-tree. Fission operator causes a change in a HCPR-tree's structure creating a new HCPR-tree and can be regarded as genetic mutation operator. The general form of Fission operator can be given as

\[
\text{Fission(HCPR}_i) \rightarrow (\text{HCPR}_i^*)
\]

where \( \text{cr(HCPR}_i^*) = \text{cr(HCPR}_i) \).

During Fission operation the \( \text{HCPR}_i \) tree chosen for restructuring is removed while the resultant \( \text{HCPR}_i^* \) tree is retained. The credit revision during Fission involves almost the same technique as that of credit revision during Fusion. Fig.2.11 depicts the application of Fission operator and the credit
Fig. 2.8. Illustration of Fusion operation (between HCPR$_i$ and HCPR$_j$ generating HCPR$_k$ (case a)) and credit revision computation.
Fig. 2.9. Illustration of Fusion operation (between HCPR_i and HCPR_j generating HCPR_k (case b)) and credit revision computation.
### (a)

**HCPR\(_i\)**

\[ \text{cr}(\text{HCPR}_i) = 92 \]

- \( F = 7 \)
  - \( \text{cr} = 40 \)
  - \( F = 2 \)
    - \( \text{cr} = 6 \)
      - \( [\text{P1}, \text{P2}] \)
- \( [\text{P9}] \)
  - \( F = 5 \)
    - \( \text{cr} = 14 \)
  - \( [\text{P40}] \)
    - \( F = 1 \)
      - \( \text{cr} = 3 \)
    - \( [\text{P15}] \)
      - \( F = 3 \)
        - \( \text{cr} = 8.5 \)
      - \( [\text{P27}] \)
        - \( F = 2 \)
          - \( \text{cr} = 6 \)
          - \( [\text{P30}, \text{P25}] \)
            - \( F = 1 \)
              - \( \text{cr} = 6 \)
- \( [\text{P17}, \text{P100}] \)
  - \( F = 1 \)
    - \( \text{cr} = 6 \)

**HCPR\(_j\)**

\[ \text{cr}(\text{HCPR}_j) = 143.75 \]

- \( F = 8 \)
  - \( \text{cr} = 50 \)
  - \( F = 6 \)
    - \( \text{cr} = 37.5 \)
      - \( [\text{P16}, \text{P70}] \)
        - \( F = 2 \)
          - \( \text{cr} = 12.5 \)
        - \( [\text{P20}, \text{P15}] \)
          - \( F = 2 \)
            - \( \text{cr} = 6.25 \)
          - \( [\text{P27}] \)
            - \( F = 4 \)
              - \( \text{cr} = 25 \)
            - \( [\text{P33}] \)
              - \( F = 4 \)
                - \( \text{cr} = 12.5 \)
- \( [\text{P17}] \)
  - \( F = 1 \)
    - \( \text{cr} = 2.5 \)
  - \( [\text{P16}, \text{P70}] \)

### (b)

**Sub HCPR\(_i\)**

\[ \text{cr}(\text{SubHCPR}_i) = 65.888 \]

- \( [\text{P1}, \text{P2}, \text{P9}] \)
  - \( F = 5 \)
    - \( \text{cr} = 14 + \left(\frac{13}{18}\right) \times 40 = 42.888 \)
- \( [\text{P15}] \)
  - \( F = 3 \)
    - \( \text{cr} = 8.5 \)
  - \( [\text{P27}] \)
    - \( F = 2 \)
      - \( \text{cr} = 6 \)
      - \( [\text{P30}, \text{P25}] \)
        - \( F = 1 \)
          - \( \text{cr} = 6 \)
- \( [\text{P15}] \)
  - \( F = 3 \)
    - \( \text{cr} = 8.5 \)

**Sub HCPR\(_j\)**

\[ \text{cr}(\text{SubHCPR}_j) = 106.785 \]

- \( [\text{P1}, \text{P12}, \text{P20}, \text{P15}, \text{P23}, \text{P28}] \)
  - \( F = 4 \)
    - \( \text{cr} = 25 + \left(\frac{12}{14}\right) \times 80.833 = 94.285 \)
- \( [\text{P33}] \)
  - \( F = 4 \)
    - \( \text{cr} = 12.5 \)

---

62
Fig. 2.10. (a) Two sub HCPR-trees randomly chosen from HCPR$_i$ and HCPR$_j$ (denoted by line rectangles). (b) Construction of sub HCPR$_i$ and sub HCPR$_j$ from the randomly chosen sub HCPR-trees in Fig. 2.10(a). (c) Illustration of Fusion operation (between sub HCPR$_i$ and sub HCPR$_j$ generating HCPR$_k$ (case b)) and credit revision computation.
\[ HCPR_j \]
\[ cr(\text{HCPR}_j) = 177 \]

\[ [P_1, P_2] \quad F = 10 \quad cr = 60 \]
\[ F = 6 \quad cr = 54 \quad [P_3, P_4, P_5] \]
\[ F = 4 \quad cr = 12 \quad [P_3] \]
\[ F = 4 \quad cr = 6 \quad [P_8] \]
\[ F = 3 \quad cr = 9 \quad [P_6] \]
\[ F = 3 \quad cr = 9 \quad [P_16] \]
\[ F = 3 \quad cr = 18 \quad [P_{12}, P_{10}] \]

\[ [P_4, P_5] \quad cr = 12 \]
\[ F = 6 \quad cr = (54/3) \times 2 = 36 \]
\[ F = 4 \quad cr = 12 \quad [P_8] \]
\[ F = 3 + 1 = 4 \quad cr = (18/2) \times 1 + 3 = 12 \]

\[ [P_{12}] \quad F = 3 \quad cr = 9 \]
\[ [P_6] \quad F = 3 \quad cr = 9 \]
\[ [P_{16}] \quad F = 3 \quad cr = 9 \]

\[ [P_{10}] \quad F = 1 \quad cr = 3 \]

\( \text{Fig.2.11. Illustration of Fission operation and credit revision computation.} \)
2.3. Partitioning of The Problem

Huang in his paper [46] has presented a framework for the credit apportionment process where all the rules contributing to the solution path in a problem-solving process are considered equally useful. Therefore the framework does not distinguish a major contributor from a minor contributor in a problem-solving process, as long as both rules contribute in deriving the solution path. A partial solution to this problem of distinguishing a major contributor from a minor contributor in a problem-solving process for the proposed system can be accomplished as below:

The main task $T$ is partitioned into $n$ subtasks (domain dependent), \{ST$_1$,ST$_2$,...,ST$_i$,...,ST$_n$\}, where ST$_i$ is the ith subtask (Fig.2.12). For each subtask ST$_i$ a weight $W_i$ is assigned (on percentage scale) by the environment proportional to its importance (based on the experience of domain expert/knowledge engineer) such that

$$\sum_{i=1}^{n} W_i = 100$$  \hspace{1cm} (9)

2.4. Credit Apportionment Scheme

The problem of assessing responsibility for system behavior was recognized and explored in pioneering work in machine learning conducted by Samuel [68] and termed the credit assignment problem in [59]. The credit assignment scheme

65
Fig. 2.12. Partitioning of the problem (task T). Here HCPR-tree-i represents the set of HCPR-trees involved in the solution path of the subtask ST_i.
presented here is based on Westerdale's [78] genetic reward scheme and Stackhouse and Zeiglar's [73] credit apportionment scheme.

As noted by Westerdale [78], the bucket brigade algorithm is a subgoal reward scheme, that is, a production $p$ is rewarded for placing the system into a state which satisfies the condition part of another production $q$, provided that $q$ leads to payoff. However, if, the subgoals are not well formulated, it may be that $q$ obtains payoff in spite of the help of $p$ rather than because of it (i.e., other states satisfying $q$'s condition may be more desirable). Even under these circumstances, $p$ gets rewarded. Westerdale suggested an alternate scheme, called genetic reward scheme in which the payoff which enters a system is divided in some other way, rather than "trickling down" through the original inference chains. Stackhouse and Zeiglar [73] used a credit apportionment based on Westerdale's [78] genetic reward scheme. Under this scheme the payoff which enters the system is divided according to the frequency of firing of a rule and rule complexity.

The credit apportionment scheme that follows supports the both inference techniques, forward and backward chaining which are suitable for the HCPRs system. The proposed scheme divides the payoff to all HCPRs participating in the solution path according to their importance and complexity. The measure of performance of the HCPR-tree set depends on the utilization of HCPRs and HCPR-trees included in the set. The credit of a HCPR-tree is the accumulated credit of its constituent HCPRs and the credit of a HCPR-tree set is the
accumulated credit of its trees.

In the discussion that follows the following notations and definitions are used for simplicity:

\[ \text{HCPR}_j : \text{The jth HCPR-tree.} \]

\[ \text{HCPR}_j(i,k) : \text{The ith HCPR at the level k in the jth HCPR-tree.} \]

\[ \text{HCPR}_j = \{ \text{HCPR}_j(0,0), \text{HCPR}_j(1,1), \text{HCPR}_j(2,1), \ldots, \text{HCPR}_j(m_j,1) \}
\]

\[ \ldots, \text{HCPR}_j(1,i), \text{HCPR}_j(2,i), \ldots, \text{HCPR}_j(m_i,i), \ldots, \text{HCPR}_j(1,n), \]

\[ \text{HCPR}_j(2,n), \ldots, \text{HCPR}_j(m_n,n) \} \]

where \( m_i \) is the number of HCPRs at the level \( i \) and \( n \) is the number of levels in the HCPR\(_j\) tree. Notice that \( m_i = 1 \) for \( i = 0 \).

\[ \text{HCPR-trees(ST}_i\} = \{ \text{HCPR}_j : j = 1,2,\ldots,n_i \} , \]

where \( n_i \) is the total number of HCPR-trees in the solution path of the ith subtask \( \text{ST}_i \).

\[ \text{HCPRs of HCPR}_j \text{ involved in the solution path} = \{ \text{HCPR}(0),\ldots,\text{HCPR}(i),\ldots,\text{HCPR}(L_j) \} , \]

where \( \text{HCPR}(i) \) is the HCPR fired at the level \( i \) in the HCPR\(_j\) tree, and \( L_j \) is the maximum level reached in HCPR\(_j\) corresponding to the subtask \( \text{ST}_i \) for a given problem \( p \). It is to be noticed that \( \text{HCPR}(i) \in \{ \text{HCPR}_j(1,i),\ldots,\text{HCPR}_j(m_i,i) \} \) and \( \text{HCPR}(i) \) is the child of \( \text{HCPR}(i-1) \).
Complexity of HCPR-tree is one of the parameters involved in calculating the contribution of the related HCPRs. To compute the complexity of a given HCPR, the following parameters have to be taken into consideration with reference to any problem p:

1. \( C_p(i) \): length of the condition part in HCPR(i).
2. \( U_p(i) \): number of censor conditions checked in HCPR(i) such that \( 0 \leq U_p(i) \leq M_i \), where \( M_i \) is the maximum number of censor conditions in the UNLESS clause of HCPR(i).
3. \( (L_j)_p \): level of HCPR tree actually reached in a particular computation such that \( 0 \leq (L_j)_p \leq N_j \), where \( N_j \) is the maximum level in the HCPR tree.
4. The length of the action part of any HCPR is assumed to be fixed and equal to 1.

Now the complexity of the HCPR for the given problem p can be computed using the following formula:

\[
\text{comp}(\text{HCPR}_j)_p = 1 + (L_j)_p + \sum_{i=0}^{(L_j)_p} (C_p(i) + U_p(i))
\]  \hspace{1cm} (10)

The complexity of the HCPR-trees(ST), the set of HCPR-trees involved in the solution path of the subtask ST, can be computed as under:

\[
\text{comp}(\text{HCPR-trees(ST)})_p = \frac{n}{\sum_{j=1}^{n} \text{comp}(\text{HCPR}_j)_p}
\]  \hspace{1cm} (11)
Assuming that \( W_i \) denotes the weight assigned by the environment to the subtask \( ST_i \), the contribution of \( HCPR_j \) (\( HCPR_j \) is in the solution path of \( ST_i \)) would be:

\[
\text{con}(HCPR_j)_p = \frac{\text{comp}(HCPR_j)_p}{\text{comp}(HCPR\text{-trees}(ST_i))_p} \times W_i \quad (12)
\]

The contribution of \( HCPR(i)_p \) corresponding to the \( HCPR_j \) is computed

\[
\text{con}(HCPR(i)_p) = \frac{C(i)_p + U(i)_p}{\sum_{k=0}^{\text{(i)}_p} (C(k)_p + U(k)_p)} \times \text{con}(HCPR_j)_p \quad (13)
\]

Now we can compute the credit assigned to \( HCPR_j(i,k) \) as its accumulated contribution in solving various problems attempted i.e.,

\[
\text{cr}(HCPR_j(1,k)) = \sum_{p \in \rho^*} \text{con}(HCPR_j(i,k))_p \quad (14)
\]

where \( \rho \) is the set of all the problems and \( \rho^* \) is the set of problems solved at any instance with \( HCPR_j(i,k) \) participation. It is to be noted that the contribution \( \text{con}(HCPR_j(i,k))_p \) gets continuously accumulated into \( \text{cr}(HCPR_j(i,k)) \) as the solution to problems in \( \rho \) is attempted repeatedly during various generations.

The credit of \( HCPR_j \) tree is then computed using

\[
\text{cr}(HCPR_j) = \text{cr}(HCPR_j(m_0,0)) + \sum_{k=1}^{N_j} \sum_{i=1}^{m_k} \text{cr}(HCPR_j(i,k)) \quad (15)
\]

In addition, for every \( HCPR \)-tree set a credit is assigned

\[
\text{cr}(HCPR:\text{-tree set}) = \sum_{j=1}^{n} \text{cr}(HCPR_j) \quad (16)
\]
where \( n \) is the number of HCPR-trees in the HCPR-tree set.

2.5. The Proposed Algorithm

The proposed algorithm consists of two main phases. The algorithms corresponding to these two phases are given below.

Phase-I

Input : Master HCPR-tree set

The master set is the initial ad hoc knowledge base which may not be able to solve all the problems but must contain enough conditions, actions, specificity and generality to allow state transitions necessary for an acceptable solutions to all the problems in \( \rho \). However HCPR-trees may not necessarily be optimal, reasonable, or even properly structured.

Output : Two working HCPR-tree sets for each problem

Phase-I is divided into two stages, where in the first stage, two ad hoc HCPR-tree sets are constructed for each problem (i.e., for every subtask), and during the second stage these two ad hoc sets undergo an evolutionary process, thereby improving their ability to solve more problems if possible.

Stage-I

A solution to all problems is attempted using the master HCPR-tree set.
Each solution path is a set consisting of HCPR-trees involved in the solution of the problem. The solution paths of the solvable problems are alternately transferred into TRansit HCPR-tree Sets TRS-I and TRS-II. Two Working Sets WS-I and WS-II are generated by merging (union) sets in TRS-I and TRS-II respectively.

In case the master set fails to solve any of the problems in the problem set, the genetic operators are applied to the master set until at least two problems are solvable so that TRS-I and TRS-II can be initialized.

**Stage-II**

1. A solution to all problems is attempted using the two working sets WS-I and WS-II. The following cases may arise:

   (a) The problem is solvable by both of the working sets.

   (b) The problem is solvable by only one of the two working sets.

   (c) None of the two working sets is able to solve the problem.

   In case (a) the credit of both of the working sets is increased. In the case (b) the credit of the set which is able to solve the problem is increased. For the case (c), the following steps are performed.

(i) Crossovers are tried between TRS-I and TRS-II by choosing one set from each and generating two new sets. If the newly generated set is able to solve the problem, the set is merged into both of the working sets. During
the crossover, any newly generated set which is not able to solve the problem is thrown away.

(ii) If the step (i) fails to produce a solution to the problem, the system attempts to solve the problem using the master HCPR-tree set. The set of HCPR-trees forming the solution path of the problem (from the master HCPR-tree set) is merged into both of the working sets.

Phase-II

Input: Working sets WS-I and WS-II for each problem set and the master HCPR-tree set.

Output: One Mature Set (MS) for each problem. Mature set is the set which is able to solve all the problems in the problem set or it is the better set of the two working sets in terms of the number of problems it can solve.

The system operates in a loop during this phase and the following steps are repeated until all problems are solved or the assigned maximum number of generations is reached.

(i) A solution to all the problems is attempted using WS-I and WS-II. The following cases may arise.

(a) The problem is solvable by both the working sets.

(b) The problem is solvable by only one of the working sets.
(c) None of the two working sets is able to solve the problem.

For case (a) the credit of both of the working sets is increased. For case (b), all of the gain by credit goes to the set with the higher credit. That is, if the set with higher credit solves the problem, its credit is increased, otherwise, the set of HCPR-trees corresponding to the solution path of the problem under the working set with lower credit is merged into the working set with higher credit. In the case (c), the master set is tried to solve the given problem. If the problem is solvable, then the solution path is merged into both WS-I and WS-II.

(ii) After every $x$ generations, all of the HCPR-trees having low credits are removed (i.e., weeding).

(iii) After every generation, the specialized forms of the crossover and mutation operators are applied to the working sets and the master HCPR-tree set.

(iv) After every $y$ generations, the Fusion and Fission genetic operators are applied to WS-I and WS-II.

The integer numbers $x$ and $y$ may be specified on the basis of the computational experience with the system. Satisfying the condition of quitting the loop, the system chooses better of the two working sets WS-I and WS-II (i.e., the one having higher credit) as the final genetically learned HCPR-tree set. This chosen set is called the Mature Set (MS).

Fig.2.13. represents detailed view of the phase-I while Fig.2.14. shows the main loop of phase-II.
Fig. 2.13. Detailed diagram of phase-I.

Fig. 2.14. Detailed diagram of phase-II.